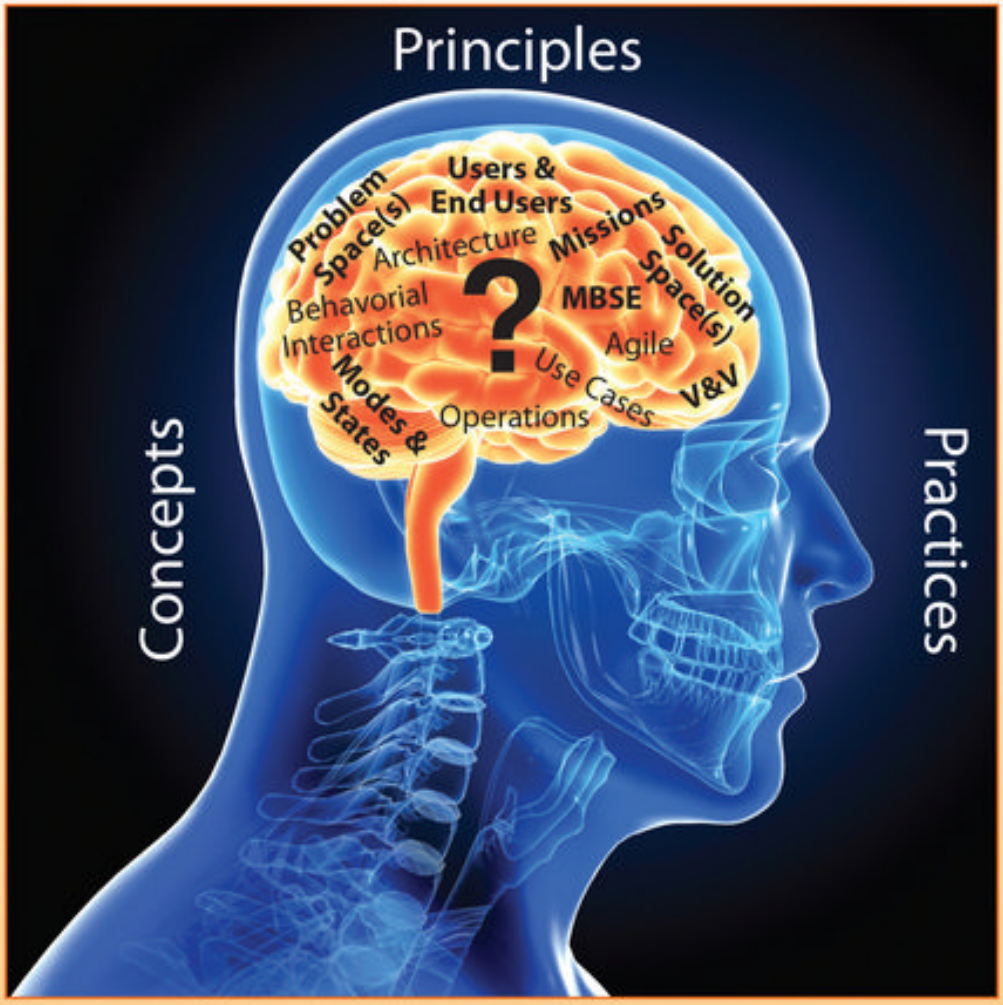
Wiley Series in Systems Engineering and Management
Andrew P. Sage, Series Editor

System Engineering

Analysis, Design, and Development



Charles S. Wasson
Wasson Strategics, LLC

Foreword by Norman Augustine
Former Chairman and CEO – Lockheed Martin Corporation
Former Under Secretary of the Army
Former Member of Princeton Engineering Faculty

WILEY

SYSTEM ENGINEERING ANALYSIS, DESIGN, AND DEVELOPMENT

WILEY SERIES IN SYSTEMS ENGINEERING AND MANAGEMENT

Andrew P. Sage, Editor

A complete list of the titles in this series appears at the end of this volume.

SYSTEM ENGINEERING ANALYSIS, DESIGN, AND DEVELOPMENT

Concepts, Principles, and Practices

CHARLES S. WASSON

WILEY

Published by John Wiley & Sons, Inc., Hoboken, New Jersey
Published simultaneously in Canada

Library of Congress Cataloging-in-Publication Data:

Wasson, Charles S., 1948-
System engineering analysis, design, and development : concepts, principles, and practices / Charles S. Wasson.
pages cm
Revised edition of: System analysis, design, and development. Hoboken, N.J. : Wiley-Interscience, 2005.
Includes bibliographical references and index.
ISBN 978-1-118-44226-5 (cloth)
1. System design. 2. System analysis. I. Title.
QA76.9.S88W373 2015
003–dc23

2014018409

Cover image courtesy of iStockphoto © cgtoolbox

Typeset in 10/12pt TimesLTStd by SPi Global, Chennai, India

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

1 2016

CONTENTS

FOREWORD

Norman R. Augustine

Arguably the most sought-after employees in engineering-oriented firms are systems engineers. This was certainly the case in the firm that I led at the time I led it, and I suspect that at least in this regard not much has changed. Of 82,000 engineers only a tiny fraction could have been categorized as "systems engineers;" nonetheless, they were the individuals who provided the "glue" in building our products and often were the ones that moved into management positions.

But, given the impact of their field, one can't help but ask why such individuals are so rare? One reason is that few universities even offer a degree in "systems engineering." (Most *high schools*, for that matter, don't even offer a *course* in what one might call "engineering.") Another reason is that it requires a rather special talent to cut across a broad set of disciplines—some of which would not even be categorized by most people as "technical." Further, in my experience, the best systems engineers are those who acquired a relatively deep understanding of at least one core discipline before moving into systems engineering. This seems to give them a grounding in dealing with the challenges one encounters when working with complex systems—but it also adds time to the educational process.

Further complicating matters, there is widespread disagreement, even in the profession itself, as to what constitutes "systems engineering." Is it an aspect of management? Does it have to do with the "ilities"—reliability, maintainability and availability? Does it have to do with the acquisition of major systems? Is it the process of conducting trade-offs between alternate approaches to carrying out a function or producing a design? Is it figuring out what something will cost? Is it the process of identifying solutions to a requirement … or is it determining what the requirement should be in the first place?

The answer is, "yes." It is all of these things … and more.

My own rather simplified definition of systems engineering is that it is the discipline of combining two or more interacting elements to fulfill a need. In this book, much greater insight will be given to the answer to this question. Many tools will also be presented that a systems engineer can use to address a broad spectrum of problems in design and analysis—all introduced in an understandable and carefully organized fashion.

One might conclude that with such a simple definition as the one I offer, systems engineering must be a fairly straightforward pursuit. Unfortunately, it is not. Consider the following: the simplest of all systems has only two elements, each of which can influence the other. Perhaps the canonical example would be (putting aside quarks and their cousins) a hydrogen atom. Furthermore, if one limits the interaction between the elements of the system to be binary ("on" or "off") but omni-interactive, it will readily be seen that the number of possible states of a two-element system will be just four. But if one expands the number of elements to merely seven or eight, the number of states virtually explodes.

Making matters still worse, many systems involve humans among their elements, adding unpredictability. All this is what makes it impossible to completely test a complex system in all of its possible states and makes the task of the systems engineer all the more critical. Further, among the humans affecting systems there are usually engineers, many of whom seem to embrace the code that "If it ain't broke … it needs more functions!"

To the designer of a component, say a fuel-control, the fuel-control is a system. Which it is. But to the designer of the jet engine into which the fuel control is incorporated, the

engine is the system. To an aeronautical engineer, the entire aircraft is the system. And it does not stop there … since to an engineer configuring an airline the system includes passengers, agents, airports, air traffic control, runways, and still more—what is often called a system of systems. Fortunately, there are techniques to deal with such challenges in systems of systems and these, too, are described in the pages that follow.

The discipline embodied in sound systems engineering practice can have an important impact on the utility of a system. For example, a few years ago a market survey found that airline passengers wished, among other things, to get to their destinations faster. To an aerodynamicist (my original field) that meant (presuming supersonic flight over land was, at least at that time, impracticable) flying faster, which in turn meant pressing even further against the sudden drag rise that occurs as one approaches the speed of sound. Thus, the effort began to develop a "near-sonic" airliner … a difficult and costly solution.

But systems engineers interpreted the passengers' desire quite differently. They deduced that what passengers really wanted was to get from, say, their homes to an office in a distant city, more rapidly. Decomposing the relevant time-line into such segments as driving to the airport, finding a place to park and clearing security, flying, recovering baggage, and driving to the destination, they concluded that any plausible increase in airspeed would be trivial in its impact on *overall* travel time and that one should focus not on a challenging aerodynamics problem but rather on such matters as expediting security inspection, handling baggage, and speeding ground transportation. The idea of a near-sonic aircraft was thus wisely, if belatedly, discarded.

As noted, this book provides the individual interested in systems engineering with a variety of techniques to deal with such problems, techniques that systems engineers (something into which I "evolved" during my career) in the past largely learned the costly way: O.J.T. Insights will be offered into such important tasks as defining requirements, decomposing requirements, managing software, root-cause analysis, identifying single-point failure modes, modeling, conducting trades, controlling interfaces, and testing for utility as opposed to simply satisfying a specification.

Many of the more important challenges facing America, and in most cases the world, are, in effect, massive systems engineering problems. These include providing healthcare; producing clean, sustainable, affordable energy; preserving the natural environment; growing the economy; providing national security; and rebuilding the nation's physical infrastructure.

In *Systems Engineering Analysis, Design and Development,* Charles Wasson has created a guide for the practitioner. This is not a philosophical treatise or an abstract, theoretical assessment. This is a book that is for the individual who faces every-day, practical challenges relating to the various aspects of systems engineering. It is not only an important teaching device, it is a reference book of lasting value.

The logic and techniques of systems engineering are truly ubiquitous in their applicability. Whether one works in engineering, venture capital, transportation, defense, communications, healthcare, cybersecurity, or dozens of other fields, understanding the principles of systems engineering will serve one well. After all, what is there in life that doesn't involve two or more elements that influence each other?

NORMAN R. AUGUSTINE

Retired Chairman & CEO, Lockheed Martin Corporation
Former Under Secretary of the Army
Former Member of Princeton Engineering Faculty

PREFACE TO THE SECOND EDITION

Welcome to the Second Edition of *System Engineering, Analysis, Design, and Development* written for anyone who is accountable for specifying, analyzing, designing, and developing systems, products, or services. This Second Edition is a landmark text intended to take System Engineering (SE) to new levels of 21st-Century System Thinking. Systems Thinking that goes beyond what some refer to as "outdated, old school, and parochial" paradigms such as "Engineering the (Hardware/Software) Box" promulgated by institutions and Enterprises.

Traditional "Engineering the Box" mindsets fail to approach SE&D from the standpoint of "Engineering the System" based on User capabilities and limitations. Contrary to public perceptions, system failures are often attributable to poor System Design – "Engineering the Box" – that influences "human error" publicized as the "root cause." The reality is system failures are typically the result of a series of latent defects in the System "Box" Design that lie dormant until the right set of enabling circumstances occur and proliferate via a chain of events culminating in an incident or accident. This text goes beyond traditional "Engineering the Box" and fosters Systems Thinking to broaden insights about how to "Engineer the System" and the "Box."

The Systems Engineering *concepts*, *principles*, *practices*, and problem-solving and solution-development *methods* presented in this text apply to any discipline irrespective of type of discipline. This includes:

1. System Engineers (SE);
2. Multi-discipline Engineers—Electrical, Mechanical, Software, BioMedical, Nuclear, Industrial, Chemical, Civil, and others.
3. Specialty Engineers—Manufacturing, Test, Human Factors (HF); Reliability, Maintainability, and Availability (RMA); Safety; Logistics; Environmental, and others.
4. System and Business Analysts.
5. Quality Assurance (QA) and Software QAs.
6. Project Engineers.
7. Project Managers (PMs).
8. Functional Managers and Executives.

System Engineering Analysis, Design, and Development is intended to fill the SE void in Engineering education and to provide the concepts, principles, and practices required to perform SE. Based on the bestselling, international award-winning First Edition, this Second Edition builds on those foundational concepts, principles, and practices. This text has three key objectives:

1. To help educate Engineers who have a vision of becoming an SE or a better SE through course instruction or self-study.
2. To equip discipline Engineers and System Analysts—EEs, MEs, SwEs, etc.—with SE *problem-solving and solution development methods* that help them better understand the *context* of their work within the overall framework of the system, product, or service they are Engineering.
3. To provide Project Managers (PMs) with an understanding of SE & Development (SE&D) to facilitate better project integration with Engineering.

During the past 70 years, Systems Engineering has evolved from roots in fields such Aerospace and Defense (A&D) and proliferated into new business domains such as energy; medical products and healthcare; transportation

—land, sea, air, and space; telecommunications; financial, educational, and many others. Worldwide awareness and recognition of Systems Engineering, its application, and benefits have placed it at the forefront of one of the most sought-after fields of study and employment. In 2009, *Money Magazine* identified Systems Engineering as #1 in its list of Best Jobs in America with a 10-year job growth forecast of 45%. Besides being #1, the criticality of this profession is in stark contrast to the second place job, which had a 10-year job growth projection of 27%.

Despite its rapidly expanding growth potential, Systems Engineering is still *evolving* in terms of its methodology, discipline, and application by Users. Its application in many Enterprises is often *ad hoc*, *experiential,* and characterized by semantics and methods that often exist in lofty marketing brochures and websites. Yet, produce limited objective evidence that SE has been performed. Based on the author's experience:

- Most Engineers, in general, spend from 50% to 75% of their total career hours on average making systems decisions for which they have little or no formal Systems Engineering coursework.
- Less than 3% of the personnel—one person out of 30—in most Enterprise SE organizations possess the requisite knowledge of the concepts, principles, and practices identified in this text.
- What most people and Enterprises perceive to be System Engineering (SE) is actually an *ad hoc, trial-and-error, endless loop*, Specify-Design-Build-Test-Fix (SDBTF) Paradigm. Compounding the problem is the fact that embedded within the SDBTF Paradigm is another *trial-and-error endless loop* Design Process Model (DPM) documented in the 1960s. Users of the SDBTF-DPM paradigm acknowledge it is *inconsistent*, *inefficient*, *ineffective*, and *chaotic* in developing systems, products, or services. Yet, they continue to employ it despite the fact that it is *not scalable* to moderate or large, complex systems projects.
- The underlying rationale to the SDBTF-DPM Paradigm is that since they have used it to to develop "systems," it must be—by definition—Systems Engineering. Based on those misperceptions, the SDBTF-DPM Paradigm becomes the "core engine" within an SE "wrapper." When the SDBTF-DPM Paradigm is applied to System Development and the deliverable system fails or the customer does not like the system, they *rationalize* the root cause to be … Systems Engineering.
- Within these Enterprises anyone who has electrically, electronically, or mechanically integrated two hardware components or compiled two software modules is "knighted" as a Systems Engineer by their manager whether they have exhibited SE skills and competence in the discipline or not … everyone is "Systems Engineer."

Frightening isn't it! Unfortunately, executives and managers are often unaware or refuse to acknowledge the existence of the SDBTF-DPM Paradigm as the defacto "SE Process" within their Enterprise.

These Enterprises are easily identifiable. Ignoring or oblivious to the problem, executives and managers will challenge the SDBTF-DPM Engineering Paradigm observation. They recite metrics that quantify how they have trained XX personnel in an SE short course, YY personnel obtained a Master's degree in SE or higher, and ZZ personnel have been certified as Systems Engineering professionals within the past year. This is mindful of an old cliché that "owning a paint brush does not make someone an artist."

Despite their proclamations, project performance issues traceable to a lack of true SE education or SE courses in Engineering education refute any evidence to the contrary. There are, however, Enterprises and professionals who do understand SE and perform it reasonably well. *What are the differences between Enterprises that perform SE well versus those where the SDBTF-DPM Engineering Paradigm thrives?*

First, SE knowledge is often learned *experientially* through personal self-study and "On-the-Job Training" (OJT). Despite its significance as a critical workplace Engineering skill, the fundamentals of SE are not taught as a course in most undergraduate Engineering programs. Undergraduate or graduate level courses that are labeled as SE: (1) often focus on Systems Acquisition and Management or (2) specialty engineering equation-based courses. These courses are fine … when staged and sequenced … after … a strong, requisite foundation in understanding what a system is coupled with the ability to perform the problem-solving and solution development to actually develop a system.

Secondly, Engineering has always wrapped itself in a cloak of equations; that's the perception of Engineering by many. 21st Century System Engineering and Development (SE&D) in industry and government demands a combination of *problem-solving and solution-development* decision-making *soft skills* that precede, enable, and facilitate the equation-based *hard skills*. Engineers erroneously "knighted" as SEs who spend their days *plugging* and *chugging* equations either have a highly specialized instance of SE, misplaced priorities, or simply do not understand what is required to develop a system on-schedule, within budget, and compliant with its technical requirements!

Missing is the requisite knowledge Engineers and System Analysts need to serve as a foundation for transforming a User's abstract operational needs into the physical realization of a system, product, or service. Most SEs will emphatically

state that is what they do. However, their *misinformed* perception of SE and actions reveal that they typically take a *quantum leap* from requirements directly to a physical solution and implementation (Figure 2.3). Due to a lack of a true system problem-solving and solution development methodology and skills, these efforts often result in failure or fall short of technical performance, especially in complex systems, and compliance to specification requirements.

Foundational SE knowledge requires competence in the following areas: (1) understanding and applying SE concepts, principles, and practices; (2) applying a proven *problem-solv*ing and *solution-development* methodology; (3) scaling SE practices to meet project resource, budget, schedule, and risk constraints; (4) structuring and orchestrating technical projects; and (5) leading multi-discipline Engineering and other types of System Development decision-making teams. Where true SE knowledge and skills are lacking, Engineering evolves into an *ad hoc*, *endless loop* of Build-Test-Fix with the peception that "if we create a design and tweak it enough, sooner or later we will get it right."

To address these and other issues, industry, government, and professional Enterprises have made great strides in establishing standards, Enterprise capability assessments, certification programs, etc. These are certainly *essential* for any type of discipline. However, they do not solve the *root problem* that exists concerning the lack of foundational SE knowledge. Specifically, *shifting*—correcting—the SDBTF-DPM Paradigm that permeates Enterprises, projects, and individual thinking due to a lack of substantive SE education beginning at the undergraduate level. *How do we solve the problem*?

This Second Edition builds on the author's experiences and incorporates readers' and instructors' feedback as well as advancements in SE. This includes (1) leading-edge topics and methods that enhance your knowledge and (2) provides a framework that supports pursuit of professional certification provided by organizations such as International Council on Systems Engineering (INCOSE), Enterprise level Capability Maturity Model Integration (CMMI) Assessments, and Enterprise Organizational Standard Processes (OSPs) traceability to ISO standards.

KEY FEATURES

Textbooks often include a "principles of" subtitle as marketing claims to lure readers. Readers read these texts from cover to cover and discover the lack of explicitly stated principles despite the claims. This text delivers on its Concepts, Principles, and Practices subtitle. The Second Edition:

- Includes approximately 365 principles, 231 examples, 148 author's notes, and 21 mini case studies that exemplify how to apply SE to the real world.
- Facilitates readability and quick location of key points of information based on icon-based visual aids used to highlight principles, heuristics, author's notes, mini case studies, cautions, warnings, etc.
- Consists of two levels of end-of-chapter exercises for undergraduate and graduate level course instructions: Level 1 Chapter Knowledge Exercises and Level 2 Knowledge Application Exercises.

Textbooks are often a one-time reading and disposed of on completion—donated to a library, sold back to a bookstore, or given away. It is the author's intent for this text to serve as a personal desk reference throughout your professional career subject to the evolution of SE standards, updates, etc. Professions, industries, and individuals evolve and inevitably change over time; however, fundamental systems concepts stand the test of time.

In summary, *System Engineering Analysis, Design, and Development* provides foundational SE knowledge based on the author's experience tempered by over 40 years in industry with some of the world's leading SE Enterprises and private consulting with small, medium, and large corporate clients. The next step is up to you and your Enterprise. Leverage these concepts, principles, and practices to achieve the next level of performance. Learn to *competently scale* this knowledge along with your own unique experience to meet each project's technical, resources, technology, budgetary, schedule, and risk constraints.

ACKNOWLEDGMENTS

Projects such as this demand a lot of time, inspiration, and support from a community of professionals that strive to advance the practice of the Systems Engineering. My sincere gratitude and appreciation to colleagues, instructors, authors, mentors, friends, and family who contributed to the success of the 1st Edition and to this 2nd Edition.

- Dr. Norman R. Augustine for his visionary leadership in Engineering worldwide and preparation of the Foreword.
- Readers, colleagues, instructors, and authors, – Brian Muirhead – Prof. Heinz Stoewer, Dr. Sven Bilen, Dr. Ricardo Pineda, Dr. Adeel Khalid, Dr. Kamran Moghaddam, Joe Stevens, Michael Vinarcik, Mark Wilson, Rebecca Reed, Dr. Marita O'Brien, Dr. Bud Lawson, Neill Radke, Christopher Unger, Matthew Eash, Dr. Kevin Forsberg, David Walden, James Highsmith, Dr. Barry Boehm, Dr. Rick Dove, Dr. Eric Honour, Dr. Bohdan Oppenheim, Dr. Stavros Androulakakis, Cork Heyning, David Swinney, John Bartucci, Charles Anding, Eileen Arnold, Dorothy McKinney, William D. Miller, Garry Roedler, James

Sturges, Sandy Friedenthal, Michael W. Engle, Dr. Terry Bahill, Dr. Herman Migliore, and Dr. Eric D. Smith.

- Mentors – Dr. Charles Cockrell, Dr. Gregory Radecky, Bobby Hartway, Danny Thomas, Dr. William H. McCumber, Dr. Wolter J. Fabrycky, Benjamin S. Blanchard, Dr. Tom Tytula, William F. Baxter, Dan T. Reed, Chase B. Reed, Bob Jones, and Kenneth King.
- Wasson Strategics, LLC's many valued clients and their visionary pursuit of Systems Engineering and Development (SE&D) excellence.
- Administrative and Clerical Support – Jean Wasson and Sandra Hendrickson for their outstanding work and performance, systems thinking, and focus on achieving excellence.
- Research Support – David Moore and his University of Alabama - Huntsville (UAH) Library staff – Michael Manasco and Doug Bolden, Victoria Hamilton – Clemson University Library, and Pam Whaley – Auburn University Library.
- John Wiley & Sons, Inc. – Brett Kurzman (Editor – 2nd Edition), Kari Kapone – Manager, Content Capture, and Alex Castro – Senior Editorial Assistant; Vishnu Priya (Production Editor); Lincy Priya (Project Manager) and team members; Nicole Hanley (Marketing); and George Telecki (Editor – 1st Edition).

Most of all … the 1st and 2nd editions of this text could not have been possible without my wife who provided reviews, comments, and unwavering support throughout this challenging project.

To Jean, my love … my sincere gratitude and appreciation … always and forever!

In summary, thank you—the readers, instructors, and professionals—for the opportunity to shift traditional, narrow-scoped, Engineering and Systems Engineering paradigms to a new level of multi-disciplined 21st Century System Thinking.

Charles Wasson
Wasson Strategics, LLC
August, 2015

ABOUT THE COMPANION WEBSITE

This book is accompanied by a companion website:

www.wiley.com/go/systemengineeringanalysis2e.

- Level 2 Knowledge Application Exercises
- Instructional Materials
  - Chapter-Based Learning Objectives
  - Links to News Articles and Technical Papers

INTRODUCTION—HOW TO USE THIS TEXT

System Engineering Analysis, Design, and Development by virtue of its broad application to any type of Enterprise or Engineered system, product, or service is written for Engineers—Hardware, Software, BioMedical Specialty, Test, Chemical, Nuclear, etc., System Analysts, Project Engineers, Project Managers, Functional Managers, and Executives who strive to achieve System Engineering & Development (SE&D) excellence. Across that spectrum are Engineers and Analysts who may be new to SE, simply interested in learning more about SE methods to apply to their own Engineering disciplines, or seasoned professionals who want to improve and advance the state of the practice of their existing skills.

This text is written to accommodate a broad range of audiences. Writing to fulfill the needs of readers across a diverse spectrum of disciplines can be challenging. Readers who are new to SE request detailed discussions; seasoned professionals request less discussion. To accommodate such a diverse audience with varying levels of knowledge and skills, this text attempts to achieve a *reasonable balance* between communicating *essential* information about SE concepts, principles, and practices while limiting the depth due to page count limitations. As a result, our discussions will *drill down* to a particular level and provide resource referrals for you to pursue via your own personal study.

SCOPE OF TEXT

Due to the broad range of *technical* and *managerial* activities required to perform Systems Engineering and Development (SE&D) coupled with the need to limit the page count, the scope of this text focuses primarily on the *technical* aspects of SE.

As its name communicates, this text is about *System Engineering Analysis, Design, and Development: Concepts, Principles, and Practices*. This text is not about designing integrated circuits or electronic circuit boards or selecting physical components—resistors, capacitors, etc. or their deratings; design of mechanical structures and mechanisms; design and coding of software; Modeling or Simulation (M&S); developing mathematical algorithms, etc. Instead, it provides the SE concepts, principles, and practices that are *essential* for discipline-based Engineers and Analysts who perform those activities to better understand the *context* of their work products in terms of its Users, requirements, architecture, design, trade-offs, etc.

PRIMARY STRUCTURE

System Engineering Analysis, Design, and Development is partitioned into three parts:

- Part 1—System Engineering and Analysis Concepts
- Part 2—System Design and Development Practices
- Part 3—Decision Support Practices

Be Advised: Each part has a specific purpose, scope, and interrelationship to the other parts as illustrated in Figure I.1. However, for purposes of this description, to understand *why* Part 1 exists, we need to first understand the scopes of Parts 2 and 3.

Part 2—System Design and Development Practices

Part 2—System Design and Development Practices—addresses multi-discipline SE *workflow* activities

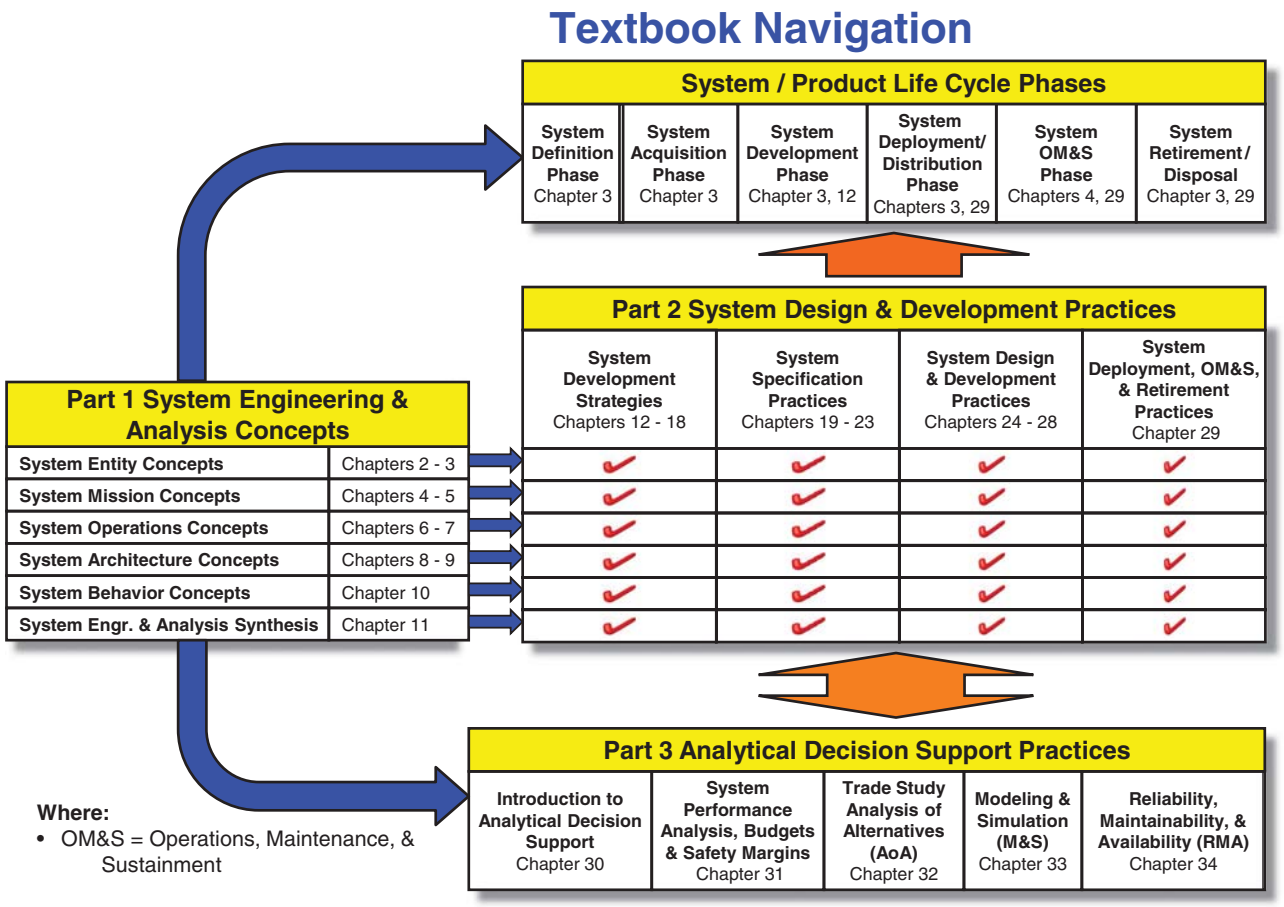


Figure I.1 Book Structure and Interrelationships

and practices required to engineer, develop, and deliver systems, products, or services. Part 2 is partitioned into four sets of SE practices that address how SE is performed not only during SE&D but also during operations performed by the User after delivery—System Deployment; Operations, Maintenance, & Sustainment (OM&S); and Retirement/Disposal. These include:

System Development Strategies

- Chapter 12 Introduction to System Development Strategies
- Chapter 13 System Verification and Validation (V&V) Strategy
- Chapter 14 The Wasson System Engineering Process
- Chapter 15 System Development Process Models
- Chapter 16 System Configuration Identification and Component Selection Strategy
- Chapter 17 System Technical Documentation Strategy
- Chapter 18 Technical Reviews Strategy

System Specification Practices

- Chapter 19 System Specification Concepts
- Chapter 20 Specification Development Approaches
- Chapter 21 Requirements Derivation, Allocation, Flow Down, and Traceability
- Chapter 22 Requirements Statement Development
- Chapter 23 Specification Analysis

System Design and Development Practices

- Chapter 24 User-Centered System Design (UCSD)
- Chapter 25 Engineering Standards of Units, Coordinate Systems, and Conventions
- Chapter 26 System and Entity Architecture Development
- Chapter 27 System Interface Definition, Analysis, and Control
- Chapter 28 System Integration, Test, and Evaluation (SITE)

System Deployment; Operations, Maintenance, and Sustainment (OM&S), and Retirement Practices

- Chapter 29 System Deployment; Operations, Maintenance, & Sustainment (OM&S); and Retirement

Part 3—Decision Support Practices

PART 3—DECISION SUPPORT PRACTICES—addresses multi-discipline SE practices such as System Analysis, Reliability, Maintainability, Human Factors, safety, etc. required to provide *timely* and *effective* decision support to the Part 2 decisionmaking practices. This includes development and assessment of rapid prototypes, Modeling and Simulations (M&S); proof-of-concept, proof-of-principle, and proof-of-technology demonstrations, to derive data to support SE&D decisions as well as to validate models and simulations. Part 3 is partitioned into the following chapters:

- Chapter 30 Introduction to Analytical Decision Support
- Chapter 31 System Performance Analysis, Budgets, and Safety Margins
- Chapter 32 Trade Study Analysis of Alternatives (AoA)
- Chapter 33 System Modeling & Simulation (M&S)
- Chapter 34 System Reliability, Maintainability, and Availability (RMA)

This brings us to the purpose of Part 1.

Part 1—System Engineering and Analysis Concepts

Most Enterprises, organizations, and projects perform the practices addressed in Parts 2 and 3. The problem is they employ the SDBTF-DPM Engineering Paradigm that is acknowledged to be *ad hoc, inefficient,* and *ineffective* as evidenced by poor performance in performing Parts 2 and 3. The reality is this is a *problem space*. PART 1—SYSTEM ENGINEERING AND ANALYSIS CONCEPTS—provides a *solution space* framework for *shifting* the SDBTF-DPM Engineering Paradigm to correct these shortcomings and serves as the foundational knowledge required to competently perform the practices in Parts 2 and 3.

Part 1 is partitioned into primary concepts that enable you to understand: *what* a system is: *why* it exists—missions; *how* the User envisions deploying, operating, maintaining, sustaining, retiring, and disposing of the system; *how* systems are structured architecturally; and *how* the User envisions system behavioral responses to mission interactions in its OPERATING ENVIRONMENT. Part 1 SYSTEMS ENGINEERING AND ANALYSIS CONCEPTS consists of the following:

System Entity Concepts

- Chapter 1 Systems, Engineering, and Systems Engineering
- Chapter 2 The Evolving State of SE Practice–Challenges and Opportunities
- Chapter 3 System Attributes, Properties, and Characteristics

System Mission Concepts

- Chapter 4 User Enterprise Roles, Missions, and System Applications
- Chapter 5 User Needs, Mission Analysis, Use Cases, and Scenarios

System Operations Concepts

- Chapter 6 System Concepts Formulation and Development
- Chapter 7 System Command and Control (C2) -Phases, Modes, & States of Operation

System Architecture Concepts

- Chapter 8 System Levels of Abstraction, Semantics, and Elements
- Chapter 9 Architectural Frameworks of the SOI & Its OPERATING ENVIRONMENT

System Behavior Concepts

- Chapter 10 Modeling MISSION and ENABLING SYSTEM Operations

System Engineering & Analysis Synthesis

- Chapter 11 Analytical Problem-Solving and Solution Development Synthesis

How to Use This Text

If you are reading this text for the first time … regardless of SE experience, you are encouraged to follow the sequence of Parts 1–3 and Chapters as sequenced. Understanding PART 1 SYSTEM ENGINEERING & ANALYSIS CONCEPTS is the critical foundation for understanding Parts 2 and 3. Figure I.1 serves as a roadmap for quickly locating and navigating the chapters.

Once you have read the text, you will be performing project work addressed in Part 2 SYSTEM DESIGN AND DEVELOPMENT PRACTICES or PART 3 ANALYTICAL DECISION SUPPORT PRACTICES. Figure I.1 facilitates navigation in the text by enabling you to easily refer back to more detailed discussions in the other Parts.

Undergraduate and Graduate Level Course Instruction

This textbook is structured to accommodate both *upper level undergraduate* and *graduate level* Engineering and other

courses. Depending on the (1) students and (2) the instructor's knowledge, skills, and industry experience, this text has also been designed to accommodate as much technical depth as the instructor wants to achieve. The instructor can treat the material as introductory or drill deeply into challenging topics.

Chapter Features

System Engineering Analysis, Design, and Development: Concepts, Principles, and Practices has been designed to incorporate specific features to facilitate readability and searches. In general, the textbook employs a common outline sequence of topics in each chapter.

- Chapter Introduction
- Definitions of Key Terms
- Approach to the Chapter (where necessary)
- Sectional Details and Discussions of Chapter Topics
- Chapter Summary
- Chapter Exercises
- References

Let's address some of the details of these features.

Definitions of Key Terms

The introduction to each chapter consists of Definitions of Key Terms that are relevant to the Chapter's discussion. Some of the definitions originate from military handbooks and standards. If you work in energy, medical, transportation, telecommunications fields, *avoid* the notion that these are not applicable to your work. As an SE, System Analyst, or Engineer, the mark of a true SE professional is the ability to work across business domains, understand the context of usage of definitions, their application, and what is to be accomplished. If your business domain or Enterprise has its own standards and definitions, *always* employ those in your work unless there is a compelling, authoritative reason to do otherwise.

Icon-Based Breakouts for Principles, Heuristics, Author's Notes, Examples, and Mini-Case Studies

SE, like most disciplines, is characterized by key points that are worthy of consideration. This includes principles, heuristics, author's notes, examples, and mini-case studies.

To facilitate readability, this text employs icon-based breakouts that also serve as easily identifiable navigation landmarks for referencing other chapters. For example, icons are encoded with XX.Y *syntax* reference identifiers where XX represents the chapter number and Y represents a sequential number indexed from the beginning of the chapter. Principle 12.4 represents Chapter 12 Principle #4, and so forth. The following is a list of icons for identifying principles, heuristics, author's notes, examples, cautions, and warnings that use this approach.

Principles represent a truth or law that governs reasoning and serves as a guide for action.

Principle I.1

Heuristic I.1 Heuristics represent "rules of thumb" that are not rigid but do provide insightful guidance that is worthy of consideration. As such, Heuristics are subject to exceptions depending on the circumstances.

Author's Note I.1

Author's Notes provide observations that highlight *subtle* or *noteworthy* aspects of a discussion concerning the context, interpretation, or application of a concept, principle, or practice. Your experiences may be different. You, your team, project, and Enterprise are wholly accountable for the decisions or lack of decisions you make and their consequences.

Examples illustrate a situation or practical application of a principle, heuristic, or SE practice.

Example I.1

Mini-Case Study I.1

Mini-Case Studies provide a brief description of a real-world situation, event, or incident that illustrates a principle, heuristic, example, or key point relevant to a topical discussion.

Reserved Words

Reserved words have unique *contexts* that differentiate them from general usage. For these terms, the text uses SMALL CAPS. For example, there is a *contextual* difference in referring to your SYSTEM (small caps) … versus … generic systems, products, or services (regular font). Reserved words occur in three categories of usage:

System Levels of Abstraction

Your SYSTEM, PRODUCT, SUBSYSTEM, ASSEMBLY, SUBASSEMBLY, and PART Levels.

System Types

Types of systems such as a System of Interest (SOI) composed of one or more Mission Systems and one or more Enabling Systems.

Environments

Your OPERATING ENVIRONMENT consisting of a NATURAL ENVIRONMENT, INDUCED ENVIRONMENT, or PHYSICAL ENVIRONMENT.

A Word of Caution I.1

Cautions are informed awareness notifications concerning conditions that represent potential risks that require special consideration. Remember—You, your team, project, and Enterprise are wholly accountable for the decisions or lack of decisions you make and their consequences.

Warning I.1

Warnings are risk-based situations that demand special attention, awareness, and recognition of decisions or conditions related to safety as well as statutes, regulations, ethics, etc. established by international, national, state, and local governments and organizations that carry severe penalties for violation. Remember – You, your team, project, and Enterprise are wholly accountable for the decisions or lack of decisions you make and their consequences.

CHAPTER EXERCISES

Chapter Exercises are provided in two forms:

- **Level 1 Chapter Knowledge Exercises**—Represent *essential* knowledge you should have learned from the chapter.
- **Level 2 Knowledge Application Exercises**—Represent *upper undergraduate level* and *graduate-level* exercises that challenge the reader's ability to apply Chapter knowledge to real world systems, products, or services. Level 2 Exercises are located on the text's companion website located at: www.wiley.com/go/systemengineeringanalysis2e

APPENDICES

This text consists of three Appendices:

- **Appendix A—Acronyms and Abbreviations**—Provides an alphabetic listing of acronyms and abbreviations used in the text.
- **Appendix B—INCOSE Handbook Traceability**—Provides a traceability matrix that links to the International Council on Systems Engineering (INCOSE) *Systems Engineering Handbook* (SEHv4, 2015) to chapters within this text.
- **Appendix C—Systems Modeling Language (SysML™) Constructs**—Provides a brief overview of SysML™ constructs used in the text. SE employs the Object Management Group's (OMG) Systems Modeling Language™ (SysML™), an extension of the OMG's Unified Modeling Language (UML™), to model Enterprise and Engineered systems, products, and services. This text uses some of the SySML™ features to illustrate SE and Analysis concepts.

As its title conveys, this text is about *System Engineering Analysis, Design, and Development*, not SysML™; that is a separate text and course. However, to facilitate your understanding, Appendix C provides a brief overview of SysML™ constructs used in figures of this text. For more detailed information about SysML™, refer to the OMG's website.

- *Note:* UML™ and SysML™ are either registered trademarks or trademarks of Object Management Group (OMG), Inc., in the United States and/or other countries.

SUMMARY

Now that we have established *How to Use the Text*, let's begin with Chapter 1 SYSTEMS, ENGINEERING, AND SYSTEMS ENGINEERING.

1

SYSTEMS, ENGINEERING, AND SYSTEMS ENGINEERING

Principle 1.1

SE Alpha–Omega Principle

SE *begins* and *ends* with the Users of a system, product, or service.

Have you ever purchased a commercial hardware and/or software product; contracted for development of a system, product, or service; or used a website and discovered that it:

- May have complied with its specification requirements but was not what you wanted, needed, or expected?
- Was difficult to use, unforgiving in accepting User inputs and errors, and did not reflect your thought patterns of usage?
- Consisted of an overwhelming number of *non-essential* features that were so distracting it was difficult to navigate?
- Buried commonly used features under several layers of linkable structures requiring numerous mouse clicks to reach and invoke?
- Has software updates that are incompatible with standard operating systems. The System Developer's customer service response was to post a question in an online "community forum." Then, wait (potentially forever) for some other "community User" to offer a solution as to how they solved the System Developer's problem?

Then, in frustration, you and millions of other Users question whether the System Developer and its designers ever bothered to communicate with and listen to the Users or marketplace to understand and comprehend:

- The jobs or missions the User is expected to perform to deliver the system's outcomes to their customers
- How the User expects to deploy, operate, maintain, sustain, retire, or dispose of the systems, products, services, or by-products required to perform those jobs or missions

Welcome to Systems Engineering (SE)—or more appropriately the lack of SE. If you talk with Users such as the ones in the examples above, you will often hear comments such as:

- Company XYZ needs to do a better job "Engineering" their systems, products, or services!
- System ABC needs some "SE!"

From an SE perspective, what emerges from a distillation of User comments are questions such as: *What is SE*? Answering this question requires understanding (1) *what is a system* and (2) *what is Engineering*. Then, *what is the interrelationship between Engineering and SE*?

System Engineering Analysis, Design, and Development: Concepts, Principles, and Practices, Second Edition. Charles S. Wasson.

Companion Website: www.wiley.com/go/systemengineeringanalysis2e

Opinions vary significantly for definitions of these terms and their context of usage. Industry, government, academia, professional, and standards organizations have worked for years to reach consensus definitions of the terms. To achieve a consensus—global in some cases—the wording of the definitions becomes so diluted and abstract that it has limited utility to the User communities the organizations serve. In some cases, the abstractness distorts User perceptions of what the terms truly encompass. For example, the definition of a *system* is a classic example.

The problem is exacerbated by a general lack of true Systems Engineering & Development (SE&D) courses that focus on *problem-solving and solution development* methods and Engineering. Unfortunately, many of the so-called SE courses focus on: (1) System Acquisition & Management - how to manage the acquisition of systems and (2) equation-based courses – "Engineering the box," not the system. This results in a major deficiency in Engineering knowledge and skills required to actually transform a User's abstract, operational need into the Engineering of a physical system, product, or service that meets those needs. *Should there be any surprise as to why User frustrations with systems, products, or services highlighted above occur*?

Given this backdrop, Chapter 1 establishes the foundational definitions for understanding what it means to perform SE addressed in Chapters 2–34.

1.1 DEFINITIONS OF KEY TERMS

- **Capability**—An explicit, inherent feature *initiated* or *activated* by an external stimulus, cue, or excitation to perform an action (function) at a specified level of performance until terminated by external commands, timed completion, or resource depletion.
- **Engineering**—"[T]he profession in which knowledge of the mathematical and natural sciences gained by study, experience, and practice is applied with judgment to develop ways to utilize economically the materials and forces of nature for the benefit of mankind" (Prados, 2007, p. 108).
- **Entity**—A generic term used to refer to an operational, logical, behavioral, physical, or role-based object within a system. Physical entities, for example, include PERSONNEL; EQUIPMENT items such as SUBSYSTEMS, ASSEMBLIES, SUBASSEMBLIES, or PARTS comprised of HARDWARE and/or SOFTWARE; PROCEDURAL DATA such as User's guides and manuals; MISSION RESOURCES such as *consumables* (water, fuel, food, and so on) and *expendables* (filters, packaging, and so on); and SYSTEM RESPONSES—performance-based outcomes—such as products, by-products, or services or FACILITIES.
- **Environment**—A general, context-dependent term representing the NATURAL, HUMAN SYSTEMS, or INDUCED Environments that in which a SYSTEM or ENTITY of Interest must operate and survive.
- **Ilities**—Specialty Engineering disciplines such as Reliability, Maintainability, and Availability (RMA); Sustainability; Safety; Security; Logistics; and Disposal.
- **System**—An integrated set of interoperable elements or entities, each with specified and bounded capabilities, configured in various combinations that enable specific behaviors to emerge for Command and Control (C2) by Users to achieve performance-based mission outcomes in a prescribed operating environment with a probability of success.
- **System Engineering (SE)**—The multidisciplinary application of analytical, mathematical, and scientific principles to formulating, selecting, developing, and maturing a solution that has acceptable risk, satisfies User operational need(s), and minimizes development and life cycle costs while balancing Stakeholder interests.

1.2 APPROACH TO THIS CHAPTER

Our approach to this chapter focuses on defining SE. Since the term SE is comprised of *System* and *Engineering*, we begin with establishing definitions for both of these terms as a precursor for defining SE.

Most definitions of a *system* are often too abstract with limited utility to the User. This text defines a *system* in terms of its attributes and success criteria—what a system is, why it exists, its compositional structure, what it accomplishes, under what conditions, and User expectations for success. Although systems occur in a number of forms such as Enterprise, social, political, and equipment, we focus on Enterprise and Engineered Systems.

One of the challenges in discussing systems is the need to differentiate systems, products, or services. We address those differences and relationships and provide examples. When systems are developed, they may be (1) new innovations (*unprecedented* systems) based on new or emerging technologies or (2) improvement on existing systems or technologies (*precedented* systems). We address the contexts of *unprecedented* versus *precedented* systems.

Based on establishment of what a *system* is, we introduce a commonly accepted definition of *Engineering* and then derive the definition of SE used in this text. Since people often are confused by the usage of *System* versus *SE*, we delineate the context of usage for these terms.

An introduction to SE is incomplete without some form of background description of its history. Rather than repeating

a set of dates and facts that have been documented in other texts, a more important point is understanding what has driven the evolution of SE. To address this point, we introduce an excellent source of SE history for those who want more detailed information. We elaborate this topic in more detail in Chapter 2.

Finally, we close Chapter 1 with a discussion of a key attribute of Systems Engineers—Systems Thinking.

Before we begin, a brief word concerning individuals and teams crafting statements—definitions, specification requirements, and objectives—to achieve consensus agreement is as follows:

When people or organizations develop definitions, attempts to simultaneously create *content* and *grammar* usually produce a result that only has a degree of acceptability. People typically spend a disproportionate amount of time on *grammar* and spend very little time on substantive *content*. We see this in development of plans and specifications, for example. Grammar is important, since it is the root of any language and communications. However, grammar is simply a mechanism to convey: (1) *content* and (2) *context*. "Word-smithed" grammar has *little or no value if it lacks substantive content or context*.

You will be surprised how animated and energized people become during grammar "word-smithing" exercises. After protracted discussions, others simply walk away from the chaos. For highly diverse terms such as a *system*, a good definition may begin with simply a bulleted list of descriptors concerning what a term *is* or *is not*. If you or your team attempt to create a definition, perform one step at a time. Obtain consensus on the key elements of *substantive content*. Then, structure the statement in a logical sequence and translate the substantive content into a grammar statement.

Let's begin our discussion with *What Is a System*?

1.3 WHAT IS A SYSTEM?

Merriam-Webster (2014) states that the term *system* originates from "late Latin *systemat-, systema,* from Greek *systēmat-, systēma,* from *synistanai* to combine, from *syn-* + *histanai* to cause to stand." Its first known use was in 1603.

There are as many definitions of a *system* as there are opinions. Industry, government, academia, and professional organizations over many decades have worked on defining what a system is in their context. If you analyze many of these definitions, most of the definitions have become so diluted due to "wordsmithing" to achieve a consensus of the user community, the remaining substantive content is almost nil. That is reality, not a critique! It is a very challenging task given a diverse set of views and levels of experience weighted toward those who are willing to participate.

The *definition* that emerges from these exercises accomplishes a different objective—obtain a consensus definition of what a User community believes a system is versus *what a system actually is* and *what its Users expect it to accomplish*. Additionally, the definitions are often *abstract* and *intermix* different types of information and levels of detail that may impress uninformed customers but are *technically incorrect*. Consider the following example.

Example 1.1 **Making Statements That Are Partially True but Technically Incorrect**

Definitions over the years loosely infer that a system is a collection of people, hardware, software, procedures, and facilities—for example, entities. Systems do encompass those entities. However, general definitions such as this crafted to achieve a consensus do not express what a system is, why it exists, who it serves, its operating conditions, required outcomes and performance, criteria for success, etc.

The intent here is not to critique established definitions. If they work for you and your organization, fine. However, let's establish a definition that expresses *what a system actually is*. This is a crucial step in establishing a foundation for understanding Chapters 2–34. Therefore, we establish the following definition of a *system*:

- **System**—An integrated set of interoperable elements or entities, each with specified and bounded capabilities, configured in various combinations that enable specific behaviors to emerge for Command & Control, C2 by Users to achieve performance-based mission outcomes in a prescribed operating environment with a probability of success.

The "system" definition above captures a number of key discussion points that define a *system*. A *system* is composed of two or more integrated entities that enable accomplishment of a higher-level purpose—emergence—that cannot be achieved by each of the entities on an individual basis. However, a *purpose* without some *measure of success*—an outcome and level of performance—has limited value to the User or its stakeholders. With the establishment of this theme as a backdrop, let's explore each of the definition's phrases individually to better understand what they encompass and communicate.

1.3.1 System Definition: "An Integrated Set of Interoperable Elements or Entities …"

Systems occur in a variety of forms that include Enterprise and Engineered Systems—equipment hardware and software, social systems, political systems, and environmental systems. This text focuses on two types of systems: Enterprise and Engineered. Let's define each of these terms:

- **Enterprise Systems**—Formal and informal industry, academic, governmental, professional, and nonprofit organizations such as corporations, divisions, functional organizations – departments such as accounting and engineering; projects, and others.
- **Engineered Systems**—Physical systems or products developed for internal use, commercial sale to the marketplace, or for contract development that require one or more Engineering disciplines and skillsets to apply mathematical and scientific principles to design and develop solutions

Since Engineered Systems are an integral part of our home and work lives, let's begin with those.

1.3.1.1 Engineered Systems Engineered Systems, in general, consist of equipment comprised of hardware and/or software, fluids (lubricants, coolants, etc.), gases, and other entities:

- Hardware entities, for example, include hierarchical levels such as PRODUCTS comprised of → SUBSYSTEMS comprised of → ASSEMBLIES comprised of → SUBASSEMBLIES comprised of → PARTS (Chapter 8).
- Software entities include hierarchical terms such as Computer Software Configuration Items (CSCIs) comprised of → Computer Software Components (CSCs) comprised of → Computer Software Units (CSUs) (Chapter 16).

1.3.1.2 Enterprise Systems Enterprise Systems are HIGHER-ORDER SYSTEMS (Chapter 9)—government, corporations, and small businesses—that:

- Employ Engineered Systems—manufacturing systems, vehicles, computers, buildings, and roads—to:
  - Produce and distribute consumer products and contract deliverable systems
  - Provide services such as retail sales; land, sea, air, or space transportation; utilities such as electrical power, natural gas, telephone, water, sanitation, and refuse; and medical, healthcare, financial, educational, and other services

As we shall see in Chapters 8 and 9, analytically:

- **Enterprise Systems** consist of hierarchical levels of abstraction (divisions, departments, branches, etc.) comprised of System Elements (Figure 8.13)—personnel, equipment (hardware and software), procedures, resources, behavioral outcomes, and facilities—that are integrated to perform Enterprise missions.
- **Engineered Systems** consist of hierarchical levels of abstraction (Figure 8.4)—SEGMENTS, PRODUCTS, SUBSYSTEMS, ASSEMBLIES, SUBASSEMBLIES, and PARTS.

Observe the terms Enterprise System *elements* and Engineered System *entities*. Application of these terms will become more important in follow-on chapters. The terms imply that these are discrete objects, which they are. However, remember the earlier point in the *system* definition … *comprised two or more entities in combination that enable accomplishment of a higher-level purpose that cannot be achieved by each of the entities on an individual basis*. The operative term *combination* means that the system elements or entities must be *integrated*—connected.

Integrating elements and entities is a *necessary* condition to leverage or exploit the combination of capabilities. However, suppose the entities are *incompatible*? Hypothetically, you could fill—*integrate*—diesel fuel or kerosene into a gasoline-based automobile's fuel tank. But that does not mean that the engine will perform. Due to the *incompatibility*, fuel station pump nozzles and vehicle fuel tank ports are purposely designed to preclude inadvertent mixing.

Being *compatible* may be a *necessary condition* for some system entities such as rigid mechanical interfaces or System Elements such as procedural consistency between equipment—hardware and software—and User or Operator Manuals. Being compatible, however, does not mean that they can communicate in a language that is intelligible and comprehensible. That leads us to the need for some systems to be … *interoperable*. Consider electronic financial transactions in which debit or credit cards, card readers, and computers must be not only *compatible* in terms of electronic protocols but also formatting in an intelligible language that enables each to understand and interpret what is being communicated—*interoperability*. For those types of systems, compatibility and interoperability are both *necessary* and *sufficient* conditions for success.

In summary, the foundation of a system begins with *an integrated set of interoperable* (*Enterprise System*) *elements* (personnel, equipment, procedures, resources, behavioral outcomes, and facilities) *or* (*Engineered System*) *entities* (PRODUCTS, SUBSYSTEMS, ASSEMBLIES, SUBASSEMBLIES, etc.). In either case, we refer to the *system* being analyzed or investigated as a System of Interest (SOI).

1.3.2 System Definition: "… Each with Specified and Bounded Capabilities …"

If a system requires *System Elements* or *entities* that are *compatible* and *interoperable*, how do we ensure that they are? This requires multi-discipline Engineering - SE - to *specify* and *bound* these operational, behavioral, and physical capabilities—*attributes*, *properties*, and *characteristics*

(Chapter 3)—via specification requirements as a starting point.

Observe usage of the term *capability*. Traditionally, the term *function*—as in *form, fit, and function*—has been used by Engineers to characterize *what* a system is expected to accomplish. However, there is a gross disparity between the true definition of a *function* and what the User expects the system to *accomplish*. Here's the difference.

A Word of Caution 1.1

Form, Fit, and Function: An Implied Catch Phrase for Failure!

The phrase *form, fit, and function*, which is deeply ingrained as a paradigm in everyday Engineering, is a well-intended concept that is subject to misinterpretation. By virtue of the sequence of terms, people sometimes interpret the phrase as the sequence of steps required to perform Engineering:

- Step 1—Design the physical system—*form*.
- Step 2—Figure out how to get the pieces to *fit* together.
- Step 3—Decide what the system must do—*function*.

Evidence of this paradigm is illustrated in Figure 2.3. PURGE the *form, fit, and function* paradigm from your mind-set! The phrase simply identifies three key attributes of a system, product, or service that must be considered, nothing more!

Simply stated, a *function* represents an *action* to be performed such as Perform Navigation. A function is a *unitless* term that does not express a level of performance to be achieved. In general, it is easy to "identify functions" via *functional analysis*—sounds impressive to uninformed customers. The challenge is specifying and bounding the level of performance a function must achieve. Although *functions* and *functional analysis* are certainly valid within their own context, from a current SE perspective, the concept of functional analysis as a primary driving SE activity is outdated. The reality is functional analysis is still valid but only as a supporting SE activity. So, *how do we solve this dilemma*?

The solution resides in the term *capability*. A capability is defined as follows:

- **Capability**—An explicit, inherent feature *activated* by an external stimulus, cue, or excitation to perform an action (function) at a specified level of performance until terminated by external commands, timed completion, or resource depletion.

From an Engineering perspective, think of a capability using a vector analogy. A *capability* (vector) is characterized by a *function* (direction) and a *level of performance* (magnitude).

In summary, this text replaces *functions* and *functional analysis* with more appropriate terms *capability* and *capability analysis*.

1.3.3 System Definition: "… Configured in Various Combinations That Enable Specific Behaviors to Emerge …"

Configuration of various combinations of System Element and entity capabilities to produce system responses for a given set of system inputs—stimuli, cues, and excitations—represents a system architecture. However, system responses vary based on the User's operational needs at different times during a mission. Consider the following example of an aircraft.

Example 1.2

Aircraft Configurations and Behaviors

For an aircraft to perform a mission, it must be capable of loading passengers and cargo; taxiing; performing phases of flight (taking off, climbing, cruising, holding, and landing); and unloading passengers to accommodate various Use Cases and Scenarios (Chapter 5). Each of these activities requires unique sets of capabilities—*architectural configurations*—provided by the (Enterprise System) Elements and (Engineered System) entities to accomplish the performance-based mission outcomes and objectives.

Observe the phrase "… *enable specific behaviors to emerge* …." Emergent behavior is a key attribute of systems that enables them to accomplish a higher-level purpose that cannot be achieved by the individual elements or entities. In general, *emergent behavior* means that the system exhibits behaviors that are not readily apparent from analysis of its individual elements or entities. Consider the following example.

Example 1.3

Emergent Behavior

As humans, we have the capability to walk, run, etc. However, there is a need to travel more efficiently in a shorter period of time. To achieve this higher-level purpose, humans created bicycles expressly for enabling a human to travel great distances more efficiently. But *how would you know that (1) a set of physical components could be assembled into a Bicycle System as a prime mover capable of rolling and steering (emergent behaviors) and (2) a human could simultaneously C2—balance, pedal, and steer (emergent behaviors)—the Bicycle System*? If we analyzed the human or the bicycle, do they exhibit or reveal the capabilities—emergent

behaviors—that enable them as an integrated system to accomplish the higher-level mission (travel more efficiently in a shorter period of time)? Similar emergent behavior examples include jet engines or aircraft that can counter the effects of gravity and fly.

Chapter 3 provides additional discussion on *emergent* behavior.

1.3.4 System Definition: "… For Command & Control (C2) by Users to Achieve Performance-Based Mission Outcomes …"

Observe that the thrust of this phrase is an expectation to accomplish something—an outcome with a level of performance. More specifically, accomplish performance-based mission outcomes. Those who work in non-Aerospace and Defense (A&D) sectors often associate the term *mission* as unique to military systems. That is factually incorrect! Enterprises, projects, and individuals—medical doctors, educators, and so on—all perform missions.

A *mission* represents an Enterprise or Engineered System outcome and supporting performance-based objectives to be achieved. Consider the following mission examples.

Example 1.4

- **Medical Mission**—Improve the health conditions of …, find a cure for …, administer intravenous drugs to a patient in accordance with a doctor's orders, and so on.
- **Transportation Mission**—Safely transport passengers via air, train, or bus from one city to another, deliver parcel packages, and so on.
- **Services Mission**—Provide cable and Internet services to customer's businesses or homes, respond to fire and medical emergencies, and so on.
- **Educational Mission**—Offer an accredited (EE, ME, SwE, ChemE, IE, etc.) Engineering degree program.

The concept of missions, however, is not limited to Enterprise Systems. Interestingly, Enterprises for decades have developed *vision* and *mission statements*. Yet, often fail to recognize that the Engineered Systems they produce for the marketplace are designed to perform *missions* to support their customers'—Users and End Users—Enterprise System missions. When a system, product, or service ceases to perform a *mission*, it has no value to its Users in terms of outcomes to be accomplished—End User satisfaction and shareholder value and revenue generation—and will likely be retired or disposed.

Chapters 4 and 5 address *missions* and *mission analysis* in more detail.

1.3.5 System Definition: "… In a Prescribed Operating Environment …"

Humans and equipment often have performance limitations in terms of what types of operating environments they can operate to accomplish a mission. This requires knowledge and understanding of (1) *where* missions will be conducted—land, sea, air, space, or combinations of these—and (2) under *what* types of conditions. Once the external operating environment is understood, it must be *specified* and *bounded* in terms of performance requirements such as temperature, humidity, shock and vibration, and salt/fog.

1.3.6 System Definition: "… With a Probability of Success"

Finally, to support a User's missions, the system must be available *on demand* to reliably conduct missions and deliver performance-based outcomes with a *probability of success*. If a system, product, or service is unable to fulfill the minimum requirements for mission success prior to its mission, then mission failure may be the consequence and other alternative systems must be considered.

1.3.7 Other Definitions of a System

As a final note, national and international standards and professional organizations as well as different authors present various definitions of a *system*. If you analyze these, you will find a diversity of viewpoints, all influenced and tempered by their personal knowledge and experiences. Moreover, achievement of a "one size fits all" *convergence* and *consensus* by standards organizations often results in weak wording that many believe it to be *insufficient* and *inadequate*. For additional definitions of a system, refer to the following standards:

- INCOSE (2015). *Systems Engineering Handbook: A Guide for System Life Cycle Process and Activities* (4th ed.).
- IEEE Std 1220™-2005 (2005)—Institute of Electrical and Electronic Engineers (IEEE)
- ISO/IEC 15288:2015 (2015)—International Organization of Standards (ISO)
- DAU (2011)—Defense Acquisition University (DAU)
- NASA SP 2007-6105 (2007)—US National Aeronautics and Space Administration (NASA)
- FAA SEM (2006)—US Federal Aviation Administration (FAA)

You are encouraged to broaden your knowledge and explore definitions by these organizations. Depending on your personal viewpoints and needs, the definition stated in this text should provide a more definitive characterization.

1.4 LEARNING TO RECOGNIZE TYPES OF SYSTEMS

Systems occur in a number of forms. High-level examples include:

Example 1.5

System Examples

- Economic systems
- Communications systems
- Educational systems
- Entertainment systems
- Financial systems
- Government systems
- Environmental systems
- Legislative systems
- Medical systems
- Judicial systems
- Corporate systems
- Revenue systems
- Insurance systems
- Taxation systems
- Religious systems
- Licensing systems
- Social systems
- Military systems
- Psychological systems
- Welfare systems
- Cultural systems
- Public safety systems
- Food distribution systems
- Parks and recreation systems
- Transportation systems
- Environmental systems

Observe that many of the example systems are subsets of others and may be interconnected at various levels to form Systems of Systems (SoS). If we analyze these systems or SoS, we find that they produce combinations of performance-based outcomes such as products, behaviors, by-products, or services. As systems, they exemplify the definition of a system introduced earlier.

1.4.1 Precedented Versus Unprecedented Systems

Enterprise and Engineered Systems, in general, are either *precedented* or *unprecedented*:

- **Precedented Systems**—Systems for which earlier versions exist and provide the basis for upgrades such as technology and performance improvements
- **Unprecedented Systems**—Systems that represent innovations and radical new designs that depart from traditional designs, for example, the introduction of hybrid vehicles

To illustrate these terms, consider the following automobile example.

Example 1.6

Automobile Application: Precedented and Unprecedented Systems

Gasoline-powered automobiles are an example of *precedented* systems. Over many decades, they consisted of a frame, body, doors, engine, inflatable ties, steering, and so on.

Then, as newer automotive technologies evolved over the past 100+ years, manufacturers added new features and capabilities that were *unprecedented*. Examples included heaters, air-conditioning, power steering, electronic ignition, electrical doors and windows, compression bumpers, air bags, entertainment systems, satellite radio and phone data communications, and hybrid engines.

1.4.2 Products as Systems

Our discussions to this point have focused on the generic term *system*. *Where do consumer products and services fit into the context of a system*? A *product* consisting of two or more entities integrated together to provide a performance-based capability, by definition, is an instance of a system. Observe that a product provides a "capability" but does not address outcome. Why? Unless preprogrammed to run autonomously, products as inanimate objects are dependent on humans to apply them to a specific situation and subsequently achieve an outcome. For example:

- A pencil is a *product*—an instance of a system—comprised of a lead, a wooden or composite holder, an attached eraser that provides a capability but no outcome on its own.
- A computer monitor is a *product*—an instance of a system—comprised of an chassis, touch screen display, motherboard, processor, sound board, and interface ports—power, video, audio, and communication ports such as USB:
  - The computer processor transmits commands and data to the monitor to display formatted information to its User.
  - In response to the display data, the User has the option to provide a stimulus via the touch screen display to select an action to be performed—command

and audio volume—that results in an outcome as verification feedback of acceptance and subsequent completion of the action.

1.4.3 Tool Context

Some systems or products are employed as tools by HIGHER ORDER SYSTEMS such as an Enterprise. Let's define what we mean by a tool:

- **Tool**—A physical product or software application employed by a User to enable them to leverage their own capabilities to more *efficiently* and *effectively* perform a task and accomplish a performance-based outcome that exceeds their own strengths—capabilities—and limitations.

Consider the following example:

Example 1.7

Software Application as a Tool

A statistical software application, as a support tool, enables a statistician to efficiently sort and analyze large amounts of data and variances in a short period of time.

Now, is a wooden log, as an entity, a system? No, however, the log is considered a tool that has the capability to deliver a performance-based outcome when applied by a human operator under specific conditions.

1.4.4 Service Systems

The preceding discussion illustrates that the outcomes produced by a system may be (1) physical such as products and by-products or (2) behavioral responses—services. *What is a service*?

- *A service* is an activity provided and performed by an Organizational or Engineered System to produce an outcome that benefits its User.

Consider the following example.

Example 1.8

Consumer Product Services

- Weight scales are a consumer *product*—an instance of a system with multiple parts integrated together as a system—that respond to a User stimulus to provide weight measurement information in pounds or kilograms as a *service* response. Observe that the service delivers an outcome–displayed weight, however, no physical products are-produced.
- A digital alarm clock as a consumer product provides a service by displaying current time and an alarm when activated and set for a specific time.

Now that we have established what a system is brings us to the next question: *what is SE*?

1.5 WHAT IS SE?

Definition of SE requires an understanding of its two constituent terms: *system* and *Engineering*. Since the preceding discussions defined a *system*, the next step is to define *Engineering* to enable us to define SE.

1.5.1 Definition of Engineering

Engineering students often graduate without being introduced to the root term that provides the basis for their formal education. To illustrate this point, consider a conversational example.

Example 1.9

The Engineer's Dilemma

- *What is your profession*?
- I'm an Engineer—SE, ME, EE, SwE, ChemE, Test, and so on.
- *What do Engineers do?*
- We Engineer things.
- *So, what is Engineering?*
- (Silence) I don't know. Our instructors and courses didn't cover that topic!
- *So, even though you have received an Engineering degree, you are unaware of how "Engineering" is defined by your profession?*

The term *engineering* originates from the Latin word *ingenerare*, which means "to create" (Britannica, 2014). Its first known use is traceable to 1720 (Merriam-Webster, 2014). Let's introduce a couple of example definitions of Engineering:

- **Engineering**—"The profession in which knowledge of the mathematical and natural sciences gained by study, experience, and practice is applied with judgment to develop ways to utilize economically the materials and forces of nature for the benefit of mankind" (Prados, 2007, p. 108).
- **Engineering**—"The application of science and mathematics by which the properties of matter and the sources of energy in nature are made useful to people" (Merriam-Webster, 2014).

The Prados (2007) definition of Engineering above originates from earlier definitions by the Accreditation Board for Engineering and Technology (ABET), which accredits Engineering programs in the United States. ABET evolved the definition from its founding in 1932 until 1964. It continued to appear in ABET publications from 1964 through 2002 (Cryer, 2014).

Two key points emerge from the introduction of these definitions:

- First, you need to understand the definition and scope of your profession.
- Secondly, on inspection, these definitions might appear to be a mundane, academic discussion. The reality is that these definitions characterize the traditional view of Engineering. That is, Engineering the "Box (Equipment Hardware & Software)" Paradigm or "Box" Engineering that contributes to systems, products, or services failures attributed to "human error" (Chapter 24) or are considered by the User to be failures due to a lack of *usability*. This is a critical staging point in differentiating the scope of the SE – "Engineering the (User-EQUIPMENT) System, which includes the (Equipment) Box," versus traditional "Box" Engineering. In that context, SE exemplifies the cliché "Learning to think outside the (Engineering) box" to develop systems, products, and services that Users actually need, can use, and lead to a reduction in human errors that contribute to system failures. As a result, this impacts Enterprise System reputation, profitability, customer satisfaction, marketplace perceptions, and subsequently shareholder value.

Now that we have established definitions for a *system* and *Engineering*, let's proceed with defining SE.

1.5.2 Definition of System Engineering (SE)

Content–Grammar Principle

Principle 1.2 *Substantive content* must always precede *grammar* to achieve successful results. Avoid negotiating content for the sake of achieving grammatical elegance and eloquence unless it precludes misinterpretation.

There are a number of ways to define SE, each dependent on an individual's, project's, or Enterprise's business domain, perspectives, and experiences. SE means different things to different people. You will discover that even your own views of SE will evolve over time. So, if you have a diversity of perspectives and definitions, *what should you do*? What is important is that you, project teams, or your enterprise should:

- Establish a consensus definition for SE.
- Document or reference the SE definition in enterprise command media to serve as a guide for all.

For those who prefer a brief, high-level definition that encompasses the key holistic aspects of SE – "Engineering the System" - consider the following definition:

- **System Engineering**—The multi-disciplined application of analytical, mathematical, and scientific principles for formulating, selecting, developing, and maturing an *optimal* solution from a set of *viable* candidates that has *acceptable* risk, *satisfies* User operational need(s), and *minimizes* development and life cycle costs while *balancing* Stakeholder interests

To better understand the key elements of the SE definition, let's address each of the phrases separately.

1.5.2.1 SE Definition: "The Multi-disciplined Application of …"

System, product, and service development typically require multiple Engineering disciplines of expertise to translate a User's *operational need* and *vision* into a deliverable system, product, or service that produces performance-based outcomes required by the User. Accomplishment of that translation process requires *multi-disciplined* integration of hardware, software, test, materials, human factors, reliability, maintainability, and logistics Engineering.

1.5.2.2 SE Definition: "… Analytical, Mathematical, and Scientific Principles …"

Constructive Assessment

Author's Note 1.1 The following discussion is intended to be a constructive assessment concerning the state of traditional Engineering and its views of SE today versus what twenty-first-century Engineering and SE demand. The time has come to shift the educational paradigm!

Although not explicitly stated, the ABET (Prados, 2007, p. 108) definition of Engineering infers that the work scope of Engineering focuses on the innovation and development of devices, mechanisms, and structures to produce one or more performance-based outcomes for the benefit of mankind. In fact, we analytically represent the boundaries a system, product, or service as a "box" such as Figures 3.1 and 3.2. Psychologically, the simple act of establishing these boundaries automatically fosters an "Engineering within the walls of the box and connections between the boxes" paradigm. As a result, discipline-based Engineering courses and instruction focus on:

- Developing systems, devices, and mechanisms that utilize materials—technology—to harness, adapt, transfer, store, convert, transform, and condition the "forces of nature" such as energy, forces, information, and operating environment conditions to produce performance-based outcome(s) that benefits mankind

Unfortunately, this physics-based *scientific* and *mathematical* paradigm fosters *misperceptions* that SE is limited to Engineering and designing:

- Mechanical structures, enclosures, and mechanisms that can withstand, survive, divert, convert, transfer, transform, and store the physical "forces of nature"
- Electrical devices, components, and mechanisms that (1) respond to electrical, electronic, optical, and acoustical stimuli, excitations, and cues to produce specific outputs and characteristics; (2) store and retrieve energy and information; (3) select and locate components on printed circuit layouts; (4) perform self-tests and diagnostics; and (5) interconnect wiring and cables to compatible and interoperable components
- Software to (1) perform algorithmic decisions and computations to C2 systems and products and provide Situational Assessments and (2) mathematically model and simulate component and physics characteristics and other phenomena

The scope of SE does in fact encompass these multi-discipline Engineering activities as illustrated on the right side of Figure 1.1. In general, Engineers graduating from accredited institutions are well educated and competent in performing these activities. However, SE encompasses more than simply this traditional Engineering view of SE as illustrated by the left side of Figure 1.1 concerning Analytical Problem-Solving and Solution Development. This requires more than simply "plugging and chugging" equations to harness the "forces of nature." As a result, most Engineers are unprepared to enter industry and government to perform these activities. We will address this point in more detail in Chapter 2.

Finally, the term *mankind* at the end of the ABET (Prados, 2007, p. 108) definition … as a beneficiary of Engineering work. The question is: *who determines what would "benefit" mankind that motivates the need to involve and initiate Engineering actions*? In general, we can say *mankind* represents the marketplace. But, *who determines what the needs are for the marketplace, in general, or one of its segments*? The answer has two contexts: consumer product development and contract-based system development detailed in Chapter 5 (Figure 5.1) or services support:

- **Consumer Product Development**—Commercial industry expecting to make a profit as a Return on Investment (ROI) develops systems, products, and services for marketplace consumers. As a result, they have to *understand* and *anticipate* what potential consumers of a system, product, or service Want, Need, Can Afford, and are Willing to Pay (Principle 21.1).
- **Contract-Based System Development and Support Services**—Industry and government analyze their own needs and either develop systems and services internally or acquire/outsource them from external contractors or vendors.

Principle 1.3

Intellectual Control Principle

One of the key roles of an SE is to maintain "intellectual control of the problem solution" (McCumber and Sloan, 2002).

Returning to the question, *who determines what the marketplace needs are*? In either of the consumer or contract development cases above, the answer is someone with a technical background preferably in Engineering who has the *interpersonal* skillset to collaborate with the Users—consumer or contract—to:

1. Understand, analyze, identify, and document their operational needs and expected performance-based outcomes.
2. Specify and bound the Problem, Opportunity, or Issue Space (Figure 4.3) that needs to be solved or mitigated.
3. Specify and bound the Solution Space(s) (Figure 4.6 and 4.7) that represents what the User Wants, Needs, Can Afford, and Is Willing to Pay (Principle 21.1) to acquire or develop.
4. Collaborate with multiple Engineering disciplines to translate the Solution Space(s) performance-based outcomes and characteristics into an architectural-driven set of multi-level specification requirements.
5. Select an overall system, product, or service that is *optimal* across all User Solution Space scenarios and conditions.
6. Plan, implement, and orchestrate the technical strategy for a project as a Project or Lead Systems Engineer (LSE) or as a development team SE.
7. Maintain intellectual control (McCumber and Sloan, 2002) of the *evolving* and *maturing* System Design Solution to ensure that it is *consistent* and *traceable* to User Solution Space(s) *source* or *originating* requirements.

These points illustrate *why* the traditional Engineering view of SE as illustrated by the right side of Figure 1.1 is short scoped. SE encompasses more than simply the design of physical systems, devices, and components.

Based on the preceding discussion, the scope of SE encompasses three areas of concentration (Figure 1.1):

- **Analytical Problem-Solving and Solution Development**—
  Example activities include *collaboration* with *external* and *internal* Users to identify, specify, and bound their operational needs and capabilities; oversight of multi-level design development and integration; assessment of System Integration and Test results for compliance to specification requirements; and conduct and review of Analysis of Alternatives (AoA).
- **Multi-discipline Engineering**—Example activities include *collaboration* with Engineers concerning the development and interpretation of requirements, design integrity, analyses and trade-offs, prototype development, and Modeling and Simulation (M&S).
- **Technical PM**—Example activities include planning, tailoring, orchestrating, and implementing the technical project including baseline and risk management, conducting technical reviews, Specialty Engineering Integration, performing Verification and Validation (V&V) oversight, and preserving the technical integrity of the project.

As a project development "system," these activities are not just discrete activities. They must be integrated at two levels: (1) the Enterprise System developing the (2) Engineered System. Remember that Figure 1.1 illustrates a project's Enterprise System performing multi-discipline SE. As with any type of system, its interfaces must be *compatible* and *interoperable* to orchestrate the interactions and bi-directional communications across each interface to achieve success.

Therefore, SE not only requires the *application of mathematical and scientific principles* addressed in the Prados (2007) Engineering definition but also encompasses *analytical principles*—both *inside* or *outside* the system, product, or service and within the Enterprise System among its system developers.

1.5.2.3 SE Definition: "… For Formulating, Selecting, Developing, and Maturing an Optimal Solution from a Set of Viable Candidates …"

Engineers and teams often exhibit a propensity to take a *quantum leap* from requirements (Figure 2.3) to a single Point Design Solution without due consideration of:

1. How the User expects to deploy, operate, maintain, sustain, retire, and dispose of a system or product.
2. An evaluation of a viable set of candidate solutions and selection.
3. User life cycle costs and risks.

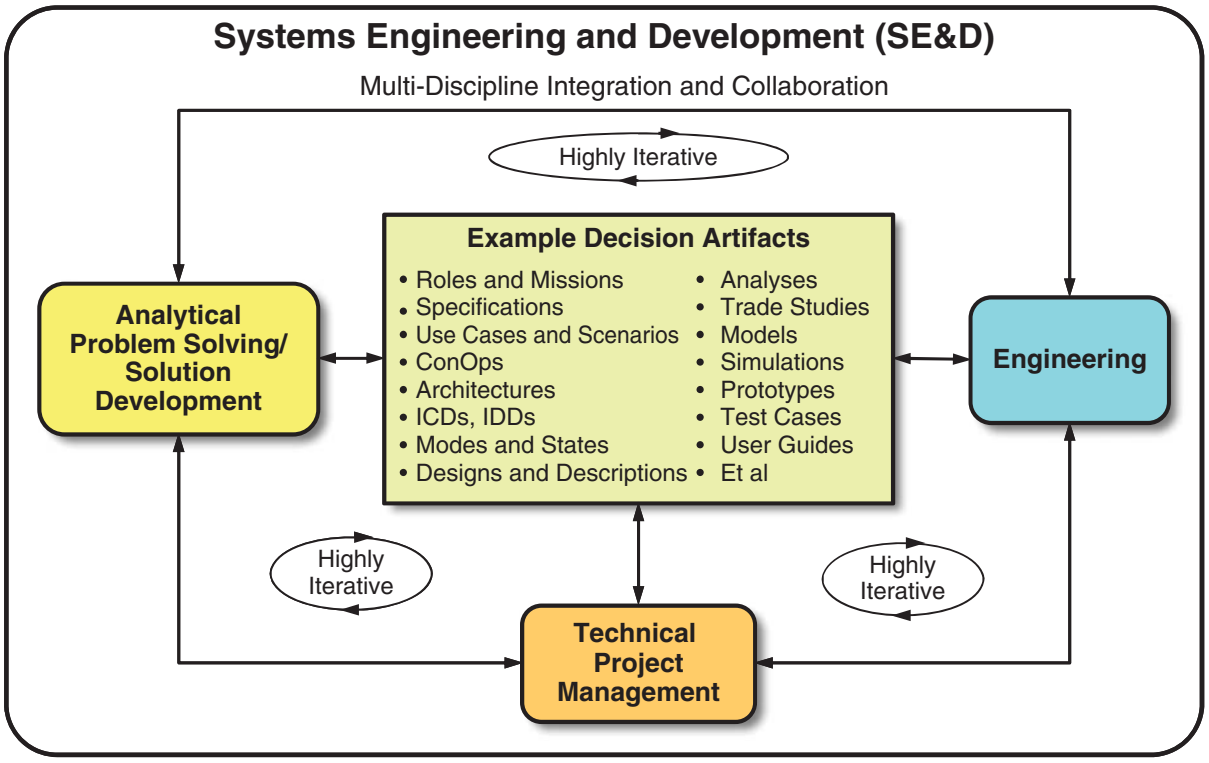


Figure 1.1 The Scope of SE and Its Relationship to Traditional Engineering

Therefore, a key objective of SE is to ensure that each design is formulated, evaluated, and selected from a set of *viable alternatives* using Multivariate Analysis or AoA. The selection may not be *ideal*; however, for a given set of constraints and operating environment conditions, it may be the best—*optimal*—that can be achieved.

1.5.2.4 SE Definition: "… That Has Acceptable Risk …"

If you ask Engineers what *level of risk* a system, product, or service should have, without hesitation, a common answer is *low risk*. The reality is customer budgets, schedules, technical requirements, and technologies may impose constraints in some cases that result in low-, medium-, or high-risk situations—whatever is *acceptable* to the User. This assumes that the System Developer has collaborated with the User to enable them to make an informed risk decision concerning the options and consequences. Therefore, under certain circumstances, low, medium, or high risk may be *acceptable* to the User. Ideally, SE tries to *mitigate* and *reduce* the risk via methods such as rapid prototyping, proof of concept and proof of technology demonstrations, and M&S methods.

1.5.2.5 SE Definition: "… Satisfies User Operational Need(s) …"

If you ask Engineers and Analysts where their requirements originate, the response is "from our contracts organization." From an Enterprise protocol perspective, that is true. However, *how do you know that the User requirements passed along by your contracts organization accurately and completely characterize the User's operational need*? Suppose you develop a system, product, or service that *complies* with those requirements and the User determines after delivery that it did not meet their operational needs. Who is to blame—legally and professionally? This brings us to a key principle of SE:

When a consumer or User—System Acquirer—purchases a system, product, or service, there is an expectation that it will achieve performance-based outcomes that characterize their operational needs. Those criteria typically characterize what is required to enable them to perform their missions. Therefore, SE technically *begins* and *ends* with the User and their End Users. Within this timeframe, the focal point of SE decision making centers on the User as a User's advocate. principle 1.1 illustrates the symbolism.

1.5.2.6 SE Definition: "… Minimizes Development and Life Cycle Costs …"

Many years ago, Enterprises and Engineers often had the view that their objective was to develop a system or product within a project's contract or task order triple constraints—cost, schedule, and technical. That was and is true, especially on Firm-Fixed Price (FFP) contracts. However, review the response again. It offers no indication of concern for the customer or User and their costs to deploy, operate, maintain, and sustain a system or product *after* it is delivered. In fact, the attitude was "we get paid to develop the system. Operations, Maintenance, and Sustainment (OM&S) costs are the User's problem." Those firms are either out of business today, consumed by their competition, or forced to change to survive.

Today, with demands on budgets—do more for less cost—Users are challenged to deal with the realities of System OM&S costs and the Total Cost of Ownership (TCO) of a system as an asset. Therefore, a key objective of SE during System Development is to *minimize* both development and User life cycle costs.

1.5.2.7 SE Definition: "… While Balancing Stakeholder Interests"

Enterprises and Engineered Systems have a variety of stakeholders to satisfy beginning with the acquisition of a system or product and continuing through its disposal. The same is true for System Developer—shareholders and suppliers. Therefore, another objective of SE is to achieve a *balance* not only in their own Enterprise interests but also to be a User's advocate. How is this achieved? Figure 1.2 illustrates how multi-discipline SE "bridges the gap" between a User's operational needs and the Engineering disciplines required to engineer a system, product, or service.

Additionally, as we shall see in Chapter 3, *stakeholders* include competitors and adversaries that have vested interests in the *success* or *failure* of a system, product, or service.

1.6 *SYSTEM* VERSUS *SYSTEMS* ENGINEERING

People sometimes get into debates about references to *System* versus *SE*. Which is correct? The answer depends on the *context* of the usage from a *project* SE or *enterprise* SE perspective.

1.6.1 Project SE

For example, a customer—System Acquirer or User—issues a contract to develop "a system." From a PM perspective, a project has (1) a Project Manager (PM) and (2) a work scope that is time-bounded with a beginning and ending for the development and delivery of the system, product, or service. The project's organizational element accountable for SE is labeled Project XYZ System Engineering (singular).

For large, complex systems that require development of multiple systems, typically project is assigned to a PM. The collective set of projects are organized underneath a Program managed by a Program Manager. At that level, the program's organizational label for SE is Program ABC Systems Engineering (plural).

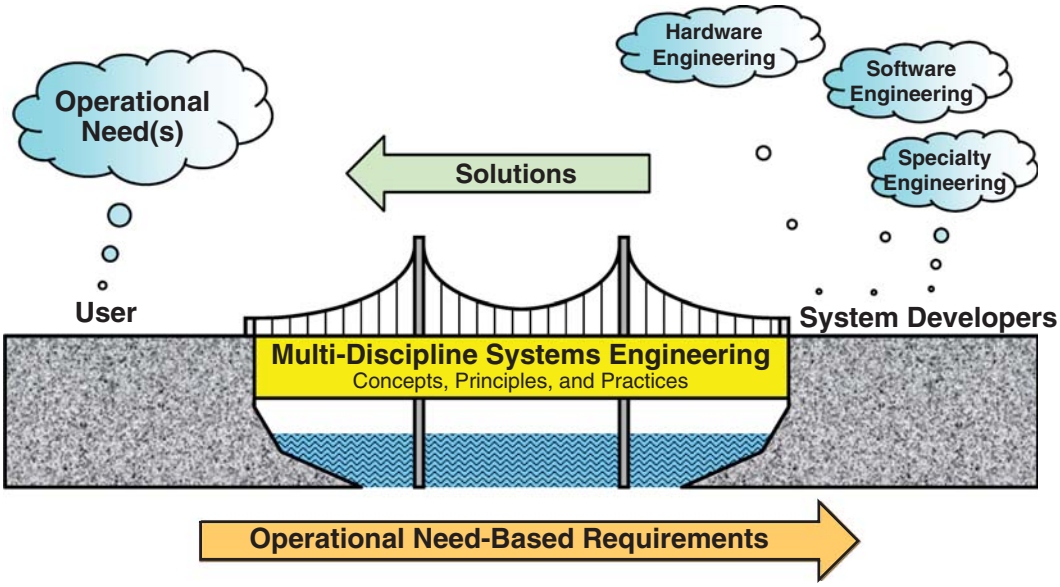


Figure 1.2 Multi-discipline SE "Bridges the Gap" between Users and System Developer Engineering Disciplines

1.6.2 Functional SE

Within most Enterprises, functional departments are established to supply personnel with Engineering discipline specialties such as EE, ME, and SwE to projects. Enterprise organizational charts often include a Systems Engineering Department. Observe the plural form *systems*. Matrix-based organizations such as departments supply Engineers with defined skillsets to perform specific tasks across multiple projects. Therefore, the term Systems Engineering (plural) is often applied to functional organizations.

1.7 SE: HISTORICAL NOTES

The earliest form of system applications began with early man with the innovation of the wooden, stone, and bone tools such as the lever and fulcrum, spear, and wheel. Systems evolved and became increasingly complex such as ground vehicle and ship transportation systems, weapons, and fortresses. The need to "engineer systems" evolved as a response to the demand to counter threats, move large objects, and develop products. Subsequently, the need to mass produce items in the late 1700s lead to the Industrial Revolution. In recent decades, larger complex systems and products drove the need to predictably and reliably develop and produce systems.

Most textbooks attempt to summarize the history of SE with facts and dates. You are encouraged to read those accounts. However, understanding what SE is today requires more than reading and memorizing facts and dates of past history. More importantly, you need to understand *how* and *why* modern-day SE has evolved as a discipline. Key points to note are:

1. During the first half of the 1900s, a new field of Systems Management emerged. Failures were attributed to poor Systems Management. As a result, rigid, inflexible management controls and processes were implemented (Johnson, 2002, pp. 1, 227–231).
2. Failures, however, continued to occur due to increasing complexity of WWII era and beyond military systems. Subsequently, industry and government came to the realization that the failures were due to poor reliability, not just Systems Management. As a result, the focus shifted to development and evolution of Reliability, Maintainability, and Availability (RMA) subsequently improving system performance.
3. During this timeframe, increasing system complexity began to drive the need to formulate a systems methodology. In turn, this led to the emergence of SE processes and methods to meet industry and government needs.

You are encouraged to read Johnson (2002) to better understand these points. Chapter exercises will delve into comparisons of recent SE processes and methods since WWII.

1.8 SYSTEMS THINKING AND SE

SE is often equated to Systems Thinking and Systems Engineers as Systems Thinkers. *What is Systems Thinking*? From the author's perspective, Systems Thinking is the ability to:

- Visualize or conceptualize any type of system—Natural, Engineered, or Enterprise—and all of its constituent levels and components, their interrelationships, and operational interactions with its OPERATING ENVIRONMENT.
- Perform a situational assessment of a system condition and level of urgency to initiate the appropriate, corrective actions in a timely manner.
- Formulate, develop, and synthesize a set of solutions that respond to User operational needs and constraints.
- Perform an AoA to evaluate and select the *optimal* solution that has *acceptable* risk to satisfy the User's operational needs and constraints for the least total life cycle cost.
- Optimize the selected solution to provide the best value—cost-performance–benefit ratio—to the User based on their operational needs, priorities, and acceptable risk.
- Observe system performance or the lack thereof, assimilate the observable facts, model, and analyze the contributory causes and effects.

Observe several key operative terms above that characterize Systems Thinking. These are visualize, conceptualize, assess, formulate, develop, synthesize, evaluate and select, optimize, assimilate, model, and analyze. To illustrate Systems Thinking under pressure, consider Mini-Case Studies 1.1 and 1.2.

Mini-Case Study 1.1

Systems Thinking in Action: The Apollo 13 Accident (Figure 1.3)

On April 11, 1970, NASA launched Apollo 13 on a lunar mission. The mission configuration consisted of a Command Module (CM) and a Service Module (SM) containing the Lunar lander. While in lunar orbit, two astronauts would enter the Lunar Module (LM), separate from the CM, land on the Moon's surface, return to the CM containing the third astronaut circling the Moon, and jettison the LM en route back to Earth.

Two days into the mission, an oxygen tank onboard exploded crippling the SM causing the lunar landing to be aborted. Challenged with being unable to visually assess the damage, the astronaut crew and Engineers on the ground had to make Situational Assessments concerning how to manage (1) redirecting a spacecraft back to Earth that was traveling away from Earth toward the Moon; (2) limited onboard power, water, heat, and breathable air resources; and (3) return of the crew home safely.

The demand for Systems Thinking became paramount. Given the situation, how do you assimilate onboard resources in the integrated CM–SM and attached LM to *synthesize* multiple power, water, heat, air, and other solutions to ensure survival of the astronaut crew? "The LM was designed to support two men on a 2-day expedition to the lunar surface. Mission Control made major revisions in the use rate of water, oxygen, and electrical power to sustain three men for the 4-day return trip to the Earth." (NASA 1970, p. 5–33) Additionally, to conserve power, the astronauts had moved to the LM as a "lifeboat" for return to Earth allowing the CM–SM to be powered down. However, to ensure a safe return, the CM–SM would have to be powered up, an action that was not intended to be performed in-flight, for the astronauts to reenter it prior to rentry.

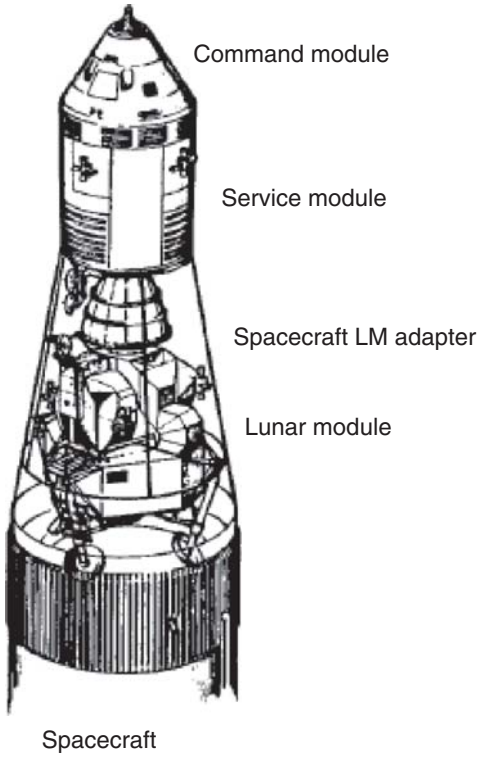


Figure 1.3 Apollo Vehicle (Source: NASA (1970))

Refer to NASA (1970) and Wikipedia (2014) for details of the solutions that illustrate how NASA applied Systems Thinking to *innovate* and *create* real-time solutions that enabled the astronauts to survive and return home safely.

Mini-Case Study 1.2

Systems Thinking in Action: Critical Software System Test

A project was developing a large software intensive system to replace an existing system. To accomplish an orderly transition and replacement, the new system operated in a surrogate "shadow mode" with the existing system to Verify and Validate (V&V) its condition as being *operationally ready*. The initial test was scheduled to occur in the early morning hours when demand for the primary system was low. Due to the system's criticality as a control center, the test was under heavy scrutiny politically, technically, and technologically.

After months of challenging work to meet an unrealistic development schedule, the software "locked up" during Pre-Test Checkout activities. Despite the efforts of many people and heavy pressure from executives and customers demanding corrective action due to careers being on the line, the contributory root cause for the lock-up could not be identified.

Frustrated, the Lead Software Systems Engineer left the control center and walked around the large parking lot several times trying to visualize and assimilate observable facts based on a mental model of the software's architecture and conceptualize a corrective action solution. During one of the laps, a "light bulb" came on in their head when they realized that a key software flag may not have been documented and set in the Pre-Test procedures. The Engineer returned to the control center, set the flag, and the software became fully operational less than 30 minutes before the crucial test.

Is Systems Thinking Unique to SEs and Engineers? Absolutely not! Systems Thinking is a personal attribute unrelated to SE or engineering. Engineers, by virtue of reputation of their interest in "tinkering and understanding how things work," are often characterized by family members and teachers as "Systems Thinkers." The same can be said about auto mechanics, food preparation in the home, PMs, and many other skills. However, there is a difference between being mechanically, electrically, electronically, or software minded—one form of Systems Thinking—and the ability to see things on a much larger, conceptual scale such as Einstein's creation of the Theory of Relativity. Systems Thinkers are present in every field—biology, chemistry, physics, medicine, politics, education, architecture, banking, military, communications, and automotive repair, not just Engineering!

1.9 CHAPTER SUMMARY

This concludes our discussion Systems, Engineering, and SE. Key points include:

1. We defined *system* is in terms of what it is, why it exists, what it is expected to accomplish, and who it benefits.
2. To define SE, we introduced the ABET (Prados, 2007, p. 108) definition for *Engineering* and coupled with the *system* definition.
3. We highlighted the scope of SE as "Engineering the (User-Equipment) System" that encompasses the "Engineering of the (Equipment) Box" by traditional Engineering that often contributes to factors that drive system failures and poor customer satisfaction.
4. We also explored examples of types of systems; distinguished between *precedented* and *unprecedented* systems; and considered the context of systems, products, and tools.
5. Since people often use the terms *SE* and *SE* interchangeably, we delineated the usage based on its project or Enterprise context.
6. Lastly, we explored one of the key attributes of SEs, Systems Thinking.

Given this introductory background, Chapter 2 will address THE EVOLVING STATE OF SE PRACTICE: CHALLENGES AND OPPORTUNITIES.

1.10 CHAPTER EXERCISES

1.10.1 Level 1: Chapter Knowledge Exercises

1. Create your own definition of a system. Based on the "system" definitions provided in this chapter:
   a. Identify your viewpoint of shortcomings in the definitions.
   b. Provide rationale as to why you believe your definition overcomes those shortcomings.

2. From a historical perspective, identify three *precedented* systems that were replaced with *unprecedented* systems.

3. What is a *system*?

4. Is a *product* a system?

5. Is a *service* a system?

6. What are examples of different types of systems?

7. What are the two primary types of systems associated with system, product, or service development?

8. What is an *Engineered System*?

9. What is an *Organizational System*?

10. What is *Engineering*?

11. What is SE?

12. SE consists of three primary aspects. What are they? Describe the interactions among the three.

13. How does the scope of *Engineering* compare with *SE* in terms of "Engineering the System" versus "Engineering the Box."

14. What is the difference between a system, a product, and a tool?

1.10.2 Level 2: Chapter Knowledge Exercises

Refer to www.wiley.com/go/systemengineeringanalysis2e

1.11 REFERENCES

Britannica (2014), *Encyclopedia Britannica*, Chicago, IL: Encyclopedia Britannica, Inc. Retrieved on 1/14/14 from http://www.britannica.com/EBchecked/topic/187549/engineering.

Cryer, Keryl (2014), *Correspondence – ABET Definition of Engineering*, Baltimore, MD: Accreditation Board for Engineering and Technology (ABET).

DAU (2011), *Glossary: Defense Acquisition Acronyms and Terms*, 14th ed. Ft. Belvoir, VA: Defense Acquisition University (DAU) Press. Retrieved on 3/27/13 from: http://www.dau.mil/pubscats/PubsCats/Glossary%2014th%20edition%20July%202011.pdf.

IEEE Std 1220™-2005 (2005), *IEEE Standard for the Application and Management of the Systems Engineering Process*, New York: Institute of Electrical and Electronic Engineers (IEEE).

INCOSE (2015). *Systems Engineering Handbook: A Guide for System Life Cycle Process and Activities* (4th ed.). D. D. Walden, G. J. Roedler, K. J. Forsberg, R. D. Hamelin, and, T. M. Shortell (Eds.). San Diego, CA: International Council on Systems Engineering.

ISO/IEC 15288:2015 (2015), *System Engineering - System Life Cycle Processes*, Geneva: International Organization for Standardization (ISO).

Johnson, Stephen B. (2002), *The Secret of Apollo: Systems Management in the American and European Space Programs*, Baltimore, MD: The Johns Hopkins University Press.

McCumber, William H. and Sloan, Crystal (2002), *Educating Systems Engineers: Encouraging Divergent Thinking*, Rockwood, TN: Eagle Ridge Technologies, Inc. Retrieved 8/31/13 from http://www.ertin.com/papers/mccumber_sloan_2002.pdf.

Merriam-Webster (2014), Merriam-Webster On-Line Dictionary, www.Merriam-Webster.com

NASA (1970), *Report of Apollo 13 Review Board*, Washington, DC: NASA, Accessed on 5/19/14 from http://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/19700076776.pdf.

Prados John W. (2007), *75th Anniversary Retrospective Book: A Proud Legacy of Quality Assurance in the Preparation of Technical Professionals*, Baltimore, MD: Accreditation Board for Engineering and Technology (ABET).

Wikipedia (2014), *Apollo 13 web page*, San Francisco, CA: Wikimedia Foundation, Inc.

2

THE EVOLVING STATE OF SE PRACTICE-CHALLENGES AND OPPORTUNITIES

Enterprises and professionals in industry and government often *erroneously* believe they are performing Systems Engineering (SE) when in fact they employ a traditional, *endless loop* Plug and Chug … Specify-Design-Build-Test-Fix (SDBTF) Engineering Paradigm. The paradigm has roots traceable to the Scientific Method taught in middle and high school science classes and a 1960s Design Process Model (DPM). The SDBTF–DPM Paradigm is common in System Development environments that are characterized *as ad hoc, chaotic, inconsistent, inefficient,* and *ineffective*. A key thrust of Chapter 2 will be learning to recognize and understand the SDBTF–DPM Paradigm that impacts SE and subsequently System Development performance.

Paradigms are simply an ingrained, cultural, mind-set - *groupthink* –or model that *filters* or *rejects* considerations to adopt or employ new innovations and ideas that may impact the existing status quo. Most paradigms remain in place until an external event or the marketplace causes a shift to a new paradigm. In the case of the marketplace, you either (1) change with the times and *proactively* make the competition; (2) succumb to internal, *reactionary* "firefighting" and ultimately go out of business; or (3) are acquired by your competition.

Chapter 2 examines the evolving state of SE practice of SE from an historical perspective. Our discussions investigate various Enterprise SE paradigms that impact overall project performance. This represents both *challenges* and *opportunities* for the future in terms of advancing the state of the SE practice.

From a System Development perspective, our context of SE practice here focuses on the need for efficient and effective *problem-solving* and *solution development methods* that apply to all Engineering disciplines, not just SE. Where these methods are lacking across Engineering disciplines, Engineers tend to dismiss them because, after all, they are not EE, ME, and SwE, practices – Not Invented Here (NIH). As a result, the multi-discipline integration required to achieve System Development performance is often *non-existent*, *ad hoc*, *inefficient*, and *ineffective*. As one Engineer stated, "… if my university had thought this was important, they would have taught me … end of story!" The end result is projects that have technical performance and compliance issues resulting in overrun budgets and schedules and diminished customer satisfaction – a source of project management frustration with Engineers and Engineering.

Traditional Engineering, in general, with the exception of Software Engineering by virtue of its nature, has always had a physics-based widget development focus. We see inferences of this in the *Engineering* definition introduced in Chapter 1. "… utilize economically, … the *materials and forces of nature* … for the benefit of mankind" (Prados, 2007, p. 108). In other words, figure out how to innovate technologies and widgets that harness and transform "*the materials and forces of nature*" to produce output(s) – Equipment boxes - that exhibit specific performance characteristics for a given set of operating environment conditions.

To illustrate this point, consider the following example.

System Engineering Analysis, Design, and Development: Concepts, Principles, and Practices, Second Edition. Charles S. Wasson.

Companion Website: www.wiley.com/go/systemengineeringanalysis2e

Example 2.1

The "Engineered" Mousetrap

A consumer requests that an engineer "build a better mousetrap." The engineer designs, builds, tests, and delivers the mousetrap to the customer.

Sometime later, the engineer sees the consumer and asks "how did the mousetrap work?" The consumer answers "Great! It did an excellent job catching mice. However, I had to spend a considerable amount of time simply trying to figure out how to bait and set the trap and later removing the mouse. There has to be a better design." The engineer responds … "You didn't say it had to be easy to use … You just asked me to build a better mousetrap!"

This example illustrates both the challenges and opportunities of the twenty-first-century Engineering in a highly competitive, global economy. This requires a different type of System Thinking that goes beyond the "box" mind-set of traditional Engineering—more specifically the need to understand the Stakeholder's Operational Needs, Technology, Cost, Schedule, and Risk; User Interactions–Usability; External Systems interfaces; and the "Forces of Nature." This is the Realm of SE as illustrated in Figure 1.1.

In the 1970s, an *unsubstantiated* rumor circulated in which industry told academia "Teach Engineers how to 'plug and chug' equations and we will teach them how to develop systems." A few large corporations developed internal SE courses. Likewise, a few major universities also offered courses. However, most Engineers did not have access to these courses or training. As a result, Engineers acquired "systems" and SE knowledge experientially via personal self-study of emerging and evolving SE standards, journal articles, etc. tempered by On-the-Job Training (OJT).

Historically, Engineering Education has been Engineering *discipline* focused—EE, ME, and SwE—as illustrated in Figure 2.1. However:

> *Where do discipline Engineers become educated in Multi-discipline Integration that serves as the centerpiece for SE in industry and government, especially in common problem-solving and solution development methods*?

The solution is multi-faceted requiring an integrated System of Systems (SoS) - academia, industry, government, professional, and standards Enterprises – environment.

> The process begins with establishing SE courses as an integral requirement for Engineering degree programs. Engineering institutions will argue that SE courses already exist but not as a degree requirement. SE courses based on the traditional Engineering instructional classroom model - *plug and chug* equations - taught by instructors with limited or no SE industrial experience do not solve the Multi-discipline Integration *void* illustrated in Figure 2.1.

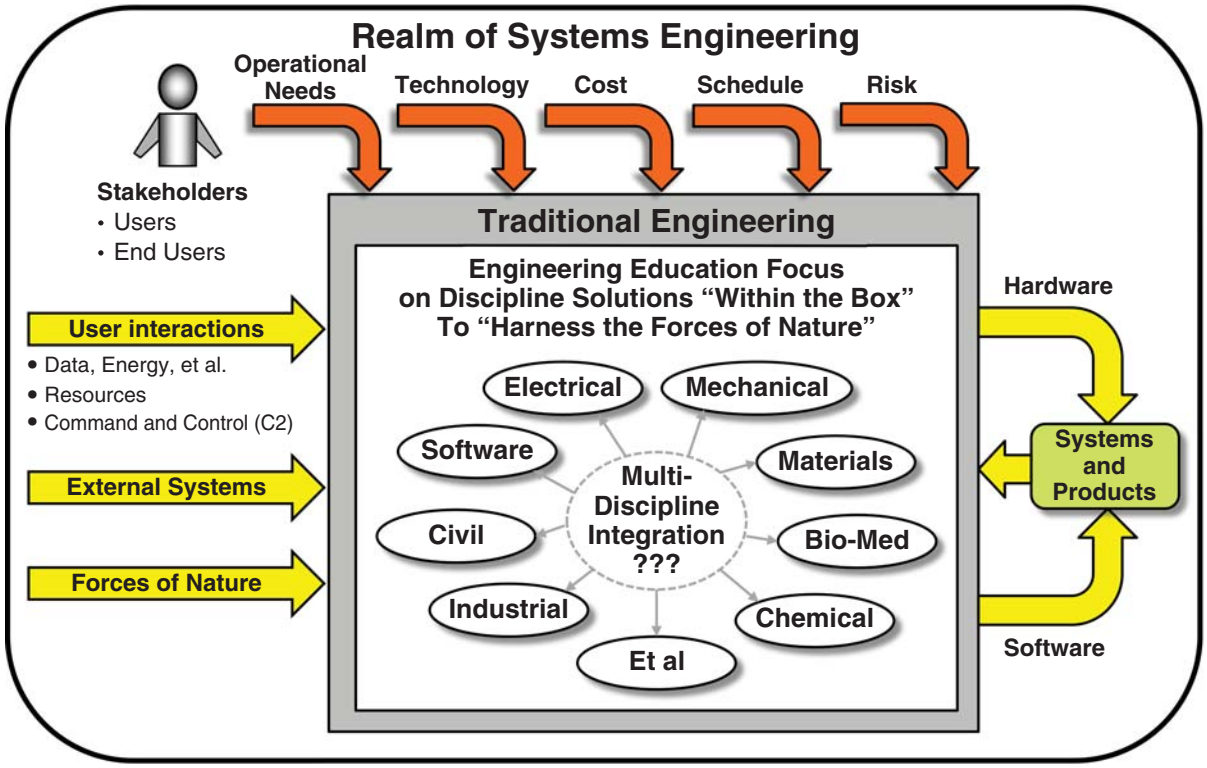


Figure 2.1 Void in the Traditional Engineering Education Model

The other non-academic aspects of the SoS solutions have been addressed by industry, government, professional, and standards organizations. However, they have not addressed a primary root cause of poor project performance—the SDBTF–DPM Paradigm, which has origins in Engineering Education.

These points provide the backdrop for Chapter 2.

2.1 DEFINITIONS OF KEY TERMS

Paradigm—An ingrained, groupthink mind-set or model that *filters* or *rejects* considerations to adopt or employ new innovations and ideas that may impact the status quo.

- **Decision Artifacts**—Physical, objective evidence of decision or result outcomes documented in work products such as plans, tasks, conference or review minutes, specifications, designs, analyses, models, simulations, demonstrations, quality conformance inspections, test results, and others.
- **Design-Build-Test-Fix (DBTF) Paradigm**—An ad hoc, trial-and-error, instructional model traceable to the Scientific Method in which engineers perform "stovepipe" engineering: (1) design an entity, (2) assemble it in the lab, (3) test it, and (4) fix, rework, or patch the design or its physical implementation. These activities are iterated in a seemingly endless loop until convergence is achieved in a final solution or schedule and cost resources become depleted. (Wasson, 2012, p. 1).
- **Paradigm shift**—A transformational change driven (1) externally by the marketplace or technology or (2) internally through visionary leadership to advance state of the practice or being from one paradigm to another over a planned period of time (Wasson, 2012, p. 2).
- **Plug and Chug Paradigm**—Represents a traditional Engineering instructional classroom teaching model for solving classical boundary condition problems in which students *Plug* a value into an equation and *Chug* out an answer (Adapted from Wasson, 2012, p. 1).
- **Specify-Design-Build-Test-Fix (SDBTF) Paradigm**—An expansion of the instructional classroom Plug and Chug … DBTF Paradigm teaching model to include specification of requirements (Wasson, 2012, p. 1).

System Acquisition and Management (SA&M) Activities performed by a System Acquirer to procure a system, product, or service; monitor, track and review its development; witness *verification* of compliance to its contract and technical specification requirements; and *validate* that the system satisfies the User's operational needs.

System Engineering and Development (SE&D) The sequential workflow activities performed by a System Developer to:

- Identify a system's Stakeholders - Users and End Users.
- Understand and analyze Stakeholder operational needs—User stories, use cases, and scenarios.
- Transform those needs into performance specification requirements.
- Incrementally select and document a multi-level System Design Solution using an Analysis of Alternatives (AoA) based of sets of viable candidates.
- Procure, develop, or modify the components.
- Integrate and test components for compliance verification to specification requirements.
- Incrementally conduct technical reviews to ensure Verification and Validation (V&V) to specification and task requirements.
- Verify and Validate (V&V) the system throughout its development.

Validation The *continuous* process of evaluating and assessing how well multi-discipline SE *activities* and *work products* – decision artifacts such as specifications, designs, or devices satisfy the respective User based on their pre-defined operational needs, constraints, and performance expectations.

Verification The continuous process of *evaluating* System Acquirer, System Developer, Services Provider, or User work products throughout the System/Product Life Cycle to *assess* compliance to pre-defined contract, mission, or task requirements.

Work Product Physical, objective evidence that planned process outcomes such as (1) *decision artifacts* – documentation and (2) deliverable systems, products, or services have been completed. This does not mean that the work product complies with a specification, plan, or task order, only evidence of completion and delivery. Objective evidence of formal verification of work product compliance, a separate issue, should be a mandatory condition for completion and delivery.

Author's Note 2.1

The *validation* definition above is a *general-purpose* description that applies to:

- System Acquirer acquisition documents such as contracts, Statements of Work (SOW), specifications, and other documents.

- System Developer SYSTEM, PRODUCT, SUBSYSTEM, ASSEMBLIES, SUBASSEMBLY, and PART specifications, designs, drawings, test procedures, and devices.
- Vendor specifications, designs, drawings, test procedures, and devices.
- System Acquirer—User evaluation of the system.
- Users—deploying, operating, maintaining, and sustaining the fielded system.

2.2 APPROACH TO THIS CHAPTER

We begin with a couple of examples that illustrate the state of SE practice in many Enterprises today. For customers of those Enterprises, project performance is often marked by overrun budgets and schedule as well as poor technical performance. We introduce results from the National Defense Industries Association (NDIA) concerning the Top Five SE issues and connect those findings to Enterprise Engineering Paradigms.

Another factor concerns the level of SE Effort—resources and funding—on projects, which is often underfunded due to limited budgets and the difficulty in quantifying the Return on Investment (ROI) for SE effectiveness. We introduce research data that illustrates how project cost and schedule performance with an *optimal* level of SE Effort improve the chances of success in meeting cost and schedule requirements.

We trace the origins of the fundamental SDBTF–DPM Engineering Paradigm back to three sources:

- The Scientific Method introduced in middle and high school science classes.
- The Plug and Chug … DBTF Paradigm acquired from academic courses and lab sessions.
- Archer's (1965) DPM.

Chapter 4 introduces the concept of *problem spaces* and *solution spaces*. Using a puzzle analogy, a *problem space* represents the puzzle and its outer boundaries to be solved (Figure 4.7). Pieces of the puzzle represent unique *solution spaces* that integrate via their interlocking boundaries solve the higher-level puzzle - *problem space*.

Viewing System Development performance as a multi-faceted *problem space*, industry, government, academia, professional, and standards organizations have attempted to implement several types of *solution spaces*. In some cases, the problem space is dynamic and evolving over time. For example, SE and SwE, as *emerging* and *maturing* disciplines, are highly interdependent not only between themselves but also with the more mature hardware Engineering disciplines such as EE, ME, CE, Civil, ChemE, Nuclear, Materials, and others.

The multi-discipline Engineering integration challenge has been exacerbated by other factors such as advancing technologies and analytical tools to support system decision-making, creation of evolving standards and capability assessment methods, improvement in Engineering education accreditation criteria, documentation of Organizational Standards Processes (OSPs), Lean Systems Engineering (LSE) and thinking, etc. Yet, System Development performance continues to be a problem. This leads to a key question: *Have industry, government, academia, professional, and standards organizations been focusing on Enterprise process "symptom solving" unaware of an underlying deficiency in Engineering education that requires "problem-solving"?*

The author proposes that all of these solutions are *necessary* to achieve project performance that is *consistent, reliable,* and *predictable*. However, they are *insufficient* in solving the root cause of the System Development performance problem. The problem traces to a lack of fundamentals of SE course that introduces system *problem-solving* and *solution development* methods *common* across all Engineering programs. For example, most Engineers are required to complete courses in Engineering Statics and Dynamics, Engineering Economy, Strength of Materials, Thermodynamics, etc. Yet, they lack *common, multi-discipline* problem-solving and solution development methods, vocabularies, meanings, etc. that enable them to immediately become productive in work environments following graduation.

Engineering programs will argue that they have offered fundamentals of SE course for years. However, many of these courses focus on (1) SA&M, overseeing how others should be performing SE, or (2) equation-based instruction, the comfort zone of Engineering. These courses are fine when appropriately introduced after a fundamentals of SE course that provides competent instruction in SE concepts, principles, and practices addressed in this text.

The educational problem is further exacerbated by a lack of course instructors who (1) have in-depth industry experience of 20+ years or more and (2) recognize, understand, and appreciate the SDBTF–DPM Engineering Paradigm and how its *abstract*, *endless loop* methodology is a key driver for poor System Development technical performance.

Given this overview, let's begin our discussion with the *State of SE and System Development Performance*.

2.3 THE STATE OF SE AND SYSTEM DEVELOPMENT PERFORMANCE

One of the best ways to illustrate the current state of SE practice is to use a couple of examples. Since the two main contributors to the System Development performance issue

are System Developers and Engineering Education, we will use one hypothetical example of each.

Mini-Case Study 2.1

Case of the Eloquent, Egotistical Proposal

A Stakeholder—User or System Acquirer—releases a Request for Proposal (RFP) to a list of potentially qualified off errors. The offerors prepare and submit their proposals. The System Acquirer's Source Selection Evaluation Team (SSET) reviews and scores each proposal that includes euphoric themes that read as follows:

Dear Mr. or Mrs. Customer:

Thank you for the opportunity to bid on Acquisition No. _____. Our Enterprise is the best there is. We have the best, well-trained, high-performing SE team and documented processes that everyone understands and follows to the letter that is compliant with Standard X …

Our Enterprise has also been assessed to be the highest level of SE capability there is … in most of the assessment areas. Our Engineering Process is tailorable and guides creation of specifications, develops designs, builds components, and integrates and tests the components. Prior to delivery, we perform V&V. If you award us the contract, rest assured the project will go smoothly (Figure 2.2, Panel 1) with on-time delivery and acceptable risk … all within the budget.

The System Acquirer informs their management they are confident they have selected a System Developer that can perform the work scope. The Acquirer awards the contract System Developer.

Beginning with the first technical review and continuing throughout the contract, the System Acquirer becomes concerned that the "rest assured the project will go smoothly" in Figure 2.2, Panel 1 actually resembles the perturbations shown in Panel 2. *Ad hoc* indecisions and subsurface *chaos* begin to emerge over time. Key milestones are missed. Budgets begin to overrun. Then, the System Developer makes bold pronouncements that they are now in System Integration and Test (SI&T) and "back on schedule."

Then … an amazing discovery occurs. The System Acquirer learns that the System Developer is actually redesigning the system … in SI&T (NDIA 2010 Issue #4 addressed later). SUBSYSTEM Teams A and B failed to baseline their Interface Control Documents (ICDs), and changes were made without informing the other team.

Fiction? As a case study, yes! However, key points in the case study occur every day. When you analyze these cases, you discover that Enterprises employ paradigms in which they truly believe they are performing SE. They perceive SE to be performing the following activities:

1. Writing requirements specifications.
2. Developing architectures and designs.
3. Developing or acquiring components.
4. Performing SI&T.
5. Verifying compliance to specification requirements.

The reality is that these activities represent the general *workflow* of the project level System Development Process

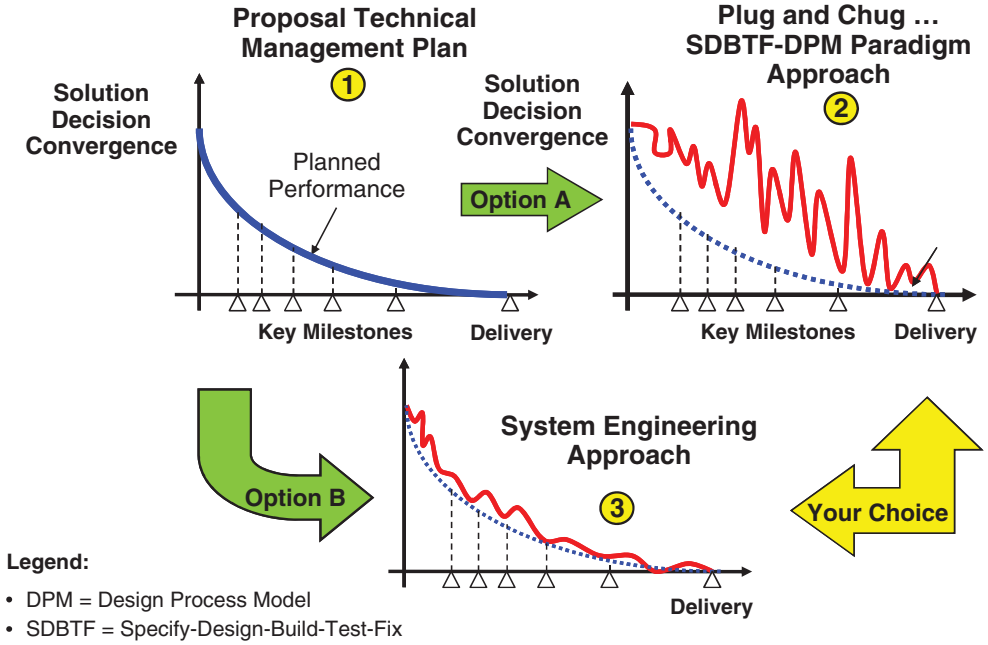


Figure 2.2 Comparison of Enterprise and Organizational SE Capabilities

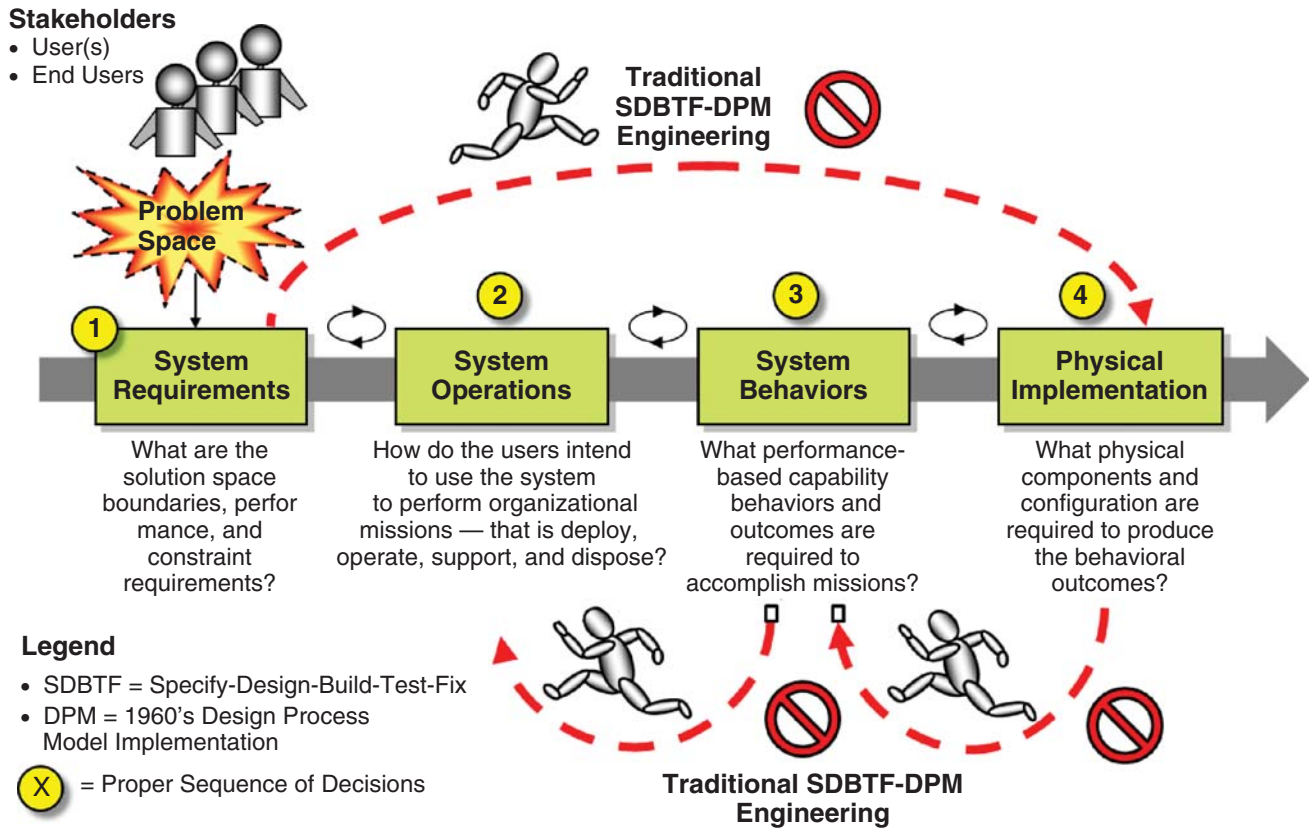


Figure 2.3 SDBTF Quantum Leaps from Requirements to Physical Design Solutions

(Figure 15.2). *They are not SE!* Although SE encompasses these activities, it is a *problem-solving and solution development* methodology that can be applied to the system or any entity within the System. Additionally, the sequence above infers that one activity must be completed before the next. *This, too, is erroneous!* Some Enterprises, projects, and Engineers approach these activities like a checklist believing SE produces documentation to check each box. Then, when (1) their SYSTEM or PRODUCT fails or (2) schedules or costs overrun, they blame the result on SE. Contrary to uninformed opinion, SE is NOT about producing documentation; it is about decision making!

Wasson (2012) observes that this paradigm is a condition referred to as the Plug and Chug … SDBTF Engineering Paradigm, an *endless loop* exercise that results in technical and programmatic risk, cost overruns, missed schedules, and latent defects in fielded systems. To illustrate, SDBTF Enterprises, functional organizations, projects, and Engineers often have the view "give us some requirements and we can design and develop the hardware and software *widgets*—whatever you need" as shown in Figure 2.3. If you observe their work patterns, Engineers often *prematurely* take a *quantum leap* or *shortcut* from Requirements (*what* is to be accomplished) to a single Physical Design Solution (*how* to physically implement the solution without consideration on interim steps). In this case, a *point design solution* is created, which may or may not be optimal. Little or no consideration is given to:

1. How the User envisions deploying, operating, maintaining, sustaining, and retiring/disposing of the system.
2. How the User expects the SYSTEM or PRODUCT to respond to external stimuli, excitations, or cues that occur after they have designed the physical implementation of the system.
3. Alternative solutions based on a set of viable candidates.

Most efforts like this often result in considerable rework or failure, overrun budgets and schedules, and risk. A bona fide SE Process methodology introduced later in Chapters 11 and 14 provides a logical decision making progression—Requirements → Operations → Behavior → Physical Implementation—in an *efficient* and *effective* manner. It focuses on decision-making *convergence* and outcomes while minimizing rework.

This leads to a key question: *If the Typical SDBTF Engineering Approach results in rework, failure, risk, and*

other effects, why is System Development performed in this manner? The answer resides in Engineering Education. This takes us to Mini-Case Study 2.2.

Mini-Case Study 2.2

Case of the Project Engineer and Engineering Education

A Lead SE (LSE) on a large complex System Development project is confronted not only with the technical and technological challenges of the system but also with personnel that have been "knighted" by their manager as SEs without any requisite education and experience. It is a major challenge that diverts energy and time from the main work of the project.

Challenged to (1) train the so-called SEs in real time during the business day, which is their functional SE manager's job, and (2) oversee the SE aspects of the technical program, the LSE decides to contact a local university with a ranked Engineering Program about teaching a continuing education course in SE after hours. The LSE schedules a meeting with the Dean of Continuing Education (DCE), presents the course proposal, and provides a course syllabus that was well received. The DCE, however, observes that since "Engineering" is part of the proposed SE course title, it should be coordinated with the Dean of Engineering. Makes perfect sense!

A couple of weeks later, the LSE has a follow-up meeting with the DCE. All smiles, the DCE noted that the Dean of Engineering found no conflict with other Engineering courses. In fact, the DCE quotes the Dean of Engineering … "… could find NO ENGINEERING in the course at all" and "proudly recited the revered research institution speech."

Leaving the meeting, the LSE returns to their office contemplating the Dean of Engineering's response. By coincidence, a green Engineering notepad is lying on the desk with its reverse side grid facing upward. Staring at the grid, the LSE comes to a startling realization expressed in Figure 2.4. The outer boundaries of the notepad represent the complex *system* development problem the LSE has to solve. Each cell symbolizes undefined *boundary condition* problems to be bounded and specified.

In contrast, the Dean of Engineering is educating Engineers to "plug and chug" Engineering *boundary condition* problems where initial conditions, assumptions, etc. are *known*. After graduation, they enter industry and government to await someone to give them clearly defined boundary condition problems to solve using the Plug and Chug … SDBTF Engineering Paradigm. Dismayed, the LSE shakes their head concerning the void between requisite skills Engineering graduates need to enter the workforce versus the reality of today's Engineering education outcomes. But that does not help the LSE solve their current problem.

In general, these two examples illustrate the state of SE and System Development practice today in some Enterprises.

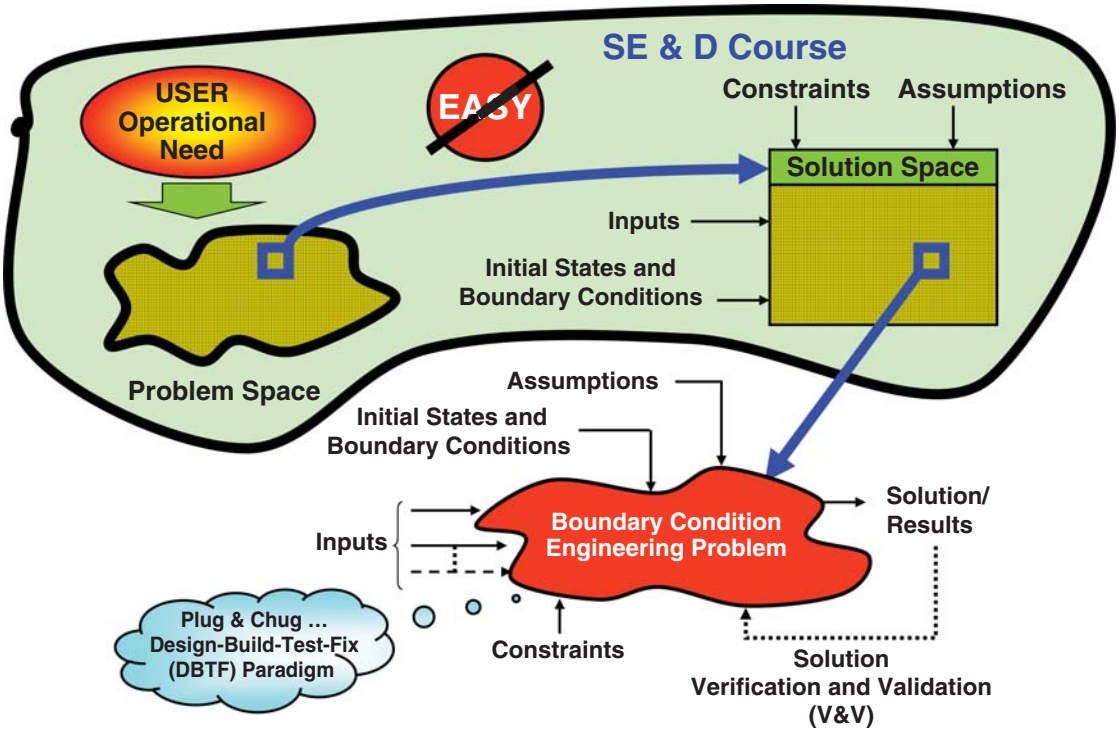


Figure 2.4 Industry and Government SE Challenges and Engineering Education Outcomes

2.4 UNDERSTANDING THE PROBLEM: ROOT CAUSE ANALYSIS

There are numerous factors that contribute to System Development performance issues such as organizational management, functional management, Project Management (PM), lack of Engineering education, especially a fundamentals of SE course, and training, etc.

To understand what is occurring in enterprises and projects, you need to understand not only the technical aspects of System Development—specifically SE—but also the Organizational Development (OD) and Instructional System Development (ISD) aspects concerning how people learn and perform. Wasson (2012) notes several enterprise performance effectors that contribute to the performance issue:

1. Ad hoc, bottom-up Engineering.
2. Misperceptions that writing specifications, developing designs, performing integration and test, and then verifying and possibly validating a system is, by definition, SE.
3. Erroneous perception that the ad hoc, Plug and Chug … SDBTF Paradigm, a convolution of the Scientific Method and a 1960s DPM, is a valid SE problem-solving and solution development methodology.
4. Incorrect portrayal of the System Development Process as being the SE Process.
5. Erroneous assumption that the SE Process applies once to the SYSTEM level.
6. Failure to recognize and understand that specifications are more than just a random "shall" statements organized via a standard outline structure.
7. Erroneous belief that SE is about creating bureaucratic documentation.
8. Failure to recognize and understand that the SYSTEM and every hierarchical ENTITY within it is characterized by four domain solutions—Requirements, Operations, Behavioral, and Physical—that when integrated comprise the overall System Design Solution.
9. Failure to recognize that SE competency requires two levels of knowledge and experience: (1) understanding the SE concepts, principles, and practices concerning the engineering of systems (undergraduate level) and (2) understanding how to efficiently and effectively tailor SE practices to achieve project objectives within technical, technology, cost, schedule, and risk constraints without compromising the integrity of the discipline (graduate level).
10. Failure to recognize that directing personnel to follow regimented organizational processes and checklists—"'Paint-by-Number' Engineering" (Wasson, 2008, p. 33)—unsupported by formal SE educational, training, and leadership will lead to Enterpriseal and project success" (Wasson, 2012, p. 4–5).

Author's Note 2.2

OSPs incorporate best practices, lessons learned, and checklists to serve as guides for planning, flexibility, and performing tasks. By education and training, bona fide SEs know and understand the process instinctively and use it as a mental reference model for decision-making.

Rigid "Paint-by-Number" processes turn SEs, System Analysts, and Engineers into *procedural robots*, not *systems thinkers*. As an example, Dr. Michael Griffin, former NASA Administrator, *cautions* about procedural interface definitions and verification without understanding the dynamic interactions (Warwick and Norris, 2010). Every project and system is different; documented processes should provide a common frame of reference while promoting *flexibility* and *agility* in thinking, action, and informed decision-making.

Objective evidence of these conditions is reflected by the NDIA SE Issues Surveys.

2.4.1 NDIA SE Issue Surveys

The NDIA System Engineering Division (SED) System Engineering Effectiveness Committee (SEEC) periodically surveys its members to assess the Top 5 SE Issues and progress in corrective actions identified in previous surveys. The most recent survey, which was conducted in 2010, identifies the following issues—*no specific priority order*:

> **NDIA 2010 SE Issue 1**—The *quantity* and *quality* of SE expertise are *insufficient to meet the demands of* the government and defense industry.
>
> **NDIA 2010 SE Issue 2**—SE practices known to be effective are not *consistently applied or properly resourced* to enable early system definition.
>
> **NDIA 2010 SE Issue 3**—Technical decision-makers do not have the right information and insight at the right time to support informed and proactive decision-making or may not act on all the technical information available to *ensure effective and efficient* program planning, management, and execution.
>
> **NDIA 2010 SE Issue 4**—Lack of technical authority can impact the integrity of developed system and result in cost/schedule/system performance impacts as the *technical solution is iterated and reworked* in later stages of the development.
>
> **NDIA 2010 SE Issue 5**—Increasingly urgent demands of the warfighter are requiring effective capabilities to be fielded more rapidly than the conventional acquisition processes and development methodologies allow.
>
> —NDIA, 2010, pp. 4–5.

You may argue these issues are unique to the defense industry and are not relevant to your industry. Commercial industry, for example, by virtue of its *competitive*, *proprietary, and intellectual property* nature does not publicly reveal issues about its own performance. For software, as an example, the Standish Group produces *Chaos Reports* that identify performance issues and metrics for Information Technology (IT) projects. Similar issues exist in commercial industry as well.

SEs and others argue that one of the reasons SE project performance is *inadequate* is due to the lack of adequate SE resources. Token resources are often allocated by projects to substantiate claims of performing SE.

2.4.2 SE Project Resources Challenges

Every technical project faces a common issue: *what is the minimum funding threshold level for SE for the project to ensure a reasonable chance of success*? In general, the answer depends on the Enterprise or project; the education, training, and competency of its personnel; project complexity; the customer; and a host of other factors.

Typically, Quality Assurance (QA) and sometimes Configuration Management (CM) are allocated an automatic percentage of the project's budget "up front" after the Project Manager (PM) takes out 10%+ as a manage reserve for contingencies and risk actions. PMs, responding to Enterpriseal command media, often pay token lip service for SE by "checking the box" and providing *insufficient* resources or funding. Since SE is not considered "touch labor" in terms of designing the SYSTEM or PRODUCT, it is often viewed as an "overhead cost" or tax.

Quantification of SE *effectiveness* and ROI is challenging. SE is like QA; it is difficult to measure the Cost of Quality; however, when QA is missing, the Cost of Poor Quality is very quantifiable such as product recalls, dissatisfied customer returns, and so on.

To address the SE effectiveness and ROI issue, Honour (2013) makes several key points based on his research that includes dissenting opinions:

1. "There is a quantifiable relationship between SE effort levels and program success" (Honour, 2013, p. 177).
2. "SE has a significant, quantifiable ROI.

> … For programs operating at near-nil SE effort, that ROI is as high as 7:1, a program cost reduction seven times as great as the added SE cost. For programs operating near the median of the interviewed programs, the ROI is 3.5:1"
>
> —(Honour, 2013, p. 178).

3. There is an optimum amount of SE for best program success.

> "For Total SE, the optimum amount of effort for a median program is 14.4% of the total program cost. For non-median programs, this value can vary roughly between 8% and 19% of the total program based on the varying program characteristics"
>
> —(Honour, 2013, p. 179).

4. "Programs typically use less SE effort than is optimum for best success."

Is there a correlation between SE Effort and meeting planned actual/planned project costs?

Honour's (2013) research indicates there is. He states "… for a median $14M program operating at 8.5% SE effort (versus 14.4% for optimal SE effort), the observed cost overrun was on the order of $1.5M; for a similar program using $200K greater SE effort, the cost overrun was only $1.0M" (Honour, 2013, p. 180). Figure 2.5 provides a correlation plot of Actual/Planned Cost normalized to 1.0 as a function of Equivalent SE Effort (ESEE) as a % of Program Cost (Honour, 2013, Figure 37, p. 110). Actual/Planned Cost > 1.0 represents a *cost* overrun.

Is there a relationship between SE Effort and meeting planned actual/planned schedule targets?

Honour's (2013) research indicates there is. Figure 2.6 illustrates results that correlate a correlation plot of Actual/Planned Schedule results normalized to 1.0 as a function of ESEE as a % of Program Cost (Honour, 2013, Figure 38, p. 110). Actual/Planned Schedule > 1.0 represents a *schedule* overrun.

Is there a relationship between SE Effort and overall success?

First, what constitutes overall success? Honour (2013, p. 43) states that Overall Success is a *subjective* measure in which interview participants were asked to *estimate* stakeholder satisfaction on a *subjective* scale. Figure 2.7 provides a correlation plot of Overall Success normalized to 1.0 as a function of ESEE as a % of Program Cost (Honour, 2013, Figure 39, p. 111). Overall Success < 1.0 represents a *degree of success*.

Author's Note 2.3

Honour's Research (2013)

Please note that Honour's research does not differentiate if data were collected from projects that employed the ad hoc Plug and Chug … SDBTF Paradigm or true SE as addressed in this text or enterprise/project SE capability.

As a final point, observe the *bow* like curves in Figures 2.4–2.6. In terms of meeting Actual/Planned Cost and Schedule targets, *reducing* or *adding* resources on either side of the *optimal* 14.4% of Program Cost for SE Effort can have *negative* performance effects.

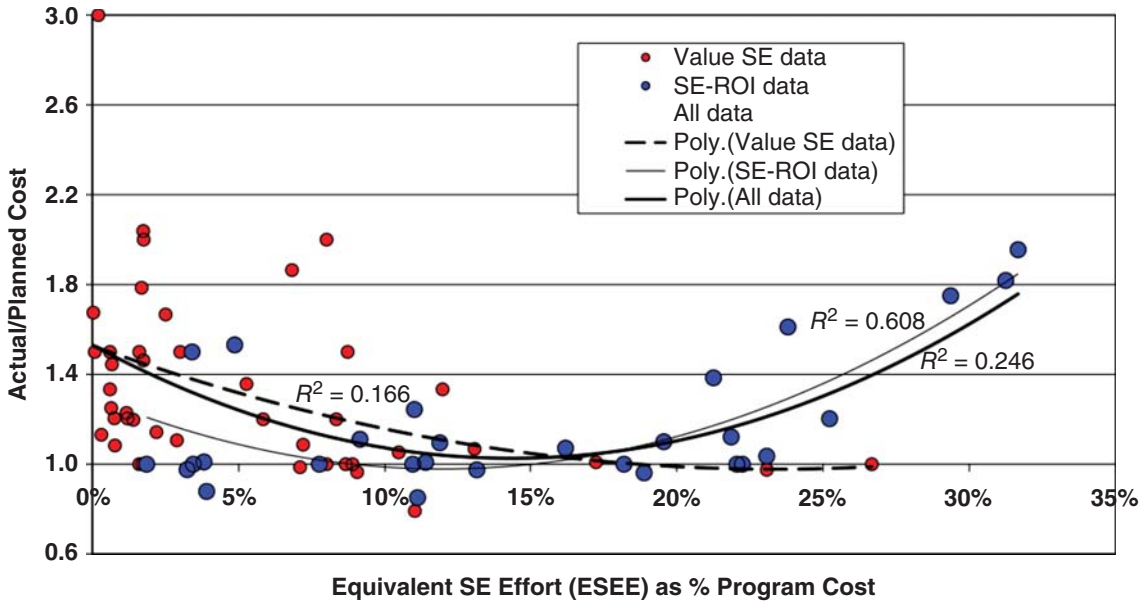


Figure 2.5 Correlation Plot: Cost Overruns (Normalized) as a Function of Equivalent SE Effort (EESE) (Percentage of Program Cost) (Source: Honour (2013). Used with permission.)

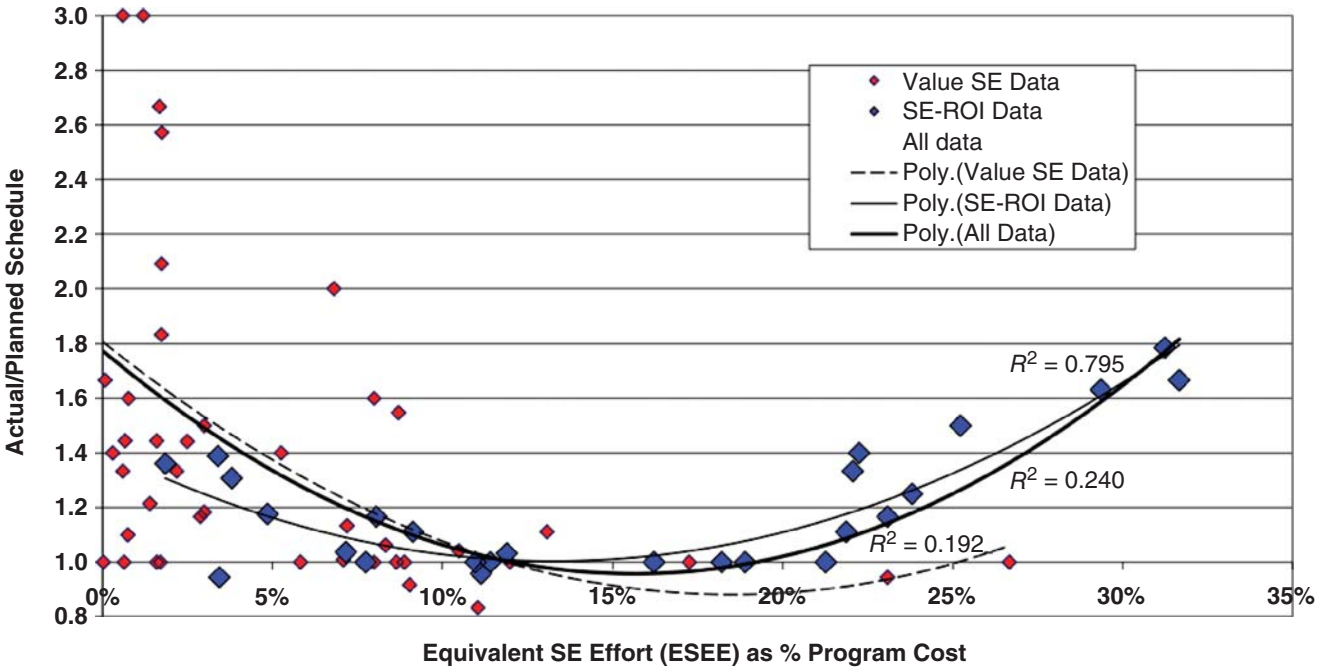


Figure 2.6 Correlation Plot: Project Schedule Overruns (Normalized) as a Function of Equivalent SE Effort (EESE) (Percentage of Program Cost) (Source: Honour (2013). Used with permission.)

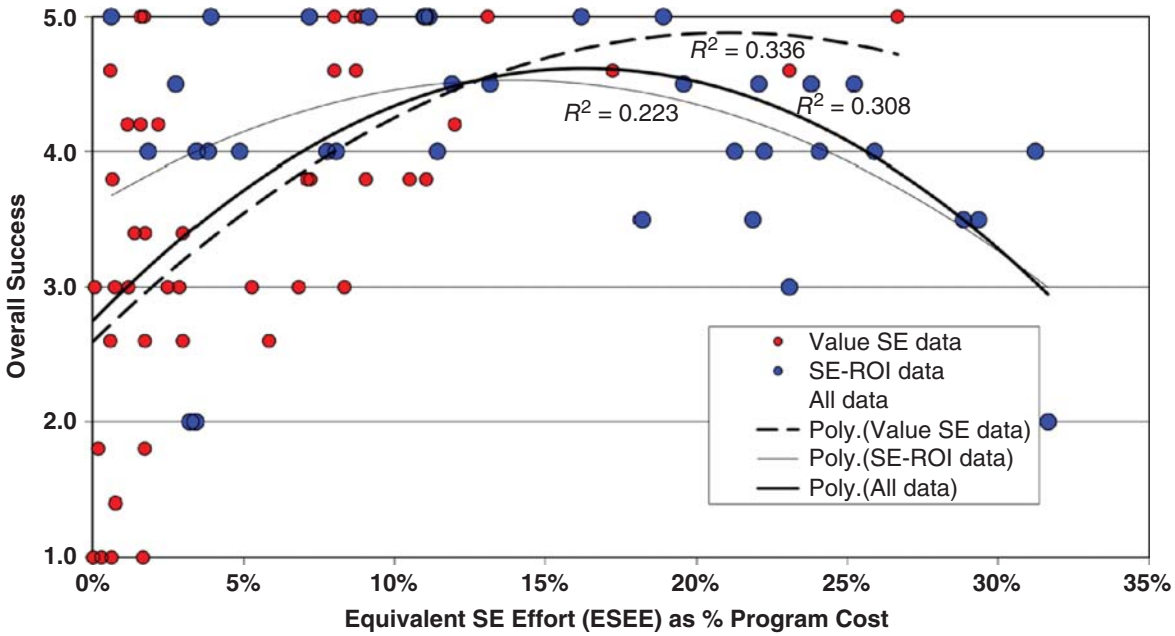


Figure 2.7 Correlation Plot: Overall Project Success (Normalized) as a Function of Equivalent SE Effort (EESE) (Percentage of Program Cost) (Source: Honour (2013). Used with permission.)

Referral For additional information concerning the *optimal* level of SE resources for a project, refer to Honour (2013) for a detailed discussion of his research.

How do we solve these problems? This brings us to our next topic, Industry, Government, Academic, Professional, and Standards Organizations Solutions.

2.5 INDUSTRY, GOVERNMENT, ACADEMIC, PROFESSIONAL, AND STANDARDS ORGANIZATIONS SOLUTIONS

Principle 2.1

User's Problem Space Principle

Thoroughly understand the *problem* or *issue* the User needs to solve, not just surface level *symptoms* that fail to reveal the underlying *root cause* and its contributors.

Most people often think of SE concepts, principles, and practices as applying only to Engineered Systems such as computers, smartphones, and aircraft. Yet, fail to recognize that the same SE concepts, principles, and practices apply to Enterprise Systems—divisions, departments, and projects—that develop Engineered Systems.

When Enterprise Systems exhibit project performance issues, SE views each issue as *problem space* (Chapter 4) that must be prioritized for corrective action by one or more *solution spaces*. This section addresses how industry, government, academia, professional, and standards organizations provided *solution space* corrective actions to deal with various types of System Development, SE, SwE, and other performance issues—*problem spaces*—over the past 50 years.

Since the emergence of modern SE in the WWII timeframe, industry, government, academia, professional, and standards, organizations have tried to address the relationship of SE effectiveness to project performance *problem space*. Honour's (2013) research in the preceding discussion illustrate one aspect Enterprises understand – application of funding resources to a problem space in the hope it will go away.

In general, fragmented *solution spaces* to project performance problems have evolved such as … we need better standards … we need more processes … we need to assess our SE capabilities to perform. The solution has been to pile on layers of standards, processes, capability assessments, etc. As a result, metrics indicate that Enterprise performance has improved … up to a point of *diminishing returns*. Yet, *System Development performance issues persist after all of this investment*.

Although the evolving solutions have been *necessary*, they have been *insufficient* in addressing at least one key issue—how SE is understood and applied. The flaw in the *solution space* discussions that follow is how the underlying Plug and Chug … SDBTF Engineering Paradigm has been allowed to migrate *unchecked* and *uncorrected* through

higher education and subsequently into industry to merge with another DPM paradigm (Archer, 1965). To better understand the evolution of industry, government, academic, professional, and standards organizations solutions, let's investigate the historical legacy.

2.5.1 SE Standards Solutions

Since the 1950s, SE standards have been developed and evolved to establish a basis for *consistently* developing systems that will deliver the required outcomes and levels of performance. Most notably, the initial releases of various standards include:

AFSCM 375-5 (1966) Air Force Systems Command Manual 375-5, *System Engineering Management Procedures.*

MIL-STD-499 (1969) Military Standard: *Systems Engineering Management.*

FM-770-78 (1979) US Army Field Manual: *Systems Engineering*.

EIA/IS 632-1994 (1994) Interim Standard: *Processes for Engineering a System.*

IEEE 1220-1994 (1995)—IEEE Trial-Use Standard for *Application and Management of the SE Process*.

ISO/IEC 15288:2015 (2015) *Systems and software engineering—System life cycle processes.*

For some of today's senior SEs, the evolution of these standards became a source of SE education. SE courses were simply not *available* or *accessible* for many unless you lived near one of the few major institutions. Johnson (2002) addressing the evolution of SE observes that system failures in the first half of the 1900s were due to a failure to "manage" the development. As such, Systems Management processes and standards became a focal point for System Development. This point is illustrated by SE standards titles in the list above. Today, education and training continue to focus on Systems Acquisition and Management and Specialty Engineering courses where anecdotal evidence suggests that most SEs spend less than 5% of their time; yet ignoring SE&D (Figure 1.1) courses where the remaining 95% of most SE's time is consumed.

Enterprises often contend that their SE Process (SDBTF) is a *problem-solving* and *solution development* methodology. In some respects, especially in Research and Development (R&D), that is true. However, the core methodology of the Scientific Method is a process of scientific inquiry and investigation requiring hypothesis development, testing, refinement, and validation. SE&D is not always an exact science due to the uncertainties and frailties in human decision-making. SE&D involves *application of technology* to a system, product, or service, not R&D. If a project requires an SE *problem-solving* and *solution development* methodology focused on *hypothesis testing*, either the problem is not well understood or the technology requires further refinement and maturity for application development.

Author's Note 2.4

Types of Engineering Problems

The nature of some complex Engineering problems is so challenging that they are characterized as:

1. *Wicked—Rittel Lecture (Pre-1967), Churchman (1967).*
2. *Vexing—These do require iterations of design solutions.*

Referral For additional information about *wicked problems*, refer to Goldman (2013, p. 6), Rittel and Webber (1973), Rowe (1998, p. 41), and Vitasek (2014).

Military organizations, for example, often release a sequence of contracts over time in which the outcomes and work products—specifications and designs—of each contract become the basis for the next phase until specification requirements are *sufficiently mature* to develop a reliable and mature system.

The preceding hypothesis testing discussion above refers to general, everyday Engineering problems that are not considered *wicked* and *vexing*. Those generally fall into the realm of R&D due to technology and solution maturity.

2.5.2 Current SE Process Paradigms

Enterprises will often contend they have a *problem-solving* and *solution development* process based on the System Engineering Process originating in MIL-STD-499B Draft (1994). Key steps of the process include Requirements Analysis, Functional Analysis/Allocation, Synthesis, and System Analysis and Control (Balance). The process has served the military well and is often referenced in textbooks and Enterprise OSPs.

Commercial Enterprises and engineers often rebuff the process as indicative of bureaucratic paper work. In terms of advancing SE knowledge and best practices, the SE Process as a paradigm needs to be shifted to a new SE Process paradigm (Chapters 11 and 14) to resolve many of its *deficiencies* and reflect new SE approaches.

Enterprises and projects today need a global SE Process that is applicable to multiple industries—public and private—that reflects current thinking, is easy to learn, and overcomes the deficiencies of the MIL-STD-499B Draft (1994) SE Process. For example:

Requirements Analysis infers that requirements exist and are available to be analyzed. This is seldom the case, especially when customers may not know what their needs or objectives are. In fact, only an abstract vision, problem, or issue may exist. Requirements Analysis is a "downstream" activity that needs to be replaced by customer-focused *systems thinking* in terms of understanding the User's problem or issue space to be solved.

Functional Analysis/Allocation became outdated in the 1980s. If you challenge this, ask software developers. Yet, SE Enterprises seemingly continue to cling to the Functional Analysis concept forever. As stated in Chapter 1, a *function* simply expresses a *unitless action* to be performed. Users expect a system, product, or service to produce performance-based *outcomes*, not *functions*! When people refer to Functional Analysis and Decomposition, that is the easy part. The challenge of SE is bounding, specifying, and allocating capabilities with performance-based magnitudes. Recognize and appreciate the difference! More on this topic will be presented in Chapter 21.

Synthesis although valid as a label, it is an *abstract* term that few understand. There are better ways of expressing the solution conceptualization, formulation, selection, and development of a solution than *abstract* terms such as Synthesis.

System Analysis and Control are clear and concise; however, System Control is a technical management issue, not a technical process.

Then, there are the *missing* process elements. The Requirements → Functions → Synthesis flow is outdated, flawed, and does not clearly reflect how SE is performed (Chapters 11 and 14). Specifically, SDBTF Enterprises prematurely take the Requirements → Functions → Synthesis quantum leap (Figure 2.3) and neglect two critical elements in the flow.

Operations—How the User intends to deploy, operate, maintain, sustain, retire, and dispose of a fielded system that leads to defining *system behaviors*.

Behavior—How the User expects the system or product to respond behaviorally to external stimuli, excitations, and cues that lead to formulation, development, evaluation, and selection of a physical implementation.

Attempts have been made to express these missing elements in terms of *views* of a system: (1) an Operational View, (2) a Functional View, and (3) a Physical View. The problem is we now have:

1. An SE Process with one set of *activities*—Requirements Analysis, Functional Analysis, Synthesis, and System Analysis and Control.
2. Another set of *views*—Operational, Functional, and Physical.

Understanding the MIL-STD-499B Draft (1994) SE Process is exacerbated by another paradigm. Enterprises, functional organizations, projects, and Engineers often believe that the SE Process is only performed once at the System level. Chapters 12 and 14 shift this paradigm to a new SE Process.

In summary, the time has come to embrace new concepts for an SE Process. What is needed is an SE Process based on a *problem-solving* and *solution development* methodology applicable to any type of Enterprise System or Engineered system, product, or service. We will introduce the foundation for new SE Process in Chapters 11 and 14 after we have established an understanding in Part I, "System Engineering and Analysis Concepts."

2.5.3 The Emergence of Software in Systems

During the 1980s, systems began to leverage microprocessor and other technologies to accommodate the need for quick modification changes that otherwise would have taken considerably longer in hardware—procurement of parts: Fabrication, Assembly, Inspection, and Test (FAIT). However, with the emergence of Software Engineering as a discipline coupled with rapid growth of software intensive systems, challenges emerged—how to develop quality software *efficiently* and *effectively*, especially in *mission-critical* and *safety-critical* systems such as manned space flight, medical systems, etc. During the early years, EEs and others often wrote the software for some hardware applications.

To address these challenges, the US DoD established the Software Engineering Institute (SEI) at Carnegie Mellon University in 1984 as a federal research center. In 1987, the SEI published the Capability Maturity Model (CMM) for Software (SEI, 2014) to assess Enterprise capabilities to develop software. The CMM continued to evolve over several years to a release of the SW-CMM Version 1.0 in 1991.

Developing quality software was just one aspect. Of particular interest to the DoD was the need to contract with Enterprises that were capable of developing, producing, and delivering quality software that was technically compliant, on-schedule, and within budget. Significant resources were invested in SW training, process development, and assessments. As a result, the quality of SW improved dramatically. However, one of the discoveries was that SW engineers were developing improved software to very poor requirements originating from SE.

From the 1950s to 1980s, SEs, typically EEs, wrote software requirements. Since the SEs and EEs were hardware

designers, they wrote the specification requirements and selected the computer hardware processor whether it was appropriate for the software application or not. These decisions were "thrown over the transom to SwEs" for implementation with little or no collaboration, opportunity for review, acceptance, or approval. Fortunately, many of these old Engineering paradigms have shifted in most Enterprises.

Recognizing the significance and criticality of software in systems today and its contributions to project failures, industry responded and formed Systems and Software Engineering divisions, departments, etc. to clearly communicate to their customers that they were addressing the SE–SW integration issue. *Hardware*, an integral part of every system, was noticeably absent. The title raised questions and appeared odd to many people, especially given the "multi-disciplinary system integration" public relations messages. For example, *did it mean Systems (Engineering) and Software Engineering*? If so, *wasn't hardware part of "multi-disciplinary systems integration"*? *Or did it mean Systems (Hardware) and Software Engineering*, *Systems (Engineering and Hardware) and Software Engineering*, *and so forth?*

2.5.4 The Evolution of OSP Solutions

Faced with global competition, the automotive industry began redefining itself by documenting processes, performing statistical process control—Six Sigma—and other methods in the 1980s. Other industries began to address the *ad hoc* and *chaotic* nature of System Development project performance based on system failures and poor project performance. They reasoned that they should be able to develop systems, products, or services the way automobile manufacturers designed and mass-produced automobiles. From an Enterprise perspective, the solution became "everyone document your (SDBTF) processes." As a result, Engineering Manuals that documented THE (SDBTF) Engineering Process including the SE Process began to emerge.

Unfortunately, due to a lack of SE education and commitment of training resources by enterprises, most of these processes turned into "Paint-by-Number Engineering" (Wasson, 2008, p. 33) exercises—perform to the processes. Many of these processes did nothing to shift the SDBTF Paradigm. If something went wrong, along came another *patch and fix* process.

In 2010, Warwick and Norris (2010) posed an interesting question to industry leaders that included Dr. Michael Griffin, former NASA Administrator: *Is it time to revamp SE*? The context of the article addressed the application and validity of SE requirements decomposition methods to *complicated* versus *complex*—dynamically changing—systems. Some view SE decomposition methods as applicable only to *complicated* systems, not *complex* systems such as Systems of Systems (SoS) due to their autonomy as well as their evolving and dynamic nature. Dr. Griffin makes key observations that seem applicable to the "patch and fix processes" current State of SE Practice:

1. "How is it that we continue to encounter failure of important and complex systems where everything thought to be necessary in the way of process control was done, and yet despite these efforts the system failed?" Griffin asks. "The answer cannot lie in continuing to do more of the same thing while expecting a different outcome." (Warwick and Norris, 2010).
2. "Adding processes is not the right answer, he believes. What is needed is a new view that the core SE function 'is not primarily concerned with characterizing the interactions between elements and verifying that they are as intended.' What's more important, he says, is understanding the dynamic behavior of those interactions." (Warwick and Norris, 2010).

To address these issues, industry, government, and professional organizations have attempted over several decades to institute standards as safeguards to ensure *consistent, repeatable,* and *predictable* SE processes. SE is not perfect by any means as illustrated by the ripples in Panel 3 of Figure 2.2. After all, the *ambiguities* and *uncertainties* of human decision-making are still the driver of any System Development project and its performance.

Yet, as Dr. Griffin observed, "adding processes is not the solution." Dr. Albert Einstein also observed the following:

> "**Insanity:** Doing the same thing over and over again expecting different results"
>
> —(Einstein).

As time progressed, industry and government began to recognize that OSPs needed to be traceable to a global SE standard such as ISO/IEC 15288 for Systems and ISO/IEC 12207 for Software. Although these standards were a major stride forward, they did not correct the project performance issues traceable to the SDBTF–DPM Engineering Paradigm.

2.5.5 The Evolution of SE Capability Assessment Solutions

In the early 1990s, several organizations developed and evolved various Capability Assessment Models (CAMs). Examples include:

In 1994, the SEI released its SE CMM (SE-CMM) Version 1.0 (SE-CMM, 1994). Recognizing the interdependence between SE, SW, people, and other drivers in project performance, the SEI released other models under the Capability Maturity Model Integration® (CMMI®) title. The CMMI® is now administered by the SEI Capability Maturity Model Institute at Carnegie Mellon University.

In 1994, International Council on Systems Engineering (INCOSE) released the initial draft of its SECAM. The SECAM Version 1.2 was released later that year (SECAM, 1996, p. 6).

In 1994, the EIA, the INCOSE, and the Enterprise Process Improvement Collaboration (EPIC) collaborated and released EIA 731-1 *SECM* (EIA 731.1, 2002, p. 1).

Eventually, the CMMI emerged as the reference model for assessing Enterprise capabilities.

Referral For additional information about the history of these models, refer to EIA 731.1 SECM (2002, p. 1), INCOSE SECAM (1996, pp. 132–139), Paulk (2004), Scacchi (2001), etc.

So, *what is the CMMI*? The CMM Institute defines CMMI as follows:

CMMI "…a process improvement approach that provides organizations with the essential elements of effective processes that will improve performance. CMMI-based process improvement includes identifying your organization's process strengths and weaknesses and making process changes to turn weaknesses into strengths" (CMMI, 2014b).

The CMMI is comprised of three models:

CMMI for Acquisition (CMMI-ACQ) Model "provides guidance to organizations that manage the supply chain to acquire and integrate products and services to meet the needs of the customer."

CMMI for Development (CMMI-DEV) Model "is used for process improvement in organizations that develop products. CMMI-DEV provides guidance to improve the effectiveness, efficiency, and quality of their product development work."

CMMI for Services (CMMI-SVC) Model "provides guidance to organizations that establish, manage, and deliver services that meet the needs of customers and end users" (CMMI, 2014a).

Author's Note 2.5

Observe the CMMI focus on processes, which are a key to ensuring *consistency, repeatability,* and *predictability* and quality in Enterprise performance. However, processes *assume* that there is a solid foundation in Engineering disciplines concerning how to *efficiently* and *effectively* engineer systems based on multi-discipline SE *methods* commonly shared across various disciplines. When this foundation is missing, processes become "Paint-by-Number" Engineering (Wasson, 2008, p. 33) discussed earlier. Instead, well-trained professionals who are *agile*, understand the multi-discipline SE process, and know *instinctively* how to apply and *tailor* the processes efficiently and effectively should be empowered to lead technical projects.

Some Enterprises and projects have the *misperception* that if you are assessed to be compliant with a CAM, you will be successful. In theory, this is true; however, recognize that a CAM assesses the *capability* to perform based on threshold capability criteria at several levels. CAM assessments *do not guarantee* project success; project personnel do. Remember that a CAM does not tell an enterprise how to conduct its business – SDBTF versus SE&D; it only assesses an Enterprise's "capability to perform" based on the CMMI Model.

This raises a *hypothetical* question. Let's assume that an Enterprise does all of the right things:

1. Documents their OSPs.
2. Verifies OSP compliance to standards such as ISO/IEC 15288 for Systems, ISO/IEC 12207 for Software, the ISO 9000 series, and so forth.
3. Achieves a CMMI maturity assessment rating in all or some of the process areas.

Why then do the System Development problems persist? Obviously, it ultimately comes down to people—Engineers, Analysts, etc.—who know how to efficiently and effectively apply SE problem-solving and solution-development methods. However, *what do Engineers, Analysts, etc. do?* The answer is what they have been educated and trained to do—SDBTF.

During a capability assessment, evaluators typically sample and assess two or three "representative" projects. Observe the term "representative." The assumption is that the assessed projects are "representative" of every project within the enterprise. The assessments consist of interviews with project personnel and Enterprise management, reviews of OSPs, and reviews of *work products* such as plans, specifications, and requirements traceability. as objective evidence of compliance. *Is it conceivable that SDBTF Paradigm Enterprises can produce the required interviews and objective evidence for an assessment—that is, do all of the right things—but still exhibit project performance issues or failures? Are these issues or failures attributable to the underlying SDBTF paradigm as the root cause but never explicitly identified as a weakness*? The answer is yes!

2.5.6 SE Tool Solutions

The evolution of better, faster, and cheaper technologies such as microprocessor-based computers and Software Engineering methods in the 1980s enabled rapid expansion of SE

tools. This enabled Engineers to link and manage multi-level specification requirements, architectures, Modeling and Simulation (M&S), etc. In some respects, you could make the following observation:

> *Our ability to model and manage systems data exceeds the capability of most SDBTF Enterprises to implement SE.* (Wasson)

Today, the concept referred to as Model-Driven Design or Development (MDD) such as Model-Based Systems Engineering (MBSE) has been expanding rapidly. Watson (2008) provides an overview of the motivation for model-driven development. People often erroneously believe that the underlying MBSE concept is new based on the promotion of the term in recent years. However, MBSE as a concept is traceable to the 1950s.

SDBTF Enterprises often view themselves as successful despite *inefficient* and *ineffective* methods. Therefore, they see SE as *unnecessary*. Rather than learn SE methods, there is a tendency by these Enterprises to bypass SE training. Instead, they purchase an MBSE tool. After all, if you have an MBSE tool, you must be, by definition, performing SE. Wasson (2011) observes that an old cliché—*Owning a paintbrush does not make one an artist*—applies to these situations.

A Word of Caution 2.1

Unless an Enterprise understands the SE concepts, principles, and practices discussed in this and other texts, an MBSE tool is nothing more than a *drag-and-drop* graphics application for creating nice pictures. Remember that an MBSE tool is only as *valid* as the competency of the Engineer creating the model, entering the data, and linking the information.

2.5.7 The SE Certification Solution

Beginning in the 1990s, the need to certify SEs began gaining momentum. In 2004, the INCOSE initiated its Professional Certification Program (INCOSE, 2014). Since the program's beginning, INCOSE has expanded it to include the following types of certifications:

CSEP—Certified SE Professional.

ASEP—Associate SE Professional.

ESEP—Expert SE Professional.

Applicants are required to meet various levels of qualifications and experiences including validation of experience and taking an exam based on a specific version of the INCOSE Handbook. Rhetorically, the challenge question and opportunity is: *does taking a certification exam solve the educational SDBTF–DPM Paradigm problem*?

2.5.8 Solutions Summary

In summary, documenting OSPs, tracing compliance to standards, assessing Enterprise capabilities, and certification are *absolutely essential* to achieving *consistent, repeatable,* and *predictable* performance. However, the challenge is that they do not correct a flawed SDBTF Paradigm that may be deeply embedded within the culture of an Enterprise.

Given this background, let's define the central problem.

2.6 DEFINING THE PROBLEM

Wasson (2012) summarizes the project performance problem as follows:

> "*Despite the formulation and development of Systems Engineering capability assessment and competency models, certifications, education and training courses, et al, system development projects continue to exhibit technical performance issues concerning the engineering of systems*"
>
> —(Wasson, 2012, p. 4).

How do we solve this problem and overcome the SDBTF Engineering Paradigm? The answer resides in SE educational and Enterprise paradigms—specifically what SE is, its methodology, and how and where it is applied, etc. There is an old cliché that states:

> *To understand the tiger, you must go into the jungle.* (Anonymous)

To understand the underlying SDBTF Paradigm, you need to have direct, day-to-day contact—work, leadership, and conversations—with Engineers and Analysts developing systems and products and observe their actions and patterns of behavior.

If you observe how Engineering is performed, patterns of behavior emerge in their daily work habits and perceptions of SE. Specifically, the perception of SE is:

> Read spec requirements, create a single point design solution, compute a few equations to determine nominal component values or performance, research catalogs to find components that fulfill or can be modified to meet design requirements, create a prototype, and conduct some laboratory tests, tweak and iterate the design over and over in an *endless loop* until it meets its specification requirements.

If you characterize this process graphically, Figure 2.8 emerges.

When new Engineering graduates enter the workforce, they soon discover that their intrinsic Plug and Chug … DBTF Paradigm is expanded into a Plug and Chug … SDBTF Paradigm. SYSTEM or PRODUCT development is

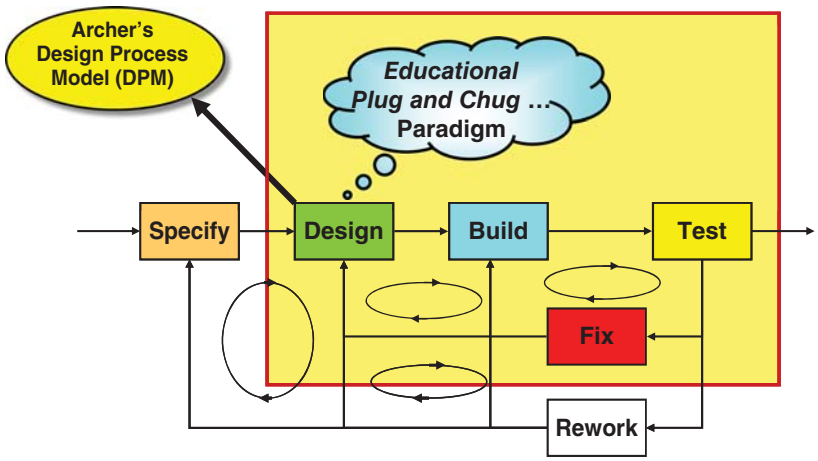


Figure 2.8 Graphical Depiction of the Plug and Chug … Specify-Design-Build-Text-Fix (SDBTF) Engineering Paradigm with Archer's Embedded Design Process Model (DPM)

performed in a seemingly *endless loop* until the evolving point design solution complies—*system verification*—with customer requirements. As a point design solution, it may or may not have been selected based on an AoA (Chapter 32). Additionally, compliance to customer specification requirements, which is a binding condition of contracts, does not necessarily mean that the SYSTEM or PRODUCT is the *right solution—system validation*—to fulfill the User's operational need(s) (Wasson, 2012, p. 13). It simply means that the system or product meets technical specification requirements much like the better like the "Engineered Mousetrap" presented earlier in Example 2.1.

You may ask: *what is the problem with SDBTF Engineering Paradigm? That's what engineers do*. You are correct. Engineering, which is dependent on human decision-making, is not an exact science. The System Development Process (Figure 15.2) does have a general Specify-Design-Build-Test workflow and inevitably requires rework to some degree, depending on the effectiveness of the Engineers applying the SE Process and making insightful, timely, decisions. However, this is not the context of the SDBTF Paradigm. The central issue is:

> The SDBTF Engineering Paradigm (Figure 2.8) is an *endless loop* set of Engineering activities erroneously employed as a *problem-solving and solution development* methodology with no identifiable outcomes other than to "design components, integrate, and deliver them on time and within budget."

At a component level, Engineers acknowledge that their *ad hoc, endless loop* (SDBTF) methods are often inefficient and ineffective. Now, imagine the chaos compounded by multi-discipline teams with no common problem-solving and solution development methodology attempt to develop moderate to complex systems with hundreds and thousands of components at various levels of abstraction (Figure 8.4).

From an SE perspective, although the SDBTF methods might work on a small scale for development of a device, they simply are not *scalable* for application to moderate to large, complicated and complex systems. Objective evidence of this condition is readily apparent when Enterprises and projects claim to adopt SE methods and begin applying them to the project without appropriate SE education and training. Then, when the first milestone is missed; customers, executives, and Project Managers (PMs) panic; and the project immediately reverts back to its primal SDBTF instincts, which exacerbates the multi-level chaos.

The SE Process Model shown in Figure 14.1 is a scalable, problem-solving and solution development methodology applicable to the SYSTEM or any hierarchical ENTITY within it. The SE Process corrects the amateurish, *quantum leap* approaches from Requirements to Physical Implementation illustrated in Figure 2.3. Two key points:

First, recognize that SE is not a linearly sequential SDBTF process. In fact, the steps are interdependent. As we will discover in Part 2, "SYSTEM DESIGN AND DEVELOPMENT PRACTICES," you have to understand the preliminary design implications at lower levels of abstraction to evaluate the *reasonableness* of achieving a system or entity's specification requirements. Also, the inference of SDBTF Paradigm (Specify-then-Design-then-Build-then-Test-then-Fix) as sequential steps is reflective of the rigid, inflexible, 1960s Waterfall Model (Figure 15.1) that created major problems for System Developers.

Secondly, the problem originates within the SDBTF "Design" box—the focal point for some SE courses. Specifically, most design methodologies, assuming they are taught at all, are typically too abstract. Case

in point: the MIL-STD-499B Draft (1994) SE Process discussed earlier and currently embraced by many industry Enterprises and government organizations is simply labeled *Synthesis*.

The reality is most design methodologies approach design as an exploration and discovery of the *unknown* -"Here Be Dragons," a label used by ancient cartographers. Objective evidence within an Enterprise or project of this condition is illustrated when they inform customers about plans to "discover" specification requirements. Exploration of the *unknowns* such as User's abstract operational needs requires a *problem-solving* and *solution development* methodology that leads to design solution outcomes, not *ad hoc, endless loops*.

By virtue of the introduction of the Scientific Method of inquiry and investigation in middle and high school education, students naturally gravitate to it as their de facto *problem-solving* and *solution development* methodology. As a result, Engineering Design becomes an iterative process of *trial-and-error* hypothesis—conceptual design—testing via laboratory prototypes, models, and simulations. Then, by a continuous process of rework and refinements. The process iterates as an *endless loop* until: (1) a final solution that complies with Acquirer specifications or derived specification requirements is achieved or (2) "terminated for convenience" by the customer due to frustration or resource depletion. Unfortunately, this paradigm is allowed to migrate *unchecked* and *uncorrected* throughout most undergraduate Engineering programs. *Why*? Two reasons are:

Reason #1—Academic instructors often lack industry SE experience to be cognizant of the problem. Ironically, executives, managers, and engineers who work in industry and government with direct exposure to the problem typically fail to recognize it as well.

Reason #2—The SDBTF paradigm is congruent with academic research and scientific inquiry methodologies, a key focus of many academic institutions (Mini-Case Study 2.2).

How did the SDBTF Engineering Paradigm originate? This brings us to our next topic, Archer's DPM.

2.6.1 Archer's Design Process Model (DPM)

In 1963–1964, Archer (1963–1964) published a series of *Design* articles in which he summarized a design method used by designers. In 1965, the collection of articles was compiled into a booklet titled *Systematic Methods for Designers* (Archer 1965). These articles not only synthesized these patterns into the DPM but also culminated in a design methodology as a frame of reference.

Archer's DPM Model illustrated in Figure 2.9 consists of Programming—Acquired Knowledge, Data Collection, Analysis, Synthesis, Development, and Communication. *Highly iterative* loop-backs provide corrective action feedback from:

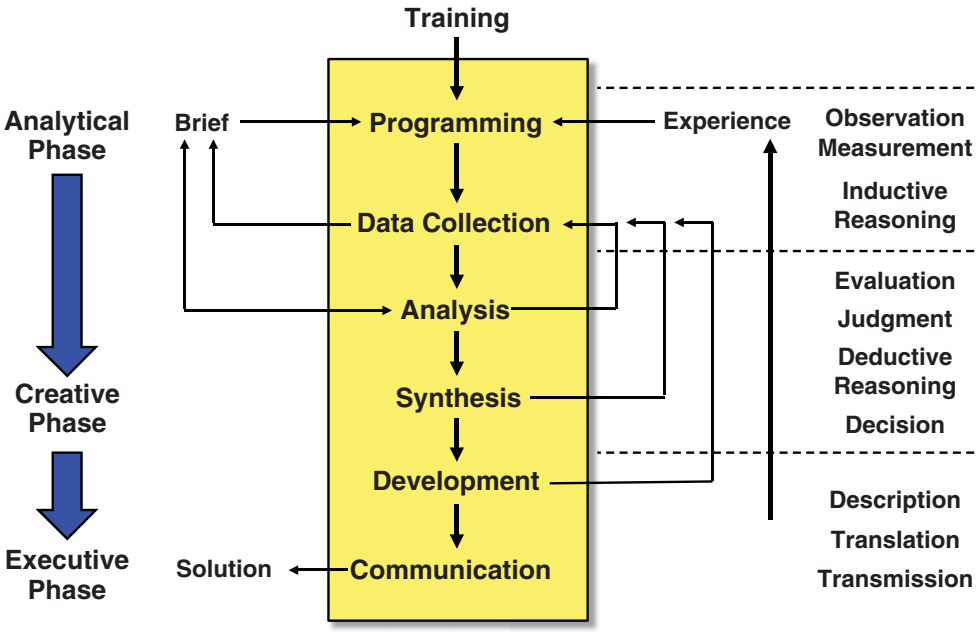


Figure 2.9 Archer's DPM (Source: Reprinted from Rowe (1998), Design Thinking, published by The MIT Press, modified from its original presentation in Archer (1965).)

- Data Collection and Analysis feedback to Brief and subsequently to Programming.
- Analysis, Synthesis, and Development feedback to Data Collection and so forth.

Observe the right side of the diagram that characterizes attributes of each phase such as Inductive Reasoning, Evaluation, Judgment, and Deducting Reasoning. While these are certainly valid, these attributes are too abstract, especially for new students, to properly perform SE&D. Given the abstractness, everyone within an Enterprise or project implicitly evolves their own version of the DPM. In principle, the DPM is fine as an investigative *problem-solving* and *solution development* methodology. However, observe that it does not preclude Engineers from *prematurely* taking the "quantum leap" from Requirements to the Physical Implementation illustrated in Figure 2.3. You may ask: *how does the SDBTF–DPM Engineering Paradigm evolve within an Enterprise*?

For decades, the Engineering SDBTF–DPM Engineering Paradigm has been ingrained as part of the Engineering culture in many Enterprises. Executives and functional managers immersed in the SDBTF culture get promoted, are often unaware of the paradigm's existence in their Enterprises or its effects on System Development performance, and *unwittingly* allow it to perpetuate culturally. "It's always worked for us … if it isn't broken, don't fix it."

Johnson (2002, p. 1) observes that Engineered System performance problems were perceived to be due to a lack of (rigid) systems management – managing the university trained "knowledge workers" coined by Drucker (1959) – via processes and other methods. However, the "knowledge workers" had little interest in rigid systems management. He adds that they promoted new idea generation using an "*undefined process* that no one could routinize." In effect, it *invalidated* the scientific management techniques that were believed to solve the Engineered System performance problems.

Within an Enterprise, objective evidence of the SDBTF Engineering Paradigm's existence comes from PMs who legitimately protest that Engineers … "never seem to come to completion designing a system or product." They lament that if project cost and schedule constraints were not an issue, Engineers would "never finish a design."

In response, Engineers respond that PMs simply do not understand what is required to design (SDBTF) a system or product. Ironically, they and their managers submitted cost estimates as part of the proposal effort. They add:

> "It's how we have always developed systems. Yes, system development is *chaotic* and *inefficient* … but we eventually get the system built after spending a lot of nights, weekends, and holidays. That's how Engineering is performed, … at least in our organization."

Then, when projects are eventually completed and delivered, award ceremonies are conducted by executives to bestow well-deserved accolades for the professionalism and dedication of outstanding personnel "who worked nights, weekends, and holidays to get the job done" (based on our *ad hoc, inefficient, and ineffective* SDBTF–DPM Paradigm). Imagine what these teams could have accomplished … if … the Enterprise had an *efficient* and *effective* problem-solving and solution development SE Process (Figure 14.1) that was not ad hoc and overcomes the problem illustrated in Figure 2.3.

The irony is that Enterprises strive to improve market share, shareholder value, and profitability. Yet, are seemingly oblivious to Enterprise performance effectors such as the SDBTF–DPM Engineering Paradigm and its impact on project and system, product, or service quality and performance that impact stakeholder satisfaction and profitability. These are the managers who proclaimed they sent XX SEs to SE training courses. The challenge question is:

1. Did the course instructors understand the SDBTF–DPM Paradigm issue and its underlying DPM?
2. Did the SE courses highlight and correct the SDBTF–DPM Engineering Paradigm?
3. Did you as a functional manager leverage the training to eliminate the SDBTF Engineering Paradigm?

The answer to all of these questions is probably NO!

This brings us to a key question, *how did the SDBTF–DPM Paradigm become rooted in industry and government Enterprises*? The answer resides in what industry and government do: *design systems to produce products and services for their respective marketplaces*. Figure 2.10 provides an illustration.

Beginning in middle and high school, students are taught the Scientific Method as a *problem-solving and solution development* methodology for scientific *inquiry* and *investigation*. On graduation, they enter higher education to acquire Engineering and other technical degrees. During their education, they are exposed to two types of educational paradigms:

- **Plug and Chug Paradigm** employed as an Engineering classroom teaching model and homework problem-solving.
- **DBTF Paradigm** acquired through refinements and corrections to laboratory experiment exercises.

To better understand the origins of these paradigms, let's explore how they migrate into the workplace. Figure 2.10 serves as a reference model.

On completion of their engineering degree, Engineering graduates enter industry or government and

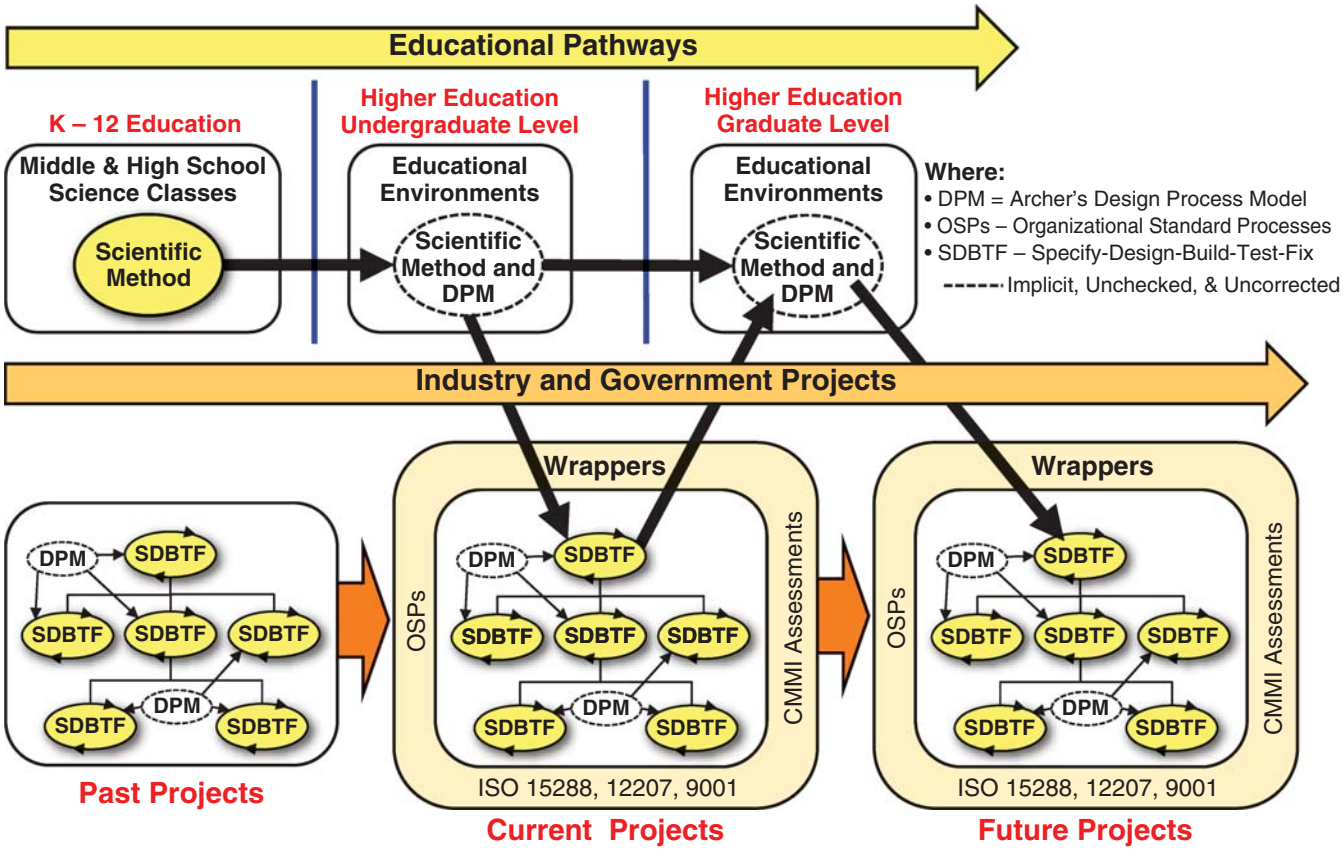


Figure 2.10 Archer's (1965) DPM as the Underlying Problem-Solving and Solution Development Methodology Embedded within the Plug and Chug … SDBTF Engineering Paradigm

migrate their knowledge and tools—Plug and Chug … DBTF—into their new jobs. There, they are assigned to Current Projects and exposed to a modified Plug and Chug … Specify-Design-Build-Test-Fix (SDBTF) Engineering Paradigm, which evolved from Past Projects.

Over time, Engineering personnel may decide to advance their SE knowledge and pursue an MS or PhD as shown in the upper portion of the diagram. Again, as in the undergraduate courses, the Plug and Chug … SDBTF Paradigm is not *explicitly* taught *per se*; however, its underlying Scientific Method and DPM methodologies may be allowed to thrive *unchecked* and *uncorrected* during graduate school. Graduates complete their MS or PhD programs and return to or continue in industry and government applying what they know … The SDBTF Paradigm containing Archer's embedded DPM.

Observe the gray Enterprise *Wrappers* band—OSPs; ISO 15288, 12207, 9001; CMMI Assessments; etc.—around Current Projects. *How does an enterprise comply with these standards and assessments and still have exhibit questionable System Development Performance*? The reality is they are *necessary* but do not correct the SDBTF–DPM Paradigm, which exists as the core problem-solving and solution development methodology of each Engineer and project, nor should they.

As noted earlier, these standards do not tell an Enterprise how to perform Engineering or develop systems, products, or services. They simply identify *process areas* that represent Enterprise levels of *capabilities to perform* that characterize attributes of successful projects. Project-to-project compliance to these standards within the Enterprise is an internal management issue. Failure to comply to these standards may necessitate a reassessment unless *mitigated* through an Enterprise program of continuous process improvement.

Today, we live in a psychological "spin" world in which Enterprises attempt to depict a *deficiency* or *shortcoming* into something positive that defies commonsense and fact-based objective evidence. Interestingly, if you talk with SDBTF Enterprise executives and managers, inquiries about how they implement SE are sometimes met with … "we have a different brand of SE … ours is different (SDBTF)!"

First, from a professional practices perspective, there is only one "brand" of SE.

Second, Enterprises that have a vision for achieving SE excellence usually have a different response and interest … "… we are trying to overcome these challenges in our organization … tell us more …"

Given the two contrasting responses above, you soon discover that those who protest the most—"we have a different brand of SE"—is a clear indication that the SDBTF Paradigm is alive and well within the Enterprise. Typically, they are unaware of the existence of the SDBTF Paradigm within their projects and its ramifications on project performance.

Functional managers often agree that the SDBTF–DPM paradigm may be implicit in their Engineering process. However, they will state that they employ the MIL-STD-499B Draft (1994) SE Process. This brings up an interesting question: *Is the MIL-STD-499B Draft (1994) SE Process any different from Archer's (1965) DPM*? To answer this question, let's make the comparison shown in Table 2.1. Observe the similarities of Archer's (1965) DPM and the MIL-STD-499B Draft (1994) SE Process. To illustrate this point, consider the similarities between Table 2.1 and Figure 2.3:

1. Steps in the MIL-STD-499B Draft (1994) SE Process and Archer's (1965) DPM tend to correlate.
2. MIL-STD-499B Draft (1994):
   - Does not preclude the *premature quantum leaps* from Requirements directly to the Physical Implementation illustrated in Figure 2.3. Although it provides a very high level Requirements → Functions → Synthesis workflow, it does not provide the logical sequencing to correct the Figure 2.3 issue. The SE Process Model addressed later in Figure 14.1 corrects this problem.
   - Implicitly assumes that its Users understand how to perform SE. For those new to SE, a process should not make assumptions about its Users.

This discussion brings us to a key point: *how do we shift the SDBTF Paradigm*? Responsibility resides in an integrated collaboration between industry, government, academia, professional, and standards organizations through a program of awareness, education, and training. What industry and government need is to shift current SE Process Paradigms to a new one. INCOSE, IEEE, ISO, and others, for example, are making great strides toward harmonizing their respective multi-discipline SE standards and handbooks. This requires:

1. Correcting the *ad hoc, endless loop* SDBTF–DPM Paradigm beginning with Engineering education to better prepare Engineering graduates for the workplace.
2. Shift current Enterprise SDBTF–DPM Paradigms through a program of education and training courses for Engineers, PMs, functional managers, and executives.

To better understand how to address the challenge, let's investigate how most Engineers learn SE.

TABLE 2.1 A Comparison of Archer's (1965) DPM and MIL-STD-499B Draft (1994) SE Process

Archer's (1965) DPM	MIL-STD-499B Draft (1994) SE Process
Training	
Programming (general knowledge)	
Data collection	Requirements analysis
Analysis	Requirements analysis
	Functional analysis
	System analysis and control
Synthesis	Synthesis
Development	System analysis and control
Communication	System analysis and control

2.6.2 The SE Learning Model in Many Enterprises & Organizations

Academic Engineering programs are required to meet specific criteria established by their respective for accreditation organizations. For example, the Accreditation Board of Engineering and Technology (ABET) establishes criteria for the United States.

At graduation, Engineering graduates from accredited institutions are expected to have demonstrated knowledge-based performance through coursework that meets or exceeds a Minimum Competency Threshold as illustrated in the upper left portion of the Figure 2.11.

Following graduation, they enter the workforce—lower portion of Figure 2.11. They soon discover that they have to learn how their industry implements not only their own disciplines but also how the Enterprise develops *systems* and *products*. In the 1970's an unsubstantiated paradigm circulated that industry told academia to "teach Engineers to plug and chug equations and we will teach them how to build systems." That requires learning concepts, principles, and practices in System Analysis, Design, and Development shown in the lower right portion of Figure 2.11. If we compare the level of Engineering Education at graduation (upper left) with what is required to perform their jobs (lower right), an Educational Void (upper right) emerges. The Educational Void represents a *deficiency* in Engineering Education at graduation in terms of entering the workforce and being able to be productive.

Over time, their Engineering competency and skills *improve* in some areas beyond the graduation criteria threshold and *lessen* in other areas (lower left) as a function of their work task assignments.

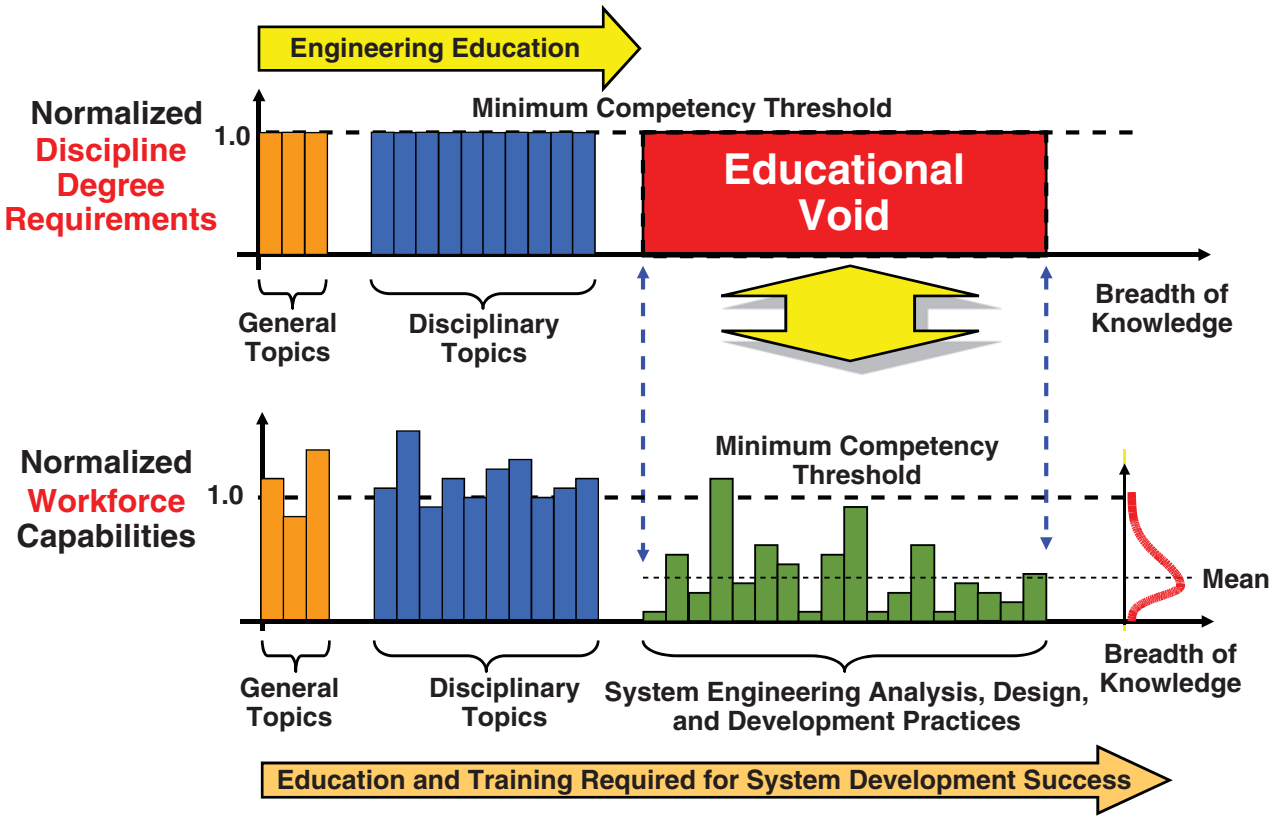


Figure 2.11 SE – The Void in Undergraduate Engineering Education

As a result of the Engineering Education deficiency, learning becomes a process of *osmosis* of listening to meeting discussion semantics that may or may not be accurate, reading standards that are sometimes abstract, participating in hallway and water cooler conversations, and occasional training. Over time, they accumulate various levels of *informal* and *experiential* knowledge as illustrated in the graphic. Exacerbating the problem are managers who learned the same way proliferating the culture by inappropriately labeling Engineers as SEs.

As senior level SEs retire, *how do Enterprises transfer SE knowledge to new SEs in these environments*? Whereas formal Engineering Education is structured to address the spectrum of topics required to achieve an Engineering degree, *informal* SE knowledge is incomplete and exists as information fragments that are acquired experientially as shown in Figure 2.12.

Formal education is based on Instructional System Development (ISD) principles that teach the *what, why, when, where,* and *how* in understanding subject matter such as SE concepts, principles, and practices as illustrated by the left side of Figure 2.12. Formal learning should be based on *best* or *preferred* practices and lessons learned via case studies, problem exercises, and other methods. Let's take a closer look at how informal SE knowledge is acquired in some Enterprises.

2.6.3 SE Experiential Learning by "Go Do" Tasking

Observe the right side of Figure 2.12. The formal ISD *whats* are replaced by "Go Dos" tasking. That is … Go Do a specification … Go Do an architecture … Go Do a design … Go Do a test procedure. What is absent as noted by the dashed lines are the *whys, where tos, when tos,* and *how tos*. Is there any wonder why Engineering task performance on projects is *ad hoc, chaotic, inconsistent, inefficient,* and *ineffective*?

This also brings up an interesting question concerning Enterprises that submit proposals—Mini-Case Study 2.1—and claim to have tailored their (SDBTF–DPM) processes. *If personnel lack an understanding of the what, why, where, when, and how to, can you effectively tailor a key OSP holistically without having requisite knowledge of the ramifications and risks of those actions*?

2.6.4 SE & Development versus System Acquisition & Management Courses

In response to the right side of Figure 2.12, Enterprises and functional managers will boldly proclaim that they train XX

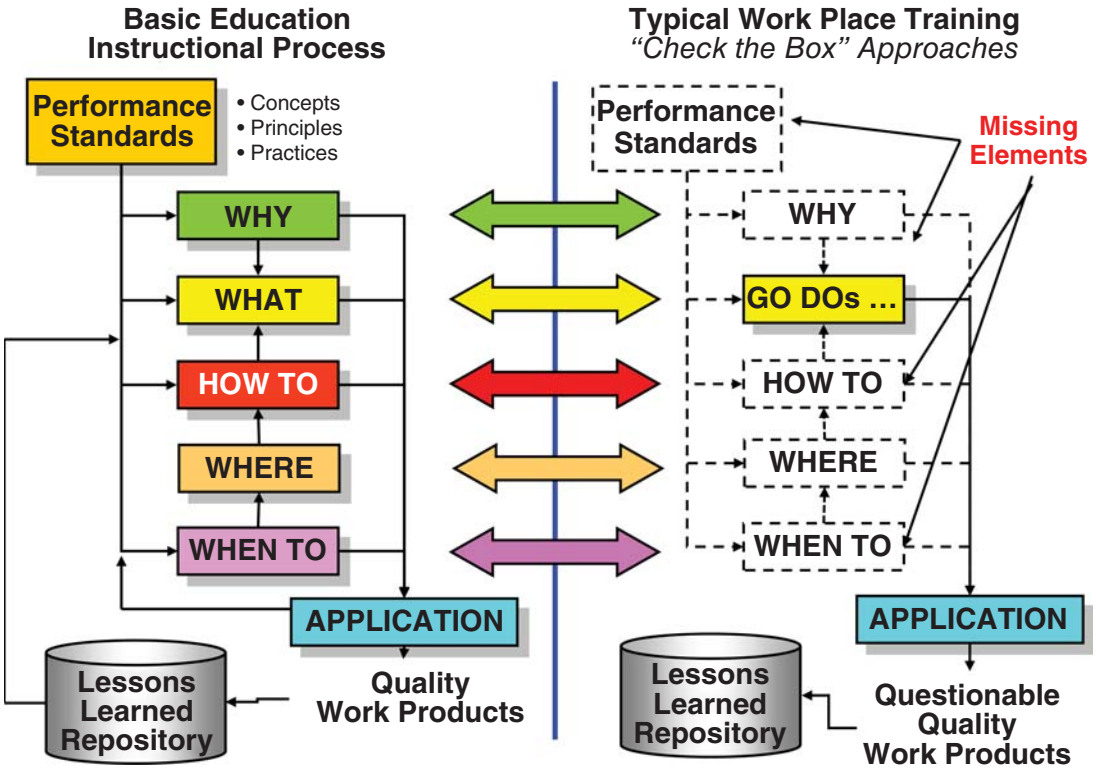


Figure 2.12 Formal Education versus Enterprise and Organizational Experiential Learning

(Qty.) Engineers every year in SE—via courses and OJT (Go Dos). *If that is the case, then WHY do we still see objective evidence of the effects of the SDBTF Paradigm in project performance that is validated by Engineers*?

A Word of Caution 2.2

At this point in our discussion, it should be apparent that titling a course as "System Engineering" does not mean that it provides the requisite instruction in SE concepts, principles, and practices required to perform Systems Engineering and Development (SE&D).

Author's Note 2.6

Less than 3% of SEs in Enterprises are Actual SEs

Based on the author's experience and anecdotal evidence—yours may be different—less than 3% of the engineers in most SE-labeled Enterprises understand and can apply SE. For a typical 30-person SE Enterprise in which everyone is arbitrarily labeled as an SE, only 1 person ± typically possesses the requisite SE concepts, principles, and practices addressed in this text. Most of those labeled as SEs are discipline Engineers performing system analysis, modeling, Specialty Engineering (RMA), Human Factors (HF), logistics, safety, security, and other tasks.

Wasson (2012, p. 20) observes that the thrust of most courses labeled as "SE" focuses on (1) SA&M—key semantics, the philosophy of SE, and what you should do to oversee others claiming to be performing SE—or (2) equations. There is a significant difference between overseeing and managing the work of other Engineers or contractors versus actually understanding how to perform SE&D. But here's the irony. We now have SEs from the SA&M courses overseeing the work of others who also … completed SA&M courses. SA&M and equation-based courses are fine when properly sequenced and matched to specific personnel task descriptions; however, both require a strong foundation in SE&D as a pre-requisite.

2.6.5 Understanding the Ramifications of the Educational Void

To better understand the Educational Void in Figure 2.11, let's take this a step further. Referring to using Figure 2.13, most engineers typically have a career that lasts on average 40 years ±. During the first 5–10 years, EEs, MEs, SwEs, and others apply their SDBTF–DPM Paradigm skills to perform their assigned project tasks based on SDBTF methods they migrated from their Engineering degree programs.

Beginning with the fifth year, their employers expect them to assume project roles with increasing responsibility.

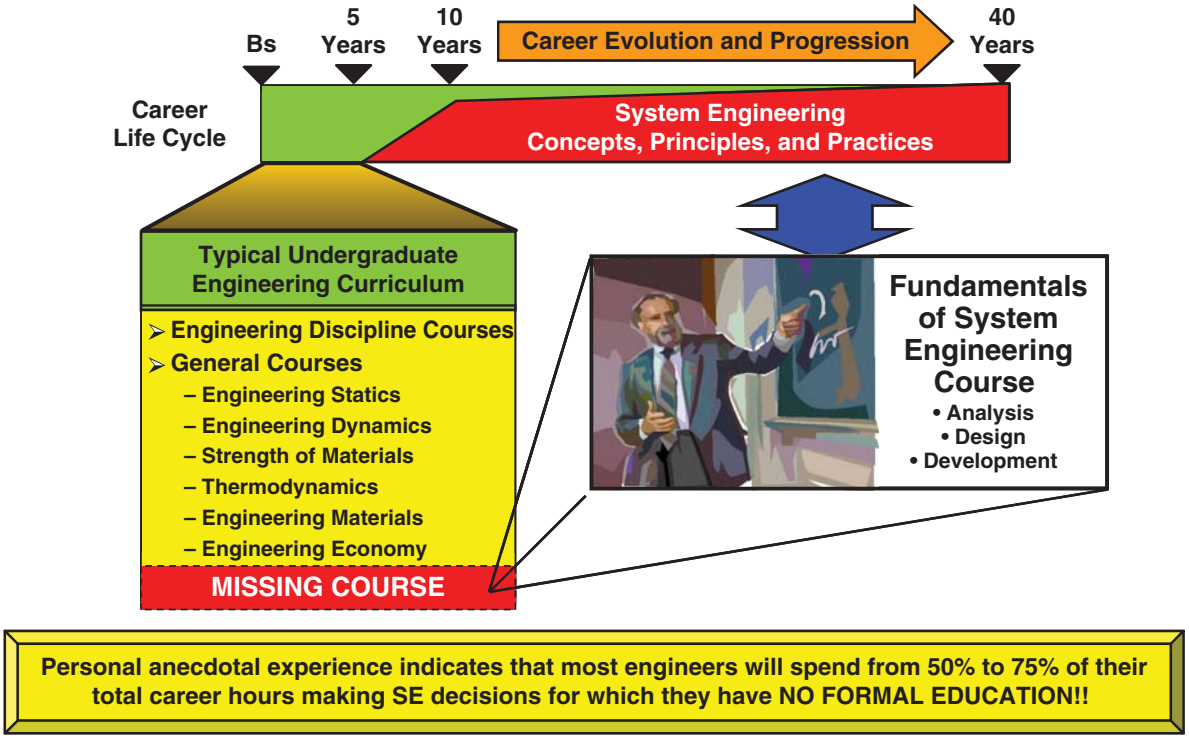


Figure 2.13 System Engineering—The Missing Course in Engineering Education

This comes with more complex project assignments including leadership roles of team that include new Engineering graduates. As a result, they spend *less time* working *discipline-specific* tasks and more time leading a *multi-discipline* team that requires *collaboration* and *interactions* with other *multi-discipline* teams.

From a career hours perspective based on the author's experience, Engineers spend 4 years on average obtaining an Engineering degree that has a "hands-on" application shelf life of 5–10 years in industry and government. Wasson (2012) observes that most engineers spend 50–75% of their *total career hours* making SE&D decisions for which they have *NO formal SE coursework*. This Educational Void (Figure 2.11) is a contributory performance effector that impacts project performance.

Today, a new SE paradigm, Lean Systems Engineering, is being embraced by industry and government. To better understand this concept and its ramifications of its application, let's briefly explore this topic.

2.6.6 Lean Systems Engineering (Lean SE)

With the introduction of quality initiatives such as Zero Defects, Total Quality Management (TQM), Design for Six Sigma (DFSS), etc. in the 1980s by the automotive industry, for example, enterprises have strived to become more *efficient* and *effective* to improve customer satisfaction and improve profitability. Other industries adopted the approaches with the belief that systems, products, and services could be designed and developed in the same manner as automobiles.

Since the 1990s, the automobile industry has made great strides in *driving out* waste and *eliminating* non-value-added processes. The same principles can be applied to the *innovation* and *creation* of systems, products, and services via multi-discipline System Engineering. What evolved is a concept referred to as Lean SE. *What is Lean SE*?

2.6.6.1 What is Lean SE?

Lean SE roots originate from the Lean Aerospace Initiative (LAI) Consortium at the Massachusetts Institute of Technology (MIT) in 2004 (Rebentisch, et al, 2004). In 2006, initiative was transferred to the INCOSE LSE Working Group (INCOSE, 2010, p. 4).

The INCOSE LSE Working Group defines Lean SE as follows:

> **Lean SE** is "The application of lean wisdom, principles, practices and tool to systems engineering in order to enhance the delivery of value to the system's stakeholders."

Oppenheim (2011, p. 3) notes:

> "The *Lean* in *Lean* SE should be regarded as the process of amending the well-established, traditional SE process with the wisdom of *Lean Thinking*, rather than replacing SE with a new body of knowledge."

What is Lean Thinking? Murman et al. (2002) define *Lean Thinking* as:

> **Lean Thinking (Lean)** "… the dynamic, knowledge driven and customer-focused process through which all people in a defined enterprise work continuously to eliminate waste and to create value" (Murman et al. 2002, p. 1).

Oppenheim (2011, p. 3) makes an additional observation:

> "Put emphatically: … Lean Systems Engineering does not mean 'less systems engineering,' but rather more SE, with better preparations of the enterprise processes, people, and tools; better program planning and frontloading; better workflow management; and better program management and leadership with higher levels of responsibility, authority, and accountability."

In 2009, Oppenheim (2009) published a paper on Lean Enablers for Systems Engineering (LEfSE) based on work accomplished conducted by the INCOSE LSE Working Group (Oppenheim, 2009). The resulting product was described as:

> "… a collection of 194 practices and recommendations formulated as "dos" and "don'ts" of SE, and containing collective wisdom on how to prepare for, plan, execute, and practice SE and related enterprise management using Lean Thinking." (Oppenheim, 2009, p. 1)

As a result of ongoing work performed by the INCOSE Lean WG, MIT, and the PM Institute (PMI), a joint MIT-INCOSE-PMI Community of Practice (CoP) agreed to collaborate and develop a guide for Lean Enablers.

2.6.6.2 Lean Enablers Guide for Managing Engineering Programs

Beginning in 2011, the Joint MIT, PMI, INCOSE—MIT-PMI-INCOSE—CoP initiated a one-year project to develop a comprehensive guide of lean enablers.

In 2012, the group released *The Guide to Lean Enablers for Managing Engineering Programs*. The guide was "extensively validated through community and practitioner feedback, multiple workshops at INCOSE and PMI conferences, and LAI-hosted web-based meetings, and surveys of the extended professional community" MIT-PMI-INCOSE (2012).

The MIT-PMI-INCOSE (2012, p. vi) Lean Enabler Guide addresses "major challenge themes in Engineering programs that Lean Enablers help to address." These include:

1. "Firefighting – Reactive program execution.
2. Unstable, unclear, and incomplete requirements.
3. Insufficient alignment and coordination of the extended enterprise.
4. Processes are locally optimized and not integrated for the entire enterprise.
5. Unclear roles, responsibilities, and accountability.
6. Mismanagement of program culture, team competency, and knowledge.
7. Insufficient program planning.
8. Improper metrics, metric systems, and KPIs (Key Performance Indicators).
9. Lack of proactive program risk management.
10. Poor program acquisition and contracting practices."

You may ask: *What is the relevance and connection between the lean enablers, SE, and the SDBTF–DPM Engineering Paradigm*? This brings us to our next topic.

2.6.6.3 Applying Lean Principles to an Enterprise Engineering Process

Enterprises either perform SE correctly; employ the *ad hoc*, *endless loop* SDBTF–DPM Paradigm; or have some hybrid variation of the two. Regarding application of Lean Principles to an Engineering Process, the question becomes: *how can we apply Lean Principles to eliminate waste and improve value for our customer*? If you are performing SE as addressed in this text, then obviously, you are embarking on a program of continuous process improvement and postured for applying Lean Principles. If this is not the case, a Word of Caution 2.3.

A Word of Caution 2.3

SDBTF–DPM Paradigm Enterprises

If your Enterprise or project is currently employing the SDBTF–DPM Engineering Paradigm, you have a major challenge. Application of Lean SE Principles to an *ad hoc, endless loop*, SDBTF–DPM Paradigm results in … another version of a *ad hoc, endless loop* SDBTF–DPM Paradigm unless a shift is made to a true SE Paradigm such as Figure 14.1.

Lean SE requires competent, insightful, and timely planning, training of personnel, updates to Enterprise and project command media, and other actions fully supported by executive commitment and resources. Changing Enterprise cultures is a long-term action; every Enterprise is different. Employ the services of a competent, qualified professional of your own choosing to support the paradigm shift transition.

Given this discussion, *how do we solve the* SDBTF–DPM Engineering Paradigm *problem*? The solution begins to a degree in K – 12 education and specifically in Engineering degree programs. This brings us to our next topic: *Engineering Educational Challenges and Opportunities*.

2.7 ENGINEERING EDUCATION CHALLENGES AND OPPORTUNITIES

SE, unlike most other Engineering disciplines, requires in-depth experience over many years across a multitude of small, medium, and large, complex System Development projects. Therein lies a problem for many Engineering educators.

Lattuca et al. (2006, pp. 6 and 12) observed that:

> "In the 1980s, employers expressed dissatisfaction with engineering graduates' professional skills."
>
> "New graduates were technically well prepared but lacked the professional skills for success in a competitive, innovative, global marketplace."
>
> "By the mid-1990s, ABET had implemented a new accreditation philosophy based on assessments of student learning and continuous improvement principles."

One educator commenting about SE education challenges once noted that competent SE course instructors need 25+ years of in-depth industrial system and product development experience. The problem is that they retire from industry, move into academia, and leave after a short time due to a lack of (1) tenure and (2) control over the course(s) they teach. In contrast, new PhDs become instructors, achieve tenure, but lack industrial system and product development experience.

To illustrate this point, the National Academy of Engineering's "Educating the Engineer of 2020" report (NAP, 2005, p. 21) states "… The great majority of engineering faculty, for example, has no industry experience. Industry representatives point to this disconnect as the reason that engineering students are not adequately prepared, in their view, to enter today's workforce."

Vest (NAP, 2005) observes that "Academics led the way in engineering science, but I don't think we have led the way in what we now call 'systems engineering.' In fact, as we observe developments in industry, government, and society, we are asking ourselves what in the world we should teach our students … Indisputably, engineers of today and tomorrow must conceive and direct projects of enormous complexity that require a new, highly integrative view of engineering systems" (NAP, 2005, p. 165). These initiatives are imperative to compete in a highly competitive global environment.

Erwin (1998) observes that projects in engineering schools tend to focus on the building aspects of systems. Then, when the projects are submitted for grading, most of the assessment is based on completion of the artifact, with design having lesser importance. He notes that this approach is often *rationalized* on the basis of allowing the students to be "creative." As a result, the student receives little or no guidance or direction concerning the design process Erwin (1998, p. 6).

Engineers by virtue of Engineering Education naturally focus on *component-centric* design solutions. Caldwell (2007) in addressing engineering curriculum reform describes the traditional engineering course presentation order as bottom-up: *Components–Interactions–Systems*. He observes that engineering courses focus on the analysis of engineering components, not integrated systems (Caldwell, 2007, pp. 92–93).

To address this problem, Engineering programs have used senior level Capstone Projects to introduce Engineering *discipline* students to *multi-discipline* design team environments that promote interdisciplinary collaboration and decision-making. Papers on some Capstone Projects often characterize the SE exposure as writing specification requirements, creating a design, identifying and assembling components, integrating and testing the components, and performing V&V. As noted earlier, these are SE activities but *not SE* problem-solving and solution development. In fact, this sequential flow is *incorrect and deficient*.

Engineering programs and instructors often lament about having limited classroom or laboratory time to cover required content. So, attempts are made to simultaneously infuse SE concepts in real time into the mainstream of a capstone project. While these attempts are well-intentioned, Schmidt et al. (2011), Nemes et al. (2011), and Corns and Dagli (2011) provide research and recommendations that illustrate why SE concepts should be introduced *prior to* Capstone Project courses (Wasson, 2012, p. 9).

Although any level of SE awareness is better than none at all, well-intentioned portrayal of a discipline based on "knowledge fragments"—namely, activities—is *negative education* from an Instructional System Development (ISD) perspective. Such is the case with SE. In preparation for entering the workforce, having an accurate and complete understanding of … what SE *is* and *is not* … is critically important for an Engineer being ready to competently apply SE methods and understand the of their assigned work within the context of a deliverable system, product, or service.

To address these educational concerns, ABET's Board of Directors in 1996 approved Engineering Criteria 2000 (EC2000), our next topic (Latucca et al., 2006, p. 6).

2.7.1 ABET Engineering Change 2000 (EC2000)

The EC2000 established a new Criterion 3 for Student Outcomes ABET (2012, p. 3). Criterion 3 states, "Student outcomes are outcomes (a) through (k) plus any additional outcomes that may be articulated by the program.

a. An ability to apply knowledge of mathematics, science, and engineering.
b. An ability to design and conduct experiments, as well as to analyze and interpret data.

c. **An ability to design a system**, component, or process to meet desired needs within realistic constraints such as economic, environmental, social, political, ethical, health and safety, manufacturability, and sustainability.
d. An ability to function on multi-disciplinary teams.
e. An ability to identify, formulate, and solve engineering problems.
f. An understanding of professional and ethical responsibility.
g. An ability to communicate effectively.
h. The broad education necessary to understand the impact of engineering solutions in a global, economic, environmental, and societal context.
i. A recognition of the need for, and an ability to engage in life-long learning.
j. A knowledge of contemporary issues.
k. An ability to use the techniques, skills, and modern engineering tools necessary for engineering practice."

Observe the word *system* in Item (c) "an ability to design a **system**, component, or process," which was a significant, positive change. However, reexamine the list. Rhetorically: *Is there anything in this list that acknowledges the existence and the need for a student outcome that corrects the Plug and Chug … SDBTF–DPM Paradigm that continues to plague industry and government System Development performance*? The key point is it's time to advance undergraduate Engineering Education to a higher level to correct the Plug and Chug … SDBTF–DPM Paradigm that migrates *unchecked* and *uncorrected* through Engineering education.

We need to do more than simply produce Engineers that still use the Plug and Chug … SDBTF–DPM Paradigm to design systems. Today, Engineering Education is challenged to shift to more "soft" courses such as communications, philosophy, and other courses in an already overloaded curriculum. This knowledge is certainly vital to the twenty-first-century Engineering. The challenge question is:

- *What is the relative value of an Engineer who can communicate well and understands philosophy but has no clue as to what a system really is, how it is structured architecturally, or the context of their Engineering task within the overall System or Product as well as its contribution to overall system performance*?

There has to be a balance that produces Student Outcomes—Engineering graduates—that can fulfill industry and government System Development needs on Day #1 of their employment and produce outcomes as indicated by the definition of Engineering "… for the benefit of mankind" (Prados, 2007, p. 108).

2.8 CHAPTER SUMMARY

In summary, we addressed the evolving state of SE and presented challenges and opportunities for advancing the state of the practice. The current state of SE practice has two key challenges:

1. Engineers graduate with degrees in a specific Engineering discipline with an Educational Void (Figure 2.11) characterized by:
   a. Little or no instruction in multi-discipline problem-solving and solution development required in today's workplace.
   b. A lack of SE courses required to perform multi-discipline SE&D.
2. Based on the author's experience:
   a. Most engineers spend from 50% to 75% of their total career hours making SE decisions for which they have NO formal education or coursework (Figure 2.13).
   b. Despite having SE job titles, less than 3% of Enterprise SEs competently exhibit knowledge of the SE concepts, principles, and practices addressed in this text. Collective SE competency resides at the Enterprise level. However, when Enterprise SE competency is dispersed–matrixed–across multiple projects, the level of SE competency on a project is dependent on the informal, experiential education, knowledge, and experience of those with "SE job titles."
   c. SE knowledge for most Engineers is acquired from informal and experiential learning based on OJT via "Go Do" tasking (Figure 2.12).
   d. SE is one of the most abused enterprise job labor categories in which managers arbitrarily label Engineers as SEs irrespective of whether they have met educational requirements and demonstrated knowledge and competency in applying SE concepts, principles, and practices.
3. Some courses labeled as SE should be labeled SA&M—overseeing and managing the work of others performing SE. Courses are needed in SE&D to provide instruction in how to actually develop systems, products, or services.
4. Industry, government, academia, professional, and standards organizations developed and evolved SE standards to:
   a. Correct SE and project performance issues.
   b. Promote consistent, repeatable, and predictable project performance.
5. A paradigm is an ingrained, groupthink mind-set or model that filters or rejects considerations to adopt or

employ new innovations and ideas that may impact the status quo.

6. Project and Engineering performance issues are often traceable to a little known Plug and Chug … DBTF Engineering Paradigm based on the Scientific Method of inquiry or investigation that:
   a. Originates in middle and high school education.
   b. Migrates through Engineering Education unchecked and uncorrected and subsequently into industry and government.
   c. Mutates into a Plug and Chug … SDBTF Engineering Paradigm with its embedded endless loop DPM (Figure 2.8) prevalent in industry and government.
7. The industrial Plug and Chug … SDBTF Engineering Paradigm is:
   a. Often confused with the System Development Strategy, which does have a generalized Specify, Design, Build, and Test workflow (Figure 12.3) over time.
   b. Perceived to be an SE problem-solving and solution development methodology. The reality is the SDBTF Paradigm is an ad hoc, endless loop set of activities that lack completion convergence and frustrate PMs who are challenged to meet cost, schedule, and technical constraints.
8. Enterprises, projects, and Engineers often erroneously perceive SE as:
   a. Writing specifications, creating a design, building or acquiring components, integrating and testing the system, and performing System V&V prior to delivery.
   b. Documentation-centric – i.e, produce documentation. The reality is SE, which is methodology-centric and outcome based, focuses on convergent, technical decision-making to select an optimal solution from a set of viable alternatives. Decision artifacts such as inputs, constraints, and recommendations, and are captured via documentation.
9. Be Top-Down Engineering. The reality is multi-discipline SE is performed:
   - Vertically—Top-Down and Bottom-Up.
   - Horizontally—Left to Right and Right to Left.
10. Lean Systems Engineering:
    a. Focuses on driving out waste in SE and eliminates non-value-added processes.
    b. Applied to an Enterprise Plug and Chug … SDBTF Engineering process often results in another Plug and Chug … SDBTF Engineering process that fails to correct the quantum leap approaches from Requirements to Physical Implementation shown in Figure 2.3.

Principle 2.2

Decision Artifacts Principle

If a key *decision* or *event* and its contributory inputs, constraints, and their sources are not documented, the decision or event never occurred.

If you shift the SDBTF–DPM Engineering Paradigm to a new SE paradigm, will it work?

First, installation of a new SE paradigm requires visionary leadership, long-term commitment, and incremental rollouts via updates to OSPs, education and training courses, and new, fresh managerial perspectives based on seasoned SE experience.

Secondly, in an ongoing study conducted by Elm and Goldenson (2012), data validate SE performance. For example, Figure 2.14 illustrates the effectiveness of SE best practices on three types of organizations categorized as having Lower, Middle, and Higher SE Capabilities (SEC) applied to performing two types of projects: (1) those with a Low Project Challenge (Low PC) and (2) those with a High Project Challenge (High PC) (Elm and Goldenson, 2012, Figure 3, p. xiv):

- **Low Project Challenge (PC) Projects** (Figure 2.14, Panel A) Enterprises with higher SE Capabilities (SEC) delivered the Highest Level of Performance 52% of the time versus 23% for Middle SEC Enterprises and 23% for Lower SEC Enterprises.
- **High Project Challenge (PC) Projects** (Figure 2.14, Panel B) Enterprises with Higher SEC delivered the Highest Level of Performance 62% of the time versus 26% for Middle SEC Enterprises and 8% for Lower SEC Enterprises.

Bottom line: Enterprises with Higher SE Capabilities have a much better probability of consistently delivering higher levels of performance regardless of the Project Challenge than Enterprises with Middle and Lower SECs.

Author's Note 2.7

Please note that Elm and Goldstein (2012) analysis of respondent survey data makes no distinction concerning the type of SE Process used—either the SDBTF–DPM Paradigm based on MIL-STD-499B Draft (1994) or the SE Process addressed in Chapter 14 that overcomes performance issues.

Do true, non-SDBTF, SE capable Enterprises have better performance results? Based on research and industry surveys by Honour (2013) and Elm and Goldenson (2012):

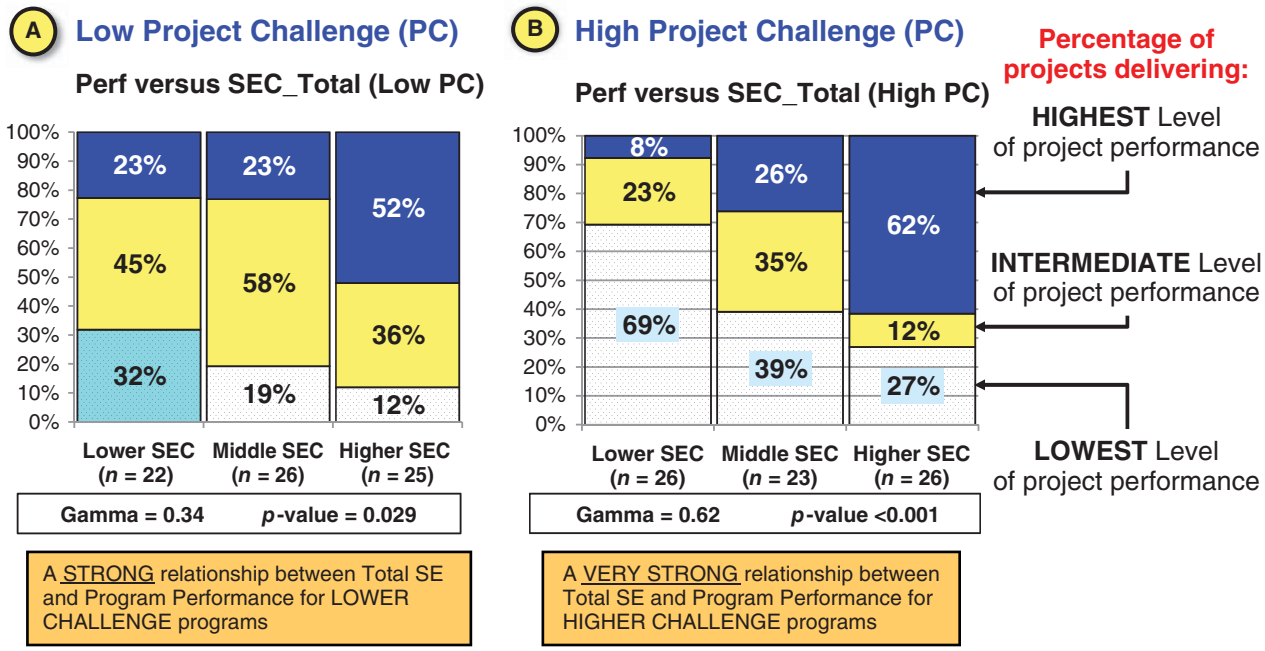


Figure 2.14 SE Best Practices Effectiveness—Project Performance versus Total SE Capability Controlled by Project Challenge (Source: Elm and Goldenson, 2012, Figure 3, p. xiv. Used with permission.)

> "Projects that properly apply systems engineering best practices perform better than projects that do not"
> —(Elm and Goldenson, 2012, p. xiv).

> "There is a quantifiable relationship between systems engineering effort levels and program success"
> —(Honour, 2013, p. 177).

To overcome the proliferation of current SE paradigms, industry, government, and academia need to seize the opportunity to correct the root problem and its contributory causes, not just create wrappers around the perimeter of the problem (Figure 2.8) expecting different results.

Up until the 1980s, many Enterprises believed that QA and Verification & Validation (V&V) were activities performed on a system or product *after* it was finished. Rejections rates were high, rework cost vast amounts of money, and scrap heaps were large. Then, challenged by global competition, organizational survival, and the need for higher profitability, executives and others became enlightened *to learn* that system, product, or service quality and performance is built in from Day #1, not at the end. Enterprise and project performance and success are at a new crossroad: continue the SDBTF Engineering Paradigm or shift to a new SE Paradigm discussed in this text.

One of the key aspects of Enterprise performance and tenets of various quality movements, DFSS, and so on is Supply Chain Management (SCM) illustrated later in Figure 4.1. Delivery of quality products and services is dependent quality principles instilled in each step of the supply chain. The SE implementation and education *supply chain* is fragmented and stovepiped! Industry, government, and academia need to *recognize and acknowledge the existence of the Plug and Chug … SDBTF Paradigm with its embedded DPM*. Then, work collaboratively to shift the SDBTF paradigm.

The challenge questions are:

1. Organizationally, do we and our Enterprise:
   a. Continue to allow the Plug and Chug … SDBTF–DPM Paradigm that everyone acknowledges is ad hoc, chaotic, inefficient, ineffective, and unscalable to moderate to large, complex projects?
   b. Shift it to a true SE paradigm (Chapters 3–34)?
2. Academically, do we:
   a. Continue with current curricula that allow the Scientific Method and the SDBTF–DPM Paradigm to migrate through Engineering Education unchecked and uncorrected?
   b. Shift to a new Engineering Education Paradigm that eliminates the Educational Void (Figure 2.11) that:
      - Prepares undergraduates and graduates to enter industry and government where real work is performed?
      - Is based on a balance between problem-solving and solution development and Engineering, not SA&M or equation-based courses, which have their place as follow-on courses?

This does not mean create to any type of course - SA&M or equations-based, put an SE label on it, and assign it to an instructor with no industry experience to teach it. The ABET in the United States, for example, via its EC2000 expressed recognition of the need for Specific Criteria beyond the General Criteria for Systems Engineering (ABET, 2011, p.3). Despite several years of having a placeholder in its annual *Accreditation Criteria*, the PROGRAM CRITERIA FOR SYSTEMS AND SIMILARLY NAMED ENGINEERING PROGRAMS section continues to state, "There are no (Systems Engineering) program-specific criteria beyond the General Criteria" (ABET, 2014, p. 20).

Given this discussion of the evolving state of SE practice and its challenges and opportunities, you should have a better appreciation for the SE concepts, principles, and practices addressed in Chapters 3–34.

2.9 CHAPTER EXERCISES

2.9.1 Level 1: Chapter Knowledge Exercises

1. What is a paradigm?
2. What is meant by a paradigm shift?
3. What is verification?
4. What is validation?
5. What is the Plug and Chug … SDBTF paradigm? What are its origins? Why is it important to get the SDBTF Paradigm corrected in Engineering education?
6. What does "Paint by Number" Engineering refer to? What are its pros and cons?
7. What is Lean Thinking?
8. What is Lean SE?
9. What are Lean SE Principles intended to accomplish?
10. Why does application of Lean SE principles to an SDBTF-DPM Engineering Process result in another SDBTF-DPM Engineering Process

2.9.2 Level 2: Chapter Knowledge Exercises

Refer to www.wiley.com/go/systemengineeringanalysis2e

2.10 REFERENCES

ABET (2012), *Criteria for Accrediting Engineering Programs – Effective for Reviews During the 2013 – 2014 Accreditation Cycle*, Baltimore, MD: Accreditation Board of Engineering and Technology (ABET) Engineering Accreditation Commission.

AFSCM 375-5 (1966), *Air Force Systems Command Manual No. 375-5, System Engineering Management Procedures, Andrews AFB* – Washington, DC: Headquarters Air Force Systems Command (AFSC).

ANSI/EIA 632 (1998), *Processes for Engineering a System*, Arlington, VA: Electronic Industries Alliance (EIA).

Archer, L. Bruce (1965), *"Systematic Methods for Designers," Design*, London: Council on Industrial Design.

Caldwell, Barrett S. (2007), "*Teaching Systems Engineering by Examining Educational Systems*," *Proceedings of the Spring 2007 American Society for Engineering Education (ASEE) Illinois-Indiana Section Conference*, March 30–31, 2007, Indianapolis, IN: Indiana University-Purdue University Indianapolis. Retrieved on 10/6/14 from http://ilin.asee.org/Conference2007program/Papers/Conference%20Papers/Session%201B/Caldwell.pdf.

Churchman, C. West (1967), "Guest Editorial," *Management Science*, Vol. 14, No. 4, 1967, pp. B-141–142. Retrieved on 10/6/14 from http://pubsonline.informs.org/doi/pdf/10.1287/mnsc.14.4.B141.

CMMI (2014a), *Solutions webpage*, Pittsburgh, PA: Carnegie Mellon University Capability Maturity Model Institute (CMMI). Retrieved on 1/12/13 from http://cmmiinstitute.com/cmmi-solutions/.

CMMI (2014b), *FAQs: CMMI Product Suite – What Is CMMI? webpage*, Pittsburgh, PA: Carnegie Mellon University Capability Maturity Model Institute (CMMI). Retrieved on 1/12/13 from http://cmmiinstitute.com/cmmi-getting-started/frequently-asked-questions/faqs-cmmi-product-suite/.

Corns, Steven, and Dagli, Cihan H. (2011), "*SE Capstone: Integrating Systems Engineering Fundamentals to Engineering Capstone Projects: Experiential and Active*," *American Society for Engineering Education (ASEE) 2011 Annual Conference Proceedings*, June 26–29, 2011, Vancouver, Canada. Retrieved on 1/9/13 from http://www.asee.org/public/conferences/1/papers/1211/view.

Drucker, P. F. (1959), *The Landmarks of Tomorrow*, New York: Harper and Row - Harper Collins.

EIA/IS 632–1994 (1994), Interim Standard: *Processes for Engineering a System*, Arlington, VA: Electronic Industries Alliance (EIA). Retrieved on 1/9/14 from http://www.acqnotes.com/Attachments/EIA-632%20%93Processes%20for%20Engineering%20a%20System%94%207%20Jan%2099.pdf.

EIA 731.1 (2002), *EIA Standard: System Engineering Capability Model (SECM)*, Arlington, VA: Electronic Industries Alliance (EIA).

Elm, Joseph P. and Goldenson, Dennis R. (2012), "*The Business Case for Systems Engineering Study: Results of the Systems Engineering Effectiveness Survey*", Special Report: CMU/SEI-2012-SR-009, Pittsburgh, PA: Carnegie Mellon University (CMU) CERT Program. Retrieved on 1/4/13 from http://www.sei.cmu.edu/library/abstracts/reports/12sr009.cfm.

Erwin, Ben (1998), "K-12 Education and Systems Engineering: A New Perspective," Session 1280, Proceedings of the American Society of Engineering Education (ASEE) National Conference, Seattle, WA.

FM-770-78 (1979) *Field Manual: Systems Engineering*, Washington, DC: U.S. Army.

Goldman, Julian M., MD, (2013), *Presentation: Medical Device Interoperability: "a wicked problem,"* Boston, MA: Mass Gen Hospital/Harvard Medical School. Retrieved on 10/6/14 from http://mdcx73-working.wikispaces.com/file/view/172_Wicked Problem.pdf/414344978/172_WickedProblem.pdf.

Honour, Eric C. (2004), *Understanding the Value of Systems Engineering*, Cantonment, FL: Honourcode, Inc.

Honour, Eric C. (2013), *Full Dissertation:* Systems engineering return on investment, Thesis – Doctor of Philosophy, Defence and Systems Institute, School of Electrical and Information Engineering, Adelaide, South Australia: University of South Australia. Retrieved on 1/9/13 from http://honourcode.com /seroi/documents/SE-ROI%20Thesis%20summary-distrib.pdf.

IEEE Std 1220-1994 (1995) *IEEE Trial-Use Standard for the Application and Management of the Systems Engineering Process,* New York: Institute of Electrical and Electronic Engineers (IEEE).

INCOSE (1996), *Systems Engineering Capability Assessment Model (SECAM), Version 1.5a*, San Diego, CA: International Council on Systems Engineering (INCOSE). Retrieved on 1/4/13 from http://www.incose.org/ProductsPubs/pdf/SECAM-SysEngCapabilityAssessModel_1996-06.pdf.

INCOSE (2014), *Certification Change History Log*, San Diego, CA: International Council on Systems Engineering (INCOSE). Retrieved on 1/5/13 from http://www.incose.org/education careers/doc/CertificationChangeHistoryLog.pdf.

INCOSE (2010), *Lean Enabler for Systems Engineering, Version 1.0*, INCOSE Webinar March 17, 2010, San Diego, CA: International Council on Systems Engineering (INCOSE). Retrieved on 1/11/14 from http://www.leanssc.org/files/201004 /powerpoint/4.22%203.45pm%20Oppenheim%20LeanEnablers ForSystemsEngineering.pdf.

ISO/IEC 15288:2008 (2008), *System Engineering – System Life Cycle Processes,* Geneva: International Organization for Standardization (ISO).

ISO/IEC 12207:2008 (2008), *Systems and software engineering—Software life cycle processes*, Geneva: International Organization for Standardization (ISO).

Johnson, Stephen B. (2002), *The Secret of Apollo: Systems Management in the American and European Space Programs*, Baltimore, MD: The Johns Hopkins University Press.

Lattuca, Lisa R.; Terenzini, Patrick T.; Volkwein, J. Fredericks; and Peterson, George D. (2006), "*The Changing Face of Engineering Education*," *The Bridge*, Vol. 36, No. 2, Washington, DC: National Academy of Engineering (NAE).

MIL-STD-499 (1969), *Military Standard: Systems Engineering Management*, Washington, DC: Department of Defense (DoD).

MIL-STD-499B Draft (1994), *Military Standard: Systems Engineering*, Washington, DC: Department of Defense (DoD).

MIT-PMI-INCOSE (2012), Oehmen, Josef, (Ed.), *The Guide to Lean Enablers for Managing Engineering Programs*, Joint MIT-PMI-INCOSE Community of Practice on Lean in Program Management, Version 1.0, Cambridge, MA: Massachusetts Institute of Technology (MIT), Retrieved on 1/3/13 from http://hdl.handle.net/1721.1/70495.

NAP (2005), *Educating Engineers of 2020 and Beyond: Adapting Engineering Education to the New Century, Appendix B*, Washington, DC: National Academies Press (NAP). Retrieved on 1/11/14 from http://www.nap.edu/download.php?record_id =11338.

NDIA (2010), *Top Systems Engineering Issues in US Defense Industries, Final-v11-9/21/2010*, Arlington, VA: National Defense Industries Association (NDIA). Retrieved on 1/6/14 from http://www.ndia.org/Divisions/Divisions/SystemsEngineering /Documents/Studies/Top%20SE%20Issues%202010%20Report %20v11%20FINAL.pdf.

Nemes, J., Hochstedt, K., Brannon, M., Kisenwether, E., and Bilen, S. (2011), "SE Capstone – Introduction of Systems Engineering into an Undergraduate Multidisciplinary Capstone Course," AC 2011-1077, American Society for Engineering Education (ASEE) 2011 Annual Conference Proceedings, June 26 – 29, 2011, Vancouver, Canada. Retrieved on 1/9/13 from http://www.asee.org/public/conferences/1/papers/1077/view

OMG (2013), *OMG Systems Modeling Language webpage*, Retrieved on 1/8/13 from http://www.omgsysml.org.

Oppenheim, Bodhan W. (2009), "*Lean Enablers for Systems Engineering*," Loyola Marymount University, July/August 2009, Hill AFB, UT: *CrossTalk*: The Journal of Defense Software Engineering. Retrieved on 1/11/14 from http:// www.crosstalkonline.org/storage/issue-archives/2009/200907/ 200907-Oppenheim.pdf.

Oppenheim, Bohdan W. (2011), *Lean for Systems Engineering with Lean Enablers for Systems Engineering*, New York: John Wiley & Sons, Inc.

Paulk, Mark C. (2004), "*Surviving the Quagmire of Process Models, Integrated Models, and Standards*". Pittsburgh, PA: Carnegie Mellon University (CMU) Institute for Software Research. Retrieved on 1/5/13 from http://repository.cmu.edu/isr/15.

Rebentisch, Eric; Rhodes, Donna H.; and Murman, Earll (2004), *Lean Systems Engineering: Research initiatives in Support of a New Paradigm, Massachusetts Institute of Technology (MIT)*, Los Angeles, CA: Conference on Systems Engineering Research (CSER). Retrieved on 1/4/13 from http://web.mit.edu/adamross/www/RHODES_CSER04.pdf.

Rittel, Horst W. J. and Webber, Melvin M. (1973). "*Dilemmas in a General Theory of Planning*," Policy Sciences 4: 155 – 169, Amsterdam: Elsevier Scientific Publishing Company. Retrieved on 10/6/14 from www.uctc.net/mwebber/Rittel+Webber +Dilemmas+General_Theory_of_Planning.pdf.

Rowe, Peter G. (1998), *Design Thinking*, Cambridge, MA: The MIT Press.

Scacchi, Walt (2001), *Process Models in Software Engineering*, Irvine, CA: University of California Institute for Software Research. Retrieved on 1/5/13 from http://www.ics.uci.edu /~wscacchi/Papers/SE-Encyc/Process-Models-SE-Encyc.pdf.

Schmidt, P., Zalewski, J., Murphy, G., Morris, T., Carmen, C., van Susante, P. (2011), "*Case Studies in Application of System Engineering Practices to Capstone Projects*,"American Society for Engineering Education(ASEE) Conference and Exposition. June 26-29, 2011. Vancouver, Canada. Retrieved on 1/11/14

from http://www.asee.org/public/conferences/1/papers/1537/download.

SE-CMM (1994), Bate, Roger; Reichner, Albert; Garcia-Miller, Suzanne; Armitage, James; Cusick, Kerinia; Jones, Robert; Kuhn, Dorothy; Minnich, Ilene; Pierson, Hal; & Powell, Tim (1994) *A Systems Engineering Capability Maturity Model, Version 1.0 (CMU/SEI-94-HB-004)*. Pittsburgh, PA: Carnegie Mellon University (CMU) Software Engineering Institute (SEI), Retrieved on 1/5/13 from http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=12037.

SEI (2104), *SEI Statistics and History webpage*, Pittsburgh, PA: Carnegie Mellon University (CMU) Software Engineering Institute (SEI), Retrieved on 1/5/13 from http://www.sei.cmu.edu/about/statisticshistory.cfm.

The Standish Group, (Periodic) Chaos Reports, Boston, MA: The Standish Group.

Vitasek, Kate (2014), "*Churchman, Rittel, and Webber: The 'Wicked Problem'*," *Outsource Magazine*, Issue #35, Spring 2014, London: EMP Media, Ltd. Retrieved on 4/25/14 from http://outsourcemagazine.co.uk/churchman-rittel-and-webber-the-wicked-problem/.

Warwick, Graham and Norris, Guy (2010), "*Designs for Success: Calls Escalate for Revamp of Systems Engineering Process*", Aviation Week, Nov. 1, 2010, Vol. 172, Issue 40, Washington, DC: McGraw-Hill.

Wasson, Charles S. (2008), Systems Thinking: Bridging the Educational Red Zone Between System Engineering and Program Management Education, 2nd Annual INCOSE Great Lakes Conference (GLC), Mackinac Island, MI, September 7–9, 2008. Retrieved on 1/11/14 from http://www.destinationmi.com/documents/Systems_Thinking_Government-Industry-Academia_WASSON.pdf.

Wasson, Charles S. (2011), Model-Based Systems Engineering (MBSE): Mirage or Panacea, Dearborn, MI: 5th Annual International council on Systems Engineering (INCOSE) Great Lakes Conference (GLC). Retrieved on 1/8/14 from http://www.omgwiki.org/MBSE/lib/exe/fetch.php?media=mbse:wassonmbsepanaceaormirage.pdf.

Wasson, Charles S. (2012), *System Engineering Competency: The Missing Course in Engineering Education*, *National Conference, San Antonio, TX: American Society for Engineering Education (ASEE)*, Retrieved on 1/8/13 from http://www.asee.org/public/conferences/8/papers/3389/view.

Watson, John C. (2008), *Motivations for Model Driven Development*, Moorestown, NJ: Lockheed Martin MS2. Retrieved on 1/11/14 from http://www.omgwiki.org/MBSE/lib/exe/fetch.php?media=mbse:motivation_for_model_driven_development_e.pdf.

PART I

SYSTEM ENGINEERING AND ANALYSIS CONCEPTS

3

SYSTEM ATTRIBUTES, PROPERTIES, AND CHARACTERISTICS

If you ask most people, including engineers, analysts, and educators, how they perceive a system, the usual response focuses on *equipment*, *hardware* and *software*.

If you ask the same group to give examples of systems, the responses include physical systems such as computers, cars, and spacecraft.

These two observations exemplify paradigms and voids in our educational and training systems, especially for engineers, scientists, and analysts. Since engineers are a critical element in engineering systems for the "benefit of mankind," one would think the concepts of "systems" and "systems thinking" would be integral elements of Engineering education.

This chapter begins our journey to help you understand *what a system is*. Given the definition of a *system* in Chapter 1, we explore what a system is as an object or entity. Imagine for a moment that the system is a multi-story building with impressive architectural structure and glass exterior that does not provide clues as to its purpose. Office building? Apartment building? High-tech manufacturing facility? Hospital? Our discussions will focus on the system – its attributes, properties, and characteristics.

3.1 DEFINITION OF KEY TERMS

- **Command and Control (C2)**—The *closed loop* process of: (1) continuously monitoring *planned* versus *actual* system, product, or service mission performance; (2) performing situational assessments to determine corrective actions; (3) issuing commands to the system to achieve the performance.
- **Emergence**—A behavioral property of a system or entity that emerges from configuring and integrating the properties and characteristics of its constituent components that may not be apparent on an individual component basis. For example, physical pieces and parts of an aircraft, when fully disassembled, may not reveal its ability to fly.
- **Entity**—A "general term to denote the system, item, software, process, or material that is the subject of a specification" (MIL-STD-961E, p. 4).
- **Fit**—An item's *compatibility* to mechanically interface with another item within a prescribed set of limits with ease and without interference.
- **First Article System**—Refer to Chapter 12 Definitions of Key Terms.
- **Form**—An item's shape, geometry, material, or surface characteristics required to support one or more interface boundary constraints.
- **Form, Fit, and Function**—"In configuration management, that configuration comprising the physical and functional characteristics of an item as an entity, but not including any characteristics of the elements making up the item" (Copyright © 2014 ISO/IEC/IEEE. Used with permission) (ISO/IEC/IEEE 24765:2010, p. 127).
- **Function**—A *unitless* operation, activity, process, or action performed by a system element to achieve a

System Engineering Analysis, Design, and Development: Concepts, Principles, and Practices, Second Edition. Charles S. Wasson.

Companion Website: www.wiley.com/go/systemengineeringanalysis2e

specific objective. Functions represent *actions* such as to move a force through a distance, analyze and process information, transform energy or physical properties, make decisions, conduct communications, and interoperate with other OPERATING ENVIRONMENT systems. A *function* bounded by a level of performance constitutes a *capability*.

- **Latent Defects**—Residual, undiscovered, defects or hazards due to: (1) specification or design flaws, errors, and deficiencies; (2) poor workmanship practices, (3) material composition impurities, imperfections, or blemishes that may impact system performance or cosmetically diminish its aesthetic value.
- **Level of Performance**—An objective, measurable parameter presenting a threshold performance that serves to bound the ability of a system to perform an action based on a set of scenario assumptions, initial conditions, and operating conditions. Examples include: physical characteristics – force, frequency, rate of change, etc.; PERSONNEL proficiency; system effectiveness and efficiency; and so forth.
- **Performance**—"The degree to which a system or component accomplishes its designated functions within given constraints, such as speed, accuracy, or memory usage" (Copyright, 2012, IEEE. Used with permission) (ISO/IEC/IEEE 24765:2010, p. 219).
- **Performance Effector**—A factors that influences a system or entity's performance-based outcome.
- **Pre-Planned Product Improvements (P3I)**—A staged strategy for upgrading a system, product, or service with new capabilities or technologies to meet specific mission objectives such as payloads, cost, weight, or performance; correct for system vulnerabilities deficiencies; and so forth.
- **Physical Attributes**—"Quantitative and qualitative characteristics of material, including interfaces; for example, composition, dimensions, finishes, tolerances, source and object code, compilation information, complexity level, data structure, platform, drivers" (ANSI/EIA-649-1998, p. 8).
- **Staging or Control Point**—A pre-defined programmatic decision event such as a conference, review, or demonstration intended to assess the progress, status, maturity, and risk of a work product such as a plan, specification, and design prior to committing resources to proceed to the next step or System/Product Life Cycle phase.
- **Sustainment**—The logistical delivery of *essential* MISSION RESOURCES (*consumables* and *expendables* (Chapter 8) such as fuel, lubricants, parts, food, water, medicines, and health) to sustain MISSION SYSTEM and ENABLING SYSTEM Operations and Maintenance (O&M) daily needs to ensure missions continue without interruption.
- **System Design Solution** – The evolving technical documentation set - system specifications; designs; drawings; system description; analyses and trade studies; models, simulations, and their results; test procedures; conference minutes; and so forth - that captures the Developmental Configuration of a deliverable system / product or a specific model or version.
- **System Element**—A label applied to classes of entities that comprise a System of Interest's (SOI's) MISSION SYSTEM and ENABLING SYSTEM(S), HIGHER ORDER SYSTEMS, or PHYSICAL ENVIRONMENT domains. As a convention, specific System Element names employ 1st letter capitalizations (e.g., PERSONNEL, EQUIPMENT) throughout the text to facilitate identification and context of usage. Refer to Chapter 8.
- **System End User**—An individual or Enterprise that benefits directly or indirectly from the outcome or results of a system, product, or service. End Users typically *do not* require system operation training. For example, an aircraft's passengers are System End Users that benefit from being transported from one airport to another.
- **System of Interest (SOI)**—An entity such as a system, product, or service with boundaries scoped for contextual analysis, research, or study purposes and tasked to perform one or more Enterprise or organizational missions with outcome-based performance objective(s) within a specified time frame, available resources, and within specified operating constraints.
- **System Stakeholder**—An individual or Enterprise that has a vested interest whether friendly, competitive, or adversarial in the outcome produced by a system, product, or service in performing its assigned mission.
- **System User**—An individual or Enterprise accountable for the C2 of a system, product, or service in performing its assigned mission. System Users *may require some degree of training and possibly certification*. For example, an aircraft's pilots, as System Users, require rigorous training and certification to safely C2 the aircraft.
- **Transfer Function**—A mathematical model that expresses an output as a function of its input(s). For example, $y = f(x_1, x_2, x_3 \ldots.)$ where y represents an output as a function of x_n.
- **Validation**—Refer to the definition provided in Chapter 2.
- **Verification**—Refer to the definition provided in Chapter 2.

3.2 ANALYTICAL REPRESENTATION OF A SYSTEM

Analytically, a system is represented as a simple entity depicted graphically as a rectangular box as shown in Figure 3.1. In general, Inputs such as stimuli, excitations, or cues are fed into a system that transforms the inputs via value-added processing – transfer functions - to produce an Output(s) such as products, by-products, services, or behavior. As a construct, symbolically the box is *acceptable*; however, we need to more explicitly understand *what* the system performs. That is, the system must *efficiently* and *effectively* add value to its input to produce a responsive output that meets the operational needs of its User.

As a simple conceptual diagram of a system, Figure 3.1 is easy to understand. However, from an analytical perspective, the diagram is missing key information that relates to how the system operates and performs within its OPERATING ENVIRONMENT. Therefore, we expand the diagram to identify these missing elements. The result is shown in Figure 3.2. The attributes of the construct, which include Desirable/Undesirable Inputs, Stakeholders, and Desirable/Undesirable Outputs, serve as a key checklist to ensure that all contributory performance effectors are duly considered when specifying, designing, and developing a system.

3.2.1 System Capabilities

We refer to the transformational processing – transfer functions - that adds value to inputs and produces an output as a *capability*. You will often hear people refer to this as the system's functions or functionality; this is only partially correct. *Functionality* expresses an *action to be performed*, not how well as characterized by *performance*. This text uses *capability* as the operative term that encompasses both the *functionality* and *performance* attributes of a system.

3.2.2 The System Analytical Construct

All NATURAL and HUMAN SYSTEMS—Enterprise and Engineered—exist within an abstraction we refer to as the System's OPERATING ENVIRONMENT. Survival for many systems ultimately depends on its capabilities—physical attributes, properties, characteristics, strategies, tactics, security, timing, and luck.

If we observe and analyze these systems and their patterns of behavior to understand how they adapt and survive, we soon discover that they exhibit a common construct or template that describes a system's interactions with their OPERATING ENVIRONMENT. We refer to the system being studied or analyzed as the System of Interest (SOI). Figure 3.2 provides a graphical depiction of the construct. Using an automobile example, an SOI, which is context dependent, could be a tire, steering system, engine, radio, or the total vehicle.

When an SOI (e.g., system, product, or service) interacts with its OPERATING ENVIRONMENT, several types of behavioral patterns emerge as key Systems Engineering (SE) principles:

Principle 3.1 **System Interactions Principle**

Systems, products, and services must be capable of *encountering*, engaging, and *responding* to external systems and dynamic conditions in their OPERATING ENVIRONMENT.

Principle 3.2 **Types of Interactions Principle**

External system encounters and interactions are characterized as *cooperative*, *supportive*, *benign*, *competitive*, *harsh*, *aggressive*, *hostile*, and *defensive*, or *combinations* of these.

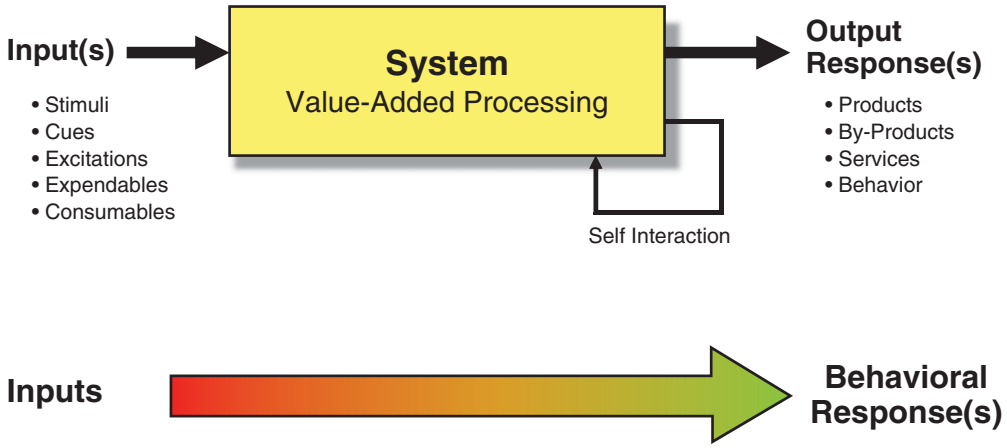


Figure 3.1 Simple Diagram of a System

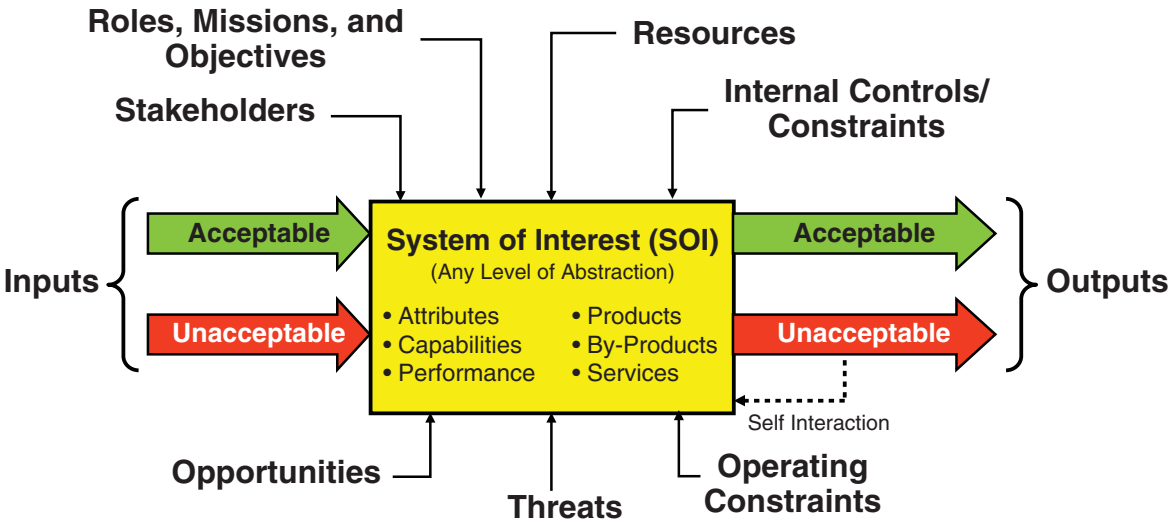


Figure 3.2 Analytical System Entity Construct

System Reactive and Adaptive Behavior Principle

Principle 3.3 Systems, products, and services must be capable of responding with *reactive* and *adaptive* behavior - non-responses, aggressive actions, protection mechanisms, or defensive countermeasures - to *stimuli*, *excitations*, and *cues* originating from external systems in their OPERATING ENVIRONMENT.

System Responses Principle

Principle 3.4 Systems produce *products*, *by-products*, *services*, *behaviors*, or *combinations* of these to accomplish mission outcome-based performance objectives and survive in their OPERATING ENVIRONMENT.

Law of Unintended Consequences Principle

Principle 3.5 Systems, products, by-products, or services responses may result in self-inflicted *adverse* or *catastrophic* conditions or effects with *negative* consequences that impact its performance, mission, or survival.

When you analyze interactions of an SOI with its OPERATING ENVIRONMENT, two fundamental types of interactions emerge:

- Peer-level, one-to-one interactions.
- Hierarchical interactions (i.e., vertical interactions under the C2 of HIGHER ORDER Systems such as Enterprise or organizational management or subject to natural forces and laws such as gravity).

HIGHER ORDER Systems Principle

Principle 3.6 Every system serves at the pleasure of or is subject to HIGHER ORDER, HUMAN SYSTEMS and NATURAL ENVIRONMENT SYSTEMS that exercise authoritative control over the system, its operation, and the conduct of its missions.

When an SOI interacts with its OPERATING ENVIRONMENT, it:

1. Performs mission task assignments established by higher level, chain of command decision authorities.
2. Interacts with external OPERATING ENVIRONMENT Systems such as HUMAN, NATURAL, and INDUCED during missions.

Author's Note 3.1 The OPERATING ENVIRONMENT is comprised of three types of PHYSICAL ENVIRONMENTS—HUMAN SYSTEMS, NATURAL, AND INDUCED ENVIRONMENTS—discussed later in Chapter 9. ITS

SOI interactions with its OPERATING ENVIRONMENT include two types of entities: (1) a HIGHER ORDER SYSTEMS Domain and (2) a PHYSICAL ENVIRONMENT Domain.

The identification of OPERATING ENVIRONMENT domains enables us to expand the *System Analytical Construct* shown in Figures 3.1 and Figure 3.2.

3.3 SYSTEM STAKEHOLDERS: USER AND END USER ROLES

System Existence Principle

Every system exists for its stakeholders based on their *perceived* operational needs.

Principle 3.7

Every system exists at the pleasure, benefit, and operational needs of its Stakeholders. When that relationship diminishes, ends, or the system is destroyed, the reason for the system, product, or service's existence has ended. We see this in marketing as illustrated by what are referred to as Consumer Product or Technology Adoption "S" curves and their segments—Early Adopters, Mass market Adoption, Mature Market, and Later Adopters (Rogers 1962).

Systems have two types of Stakeholder roles - Users and End Users. Observe usage of the term role. Some systems may have an operator that serves in both roles—System User and System End User. Let's explore and delineate each of these types of roles and then discuss how System Users can also be End Users.

3.3.1 System User(s) Role

A System User(s) (Role) operates and performs C2 of a system, product, or service to accomplish a mission or task order. Typically, System Users require some level of fundamental and proficiency training and may require certification and licensure. Systems such as an aircraft may also require one or more System Users – pilots – to fly the aircraft. For example, a commercial aircraft has a Pilot, a First Officer (Co-pilot), a head Flight Attendant, and Flight Attendants that perform C2 of a variety of flight operations such as communication, aviation, navigation, passenger safety, food preparation, and passenger food service. In performing their respective flight duties, they manipulate controls and mechanisms to accomplish task-based outcomes. For example, an aircraft's passengers, as System Users, learn to C2 the lighting, air direction/flow, and flight attendant call button above their seat with *minimal* or *no* training. In contrast, a pilot, as a System User of the Aircraft System, requires specialized aviation and aircraft knowledge, training, and experience as well as certification and licensure to safely C2 the aircraft.

3.3.2 System End User(s) Role

A System End User (Role) benefits *directly* or *indirectly* from a system's performance-based missions and outcomes. End User roles may or may not require limited training. For example, a passenger on a commercial airline, as an End User of the Aircraft System, benefits from being transported from one airport to another.

The preceding System User and End User examples illustrate a specific role-based context. A key question emerges: *Can a system's operator or maintainer be a System User and a System End User and vice versa*? The answer is, yes. Let's explore this point further.

3.3.3 System Users as End Users

To illustrate how System Users can also be End Users and vice versa, let's expand on those same examples. Observe in each case how PERSONNEL perform missions with dual roles as System Users and End Users.

3.3.3.1 Aircraft Pilot/First Officer as System Users and End Users The Pilot/First Officer, as System End Users of the Aircraft System, benefit from its real-time flight operations outcomes such as flying qualities; cockpit performance displays; cautions, alerts, and warnings.

After assimilating and evaluating this information as System End Users, the Pilot/First Officer, perform C2 as System Users of the aircraft to make corrective actions—ascend, descend, or turn—and take evasive actions related to weather or other aircraft.

3.3.3.2 Airline Passengers as System Users and End Users An aircraft passenger, as a System User of the Aircraft System, *activates* – C2 - the aircraft's flight attendant Call Button as a Request for Service (RFS). When the Flight Attendant responds to the RFS, the passenger, as a System End User of the Aircraft System, benefits from the services provided by the Flight Attendant.

3.3.3.3 Flight Attendants as System Users and End Users Flight attendants, as System End Users of the Aircraft System, benefit from receiving a passenger's RFS via the seat's Call Button identifying a specific passenger requiring service. The Flight Attendant responds to the passenger's RFS. On arrival at the passenger's seat, the Flight Attendant, as System User of the Aircraft, *deactivates* – C2 - the Call Button above the passenger's seat. The deactivation turns off the Call Button light as feedback that benefits the Flight Attendant as a System End User.

The preceding discussion introduces the concept of System Users and End Users as System Stakeholders. We will expand our discussion of Users and End Users further in Chapter 4.

Heading 3.1

People use the terms *attributes*, *properties*, and *characteristics* as part of our vocabulary. *However, what do these terms mean*? To the casual observer researching definitions of these terms, most dictionaries define these

terms by referencing one of the other terms (i.e., circular referencing). The sum of a system's attributes, properties, and characteristics uniquely identifies and distinguishes a system, product, or service from others of the same classification. Let's begin with System Attributes.

3.4 SYSTEM ATTRIBUTES

The term *attributes* describes a system's quality traits or physical features that may be *objective*—observable and measurable—or *subjective*. Examples include system/product name, model, serial number, contract number, unit cost, and fixed-wing aircraft versus rotorcraft.

All NATURAL and HUMAN SYSTEMS—Enterprise and Engineered—have unique sets of attributes (traits). Examples include roles, behavioral patterns, temperament, and appearance, even within the same species. In general, key attributes of uniqueness include the items such as the examples listed in Table 3.2.

3.5 SYSTEM PROPERTIES

The term *property* represents unique *observable* and *measurable* features of a system or product that may be *physical, emergent*, or *intangible*. Let's define each type.

Physical properties characterize the physical state of a system or component such as size, geometrical shape, and surface. Examples include:

TABLE 3.1 Examples of System Properties

Physical Property	Example Parameters
Size	Length, width or depth, height, area
Geometrical shape	Square, rectangular, irregular, spherical
Optical	Luminance, reflectivity, irradiance, opacity, spectral frequency, intensity
Thermal	Color temperature, absorption, insulation, coefficient of expansion/contraction
Mechanical	Mass, density, hardness, brittleness, force, velocity, acceleration/deceleration, momentum, pressure
Electrical	Charge, voltage, current, resistivity
Aerodynamic	Forces of flight—lift, drag, weight, thrust

One of the challenges in identifying physical properties, especially for color, is how humans perceive color and the influence of reflectivity based on surface properties of an object.

Mass properties are those properties of a system or product that characterize its physical implementation. In addition to the physical properties noted in the preceding discussion—materials, weight, and size—examples include center of gravity (CG), reference axes, weights and balances, moment of inertia, density, and so forth. Refer to Boynton and Wiener (2000) for examples of calculations.

Emergent properties are those behavioral properties of a system or product that cannot be derived from lower level properties until it is integrated, configured, and operational. Let's elaborate this point further.

Imagine for a moment that a jet aircraft had been disassembled and all the thousands of individual parts were laid out on a tarmac. Based on inspection and investigation of these components, *would it be apparent if they were to be integrated together and configured they would enable a human to fly and C2 the aircraft*? Emergence serves as a key concept that describes how humans can exploit the capabilities of the physical components to create a system that exhibits properties that are greater than the properties of each component's performance-based outcomes. Consider the following example:

Example 3.1 **Bicycle Emergent Properties Example**

Based on its mechanical properties and those of a human, who would think that we could Engineer a bicycle to allow a human to learn to balance the integrated Bicycle-Rider System to achieve and maintain stability sufficient to cycle down the road at a velocity, longer duration, and distance greater than a human can run for a given amount of energy expended?

In each of these cases, the integrated system exhibits *emergent properties* that are not apparent from analysis of the properties of the individual components.

Our discussion of what emergent properties are leaves an unanswered: *are emergent properties engineered or discovered*? Sometimes it can be either.

Returning to our aircraft example, we can say that humans have always been fascinated with being able to defy gravity and fly … on Earth … and to distant planets and solar systems. Although as humans we lack the ability to physically fly, we do have the ability to apply *Systems Thinking* (Chapter 1)—observe, envision, reason, solve problems, and assimilate properties of physical objects—in a way that allows us to exploit those properties to achieve higher level objectives than we can achieve on a personal basis. So, in that context you can say that we can ultimately "engineer" systems that exhibit emergent properties.

Observe that the previous statement said "ultimately" engineer. This is where SE becomes paramount. Consider the

TABLE 3.2 Attributes Common to Most Human Systems—Organizational and Engineered

ID	Attribute	Description
3-1	System stakeholders	Every system has at least one or more benefactors such as owners, administrators, operators, maintainers instructors, and End Users who promote and benefit from its behavior, products, by-products, or services
3-2	System Life Cycle	Every system, product, and service consists of a life cycle
3-3	System operating domain	Every system has an operating domain or "sphere of influence" that bounds its operating range, area coverage, operations, and effectiveness. Humans have learned to extend the area of coverage by employing other assets that enable a specific system to "amplify" its range **Example** An aircraft has a specific operating range under specific operating conditions such as fuel, payload, and weather. Deploying refueling sources, airborne tankers, and maintenance facilities along its mission flight path can extend the range
3-4	System frame of reference	Every system at any point in time has a *frame of reference* that serves as its permanent or temporary: • Base of operations for its operating domain • Basis for navigation **Example** • An aircraft may be assigned to a permanent home base that serves as the center of its operations. The aircraft may be ordered to perform special (temporary) assignments from a base in another global region • The Apollo Space Program used the Kennedy Space Center and the Earth as frame of reference for launch operations
3-5	Higher order systems	Every system operates within a HIGHER ORDER System (Chapter 9) that may authoritatively govern, direct, constrain, or control its operation and performance
3-6	Purpose-based role	Viewing the universe as a System of Systems (SOS), every HUMAN SYSTEM serves its Stakeholders based on a reason for its existence as envisioned by its original System User
3-7	System missions	Every system performs missions or tasks in fulfillment of its purpose to achieve performance-based outcome objectives established by its HIGHER ORDER Systems and Users
3-8	Mission goals and performance objectives	Each system has a set of goals and objectives, preferably documented, supported by one or more specific objectives that are quantifiable, measurable, testable, and verifiable. Goals and objectives provide the fundamental basis for resource expenditures and investments by the System Owner and Shareholders based on a planned set of multi-faceted accomplishments and an expected ROI
3-9	System operating constraints and conditions	Every system performing its assigned mission or task is subjected to a set of operating constraints and conditions controlled by HIGHER ORDER Systems
3-10	Operational utility	Every system must produce performance-based outcomes that are relevant to its application, ease of use, touch and feel, usefulness
3-11	Operational suitability	Every system has a level of *operational suitability* to the User in terms of suiting or meeting their mission needs and system application
3-12	Operational Usability	Every system is characterized by its ease of learnability and use that enables the User to perform their mission with minimal human error.
3-13	Operational Availability	Every system requires a level of availability to start-up on demand when required by its User
3-14	System success criteria	Each system and mission/task requires a set of criteria that the System Owner and Shareholders agree represent goals and results-oriented objectives for mission success
3-15	System reliability	Every system is characterized by a probability of success in contributing to mission objectives for a given set of OPERATING ENVIRONMENT conditions, scenarios and mission duration
3-16	System capacity	Every system requires some level of capacity to store personnel, energy, fuel, food, data, equipment, tools, and so forth

(*continued*)

TABLE 3.2 (*Continued*)

ID	Attribute	Description
3-17	System energy	HUMAN SYSTEMS—Enterprise and Engineered—require an energy source to provide responses to incoming stimuli, excitations, or cues. The source might be replaceable, restorative, and regenerative
3-18	Operational effectiveness	Every system has a cost, technical effectiveness, and probability of success related to accomplishing its missions **Example** Consider the *system effectiveness* of an educational system, healthcare system. The challenge is effectiveness from *which* Stakeholder's perspective
3-19	System efficiency	Every system has a degree of efficiency in transforming or converting energy, processing raw materials, information, or responding to stimuli, cues. As engineers, we assign an efficiency metric that mathematically expresses a ratio of the quantity of output produced for a known quantity on input
3-20	System sustainment	Every system, product, or service requires resources such as personnel, funding, consumable, and expendables; corrective and preventive maintenance; and support such as spares, supplies, and training to ensure success in accomplishing its mission
3-21	System promotion	Some systems, namely, businesses, promote their systems in anticipation of future sales via demonstrations and advertising
3-22	System concealment	Some systems may employ camouflage or stealth methods to avoid detection, visibility, or existence
3-23	System threats	Every system and its missions may be threatened by competitors or adversaries within its Operating Environment that may exhibit friendly, benign, aggressive, hostile intentions or actions
3-24	System protection	Every system requires a level of protection to minimize its vulnerability to external threats
3-25	System security	HUMAN Systems—Enterprise and Engineered—may require a level of security such as physical security (PHYSEC), communications security (COMSEC), operational security (OPSEC), and information security (INFOSEC)
3-26	System architecture	Every system consists of a multi-level, operational, logical (functional), and physical structure or architecture that provides the framework for its form, fit, and function
3-27	System capabilities	Every system, by definition, has inherent capabilities such as processing, strengths, or mathematical transfer functions that enable it to process or transform inputs such as raw materials, information, stimuli, and provide a response in the form of behavior patterns, products, and by-products System capabilities, like operating domains, can be extended using tools or other systems
3-28	System concept of operations (ConOps)	Every system requires a Concept of Operations (ConOps) as envisioned by its System Owner, System Developer, and/or System Maintainer that expresses HOW the system will be deployed, operated, maintained, sustained, retired, and disposed
3-29	System application	Every system is designed for applications such as single use, reusable, or multi-use
3-30	Operating norms, standards, and conventions	Every system employs a set of operating norms, standards, and conventions that govern its operations, behaviors, morals, ethics, and tolerances
3-31	System description	Every system requires an operational, behavioral, and physical description that characterizes its system architecture, System Elements, interfaces, behavioral responses, and physical implementation Each of these characteristics is represented by system capabilities and Engineering performance parameters that must be captured and articulated as requirements in the System Performance Specification (SPS)
3-32	System operating constraints and conditions	Every system has *operating constraints* and *conditions* that may be physical (capabilities), imposed by higher order authority—international, governmental, environmental, social, economic, financial, and psychological
3-33	Modes of operation	Every system consists of *modes* of operation that enable its Users to safely perform Command and Control (C2) to achieve performance-based objectives and outcomes
3-34	States of operation	Every system consists of *states* of operation that relate to its deployment, state of operational readiness, or physical condition

TABLE 3.2 (*Continued*)

ID	Attribute	Description
3-35	Current operating condition	Every system is characterized by a physical condition that affects its ability to successfully perform its missions
3-36	Operational status	Every system and its components have an *operational status* related to its current operation such as On/Off, Enabled/Disabled, Activated/Deactivated, Energized/Deenergized, Open/Closed, failed, degraded, calibrated, and aligned
3-37	System readiness	Every system has an Operational Health Status that represents its current state of readiness to perform or support User missions
3-38	System sensors	Human Systems—Enterprise and Engineered—require some form of sensory receptors that enable it to detect *external* stimuli, excitations, or cues and process inputs or internal status or operating condition
3-39	System behavior patterns	Every system is characterized by patterns of behavior that represent responses to interact with its OPERATING ENVIRONMENT and conditions
3-40	System responsiveness and sensitivity	Every system possesses time and performance-based behavioral capabilities that characterize its ability to respond to or process raw materials, stimuli, excitations, or cues and provide a response
3-41	System interfaces	Every system has internal and external interfaces that enable it to interact with its OPERATING ENVIRONMENT as well as itself
3-42	System pedigree	Every system has a pedigree derived from a legacy system designs, technologies, and improvements to those designs to correct for design flaws, defects, deficiencies, and errors
3-43	Mission resources (system inputs)	Every system requires resource inputs such as tasking, expendables, consumables, and operator actions that can be transformed into specific actions required to stimulate, motivate, maneuver, propel, process, and output behavioral and physical responses
	System technology	Every system is implemented with a type of technology that has a performance-based shelf life and utility
3-44	System products, services, and by-products	Every system produces: • Value-added products and/or performs services that benefit its Stakeholders • By-products that may impact system performance and/or its OPERATING ENVIRONMENT **Example** By-products include heat, waste products such as trash, exhaust, thermal signatures, colorations
3-45	Procedural data	Every Human System requires procedural data that describe safe operating procedures related to equipment, services, operator interfaces, and interfaces with external systems
3-46	System vulnerability	Every system has some form *vulnerability* that represents uncertainties or shortcomings in its operational, behavioral, and/or physical characteristics Vulnerability includes physical, psychological, social, economic, security, privacy, and other factors **Example** Military tanks have additional layers of protection to minimize the impacts of direct hits. Internet sites have vulnerabilities to computer "hackers"
3-47	System lethality	Offensive military systems are characterized by their lethality—their potential to destroy or inflict damage, disable, neutralize, or otherwise cause harm to a threat or target
3-48	System survivability	Every system consists of operating tactics and physical characteristics that enable it to survive encounters with external systems in its OPERATING ENVIRONMENT
3-49	System fault tolerance	Every system has degrees of fault tolerance that enable it to perform missions and achieve mission objectives while operating at a degraded level of performance for a given set of internal or externally induced or malfunctions
3-50	System agility	Every system requires a level of agility to respond to opportunities and threats in its OPERATING ENVIRONMENT to ensure its survival and success
3-51	System C2	Every system requires C2 of its operations, processing, behaviors, and actions
3-52	System stability	Every system requires a level of stability to accomplish mission operations

(*continued*)

TABLE 3.2 (*Continued*)

ID	Attribute	Description
3-53	System operators	Every system requires one or more human operators or capabilities to C2 its missions and operations
3-54	System maintainers	Every system requires maintainers to perform preventive and corrective maintenance actions to ensure mission performance is achieved.
3-55	System instructors	Some systems, especially complex systems, require instructors to ensure that the Users are proficient in safely operating the system to perform missions
3-56	System aesthetics	Every system possesses psychological or appearance characteristics that are aesthetically pleasing or appealing to the senses of its Stakeholders
3-57	System latent defects	Every system is unique in its development and often includes residual, undiscovered, latent defects—design flaws, errors, and deficiencies; workmanship and material defects imperfections or blemishes; that may impact system performance or cosmetically diminish its aesthetic value
3-58	System risk	Every system, product, or service has an element of risk related to mission operations and its OPERATING ENVIRONMENT that include a: • Probability of occurrence • Consequence(s) of failure
3-59	System Environmental, Safety, and Health (ES&H)	Every Human System imposes some Level of ES&H risks to system personnel—operators, maintainers, private and public property, and the environment
3-60	System Total Cost of Ownership (TCO)	Every Human System has a TCO that is cumulative over its life cycle and includes *non-recurring (NRE)* and *recurring* engineering development costs plus deployment, OM&S, and retirement/disposal costs

ABET definition of *Engineering* and the Wasson definition of *SE* stated earlier in Chapter 1:

1. ABET definition of *Engineering*—"… knowledge of the mathematical and natural sciences gained by study, experience, and practice is applied with judgment to develop ways to utilize economically the materials and forces of nature for the benefit of mankind (Prados, 2007, p. 108)."
2. Wasson SE is the "multi-disciplinary application of analytical, mathematical, and scientific principles to problem solving and solution development based on formulating, selecting, and developing an optimal solution that has acceptable risk, satisfies User operational need(s), and minimizes development and life cycle costs while balancing Stakeholder interests."

We can say that humans can apply Engineering and SE methods to innovate and create solutions that do have emergent properties.

Can emergent properties be *discovered*? Yes. Consider the following examples:

Example 3.2 **Cause and Effect Emergent Behaviors**

The medical profession sometimes discovers that drugs originally developed for certain types of medical conditions can have *positive* or *negative* effects on other conditions. In cases where the results of positive, emergent properties open new opportunities for further research and healing.

Example 3.3 **Campus Sidewalks – Emergent Behaviors**

Architects developing a new campus may sometimes perform an analysis of potential high traffic areas for installing sidewalks, wait several weeks or months, assess worn footpaths across grassy areas, and then install sidewalks in those areas. Analyses are fine but have limitations; however, the student Users exhibit emergent properties that may not have been apparent in the initial analyses or planning.

Intangible properties are those properties of a system or product that have an intrinsic, psychological value to stakeholders. For Example, nuclear, biological, and chemical weapon systems act as a *deterrence* to adversaries. Conversely, marketing creates website designs, product features, etc. having intangible properties that appeal to the attention of Users and result in their selection over other systems or products.

3.6 SYSTEM CHARACTERISTICS

The term *characteristics* refers to the operational, behavioral, and physical performance that is observable, measurable, and uniquely identifies a system's performance.

When we characterize systems, especially for marketing or analysis, there are four basic types of characteristics we consider: (1) general characteristics, (2) operating or behavioral characteristics, (3) physical characteristics, and (4) system aesthetics.

Every system consists of high-level *general characteristics* that enable us to describe its key features. We often see general characteristics stated in marketing brochures where key features are emphasized to capture a client's or customer's interest. General characteristics often have some commonality across multiple instances or models of a system. Consider the following examples:

Example 3.4

Automobile General Characteristics
Handling qualities such as swerving and cornering, two-door or four-door models; convertible or sedan; air-conditioned comfort; independent suspension; tinted windows, 22 mpg city, 30 mpg highway

Aircraft General Characteristics
Flying or handling qualities such as stability and control, 50-passenger, 2000 nautical mile range, IFR capabilities

Enterprise or Organization General Characteristics
200 employees; staff with 20 PhDs, 50 MS degrees, and 30 BS degrees; annual sales of $500M per annum

Network General Characteristics
Client–server architecture, PC and Unix platforms, firewall security, remote dial-up access, Ethernet backbone, network file structure (NFS)

At a level of detail below the General Characteristics, systems have *Operating Characteristics* that describe system features related to usability, vulnerability, survivability, and performance for a prescribed Operating Environment. Consider the following examples:

Example 3.5

Automobile Operating Characteristics
Maneuverability, turn radius of 18 ft., 0 to 60 mph in 6 seconds

Aircraft Operating Characteristics
All-weather application, speed

Network Operating Characteristics
Authorized access, access time, latency

Every system is characterized by *nonfunctional* physical attributes such as size, weight, color, capacity, and interface attributes. Consider the following examples:

Example 3.6

Automobile Physical Characteristics
2000 lbs, curb weight 14.0 cu. ft. of cargo volume, 43.1 of inches (max) of front leg room, 17.1 gals fuel capacity, 240 horsepower engine at 6250 rpm, turbo, available in 10 colors

Enterprise or Organization Physical Characteristics
20,000 sq. ft. of office space, 300 PhDs, 300 networked computers, 100,000 sq. ft. warehouse

Network Physical Characteristics
1.0 Mb Ethernet backbone, topography, routers, gateways

Observe usage of the term "non-functional" in the previous paragraph and examples above. Recall that a *function* represents an action to be performed. Non-functional indicates that an action is not performed such as size, weight, color.

In summary, *general*, *operating*, *behavioral*, and *physical* characteristics are *objective* performance parameters that are observable and measureable. However, what about *subjective* characteristics? We refer to these as system *aesthetic characteristics* because they relate to the "look and feel" appearance of a system that appeals to the User's, Acquirer's, or System Owner's preferences. Thus, some buyers make independent decisions, while others are influenced by external systems (i.e., other buyers) in matters relating to community or corporate status, image, and the like.

3.7 THE SYSTEM'S STATE OF EQUILIBRIUM AND THE BALANCE OF POWER

Principle 3.8

System Equilibrium Principle

Every system, depending on its condition, exists in a state of equilibrium with its Operating Environment to ensure survival.

Collectively, a system's attributes, properties, and characteristics integrate to create its unique identity and enable it to *operate* and *survive* within a given Operating Environment. When the system does survive, future survival depends on its ability to exist, evolve, and change in a "state of equilibrium" relative to its Operating Environment. In general, we refer to this as the "balance of power."

The state of equilibrium depends on how a system exists through its own (1) level of dominance or (2) subordination to and protection by other systems. At any instance of time prior to, during, and following an engagement or

encounters with other systems in its OPERATING ENVIRONMENT, a system has an *initial state*, an operating condition, strengths, weaknesses, or stabilization, and a *final state* that are determined by the balance of power and results of the interaction.

3.7.1 Pre-requisite Conditions Leading to an Encounter

System stability, integrity, and consistency of performance require that transitions between system phases, operations, and tasks have clean, unambiguous transitions with no *unintended consequences*. Thus, systems are assumed by designers to have pre-requisite operating conditions that lead them to the present time or operational need that requires encounters and interactions with external systems in its OPERATING ENVIRONMENT.

3.7.2 Initial Operating Conditions and State

A system's initial operating condition and state consist of the physical integrity of its components and operational state of readiness at a specific instant in time. Since analyses often require the establishment of basic assumptions for investigating some facet of system phases, operations, or task, *initial conditions* serve as a "snapshot" or starting point that captures the assumptions. To illustrate this concept, consider the following example:

Example 3.7

The aircraft took off with a crosswind of 15 knots.

The early morning rush hour began as a blizzard moved through the area with 30 mph wind gusts.

3.7.3 Static Conditions

When we analyze systems, a key basis for the analysis is often the physical state of the system at a given "instance time." Engineering statics are used to characterize a system's current orientation, such as state vector or orientation within a larger system. From an overall system perspective, an aircraft sitting in a hanger, an automobile in a driveway, a network computer system with no message traffic, and a lighting system in the On or Off State, all represent a system in a static state. In contrast, lower level system components may have a *static* states while the system as a whole is in a *dynamic* condition and vice versa. For example, an aircraft's wing flaps and landing gear may be set in a *static state* for landing despite the aircraft experiencing *dynamic* wind conditions that must be controlled by other flight control surfaces.

3.7.4 Dynamic Conditions

Every NATURAL AND HUMAN SYSTEM conducts missions in its OPERATING ENVIRONMENT in some form of dynamic, physical state (Chapter 7). *Dynamics* are characterized by an infinite number of time-dependent system static snapshots over a defined timeframe and OPERATING ENVIRONMENT conditions. The dynamics may range from slow changes (rock anchored on a hillside) to moderate changes (temperature variations) to violent, sudden changes (wind shear, earthquakes, or volcanoes).

Dynamic conditions also include inconsistencies, perturbations, and instabilities in the balance of power in the local or global environment. Mankind has always been intrigued by the study of dynamics and their effect on behavior patterns of the Earth, weather, oceans, stock market, and people. In today's world, research shifts to dynamic, complex systems with a focus on predicting dynamic behavior and its impacts on the economy and political elections. Thus, *predicting* the *dynamics* of economies, consumer preferences, market trends, technologies, social networks, and how they influence each other are major research topics and have tremendous market potential.

3.7.5 System Stabilization

Principle 3.9

System Stabilization Principle

Every system exhibits a level of *stability* that requires the User and system to Monitor, Command, & Control (MC2) its performance to successfully accomplish mission objectives.

All NATURAL and HUMAN SYSTEMS must maintain a level of *stability* to ensure their survival and longevity. Otherwise, the system can easily become *unstable* and potentially become a threat to itself, its operators and maintainers, and the public. Therefore, systems should have inherent design characteristics – robustness - that enable them to *stabilize* and *control* their responses to dynamic, external stimuli, excitations, or cues.

Stabilization is ultimately dependent on having some form of calibrated reference that is stable, dependent, and reliable. For HUMAN SYSTEMS such as systems or products, stabilization is achieved by employing devices such as inertial navigation gyroscopes, the Global Positioning System satel (GPS), quartz crystals for electronic watches, and reference diodes for voltage regulators. In each of these cases, the system stabilization is accomplished by sensing current free body dynamics; comparing them with a known, calibrated reference source; and initiating system feedback control actions to correct any variations.

3.7.6 The Balance of Power

Taking all of these elements into account, system *existence* and *survival* are determined by its ability to:

- Cope with the statics and dynamics of its OPERATING ENVIRONMENT.
- Sustain a level of capability and stabilization that harmonizes with its adjacent systems—the balance of power or state of equilibrium.

The balance of power of systems, coupled with humankind's general desire for peace and harmony, requires systems to comply with standards imposed by society. Standards in this context refer to explicit and implicit, self-imposed expectations by society such as laws, regulations, ordnances, codes of conduct, morals, and ethics. Thus, system survival, peace, and harmony are often driven by a system's compliance to these standards. System adherence to these standards involves two terms that are often interchanged and require definition. The terms are *compliance* and *conformance*.

3.7.6.1 The Consequences of Noncompliance When a system fails to adhere to established standards or norms of society, it may place itself be at risk. Society's response to a lack of compliance generally involves *formal* or *informal* notification, establishment that noncompliance occurred, adjudication of the degree or noncompliance, and sentencing in accordance with prescribed consequences or penalties.

For systems such as ships, aircraft, and automobiles, intentional or unintentional noncompliance with the HUMAN SYSTEMS, INDUCED, and NATURAL ENVIRONMENTS can be very *unforgiving* or, even worse, *catastrophic*.

3.7.6.2 Levels of System Interactions System interactions with its OPERATING ENVIRONMENT occur at two levels: *strategic interactions* and *tactical interactions*. Let's explore each of these in detail.

3.7.6.2.1 Strategic Interactions HUMAN SYSTEMS exhibit a higher level of behavior that reflects a desire to advance our current condition as a means of achieving higher level vision. To achieve the higher level vision, humans must implement a well-defined strategy, typically long term, based on stimuli and information extracted from the OPERATING ENVIRONMENT. We refer to implementation of this long-term strategy as *strategic interactions*. These strategic interactions are actually implemented via a series of premeditated missions—*tactical interactions*—with specific mission objectives.

3.7.6.2.2 Tactical Interactions All life forms exhibit various types of tactics that enable the system to survive, reproduce, and sustain itself. We refer to a system's implementation of these tactics within the confines of its OPERATING ENVIRONMENT as *tactical interactions*. In general, this response mechanism focuses all existing survival needs in the short term on obtaining the next meal.

3.7.6.3 System Interaction Analysis and Methodology Depending on the *compatibility* and *interoperability* of an interface, consequences of an encounter or engagement with an external system may be *positive*, *neutral*, or *negative*. As an SE or System Analyst, your mission is to:

1. Develop a thorough understanding of the engagement participants (systems).
2. Analyze the System Use Cases (UCs) and scenarios (Chapter 5) by applying natural and scientific laws of physics to thoroughly understand the potential outcomes, ramifications, and consequences.
3. Specify system interface requirements that ensure engagement interactions *compatibility* and *interoperability* success within cost, schedule, and technology constraints.

3.7.6.3.1 System Adaption to Its OPERATING ENVIRONMENT Most systems are specified and designed to perform in a prescribed OPERATING ENVIRONMENT. There are situations whereby a system is transferred to a new geophysical location or environment. When this occurs, the system must adapt to its new OPERATING ENVIRONMENT. Consider the following examples:

Example 3.8 **System Adaption Example**

As part of a strategy for climbing a high mountain, mountain climbers travel to a series of base camps to satisfy logistics requirements and allow their bodies time to acclimate to the thin air environment over a period of several days.

Since humans can only survive in specific types of Earth environments, SEs must understand those conditions and constraints and recreate a similar environment to enable us to expand our base of operations and operating range such as changing geographical locations to hostile or harsh conditions or engage in space travel.

Some Engineered Systems are designed for *adaptive control* to accommodate varying parameter conditions. For example, an autopilot as a controller requires a control law that enable it to C2 the aircraft based on parameter estimation such as the aircraft's changing mass due to in-flight fuel consumption.

Heading 3.2

Given an understanding of a system's attributes, properties, and characteristics as well as its Stakeholders—Users and End Users—we shift our discussion to its System/Product Life Cycle.

3.8 SYSTEM/PRODUCT LIFE CYCLE CONCEPTS

Principle 3.10

System Life Cycle Principle

Every NATURAL AND HUMAN SYSTEM exhibits a system life cycle that characterizes its staged evolution from conception to disposal.

Enterprises and Engineered systems, products, or services are characterized by a System/Product Life Cycle. The life cycle represents the evolution of a system beginning with its conception; acquisition; development; deployment and production; OM&S; retirement; and disposal.

The System/Product Life Cycle serves as both a roadmap for understanding and communicating how NATURAL AND HUMAN SYSTEMS evolve through a progression of sequential life cycle phases. For HUMAN SYSTEMS, the roadmap provides a framework for: (1) assessing existing system capabilities and performance relative to threats and opportunities; (2) defining, procuring, and developing new systems, products, and services of upgrades to respond to the threats and opportunities; and (3) implementing new system or upgrades to achieve mission objectives that counter or leverage the threats and opportunities.

Engineered systems, products, and services originate from a point of conception based on innovation and end when the system becomes too costly to maintain, is obsolete, or no longer fills an operational need. Since this text focuses primarily on systems and products developed under contract or commercially for the marketplace, we will refer to as the System/Product Life Cycle shown in Figure 3.3.

The System/Product Life Cycle represents how an Enterprise views the conception to disposal life cycle of a system, product, or service as an asset used to perform missions. Engineered Systems, for example, are conceptualized, planned, organized, scheduled, estimated, procured, deployed, operated and supported, and retired from *active service* using this framework. Their development is marked by a *control point* or *staging event* such as a key decision that authorizes and enables progression to the next phase. NATURAL SYSTEMS follow similar constructs with life phases.

Author's Note 3.2

There are a number of ways to define a System/Product Life Cycle. Ten people will have 10 different versions of this graphic. You and your Enterprise or organization should choose one that best represents your Enterprise's needs.

Over several decades, various government and professional Enterprises have developed life cycle models. Examples include:

- US Department of Defense (DoD)
- International Council on Systems Engineering (INCOSE)
- Institute of Electrical and Electronic Engineers (IEEE)
- American National Standards Institute (ANSI)
- Electronic Industries Alliance (EIA)
- International Organization of Standards (ISO)

During the late 1990s, government and industry world-wide began to recognize and appreciate the need for a consensus standard for life cycle processes. For example, the ISO/International Electrotechnical Committee (IEC) initiated activities to develop the standard that has become known as:

- ISO/IEC 15288 *Systems and software engineering—System life cycle processes*

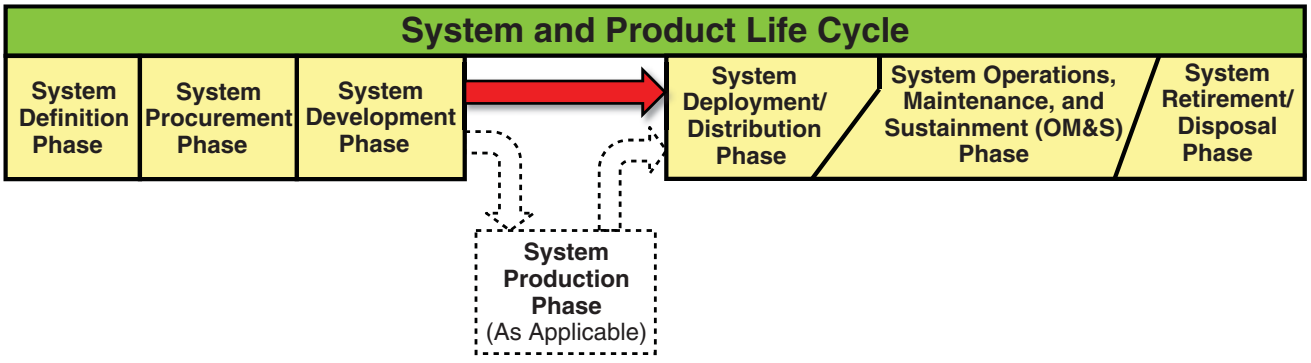


Figure 3.3 System/Product Life Cycle Overview

As a result, organizations such as INCOSE and others establish and link their standards for engineering processes to ISO/IEC 15288.

Traditionally, most organizations refer to the various "segments" of their life cycle as *phases*. In contrast, the ISO/IEC 15288:2008 life cycle identifies *stages* that include Concept, Feasibility, Development, Production, Utilization and Support, and Retirement.

One of the challenges in establishing a life cycle is the use of terms systems, hardware, software, and stakeholders within a Community of Practice (CoP). For example, by naming convention, the ISO/IEC 15288:2008 stages vary from explicit names such as Development, Production, Support, and Retirement to less explicit names such as Concept, Feasibility, and Utilization.

Author's Note 3.3

Referral

For additional information concerning the ISO/IEC 15288:2008 Life Cycle Model stages, refer to Lawson (2010) Chapter 3.

This text employs the Wasson System/Product Life Cycle and life cycle *phases* as illustrated in Figure 3.3. Table 3.3 provides a mapping between the two life cycle models. Please note: if you need to use the ISO/IEC 15288:2008 Life Cycle Model, you should do so.

In general, life cycle models serve as an Enterprise framework for planning project management activities and developing Engineered Systems. Over the years, SE has created various life cycle model frameworks. From a SE perspective, the semantics and value of a life cycle model is more than simply depicting how to structure an end-to-end workflow like a production line. The framework provides the infrastructure for SE analysis and subsequent specification requirements development. As a result, there must be harmonization of both project management and SE needs in creating a life cycle model with semantics and segmented workflow that supports both sets of objectives.

As an example, the ISO/IEC 15288:2008 Life Cycle Model transitions from Development or Production Stages directly into the concurrent Utilization and Support Stages. The reality is that commercial products and contract Engineered Systems completing Development or Production must be deployed or distributed to the field or marketplace. When deployed, they may enter an optional Storage Operations as illustrated later in Figure 6.4.

Additionally, the term Utilization leads to the question "by whom"? System Stakeholders? Operators? Maintainers? So, for educational purposes, the Wasson System/Product Life Cycle model consists of a System Operations, Maintenance, and Sustainment (OM&S) Phase.

TABLE 3.3 Mapping of ISO/IEC 15288:2008 to the Wasson System/Product Life Cycle Model

ISO/IEC 15288:2008	Wasson System/Product Life Cycle Model
Concept Stage	System Definition Phase
Feasibility Stage	System Acquisition Phase
Development Stage	System Development Phase
Production Stage	System Production Phase
	System Deployment/Distribution Phase
Utilization and Support Stages	System Operation, Maintenance, and Sustainment (OM&S) Phase
Retirement Stage	System Retirement/Disposal Phase

Traditionally, organizations often use the term *support* for simplicity. However, *support* is an abstract term that suppresses the scope of its activities. This too leads to a learning paradigm that may ignore a key activity, *sustainment*, that has gained recognition and rightfully so in recent years. The point is that although a User's Enterprise has a requirement to maintain and support a system, product, or service in the field, a *necessary* condition, it is *insufficient* unless it has have *sustainable* supply chains. Military campaigns, for example, exemplify this point. Therefore, the Wasson System/Product Life Cycle Model consists of a System OM&S Phase to provide a more explicit and complete meaning of what is required.

Commercial and other organizations often establish evolutionary *stage-gate* life cycles that focus on the productization of new technologies and maturation of a system, product, or service to ensure its readiness for the consumer marketplace. As an example, the US Department of Energy (DOE, 2007, p. 3) identifies five stages for "managing risk through project decision-making" as shown in Figure 3.4.

Based on this introduction to the Wasson System/Product Life Cycle Model, let's scope the activities that comprise each phase.

3.8.1 The System Definition Phase

The System Definition Phase begins with recognition by the User's Enterprise that a new system or upgrade to an existing—legacy—system, product, or service is required to satisfy an operational need. The operational need may be derived from (1) mission opportunities, (2) threats, or (3) projected system capability and performance "gaps" or deficiencies.

When a decision is made to acquire a new system, the User analyzes existing system operational needs and defines requirements for a new system, product, or service. In some instances, the User may enlist the services of a System Acquirer (Role) to procure the system and serve as the User's technical and contract representative during its acquisition and development. The System Acquirer is a role that may be performed by the User or contracted to an external Services

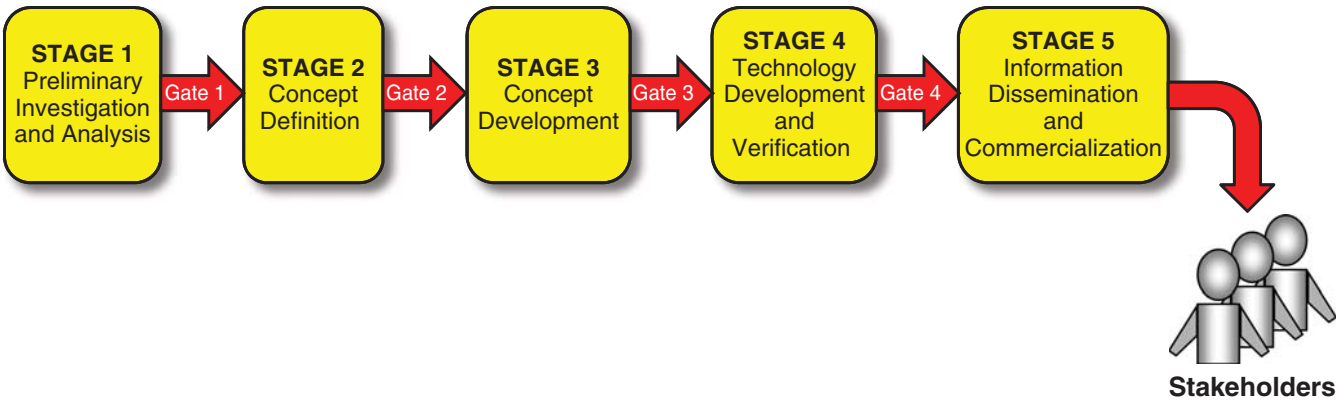


Figure 3.4 US DOE Stage-Gate Process Example *Source*: DOE (2007), Stage-Gate Innovation Management Guidelines

Provider to represent the User's interests contractually and technically during a system acquisition.

There are several reasons why Users employ the services of a System Acquirer. The User's Enterprise may be:

- Non-technical and employs high-tech systems but lack the expertise to develop new systems or upgrades internally.
- Technical but does not have the staff resources readily available to develop a new system or upgrade.
- Technical but lacks the specialized expertise or technology required to develop a new system or upgrade.

The System Acquirer, if applicable, assists the User as their technical representative in analyzing what is referred to as the Opportunity or Problem Space (Figure 4.7) that created the need. The System Acquirer, in collaboration with the User, bounds the Solution Space in the form of a set of System Performance Specification (SPS) requirements to serve as the basis for a system development contract.

When the System Definition Phase has reached sufficient maturity, the Acquirer initiates the System Acquisition Phase.

3.8.2 The System Acquisition Phase

The System Acquisition Phase consists of those activities required to formally procure the new system or upgrades to the existing system. These activities include:

1. Qualifying potential system, product, or service vendors based on a Request for Information (RFI) that elicits its qualification of capabilities, technical approaches, and subsequent down selection to a list of Offeror candidates.
2. Release of a Request for Proposal (RFP) or Quote (RFQ) from qualified vendors (Offerors)
3. Selecting a preferred vendor (Offeror).
4. Contracting with the vendor to develop the system, product, or service.

On Contract Award, two transitions occur:

- The System/Product Life Cycle Model transitions from the System Acquisition Phase to the System Development Phase.
- The selected vendor transitions from an Offeror role to a System Developer, System Integrator, or Services Provider role.

3.8.3 The System Development Phase

The System Development Phase (Figure 12.2) consists of those activities required to translate the contract SPS requirements into a physical, deliverable system. Key System Development Phase activities include:

1. System Engineering Design
2. Component Procurement and Development
3. System Integration, Test, and Evaluation (SITE)
4. System Verification
5. System Baseline Authentication
6. System Validation—Operational Test and Evaluation (OT&E)

Throughout the phase, the multi-level System Design Solution – specifications, designs, drawings, etc. - evolves through a progression of maturity phases. Each phase of maturity typically consists of a major technical design review (Chapter 18) with *entry* and *exit criteria* supported by analyses, prototypes, and technology demonstrations. The reviews culminate in design baselines that capture *snapshots* of the evolving and maturing Developmental Configuration

(Chapter 16). When the System Design Solution is formally approved, the Developmental Configuration provides the basis for component acquisition and development. We refer to the initial system(s) as the *first article* of the Developmental Configuration.

Acquired and developed components are inspected, integrated, and verified against the respective design requirements and performance specifications at various levels of integration. The intent of verification (Chapter 13) is to answer the question: *Did we develop the system in compliance with the specification requirements*? The integration culminates in a System Verification Test (SVT) (Chapter 18) that proves the system, product, or service fully complies with the contract SPS. Since the System Development Phase focuses on the creation of the system, product, or service from Contract Award through SVT, we refer to this as Developmental Test and Evaluation (DT&E) (Figure 13.6).

DT&E for *commercial* and *consumer* products (Figure 5.1) involve more than simply verifying compliance to an SPS. Consumer product safety is a critical issue from both *usage* and *human consumption* perspectives. As a result, additional verification may be required by an independent testing organization such as Underwriters Laboratories (UL®) in the United States and Conformité Européenne (CE) in Europe that a product meets the "essential" requirements of product safety established by governmental organizations and standards. Examples include the:

- US Consumer Product Safety Improvement Act of 2008
- US Food and Drug Administration (FDA)
- US Department of Agriculture (USDA)
- US Environmental Protection Agency (EPA)
- European Union (2001), the European Council Directive on General Product Safety 2001/95/EC

When the *first article* system(s) of the Developmental Configuration has been *verified*, at least two options may be available, depending on contract requirements. The system may deploy to:

- Another location for *validation* testing by the User or an Independent Test Agency (ITA) representing the User's interests
- The User's designated field site for installation, checkout, and commission for *active service*

Validation testing (Chapter 13), which is referred to as Operational Test & Evaluation (OT&E) enables Users to determine if they specified and procured the *right* system or product to meet their operational needs. Any deficiencies are resolved in accordance with the Terms and Conditions (Ts&Cs) of the contract.

After an initial period of operational field use to correct defects such as residual *latent defects*—design flaws, errors, deficiencies, and discrepancies—as well as collect field data to validate system performance, a decision is made to begin the System Production Phase, if applicable. If the User does not intend to place the system or product in production, the System Acquirer and User formally accept system delivery, thereby initiating the System OM&S Phase.

3.8.4 The System Deployment/Distribution Phase

When a system, product, or service completes its System Development Phase, the next step is to deploy or distribute it to Users or consumers. In general:

- Systems that are developed under contract between two or more parties are deployed to the User's designated field site or *staging area* for *storage* or Installation and Checkout (I&CO).
- Consumer product systems are delivered from manufacturers to distributors that serve as distribution points or channels for subsequent delivery to retail and discount stores and sold to consumers.

When the systems are finally installed and ready for operation or have been purchased by consumers, they enter the System OM&S Phase of their life cycle.

3.8.5 The System Production Phase

The System Production Phase consists of those activities required to produce small-to-large quantities of the system or product. The initial production typically consists of a Low-Rate Initial Production (LRIP) to *verify* and *validate* that:

- Production documentation and manufacturing processes are *mature* and *complete*.
- *Latent defects* such as design errors, design flaws, or poor workmanship are eliminated.

It is important to note here that simply *verifying* and *validating* the First Article systems in the System Development Phase indicates compliance to System Acquirer specifications and satisfaction of User operational needs. However, this does not mean that the verified Developmental Configuration can be cost effectively mass produced.

The System Production Phase is a form of enhanced development in which Product Engineering Teams (PDTs) investigate ways of improving the Developmental Configuration design and component selection to achieve a lowest cost solution without sacrificing reliability, maintainability, and safety of the original Development configuration. This may require *reverification* and *validation* against production specifications—Production Design Verification.

Once the production design is verified, subsequent verifications on a specific instance of the system consist of production tests to simply demonstrate that the system is operable and eliminate any latent defects such as *poor workmanship* or *faulty components*. As production systems age over time, projects typically institute a series of incremental Pre-Planned Product Improvement (P3I) upgrades and retrofits to the fielded system via new production contracts.

When the production design has completed V&V based on field tests of system production samples, full-scale production, if applicable, may be initiated. Since the system and production engineering designs have already been verified, each production system is:

- Inspected
- Verified against key SPS requirements
- Deployed - Deployment Phase - to the User's designated field site(s) for use—OM&S Phase or Storage Phase (Optional)

Author's Note 3.4

There is a difference between System Design Verification and Production System Verification. Once a design has been verified, the design is effectively complete pending any unknown latent defects – design flaws, errors, or deficiencies; or material composition defects or degradations - that may emerge over time. The only remaining variable is elimination of poor workmanship, faulty materials and components, which are unique to a specific instance of a Production System. Refer to Chapter 13, for a more detailed discussion.

3.8.6 The System OM&S Phase

The System OM&S Phase consists of User activities required to operate, maintain, and sustain the system including training for system Users to perform its operational mission. If the system is directed to change physical or geographic locations in preparation for the next mission, the system is redeployed. On deployment, the system, product, or service begins active duty.

Throughout the system's operational life, refinements and enhancement upgrades may be procured and installed to improve system capabilities and performance in support of Enterprise or organizational missions. The system configuration at initial delivery and acceptance represents the Initial Operational Capability (IOC). System upgrades, referred to *as incremental builds* (i.e., Build #1, Build #2), are released and incorporated into the fielded system or product until the system reaches a planned level of maturity referred to as Full Operational Capability (FOC).

Although most systems have a planned operational service life, the expense of maintaining a system via upgrades and ability to upgrade the existing system with new technologies is not always cost effective. As a result, the User may be forced to procure a new system, product, or service to replace the existing system. Where this is the case, a new system life cycle is initiated while the existing system is still in active service.

As the first articles of the new system are placed into *active service*, a transition period occurs whereby the legacy (i.e., existing) system and the new system operate simultaneously in the field. Ultimately, a decision will be made to *deactivate* and *phase out* the legacy system from active service. When system phase-out occurs, the legacy system's Retirement Phase begins.

3.8.7 The System Retirement/Disposal Phase

The System Retirement/Disposal Phase consists of those activities required to phase out an existing or legacy system from active service. During retirement, each system or the lot of systems may be dispositioned for sale, lease, storage or disposal. Disposal alternatives include storage for future re-commissioned use, disassembly, destruction, burning, and burial. System disposal may also require environmental remediation and reclamation to restore the system's field site or disposal area to its natural state.

3.8.8 Life Cycle Models in Real-time Operations

As stated earlier, life cycle models provide a basic strategy for generalized project management workflow over time. They do not, however, reflect the framework of real-time system operations that serve as the analytical basis for SE identifying and deriving operational capabilities that will be translated into SPS requirements. Figure 3.5 provides an illustration.

Let's assume a system, product, or service is ready to exist the System Development/Distribution Phase. Two options as noted by the bubble identifiers are available:

- Option 1—Deliver the verified system, product, or service to the System Deployment Phase for delivery to the User.
- Option 2—Transition the Developmental Configuration to the System Production Phase.

3.8.8.1 System Production Phase Operations Production systems are developed and produced in LRIP or mass quantities. On completion of the System Production Phase, the system is transitioned (Option 6) to the System Deployment Phase.

3.8.8.2 System Deployment Phase Operations The system, product, or service is transported (Option 3) from the System Developer's facility to the User's designated field site for storage, operation, and consumer product distribution.

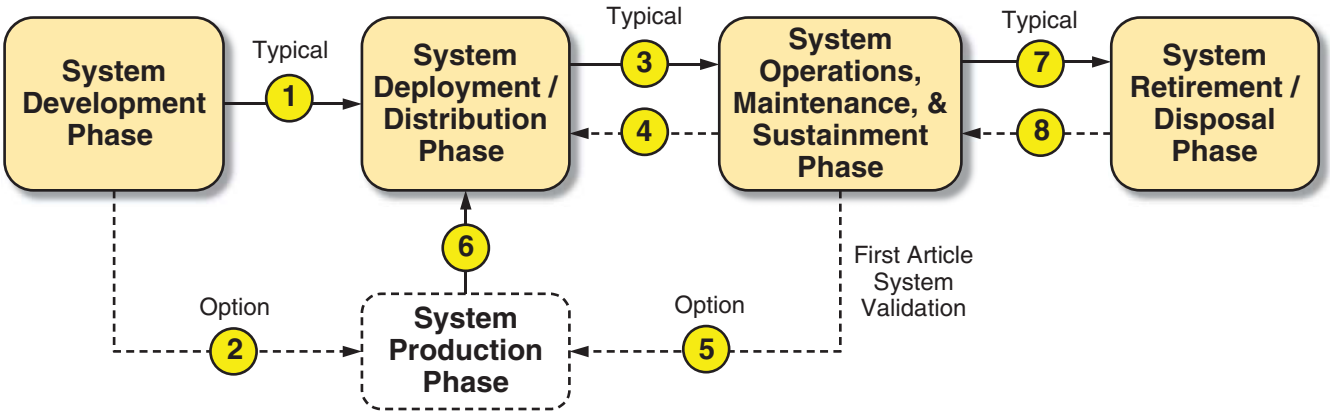


Figure 3.5 System Usage Pathway Options Through Its Lifecycle.

3.8.8.3 System OM&S Phase Operations Once the system, product, or service placed into *active service*, it performs operations, undergoes maintenance actions, and has its missions sustained. During its *active service* life, several options are available to its Users:

- Option 4—Transition back to the System Deployment Phase for redeployment to another User site and reintroduction, Option 3, to active service
- Option 5—Transition the fielded system, Developmental Configuration, after a period of field use and validation to the System Production Phase.
- Option 7—Decommission and phase-out for transition to the System Retirement Phase.

3.8.8.4 System Retirement/Disposal Phase During the System Retirement Phase, the system, product, or service may be stored or shelved in inventory until a decision is made to return it to *active service* (Option 8) or to dispose of it as an Enterprise asset.

Heading 3.3

Based on the preceding discussion, it should be obvious that a system life cycle model is more than simply identifying a generalized project management workflow. As a unifying framework for overall project success, the model should support all types of project technical uses such as SE and System Analysis. The infrastructure exhibited in Figure 3.5 provides the basis for SE and System Analysts to derive System life cycle operational capabilities addressed in Chapter 7, "System Command and Control (C2) Phases, Modes, and States of Operation."

3.8.8.5 System Phase-Out versus Retirement Some Enterprises refer to the System Phase-Out as the System Retirement Phase. The inference is an *ending* of the active service life of a system, product, or service and it is. Observe that we said *active service life*, not *end of life*. There are several scenarios that represent what might occur to a system.

- **Scenario #1**—A downturn in economic conditions may result in an excess inventory of Enterprise assets requiring them to be placed in storage until the conditions improve. In this context, a system may be placed in storage as illustrated in Figure 6.4. For example, commercial and military aircraft are sometimes stored in dry desert conditions until needed. Then, returned for use.
- **Scenario #2**—A system or product is no longer needed and will be dispositioned for disposal by the Enterprise.

The two scenarios above represent the transition of a system, product, or service to a new state—Storage or Disposal.

3.8.8.6 Nested Operational Life Cycles Now that we have a basic understanding of a system or product's life cycle, we shift our attention to understanding how a system's life cycle fits within the context of an Enterprise.

3.8.8.7 Understanding the Enterprise Aspects of System Life Cycles Systems, products, or services are owned as part of the higher level Enterprise System that also has a system life cycle. Therefore, we have multiple levels of embedded system life cycles as illustrated in Figure 3.6.

To better understand the last point, suppose that a User has quantities of a product or system, including various versions in inventory. At some point in time, the User may decide to replace a specific product or a group of products.

Example 3.9

For example, an airline might decide to replace a specific aircraft by tail number or replace an entire fleet of aircraft over a period of time. Each aircraft, which has its own life cycle, is part of a much larger Airline System that owns a fleet of aircraft.

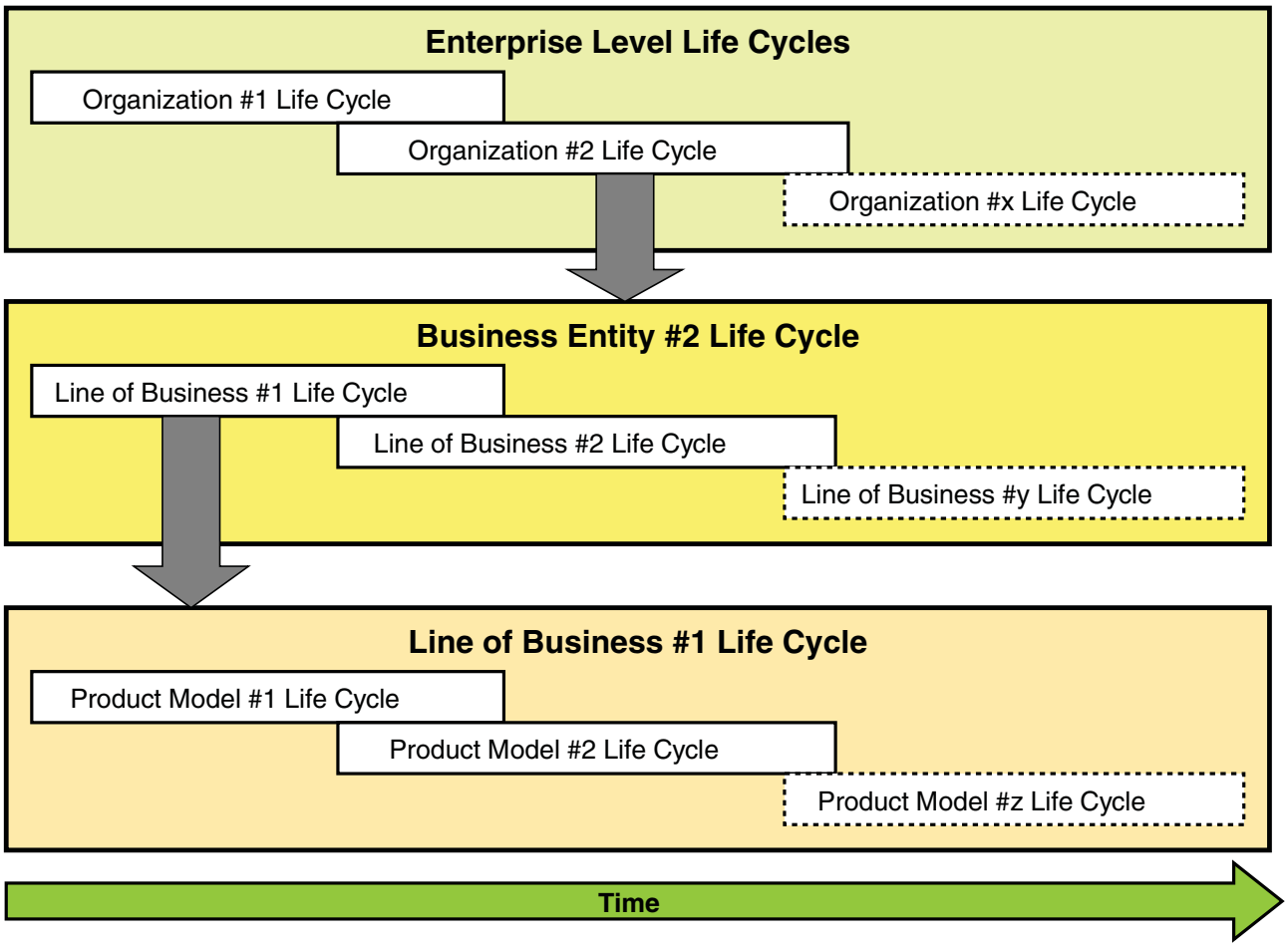


Figure 3.6 Enterprise Organizational, Line of Business (LOB), and Product Model Life Cycles

To illustrate the point of this example, Figure 3.6 provides an example. Assume we have an Enterprise that has evolved over a number of years. Historically, we can state that the business came into existence as Enterprise Entity #1. As the business entity grows, it changes its name and becomes Enterprise Entity #2.

If we examine the system life cycle of Enterprise Entity #2, we might find that the organization evolves through several Lines of Business (LOBs): LOB #1, LOB #2, and so on. Within each LOB, the organization has a core product line that consists of Product Model #1, which evolves into Product Model #2. Observe the overlapping of Product #1 and Product #2 life cycles. The evolution of this product line continues until the organization decides to terminate the product or LOB. *How is this concept applied to the real world*?

3.8.8.8 Application of System Life Cycles We can apply the concept of system life cycles within system life cycles to an example such as a small engine developer. The organization, which has a life cycle, may evolve through a number of business life cycles such as small business, corporation, and so on. During Enterprise Life Cycle #2, the organization may develop several LOBs two-cycle engines, four-cycle engines, and so on to support marketplace opportunities such as lawn mowers, edgers, and small tractors. The Enterprise four-cycle engine LOB may evolve through Product Model #1 and Product Model #2. Each product model builds on its predecessor—*precedented* system—to improve capabilities and performance to meet marketplace needs.

The preceding discussion of a System/Product Life Cycles relate to project management workflow and the development of systems, product, or services. However, life cycle models are not restricted to the workflow. The same analogy applies to Users of system, product, and service. Their Enterprises evolve through similar life cycles. The differences occur when Product Model #1:

1. Fails.
2. Becomes too costly to operate, maintain, and sustain.
3. Is predicted to be vulnerable to system threats.

4. Lacks the specific level of capability or performance to meet predicted Enterprise needs.

Why is this relevant to SE? As a Systems Engineer (SE), you need to understand what:

1. LOB your User is engaged in.
2. Opportunities, problems, or issues your User is chartered to address as part of its LOB. We refer to this as the opportunity space and specific targets as Targets of Opportunity (TOO)
3. Missions your User performs to support the LOB (we refer to this as the solution space)
4. Capabilities your User requires to support solution space missions now and in the future
5. Existing systems, products, or services your User employs to provide those capabilities
6. Deficiency gaps—or opportunities—exist in the current system, product, or service and how you and your Enterprise can cost effectively eliminate those deficiencies with new technologies, systems, products, or services

Based on this knowledge and understanding, the SE's role as a *problem solver–solution developer* becomes crucial. The challenge is: *how do SEs work with Users and Acquirers to*:

1. Collaboratively identify and partition the Opportunity Space into one or more Solution Spaces?
2. Technically bound and specify each Solution Space in terms of capability and performance requirements that are legally sufficient to procure systems, products, and service?
3. Verify that the new system complies with those requirements?
4. Validate that the system developed satisfies the User's original operational needs?

The remainder of this book is intended to answer these questions.

Heading 3.4

As we close this chapter, there is one remaining topic that is paramount to a system's use, its acceptance by the System Stakeholders—Users and End Users. The most elegantly designed system is of limited value if the stakeholders do not want to use it. This brings us to our final topic, System Acceptability.

3.9 SYSTEM ACCEPTABILITY: CHALLENGES FOR ACHIEVING SUCCESS

The degree of success of any Engineered System and its mission(s) ultimately depends on four factors:

- **Success Factor #1**—Marketplace Introduction and Timing
- **Success Factor #2**—System Feasibility and Affordability
- **Success Factor #3**—User Perception of Benefits – Return on Investment (ROI)
- **Success Factor #4**—Total Cost of Ownership (TCO)

The Success Factors listed above are seldom *optimum* simultaneously. Though appearing to be equal, psychologically *subjective measures* tend to take precedence over *objective measures* of system success. Simply stated, system success ultimately comes down to whether the User and Stakeholder decision authorities "like" the system or not and are willing to use and sustain its operation. For example:

- *Subjective* aesthetics include look, feel, and perception - within the peer community often promote User acceptance of a system that may only be partially successful.
- Conversely, the User for the same *subjective* reasons may reject an *objectively* successful system.

Success Factor #1: Marketplace Introduction and Timing

History is filled with examples of systems or products that were delivered to the marketplace prematurely or too late. You can innovate and develop the best widget or electronic mousetrap (Example 2.1). However, if the marketplace is not mentally or skillfully ready for the device or can afford it, your efforts and investments may be futile—timing is critical for User acceptance! The same is true for proposing new systems or capabilities to Users. Users may *want* and *need* a system yet lack sufficient funding (Figure 21.4). In other cases, their funding may be placed "on hold" by decision authorities due to a lack of consensus regarding the maturity of the system definition, understanding of the system's requirements, or technology. For this reason, most Enterprises develop a series of decision "gates" (Figure 3.4) that qualify the maturity of a business opportunity and incrementally increase the level of commitment, such as funding. The intent is to ensure that the *right* system or product solution is introduced at the *right* time for the *right* price and is readily accessible when the User is ready to purchase. Therefore, Enterprises must do their research, collaborate, and work proactively with the Stakeholders – Users and End Users - to ensure that system timing is *right*. This leads to the next point: User system/product feasibility and affordability.

Success Factor #2: System Feasibility and Affordability

If a determination is made that the timing for a system, product, or service is *right*, the next challenge comes

in determining if the system, as currently specified, can be feasibly developed and produced with existing technologies within the planned development and life cycle budget at acceptable risk for the User or System Acquirer.

As an SE, chances are you may be required to provide technical support to business development teams working on a new system or product acquisition. If not, you may be supporting an SE who is. From a technical perspective, the multi-disciplinary SE team is expected to conceptualize, mature, and propose technical solutions to satisfy the system feasibility questions noted in the preceding text.

If you choose to avoid business development support, others within your Enterprise may potentially formulate a *risky* or *undesirable* solution or commitment that you have to live with later. Conversely, if the others solicit engineering support and you choose to ignore them, you may be stuck with the consequences of your own inaction. Therefore, proactively support and technically influence business development activities and decision-making. It's a *win–win* for all stakeholders.

Success Factor #3: User Perception of Benefits – Return on Investment (ROI)

Principle 3.11

User Benefits Principle

Every system, product, or service must provide six benefits to Stakeholders to be considered worthy of their consideration for missions:

- Operational Utility
- Operational Suitability
- Operational Availability
- Operational Usability
- Operational Effectiveness
- Operational Efficiency

The ultimate test of any system, product, or service is its *mission* and *system effectiveness* in performing User missions and accomplishing mission objectives. Failure to perform within prescribed OPERATING ENVIRONMENT conditions and constraints places operational, financial, and survival risks on the Users, their Enterprise, and the public. One of System Engineering's greatest challenges for SEs is being able to translate *operational* and *system effectiveness* objectives into meaningful capability and performance requirements that developers understand and can implement. The challenge is exacerbated by a lack of formal Engineering education and training (Figures 2.11–2.13).

The ultimate test for a system, product, or service resides in its capabilities to produce performance-based outcomes that meet Enterprise and mission objectives. When you develop systems, there are six basic questions the User, Acquirer, and System Developer need to answer:

- **Stakeholder Decision #1 – Operational Utility**—If we invest in the development of this system, product, or service, will it be *operationally useful* to the User in accomplishing their Enterprise missions?
- **Stakeholder Decision #2 – Operational Suitability**—If the system has *operational utility*, will it be *operationally suitable* for the User's mission application(s) and integrate easily into their business model?
- **Stakeholder Decision #3 – Operational Availability**—If the system has *operational utility* and is *operationally suitable* for the application, will it be *operationally available* "on demand" to perform the mission when tasked?
- **Stakeholder Decision #4 – Operational Usability**—If the system has *operational utility* and is *operationally suitable* for the application, will it be *operationally usable* – easy to understand and operate - by its Users without inducing human errors (Chapter 24)?
- **Stakeholder Decision #5 – Operational Effectiveness**—If the system has *operational utility* to the User, is *operationally suitable* for the application, and will be *operationally available* to perform its mission, will it be *operationally effective* in accomplishing mission objectives?
- **Stakeholder Decision #6 – Operational Efficiency**—If the system has *operational utility* to the User, is *operationally suitable* for the application, will be *operationally available* to perform its mission, and have operational effectiveness in achieving mission objectives, will it be *operationally efficient* to operate?

Let's explore each of these decisions further.

3.9.1 Operational Utility

Principle 3.12

Operational Utility Principle

Every system, product, or service must be *operationally useful* to enable its User to C2 the system and perform Situational Assessments with the least number of human errors.

Users expect systems and products to have a level of *operational utility* that enables them to accomplish the Enterprise missions and achieve the stated goals and objectives. These are nice words, but what does *operational utility* really mean?

A system, product, or service having *operational utility* is one that:

- Is the right solution for use in Enterprise or organizational missions.
- Achieves mission outcome(s) and objectives.

So, if a system satisfies these operational utility criteria, *how do we determine operational suitability?*

3.9.2 Operational Suitability

Principle 3.13

Operational Suitability Principle

Every system, product, or service must be *operationally suitable* for the User's mission application—the right tool for the job.

Operational suitability characterizes *how well* a system or product:

- Suits a User's specific application in a given Operating Environment and conditions—that is, the *right* system, product, or service for the job or task to be performed.
- Integrates and performs within the User's Enterprise systems.
- Does not pose any *unacceptable* safety, environment, or health hazards or risks to its operators, the public, or environment

Some systems and products may have *operational utility* for some applications but simply are not *operationally suited* to a specific User's intended application and Operating Environment. Consider the following example:

Example 3.10

Operational Suitability Example

From a transportation perspective, vehicles such as automobiles may have *operational utility* to a User commuting to work and transporting children to school. However, if the User plans to use the vehicle Off-the-Road in a rugged, harsh environment, only specific types of vehicles may be *operationally suitable* for that type of mission application. If the User intends to carry heavy loads, only specific types of trucks may be *operationally suitable* for that type of mission application.

3.9.3 Operational Availability

Principle 3.14

Operational Availability Principle

Every system, product, or service must be *operationally available* on demand to perform missions when required by its User.

Operational availability means that the system, product, or service is capable and operationally ready "on demand" to perform a mission when tasked. *Operational availability* becomes a critical metric for assessing the level of *operational readiness* to perform missions. System availability is a function of system reliability and maintainability (Chapter 34). Consider the following example:

Example 3.11

Operational Availability Example

When an emergency situation such as an accident or fire occurs, 911 calls to police, fire departments, and emergency medical responders test the respective Enterprise's system *availability* – Personnel, Equipment, and so forth - to respond to emergencies and disasters in the least amount of time. "On demand" timing is crucial in life threatening situations. The same is true when you crank your automobile to travel to work.

3.9.4 Operational Usability

Principle 3.15

Operational Usability Principle

Every system, product, or service must be *operationally usable* - easy to understand and operate – according to the User's *mental models*, knowledge, and skill levels without inducing human errors that affect mission or system performance.

Engineers can design and develop a system, product, or service that possesses all the right technical operational attributes – operational utility, suitability, availability, effectiveness, and efficiency – based on their mental models. However, it is the User who must be comfortable in using the system, product, or service. If the System Design human interfaces – displays, comfort, ease of operation, etc. - do not comply with the User's mental models (Chapter 24) and skill levels thereby minimizing human error, it has limited or no value. Learn to think based on the User's mental model, not the Engineering designer's mental model.

3.9.5 Operational Effectiveness

Principle 3.16

Operational Effectiveness Principle

Every system, product, or service must be *operationally effective* in producing the required mission outcome(s).

Enterprises and Users are chartered with specific goals, missions, and objectives. For example:

- A new vaccine is 99% effective in eliminating a specific type of virus.

- New surgical devices (Figure 25.9) enable surgeons to perform minimally invasive surgery resulting in faster patient recovery times.

If the system, product, or service does not achieve or is only marginally *operationally effective*, it is of limited or no value to the User.

3.9.6 Operational Efficiency

Principle 3.17

Operational Efficiency Principle

Every system, product, or service must be *operationally efficient* in delivering the required mission outcome(s) in the least amount of time and cost.

You can develop the best system, product, or service that has *operational utility, suitability, availability, and effectiveness*; however, if it is not operationally efficient in terms of *cost effectiveness*, it has limited value. For example, if a system, product, or service is developed but is simply *unaffordable* for the User to purchase, operate, and maintain, it is totally useless to the User.

Success Factor #4: Total Cost of Ownership (TCO)

Developing a system, product, or service that has operational utility, suitability, availability, effectiveness, and efficiency focuses on system, product, or services outcomes and objectives. However, *what if the Total Cost of Ownership (TCO) of a system, product, or service's lifecycle is simply unaffordable*? For example, you might be able to splurge and afford a luxury automobile to rent for a short family vacation; however it would be *unaffordable* to purchase and too expensive to operate and maintain over a period of ownership of a few years. The concept of life cycle cost emerged during the 1960s when the US DoD began to recognize the significance of operations and support costs relative to system acquisition costs.

People are often surprised to learn that approximately 70% of the TCO for a system occurs during the System OM&S Phase of its life cycle. As an example, Eisenberger and Lorden (1977, p. 103) cite OM&S as 72% of the TCO. Dallosta and Simcik (2012, p. 35) indicate costs for military systems over a 30+ year life cycle have probability frequency distributions as follows:

- System Acquisition Costs account for 20% to 40% of TCO.
- Operations and Support Costs account for 60% to 80% of TCO.

What is interesting about the OM&S metric is the nominal value continues to hover around 70% of TCO based on two different sources over a 35-year period—Eisenberger and Lorden (1977, p. 103) and Dallosta and Simcik (2012, p. 35).

3.10 CHAPTER SUMMARY

During our discussion in this chapter, we introduced the key concepts that define a system, product, or services attributes, properties, and characteristics. Key topics included:

- Systems interact with themselves and with external systems within their Operating Environment (Figure 3.1 and 9.2).
- System Stakeholders consist of System Users that C2 a system or product and End Users that benefit from the mission outcomes of a system:
  - Mission System and Enabling System (Chapter 3) operators are defined as *Users* of a system if they Command and Control (C2) the system, product, or service to achieve mission outcomes and objectives.
  - Mission System and Enabling System operators (Chapter 3) are defined as *End Users* if they derive benefits from the system such as comfort – environmental control, lighting, seating, or entertainment; Situational Assessment information such as system performance – speed, battery voltage, oil pressure, and so forth.
- System attributes, properties, and characteristics enable us to characterize the unique identity for system, product, or service.
- Systems have a *state of equilibrium* and *stability* within their Operating Environment that determines their position in the balance of power and ultimately survival.
- A System/Product Life Cycle model depicts the *stages* or *phases* of evolution of a system from conception through disposal.
- In the eyes of its owners and Users, *affordability; operational utility, suitability, usability, availability, effectiveness, and efficiency; and TCO* are major drivers that ultimately determine system acceptability and success.

3.11 CHAPTER EXERCISES

3.11.1 Level 1: Chapter Knowledge Exercises

1. What is a SOI?
2. What is a capability?
3. What is the difference between a *capability* and a *function*?
4. What is the top level system analytical construct?
5. What do systems interact with?
6. What is a system *attribute*?

7. What is a system *property*?
8. What is a system *characteristic*?
9. What makes a system, product, or service unique?
10. What influences a system and its results?
11. What are some types of system characteristics?
12. What constitutes a system's state of equilibrium and stability?
13. What is a system life cycle?
14. From an SE perspective, what criteria should a life cycle meet?
15. What is the System Definition Phase, when does it start, and when does it end?
16. What is the System Procurement Phase, when does it start, and when does it end?
17. What is the System Development Phase, when does it start, and when does it end?
18. What is the System Production Phase, when does it start, and when does it end?
19. What is the System Operations, Maintenance, and Support (OM&S) Phase, when does it start, and when does it end?
20. What is the System Retirement and Disposal Phase, when does it start, and when does it end?

3.11.2 Level 2: Knowledge Application Exercises

Refer to www.wiley.com/go/systemengineeringanalysis2e

3.12 REFERENCES

ANSI/IEEE 649-1998 (1998), *National Consensus Standard for Configuration Management*, Electronic Industries Alliance (EIA).

Boynton, Richard and Wiener, Kurt (2000), *How to Calculate Mass Properties*, Berlin, CT: Space Electronics, Inc.

Dallosta, Patrick M. and Simcik, Thomas A. (2012), Designing for Supportability, Defense AT&L: Product Support Issue, March – April 2012. Accessed 3/10/13 http://www.dau.mil/pubscats/ATL%20Docs/Mar_Apr_2012/Dallosta_Simcik.pdf

DOE (2007), *Stage-Gate Innovation Management Guidelines*, Version 1.3, Figure 3, p. 3, Industrial Technologies Program, Washington, DC: Department of energy (DOE). Accessed 5/19/14 from http://www1.eere.energy.gov/manufacturing/financial/pdfs/itp_stage_gate_overview.pdf.

Eisenberger I. and Loreden G. (1977), DSN Progress Report, *Life Cycle Costing: Practical Considerations*, NASA JPL, May and June 1977. Accessed 3/10/13 http://ipnpr.jpl.nasa.gov/progress_report2/42-40/40M.PDF

European Union (2001), *the European Council Directive on General Product Safety 2001/95/EC*.

ISO/IEC 15288: (2008). *System Engineering -- System Life Cycle Processes. International Organization for Standardization (ISO)*. Geneva, Switzerland.

ISO/IEC/IEEE 24765:2010 (2012), *Software and Systems Engineering Vocabulary*, New York, NY: IEEE Computer Society. Accessed on 5/19/14 from www.computer.org/sevocab.

Prados John W. (2007), *75th Anniversary Retrospective Book: A Proud Legacy of Quality Assurance in the Preparation of Technical Professionals*, Baltimore, MD: Accreditation Board for Engineering and Technology (ABET).

Rogers, Everett M. (1962), *Diffusion of Innovations*, Glencoe, NY: Free Press (now Simon and Schuster) US Consumer Product Safety Improvement Act of 2008.

4

USER ENTERPRISE ROLES, MISSIONS, AND SYSTEM APPLICATIONS

Engineers graduate every year with Engineering degrees and go to work in industry and government. They begin their careers within projects developing systems for which they typically have little or no System Engineering education and training as discussed in Chapter 2.

- If you ask engineers an open-ended question: "where do your requirements originate?" most will respond with a project-centric protocol and say "from our Contracts organization."
- If you ask them where the Contracts organization gets the requirements, they will respond "from our customers."
- If you ask them where their Customers get their requirements, they will respond with "they write some requirements in a specification and send it out with a Request for Proposal (RFP)"

If you sift through these responses, you begin to observe that most engineers do not understand how their system requirements originate and perceive that the Customer sits down one day with a word processor and writes a few requirements for a new system "off the top of their head."

The reality is: Customers do not magically decide one day to acquire a new system, product, or service and start writing requirements. Unfortunately, this is sometimes the case for those who do not fully understand how to develop systems.

Consider a heavy construction company that owns massive haulers to carry rocks, bulldozers, front-end loaders, and water trucks. *Do you think the owner arbitrarily decides to procure these types of vehicles simply because they have a fascination with vehicles*? Absolutely not! These vehicles exist because the organization has a business mission to support its customers. The organizational mission requires performance of a diversity of mission tasks, each requiring a different type of system—vehicle—to produce outcomes that contribute to the organization's overall business mission.

Chapter 4 introduces ENTERPRISE ROLES, MISSIONS, AND SYSTEM APPLICATIONS Concepts. Our discussions and topical sequences include the following:

- User Enterprise Roles and Missions
- Duality of Enterprise Roles as MISSION SYSTEMS and SUPPORT SYSTEMS
- Problem, Opportunity, and Solution Space Concepts
- Evolution of System Capabilities and Requirements

4.1 DEFINITIONS OF KEY TERMS

- **Countermeasure**—An operational capability or tactic employed by a system to camouflage its identity, deceive or defeat adversarial or hostile system's capabilities, or minimize vulnerability by protecting itself from unauthorized access.
- **Counter-Countermeasure (CCM)**—An operational capability or tactic employed by a system to *neutralize* another system's threats or countermeasures.
- **Mission**—A purposeful action or task directed toward accomplishing a specific objective-based outcome and level of performance.

System Engineering Analysis, Design, and Development: Concepts, Principles, and Practices, Second Edition. Charles S. Wasson.

Companion Website: www.wiley.com/go/systemengineeringanalysis2e

- **Mission Needs Statement (MNS)**—A general description of the operational capabilities required for a new system, product, or service or upgrade to meet mission requirements.
- **Mission Objectives**—Performance-based outcomes to be achieved by a mission within a specified timeframe and operating constraints.
- **Mission Profile**—A time phased description of operational events and environments an item experieces from beginning to end of a specific mission. It identifies the tasks, events, durations, operating conditions and environment of the system for each phase of a mission. (MIL-HDBK-1908B, p. 23).
- **Opportunity Space**—A gap or vulnerability in a system, product, or service capability that represents an opportunity for (1) a competitor or adversary to exploit or (2) a supplier to offer solutions.
- **Problem Space**—An abstraction within a system's Operating Environment that represents an actual, perceived, or evolving gap, hazard, or threat to an existing capability. The potential threat is perceived either to pose some level of financial, security, safety, health, or emotional risk to the User or to have already had an adverse impact on the individual or Enterprise and its success. One or more lower level Solution Space systems, products, or services resolve the problem space.
- **Problem Statement**—A brief, concise, statement of fact that clearly describes an undesirable event, issue, state, or condition without identifying the source or actions required to solve the problem.
- **Situational Assessment**—An objective evaluation of current Strengths, Weaknesses, Opportunities, and Threats (SWOT) of a System of Interest (SOI) relative to its operating conditions and outcome-based objectives. Results of a situational assessment document the prioritized mission operational needs for the organization.
- **Solution Space**—A bounded abstraction that represents a capability that, when implemented, is intended to satisfy all or a portion of a higher level Problem Space.
- **Statement of Objectives (SOO)**—A statement of User performance-based objctives to be achieved by a mission, system, product, or service.
- **Strategic Plan**—An outcome-based, global or business domain document that expresses an Enterprise's vision, mission, and objectives of: (1) *where* it wants to be at some point in time and (2) *what* it wants to accomplish in the long term, typically five years or more hence. The challenge for most organizations is: What business or Line of Business (LOB) are you currently in *versus* what do you want to be in five years from now *versus* what LOB you should be in?
- **Strategic Threats**—External systems that have long-term plans to exploit opportunities that leverage or enhance an Enterprise's reputation or equity to achieve a long-term vision and upset the "balance of power." For example, an Enterprise has a long-term vision to predominate a software market.
- **System Adaptation**—The ability of a system to acclimate *physically* and *functionally* to a new Operating Environment with a minimal degree of degradation to its capabilities.
- **System Threat**—An external entity that has the potential to *cause* or *inflict* varying degrees of harm on another entity and its mission, capabilities, or performance. A system threat is any interaction by an external system that is *hostile* or adversarial and impedes the operation and performance of your system in accomplishing its intended mission.
- **Tactical Plan**—A near-term, mission-specific plan that expresses how the Enterprise's leadership approach to deploy, operate, and support existing assets—such as people, products, processes, and tools—to achieve organizational objectives allocated from the strategic plan within timeframe and resource constraints, typically one year or less.
- **Tactical Threats**—External systems that pose a potential, short-term hazard to another organization or system and its mission. For example, counter a competitor's advertising campaign.

4.2 APPROACH TO THIS CHAPTER

Principle 4.1

Customer Needs Principle

Success in providing systems, products, or services solutions to a highly competitive, global marketplace requires two levels of knowledge:

- Understanding the operational needs of your Users—customers.
- Understanding what the User's customer expects of them.

Chapter 4's title may appear to be about as far from Engineering as one could imagine. *What do User Enterprise Roles, Missions, and System Applications have to do with creating a system design and selecting components*? The reality is that systems, products, or services exist because a User has an operational need to be filled on an Enterprise or individual consumer basis, accomplish one or more

performance-based outcomes, and receive some form of Return on Investment (ROI) if they invest resources to procure it. This could be a consumer searching for a new tablet computer, smartphone, Enterprise updating its accounting system, a space agency planning a mission to Mars, and so forth.

Success in developing systems, products, or services for Enterprises and requires that you understand the *who*, *what*, *when*, *where*, and *how* they will purchase. For Enterprises making major capital investment expenditures, the process often begins years in advance through some form of budgeting process. When the decision is made to acquire a new system or upgrade to an existing one, you need to understand how to satisfy those needs, set a competitive price, and then deliver on your commitments. But the process does not begin here. You need to understand what is motivating or driving your customers' operational needs based on what their customers expect of them.

Our discussion begins with an introduction to the types of Enterprises that acquire systems, products, or services. Every Enterprise, irrespective of its business domain, serves as an SOI and performs two roles: (1) a MISSION SYSTEM role that produces physical products or performs services and (2) an ENABLING SYSTEM role that delivers those systems, products, or services to meet the operational needs of their customers—the Users. However, Users have to satisfy the operational needs of their customers—the End Users—that benefit from the systems you and your organization develop for the marketplace.

To meet the operational needs of your customers, you need to understand how Enterprises assess their operational needs to identify "gaps" in their organizational or system, product, or service capabilities and how they intend to eliminate the gap. Each gap effectively becomes a Problem Space that requires one or more Solutions Spaces, each of which becomes its own contextual Problem Space to be decomposed into lower level Solution Spaces at lower levels of abstraction and so forth.

We conclude Chapter 4 with a discussion of the need for an Enterprise to time the development of new systems, products, or services to ensure delivery prior to the emerging capability gap growing to a stage in which the business becomes vulnerable to competitive or adversarial threats and ceases to exist.

4.3 USER ROLES AND MISSIONS

Principle 4.2

System Existence Principle

Every system, product, or service has a purpose and exists for the benefit of performing missions for its Stakeholders—User(s) and End User(s). Failure of either one or both represent system *obsolescence* leading to system retirement and disposal.

Every HUMAN SYSTEM—Enterprise and Engineered—has a purpose or reason for its existence: Enable its User(s) to accomplish their performance-based mission outcome(s) and supporting objectives. As a result, every type of Enterprise and Engineered System serves at the pleasure of its Users in achieving their missions. When a system: (1) no longer has a mission; (2) becomes cost prohibitive to operate, maintain, or sustain (OM&S); or (3) is no longer *efficient* or *effective* in accomplishing Enterprise performance-based mission outcomes, it serves no one and no longer provides value. As a result, it is decommissioned and disposed.

This section introduces the concept of Enterprise roles and missions, system roles and Stakeholders—Users and End Users—Enterprise capability gaps, and Problem/Opportunity–Solution Spaces. We explore the roles of Enterprises that employ systems, products, and services and how physical systems—namely, assets—are acquired to perform in support of Enterprise System roles, missions, and objectives.

The success of HUMAN SYSTEMS—Enterprise and Engineered - in achieving success is determined by *how well* the system is specified, designed, developed, integrated, verified, validated, operated, supported, and sustained. This requires that System Stakeholders—Users and End Users - have a vested interest in the *operational* and *cost-effectiveness* of the mission results. We conclude Chapter 4 by identifying the primary system stakeholder roles and their contributions, sometimes *positive*—sometimes *negative*—to system mission performance and outcomes.

4.3.1 User and End User Organizational Roles and Missions

User and End User Enterprises and their systems perform roles that reflect their chartered missions and objectives. Table 4.1 provides examples of system roles and missions. Let's explore the context of organizational roles further.

4.3.1.1 Enterprise System Roles Context

Principle 4.3

The Customer and Customer's Customer Principle

To fully understand and support your customer as a System User, you must understand what their customers—the System's End Users—expect from them and the enabling role your system, product, and service contributes to the accomplishment of those results.

TABLE 4.1 Example System Roles and Missions

System	Role-Based Mission Description
Legislative	Establish societal compliance guidance and constraints in the form of local, state, and federal laws, statutes, regulations, ordinances, and policies that govern individuals, organizations, or Enterprises
Judicial	Adjudicate individual, organizational, or Enterprise compliance with established laws, statutes, regulations, ordinances, and policies
Military	Perform cooperative, emergency, peacekeeping, deterrence, and wartime roles that ensure the survival of a country and protect its constitution, security, and sovereignty
Transportation	Provide transportation services that enable Users, customers, and products to move safely and efficiently from one location to another by land, sea, air, or space or combinations of these
Civic	Perform public services that promote the goals and objectives of a community-based organization
Educational	Provide educational opportunities for people to gain specialized knowledge and enhance their skills to prepare them for becoming contributing members of society
Medical	Provide medical consultation, therapy, diagnostic, surgical, and treatment services
Resource	Provide resources (e.g., time, money, fuel, electricity) commensurate with performance and risk to support the missions, goals, and objectives of an individual, organization, or Enterprise with an expectation of a Return on Investment (ROI)
Producer	Produce large or mass quantities of a system design or product in accordance with requirements and standards for the marketplace
Construction	Provide construction services that enable system developers to implement facilities or sites that enable Users to deploy, operate, support, train, and dispose of systems
Agricultural	Provide nutritional food and agricultural by-products to the marketplace that are safe for human and animal consumption and safe for the environment
Food	Serve customer daily food consumption needs such as grocery stores, markets, restaurants, distributors
Public utilities	Provide community water, sewer, refuse pick-up, electrical power, natural gas, communication and other services
Retail or wholesale business	Supply consumer *products* and *services* to the marketplace
Consulting and Technical Services	Monitor the performance of other systems, evaluate the performance against established standards, record objective evidence, and control system performance
Research and Development	Investigate the research and development, productization, or application of new technologies for systems

Donne (1623) in his poem "No Man Is an Island" exemplifies Enterprise and Engineered Systems and their dependence on others. SEs must understand their User's/customer's needs to successfully develop the systems, products, or services required. Implicit in this point is recognition that Enterprise and Engineered Systems are dependent on unbroken "supply chains." Each step of the supply chain is required to deliver systems, products, and/or services that comply with Fitness for Use Criteria such as specifications to ensure their existence and survival. Let's explore this point further.

4.3.1.2 Understanding Enterprise Supply Chain Roles

Principle 4.4

Dual Producer–Supplier Roles Principle

Every system, product, or service performs two contextual roles: a MISSION SYSTEM (Producer) role and an ENABLING SYSTEM (Supplier) role.

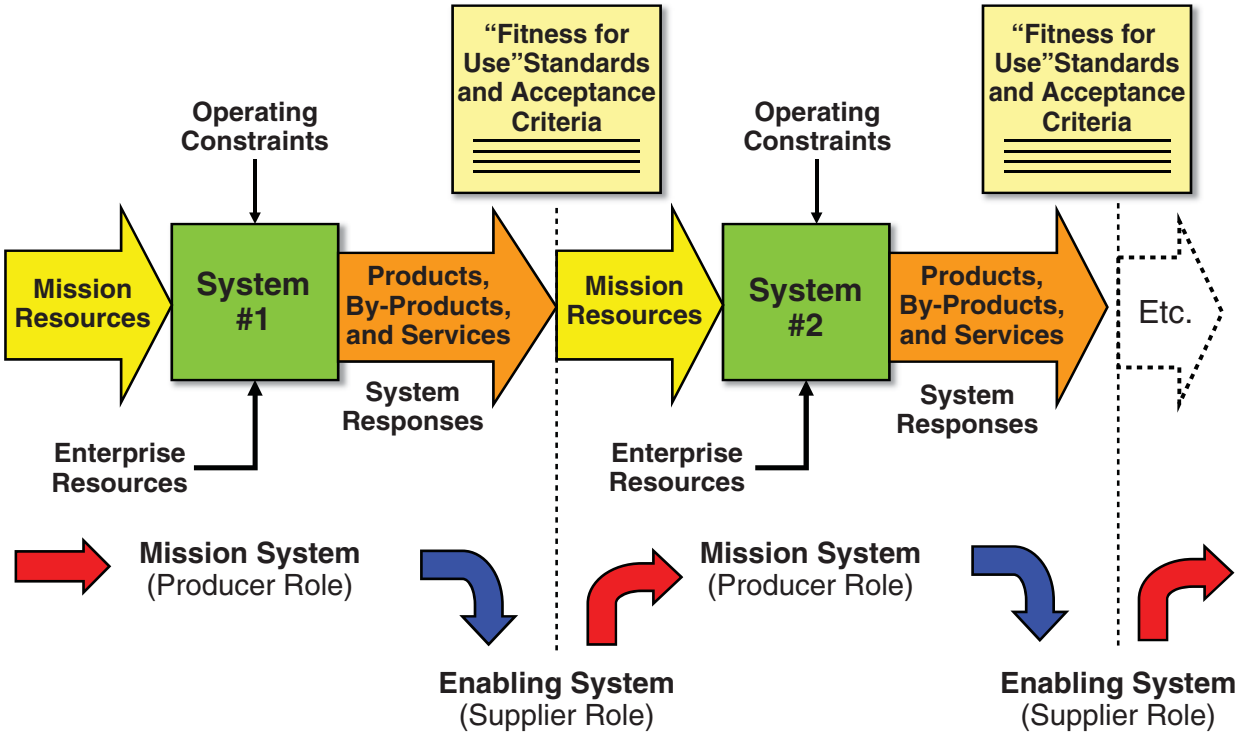


Figure 4.1 Understanding the MISSION SYSTEM (Producer) and ENABLING SYSTEM (Supplier) Roles

HUMAN SYSTEMS—Enterprise and Engineered—consist of integrated supply chains of systems in which an Enterprise produces systems, products, and services to support another "downstream" system. Figure 4.1 provides an illustration. Two key points are:

1. A MISSION SYSTEM (Producer Role) exists to perform and accomplish specific objectives defined by contract, tasking, or personal motivation and produces systems, products, by-products, and/or services as performance-based outcomes.
   This leads to a follow-up question: *Who benefits from accomplishment of these objectives*? This leads to the second contextual role.
2. Each MISSION SYSTEM (Producer Role) serves as an ENABLING SYSTEM (Supplier Role) to its Users performing their MISSION SYSTEM roles.

If we investigate the tandem roles of the MISSION SYSTEM and ENABLING SYSTEM(s) every Enterprise and Engineered System SOI—such as divisions, departments, subsystems, assemblies, subassemblies, and parts - provides capabilities that produce value-added products, by-products, and/or services required for their Users' SOIs. Figure 4.1 illustrates the dual role supply chains.

Author's Note 4.1

Value-Added Processing

The term "value-added processing" emerged from the 1980's. Due to its overuse or misapplication in marketing literature, the phrase became a meaningless cliche. Every process within a system, product, or service must produce performance-based "outcomes", which is *necessary*. However, even the term "outcome" tends to obscure what is being produced thereby making it an *insufficient* condition. Why? Every process and step within the process must "add value" to the preceding step. If it doesn't, the process or step should be eliminated. Remember—the term "process" has a human administrative and manufacturing connotation. The same is true for the human operator interaction with an EQUIPMENT Element such as a car. Model-Based Systems Engineering (MBSE) that represent logic and computations in a system is focal point for eliminating non-value added processes.

System #1 performs a MISSION SYSTEM role as a Producer of value-added products, by-products, and services to meet consumer marketplace or contract requirements. The marketplace or contract establishes *Fitness-for-Use* standards and *acceptance criteria* to meet the needs of System #2. As a Supplier of products, by-products, and services, System #1

serves as an ENABLING SYSTEM to System #2 performing its MISSION SYSTEM role.

4.3.1.2.1 "Fitness-for-Use" Standards and Acceptance Criteria

System Input and Output (I/O) Fitness-for-Use Principle

Principle 4.5 Every system Input/Output (I/O) must comply with pre-defined *fitness-for-use* performance standards and acceptance criteria established by its Stakeholders - Users and End Users.

Customers, Users, or System Acquirers of systems, by virtue of *validated* operational needs, have *minimum* requirement thresholds and expectations that must be met to ensure the products, by-products, and services they acquire are "acceptable for use." *Acceptable* in terms of technical capability, quality, and safety and are not detrimental to the environment, human safety, and health. To be considered for acceptance as completion of a contract or task, a deliverable system, product, or service is required comply with Fitness for Use standards specified in a contract, specification, and so forth.

Observe that Fitness for Use standards also apply to system inputs and system outputs. Figure 3.2 illustrated this concept as *Acceptable and Unacceptable Inputs* and *Acceptable and Unacceptable Outputs*.

4.3.1.2.2 System Producer–Supplier Relationships The construct depicted in Figure 4.1 represents the fundamental Producer–Supplier Supply Chain relationships. If we analyze each system within the Supply Chain, we discover that every system has mission objectives to achieve. The mission objectives focus on performance-based outcomes—behavior, products, by-products, and services—provided by an SOI to satisfy customer operational needs and provide an ROI to the supplier's Enterprise Stakeholders. Therefore, a system fulfills two roles:

1. A MISSION SYSTEM role to produce value-added products, by-products, and services.
2. An ENABLING SYSTEM role to deliver those products, by-products, and services to other systems.

4.3.1.2.3 Organizational Mission System (Producer Role)

MISSION SYSTEM (Producer Role) Principle

Principle 4.6 In performing its MISSION SYSTEM (Producer Role), every System performs missions or tasks to produce performance-based outcomes (e.g., systems, products, by-products, services, or combinations of these) to benefit its Users and their End Users.

MISSION SYSTEM roles are performed by HUMAN SYSTEMS—Enterprise and Engineered—that are assigned specific missions to produce systems, products, and/or services outcomes and deliverables that comply with mission objectives. Consider the following example.

NASA's Space Shuttle as a MISSION SYSTEM (Producer Role)

Example 4.1 As a MISSION SYSTEM (Producer Role), NASA's Space Shuttle performed space operations to accomplish mission outcome-based objectives—deploy satellites; conduct scientific experiment missions; and ferry astronauts, food, supplies, and refuse to and from the International Space Station (ISS).

4.3.1.2.4 User Organizational ENABLING SYSTEM (Supplier Role)

ENABLING SYSTEM (Supplier Role) Principle

Principle 4.7 As an ENABLING SYSTEM (Supplier Role), every System *delivers* products, by-products, or services to meet the needs and Fitness for Use standards of its Stakeholders - Users and End Users - performing their MISSION SYSTEM Roles.

ENABLING SYSTEMS—Enterprise and Engineered—ensure that its User and End User operations are sustained to perform missions. Consider the following examples.

NASA's Space Shuttle Supplier Role as an ENABLING SYSTEM

Example 4.2 As an ENABLING SYSTEM (Supplier), NASA's Space Shuttle deployed satellites to space for Users, collected data from experiments for scientific investigators, and transported astronauts and cargo between the Kennedy Space Center (KSC) and the ISS performing its MISSION SYSTEM Role.

Aircraft Systems as ENABLING SYSTEMS

Example 4.3 As an ENABLING SYSTEM (Supplier), an aircraft safely and comfortably transports passengers performing their MISSION SYSTEM roles and cargo between two airports.

In support of the aircraft performing its MISSION SYSTEM (Producer) Role, ENABLING SYSTEMS such as baggage handlers, mechanics, ticket and gate agents, Ground Support Equipment (GSE), and others perform their own MISSION

SYSTEM (Producer) Roles to prepare the aircraft for a safe flight, replenish expendables and consumables, and load or unload cargo and passengers.

4.3.1.3 Why User Enterprise and Engineered System Roles Are Important to SE You may be asking *why* and *how* Enterprise and Engineered System roles are important to SE and the Engineering of systems. Physical systems, such as hardware, software, and courseware, exist because higher level Enterprise Systems, such as Users, employ and leverage physical system capabilities to achieve organizational goals, missions, and objectives within budgetary cost and schedule constraints.

As an SE, recognize, understand, and appreciate *how* the User intends to deploy and employ a system in a prescribed OPERATING ENVIRONMENT. Ask yourself: *How does a system, product, or service contribute to an organization's transportation role*? For example, if you are in the airline business, you provide or contract for: reservation and ticketing services, check-in and baggage handling, aircraft, gate facilities, special services, or security. All these require physical systems, as well as integrated hardware and software, to perform the Enterprise's role. The airline has the options to:

1. Develop a system, product, or service.
2. Procure the system, product, or service from external vendors.
3. Outsource (e.g., contract, lease) for the systems, products, or services.

In any case, each Enterprise organizational element such as a division, department, and so forth is allocated goals, objectives, missions, or performance requirements that contribute to achieving the element's mission and objectives.

When an organization, such as an airline, initiates operations, large numbers of personnel must be an integral part of the planning, implementation, Operation, Maintenance, & Sustainment (OM&S) activities. Each Stakeholder—pilots, flight attendants, gate attendants, baggage handlers, and food caterers—has a contribution and vested interest in the performance-based outcomes and successes of the airline's role and its embedded systems.

4.3.1.4 MISSION SYSTEM–ENABLING SYSTEM Supply Chain Relevance to Engineering The Producer–Supplier Supply Chain concept may leave the impression that it applies only to Enterprises. *Not true*! Producer–Supplier Supply Chains apply to every aspect of engineering and its disciplines such as processes. Observe how Enterprises and their Engineered Systems serve as MISSION SYSTEM and ENABLING SYSTEM Roles in the examples below:

- **Electronics Design Supply Chain**—As a MISSION SYSTEM (Producer), a digital circuit designer transforms a set of specification requirements into a physical device such as a circuit board that has been engineered to produce the specified outputs and characteristic responses for a given set of inputs and OPERATING ENVIRONMENT conditions. As an ENABLING SYSTEM (Supplier), the device's output is routed via cables or wiring to downstream electronics to other internal/external systems for further processing.
- **Mechanical Design Supply Chain**—As a MISSION SYSTEM (Producer), a mechanical engineer designs a structure to support installation of User systems to operate under varying wind loads and OPERATING ENVIRONMENT conditions. As an ENABLING SYSTEM (Supplier), the mechanical designer collaborates with the Users to ensure that the structure produced will accommodate their MISSION SYSTEM operational needs.
- **Software Design Supply Chain**—As a MISSION SYSTEM (Producer), a software application consists of an algorithm that computes results for a given set of inputs and conditions. As an ENABLING SYSTEM (Supplier), the software application stores results in a memory location for subsequent retrieval by other software applications.
- **Chemical Process Design Supply Chain**—As a MISSION SYSTEM (Producer), a chemical is used as a catalyst to produce an outcome—a reaction—under specified conditions.

Heading 4.1 Based on the introduction of the MISSION SYSTEM–ENABLING SYSTEM Supply Chain concept leads to a need to discuss *accountability* for each SOI and its Mission System(s) and Support System(s) roles. Each of these has its own sets of Stakeholders - Users and End Users - that are accountable for SOI performance or application of its performance-based outcomes.

4.3.2 Stakeholder User and End User Roles

Engineered Systems, from conception through retirement, require some level of human operation, intervention, and support, either directly and indirectly. Stakeholders—Users and End Users—with vested interests in a system, product, or service expect to contribute to its conceptualization, funding, procurement, design, development, integration, operation, support, and retirement of every system. Depending on the size and complexity of the system, including risks and importance to the User, Stakeholder roles may be performed by an individual, an organization, or some higher level Enterprise. Consider the following examples of System Stakeholder roles:

Example

- System Advocate or Proponent

- System Shareholder
- System Administrator
- System Owner
- System User(s)
- System End User(s)
- System Architect
- System Acquirer
- System Developer
- Services Provider
- Independent Test Agency (ITA)
- Mission Planner
- System Analyst
- System Support
- System Maintainer
- System Instructor
- System Critic
- System Competitor
- System Adversary
- System Threat

Let's introduce and define each of these roles. Table 4.2 provides a brief description of each stakeholder role.

The context of stakeholder roles depends on an SOI's context. Recognize that the personnel—operators and maintainers, organizations, or Enterprises that perform these roles may also be Stakeholders and have different roles relative to other systems. As an example, the System Advocate for one system may serve as a System Owner for several other systems.

4.4 UNDERSTANDING AND DEFINING USER MISSIONS

User Enterprise roles, missions, and objectives establish the driving need for mission and system capabilities and performance requirements. Each role, mission, and objective serves as the benchmark frame of reference for scoping and bounding *what is* and *is not* relevant as an organization's mission.

Understanding the problem, issue, or opportunity the User is attempting to solve, resolve, or exploit is key to understanding *why* a system exists and *what* purpose it serves within the System Owner's Enterprise. Let's briefly explore some of the types of Enterprise Missions.

4.4.1 Types of Enterprise Missions

Organizations conduct various types of missions that require assets such as systems, products, or services to fulfill mission objectives. High level types of missions include:

- Educational
- Humanitarian
- Medical
- Transportation
- Government
- Educational
- Delivery Services

Military organizations, for example, perform missions such as:

- Search and Assist
- Search and Rescue
- Search and Retrieve or Recover
- Search and Destroy

The key point here is to fully understand the Enterprise purpose, roles and missions, performance-based outcome objectives, level of urgency, resource and time constraints, and windows of opportunity. Although these points may seem bland, they serve as critical *decision points* for the development of systems, products, or services addressed in the remainder of this text. Let's briefly explore each one.

4.4.2 Enterprise Strategic Planning for Engineered Systems

From an Enterprise perspective, we model the strategic and tactical planning process as illustrated in Figure 4.2. In general, the process consists of a Strategic Planning Loop and a Tactical Planning Loop. These two loops provide the basis for our discussion.

4.4.2.1 The Strategic Planning Loop The seed for long-term Enterprise growth and survival begins with an organizational vision. Without a vision and results-oriented plan for action, the organization's founders would be challenged to initially or continually attract and keep investors, investment capital, and the like.

4.4.2.2 The Enterprise's Operating Environment The path forward for most organizations begins with a domain analysis of the OPERATING ENVIRONMENT consisting of Targets of Opportunity (TOOs) and threat environment. The analysis task, which is scoped by the organizational vision, produces a *Market and Threat Assessment Report*. The report, coupled with the long-term organizational vision of what is to be accomplished, provides the basis for developing the organization's Strategic Plan.

As the organization's capstone planning document, the Strategic Plan defines where the Enterprise expects to be five

TABLE 4.2 System Stakeholder Role Definitions

Role	Role Description
System Advocate or Proponent	An individual, organization, or Enterprise that champions the system's cause, mission, or reason for existence The System Advocate may derive tangible or intangible benefits from their support of the system or they may simply believe the system contributes to some higher level cause the System Advocate supports
System Shareholder	An individual, organization, or Enterprise that "owns" all or equity shares in the system and its development, operation, products, and by-products—either directly or indirectly
System Owner	An individual, organization, or Enterprise that is legally and administratively responsible and accountable for the system; its development, operation, products, and by-products; outcomes; and retirement
System User(s)	An individual, organization, or Enterprise that operates, commands and controls (C2) a system or provides inputs—data, consumables and expendables, raw materials, or pre-processed materials. As a result of system usage under their control, Users control system outputs to provide results such as information, data, reports, and End Users to perform their tasks or make decisions. Users may require proficiency training and possibly certification
System End Users	An individual, organization, or Enterprise that derives direct *benefits* directly or indirectly from a system and/or its products, services, or by-products. End Users may or may not require training
System Acquirer	An agent or agency selected by the User to serve as their acquisition and technical representative to: 1. Define and specify the system 2. Select a System Developer or Services Provider 3. Provide technical assistance to assess System Developer or System Service Provider performance, progress, maturity, status, and risk 4. Provide contractual oversight for the execution of the contract and delivery of a verified and validated system to the User
System Developer	An individual, organization, or Enterprise responsible for developing and delivering a *verified* system solution based on operational capabilities and performance bounded and specified in a System Performance Specification (SPS)
System Architect	An individual, organization, or Enterprise that visualizes, conceptualizes, and formulates the system, system concepts, missions, goals, and objectives. Since SE is viewed as multi-disciplined, the system architect role manifests itself via hardware architects, software architects, and instructional architects
Services Provider	An individual, organization, or Enterprise chartered or contracted to provide services to operate the system or support its operation
Independent Test Agent/Agency (ITA)	An individual, organization, or Enterprise responsible for *verifying* and/or *validating* that a system will meet the User's documented operational mission needs for an intended and prescribed operating environment
System Administrator	An individual, organization, or Enterprise responsible for the general operation, configuration, access, and maintenance of a system
Mission Planner	An individual, organization, or Enterprise that: 1. Translates mission objectives into detailed tactical implementation plans based on situational analysis and system capabilities and performance relative to SWOT 2. Develops a course of action, countermeasures, and required resources to achieve success of the mission and its objectives
System Analyst	An individual, organization or Enterprise that applies analytical methods and techniques (e.g., scientific, mathematical, statistical, financial, political, social, cultural) to provide analysis and meaningful data to support informed decision-making by Mission Planners, System Operators, and System Maintainer personnel
System Support	An individual, organization, or Enterprise responsible for supporting the system, its capabilities, and/or performance at a sustainment level that ensures successful achievement of the system's mission and objectives. System support includes activities such as maintenance, training, data, technical manuals, resources, and management

(*continued*)

TABLE 4.2 (*Continued*)

Role	Role Description
System Maintainer	An individual, organization, or Enterprise accountable for ensuring that the Equipment System Element is properly maintained via preventive and corrective maintenance and system upgrades
System Instructor	An individual or organization accountable for training system operators or maintainers to achieve a standard level of performance based on proficiency in achieving the system mission and its objectives
System Critic	An individual, organization, or Enterprise with competitive, adversarial, or hostile motivations to publicize or promote the shortcomings of a system to fulfill its assigned missions, goals, and objectives in a cost effective, value-added manner and/or believes the system is a threat to some other system for which the System Critic serves as a System Advocate
System Competitor	An individual, organization, or Enterprise whose missions, goals, and objectives compete to capture similar mission outcomes **Example** Examples include market share and physical space
System Adversary	An individual, organization, or Enterprise that exhibits hostile behavior or actions whose interests, ideology, goals, and objectives are to: 1. Counter to another system's missions, goals, and/or objectives 2. Exhibit behavioral patterns and actions that appear to be threatening
System Threat	A competitive, adversarial, or hostile individual, organization, or Enterprise actively planning and/or executing missions, goals, and objectives that may be counter to another system's missions, goals, and/or objectives

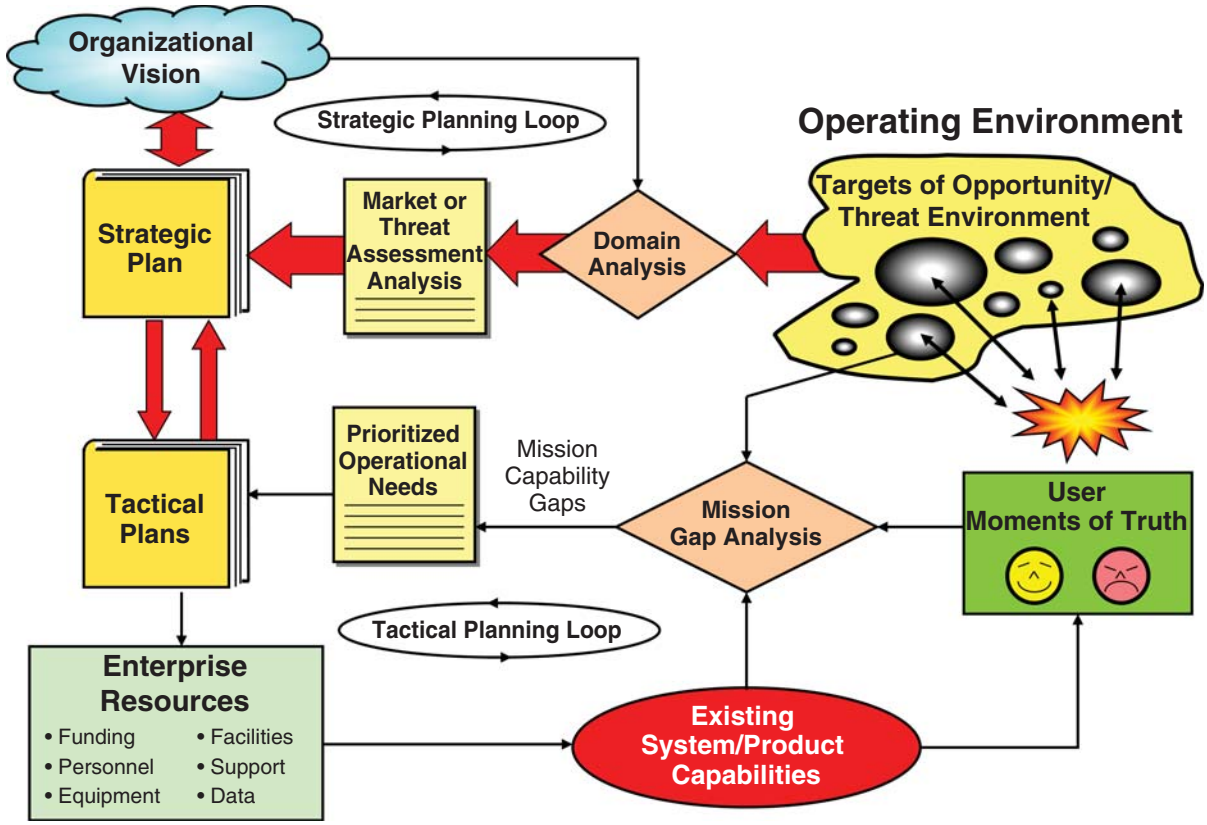


Figure 4.2 Operational Needs Identification Process

years or more from now. The plan identifies a set of long-term objectives, each of which should be *specific*, *realistic*, *measurable*, *achievable*, and *verifiable*. As with any system, strategic planning objectives require performance-based metrics that serve as benchmarks for assessing *planned* versus *actual* performance and progress.

Author's Note 4.2

The Global Nature of the Strategic Plan

It is important to underscore the global nature of the Strategic Plan. The approved document forms the frame of reference for initiating annual organizational Tactical Plans that focus on LOB-specific missions and objectives for their assets—systems, products, or services.

4.4.2.3 The Tactical Planning Loop Once the Strategic Plan is established, the key question is: *How do we get from where we are now to five years from now*? The answer resides in creating and maintaining incremental, short-range Tactical Plans that elaborate near-term—one year—objectives and actions required to achieve strategic planning objectives.

Executive management decomposes strategic objectives into tactical objectives and assigns the objectives to various organizational elements. Performance-based metrics or Measures of Performance (MOPs) benchmark the required performance of each tactical objective. The MOPs serve as benchmarks for assessing planned versus actual progress in achieving the objectives.

4.4.2.4 Tactical Plans In response to the tactical objectives, each organizational element develops a Tactical Plan that describes *how* the each Enterprise's leadership plans to achieve the objectives relative to the MOP benchmarked. In terms of *how*, the Tactical Plan describes what types of Enterprise systems, products, or services as *assets* will be required, acquired, deployed, operated, maintained, sustained, retired, and disposed. Thus, each Enterprise division, department, and so forth requires a specified level of capabilities and performance to support accomplishment of the tactical objectives. To illustrate an aspect of a tactical plan, consider the following example.

Example 4.4

Fleet Availability

Assume an Enterprise requires a fleet of 10 delivery vehicles with a fleet *operational availability* of 0.95 (Chapter 34) and you only have 6 vehicles that are operational due to the need for major repairs. The tactical plan describes the strategy concerning how the Enterprise intends to:

1. Acquire at least four additional vehicles (e.g., purchase, lease, rent).
2. Operate the vehicles to achieve organizational objectives.
3. Maintain the vehicles to achieve an operational availability of 0.95.

4.4.2.5 System Resources System assets as resources in inventory and their current operating conditions represent the existing Enterprise capabilities. Assuming the Enterprise has a realistically achievable Strategic Plan and supporting Tactical Plans, these documents and enabling Enterprise systems have a shelf life. Competitive or hostile threats, as well as opportunities, evolve over time. As a result, two types of situations occur:

- Threat capabilities begin to exceed your Enterprise's organizational capabilities.
- As new opportunities arise, each organizational asset requires a projected level of performance to defend against threats or capitalize on the opportunities.

Given that the Enterprise systems, products, or services have a *shelf life* due to materials degradation or product obsolescence, the Enterprise may experience gaps between current capabilities and they *operationally need* to survive external threats or capitalize on opportunities.

As the Tactical Plans for achieving specific missions mature and are approved, Enterprise Systems such as Personnel and Engineered Systems are funded to achieve the mission objectives. The updates may include:

1. Deployment of a new system, product, or service.
2. Upgrades, enhancements, and refinements to existing systems, products, or services.
3. Updates to Enterprise doctrine and command media revisions.
4. Personnel training and skills enhancement.
5. Revisions to operational tactics.

Depending on the *level of urgency* for these capabilities, a tactical plan may require several days, weeks, months, or a year to implement and bring the organizational capabilities up to required level of performance. Consider the following example.

Example 4.5

Evolution of System Capabilities Over Time

An Enterprise fields a system or product with an Initial Operational Capability (IOC) and incrementally upgrading via a series of "builds" until a Full Operational Capability (FOC) is achieved at some point in the future (Figures 15.5 and 15.6).

Principle 4.8

Mission Outcome(s) Principle

Mission outcome(s) identify what has to be accomplished to eliminate, minimize, or control an emerging Problem Space or exploit an Opportunity Space.

Additionally, until the required capabilities are firmly established, interim operational tactics may be employed to project a perception to adversarial or competitive threats to a capability that may only exist in virtual space. History is filled with examples of decoy systems or products that influence competitor, adversary, or customer perceptions of reality until the actual system, product, or service capability is fielded.

Heading 4.2

At this point, recognize that the Enterprise was established to capitalize on TOOs in the marketplace. As the Enterprise delivers or employs those products or services in the OPERATING ENVIRONMENT, it must continually assess a system's operational *utility*, *suitability*, *availability*, and *efficiency* and *effectiveness*. The analysis collects and analyzes data from User interviews, observations, lessons learned, Trouble Reports (TRs), and deficiencies, for example, comprising the mission capability gaps. The bottom line is:

1. Here's *what* we set out to accomplish with our products and services.
2. Here's *how* they performed in the marketplace.
3. Here's *what* our customers told us about their perceptions and level of satisfaction with our systems, products, or services.
4. Here's the scorecard on performance results.

4.4.2.6 Existing System/Product Capabilities Effective mission gap analysis requires a realistic, introspective assessment of the existing Enterprise system, product, or service capabilities. Enterprises employ media relations to project a positive image to the marketplace. As a result, a "perception" is created that the Enterprise may appear to be much stronger than the existing capabilities, actions, and performances indicate. Depending on the situation, serious business ethics may be at issue with significant consequences.

For internal assessment purposes, the *path to survival* demands objective, unbiased, realistic assessments of system, product, or service capabilities. Otherwise, the Enterprise places itself and its missions at risk by believing their own rhetoric. This paradigm includes a concept referred to as "group think" in which Enterprise management synergizes their thought processes to a level of belief that ignores and defies fact-based reality.

4.4.2.7 Organizational and Stakeholder Moments of Truth

Principle 4.9

Stakeholder Moments of Truth Principle

Every time a System Stakeholder—User and End User—*encounters* and *interacts* with your organization and its systems, products, or services is a "moment of truth" that result in *positive* or *negative* experiences, outcomes, and consequences that impact business with you in the future.

Carlzon (1989) describes every customer's *encounter* and *interaction* with your Enterprise and its systems, products, or services as "moments of truth." As a result, those physical encounters and interactions may have *positive*, *benign*, or *negative* encounters as part of the normal course of day-to-day business operations.

Observe *Stakeholder* as an operative term in Stakeholder "moments of truth." The preceding paragraph addressed the topic from the User and End User perspectives. If the systems, products, or services your Enterprise develops are *poorly* received "moments of truth" by those Stakeholders, imagine the complaints and messages the marketplace sends to your stockholders, executives, and others either directly or via falling stock prices.

How do Enterprises stay informed about Stakeholder operational needs and expectations of their systems, products, or services? Any candid User assessments of your system, product, or service such as *likes*, *don't cares*, or *dislikes* as well as comparisons with others in the marketplace provide invaluable information related to your own capability gaps and opportunities in the marketplace. Field engineers, User interviews and feedback, or broader-based User community surveys serve as key collection points for moments of truth data. The interviews capture User experiences, lessons learned, and best practices based on direct physical interactions with the TOOs or threat environment. Leverage this information and knowledge to identify and create new systems, products, services, or upgrades to them.

4.4.2.8 Enterprise Capability Gap Analysis Mission *gap analysis* focuses more in-depth on an SWOT or gap analysis between the organization's existing system, product, or service capabilities, operational state of readiness, and TOOs or threats. The analysis includes conducting *what if* scenarios; assessment of operational strengths and definition of Measures of Effectiveness (MOEs) and Measures of Suitability (MOSs) (Chapter 5); and leveraged capabilities. Based on the mission gap analysis results, prioritized operational needs are documented and serve as inputs into tactical plans. It is important to note here that gap analysis should reflect two types of information:

1. The paper analysis comparison.

2. Real world field data based on actual physical interactions between the existing system or product and the TOOs or threat environment.

The paper analysis is simply an abstract analysis and comparison exercise based on documented evidence such as "brochureware" in trade journals, customer feedback, surveys, intelligence, and problem reports. The analysis may be supported by various validated models and simulations that can simulate the effects of interactions between the existing system, product, or service capabilities and TOOs or threats. Though potentially lacking in physical substance and validation, the paper analysis approach should convey a level of risk that may have an impact on the organization, physical assets, human life, property, or the environment.

4.5 UNDERSTANDING THE USER'S PROBLEM, OPPORTUNITY, AND SOLUTION SPACES

Problem, Opportunity, and Solution Spaces Principle

Principle 4.10 Understand the User's Problem/Opportunity and Solution Spaces. A worst-case scenario is writing perfectly stated specification requirements for the *wrong problem*.

The concept of identifying and bounding a Problem Space and Solution Spaces is one that you occasionally hear in buzzword vocabularies. Sounds great! Impresses customers! However, if you ask the same people to differentiate the Problem Space from the Solution Space(s), you either get ambiguous answers or a lot of animated arm-waving rhetoric.

Most successful missions begin with a thorough identification and understanding of the Problem, Opportunity, and the Solution space(s). One system's Problem Space may be an Opportunity Space for another that desires to capitalize on the *potential* or *emerging* weakness.

4.5.1 Understanding Operating Environment Opportunities

Problem/Opportunity Space Principle

Principle 4.11 One Enterprise's or system's Problem Space is an Opportunity Space for a competitor or adversarial system.

HUMAN SYSTEMS, from an Enterprise perspective, exploit opportunities and respond to threats. Enterprises assign missions and performance objectives—financial, market share, and medical—that support the founder's or system owner's vision of capitalizing on opportunities or neutralizing threats.

Survivalist opportunity and threat motives are common throughout the Enterprise's OPERATING ENVIRONMENT. Depending on one's perspective, some refer to this as the "natural ordering of systems." The animal kingdom that exists on the plains of Africa is an illustrative example.

Plains of Africa Fable (Anonymous)

Example 4.6 On the plains of Africa, the lion awakes and wonders if it will eat that day—a Problem Space for the lion. Elsewhere, the gazelle awakens and wonders if it will survive the day without being eaten by the lion—the gazelle's Problem Space. From the lion's perspective, the gazelle represents an Opportunity Space and, if sighted nearby within stalking distance, becomes a Solution Space for the lion's survival.

To better understand the concepts and relationships of Problem, Opportunity, and Solutions Spaces, consider the illustration shown in Figure 4.3.

Assume that an Enterprise System such as an airline possesses a set of assets such as airplanes, flights, reservation system, baggage handling system, and so forth. Over time, if the Enterprise does not *continuously improve* those capabilities to offer newer, competitive services to its customers, the competition will. As a result, an Emerging Capability Gap begins to evolve that may be *unnoticeable* or have *limited* public exposure.

Engineered systems, as inanimate objects, are unaware of their plight. However, the User's Enterprise should be increasingly concerned. As the Emerging Capability Gap grows, it is viewed as a Problem Space for the User's Enterprise to fill via corrective action that may have a level of urgency. Sometimes the "gap" occurs in the form of system or product *failures* that require public acknowledgement, notification to Users and End users, and product recalls for immediate corrective action to mitigate risks and consumer safety concerns.

While this is occurring, the User's Enterprise's Competitors/Adversaries may observe, learn, or anticipate the Emerging Capability Gap. Whereas the Owner/User Organization sees the Emerging Capability Gap as a Problem Space, its competitors see the gap as a TOO as indicated by the raptor icon.

The term Problem Space has two contexts—(1) a User–Acquirer's perspective and (2) a System Developer perspective:

- A Problem Space for the User–Acquirer represents an Opportunity–Solution Space for the System Developer.
- In turn, a System Developer's Problem Space of finding a design solution becomes an Opportunity Space for subcontractors, vendors, and consultants to offer solutions.

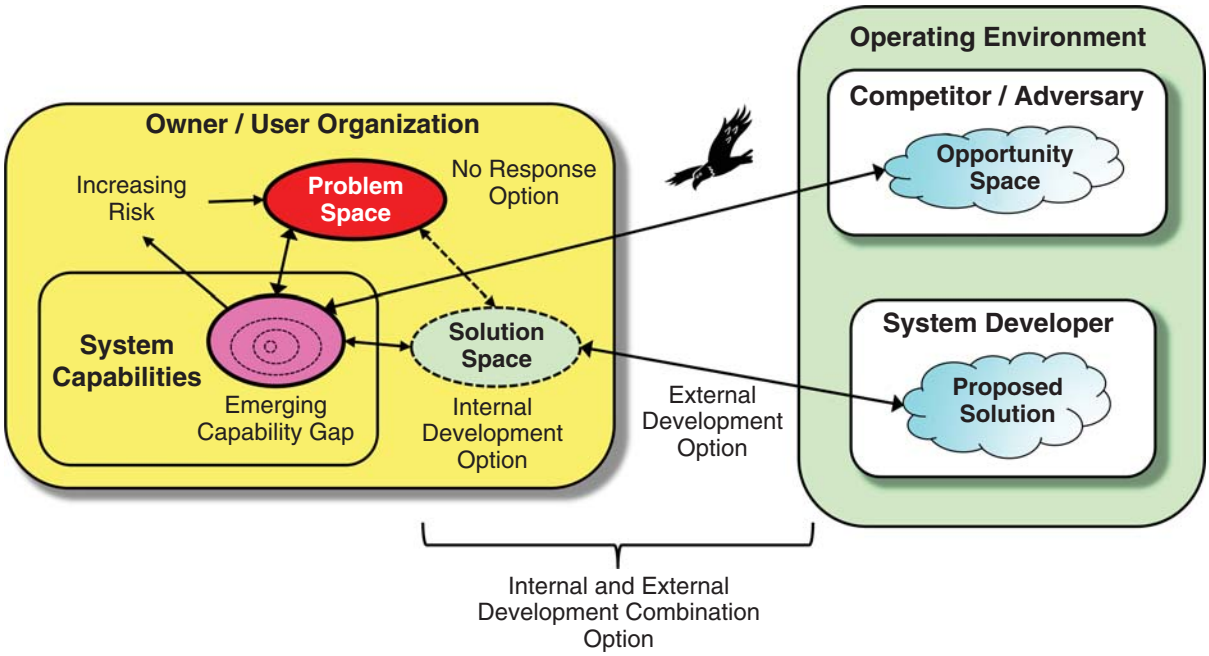


Figure 4.3 Understanding the Problem, Opportunity, and Solution Spaces

When this condition occurs, the User's Enterprise has three potential options:

1. Fill the gap via internal system or upgrade development.
2. Contract with an external System Developer to develop a new system or upgrade the existing system.
3. Combinations of internal and external development.

In highly competitive situations, the Enterprise may be required to camouflage the gap until it can be filled. In other cases, such as product recalls, the Enterprise is required to acknowledge the gap via public announcements to raise consumer awareness of the situation and build confidence that actions are being taken to correct any deficiencies or safety issues.

4.5.1.1 Types of System Opportunities Opportunities generally are of three basic types:

1. Time-based (i.e., waiting for the right time).
2. Technology-based (i.e., waiting on technology maturity).
3. Location-based (i.e., waiting for a lease to expire).

Let's explore both of these further.

4.5.1.2 Time-Based Opportunities Time-based opportunities can occur randomly or predictably. Random opportunities are sometimes viewed as "luck." Predictable opportunities are dependent on periodic or repeatable behavioral patterns (i.e., knowledge applied to practice) that enable an aggressor system to capitalize on a situational weakness.

4.5.1.3 Technology-Based Opportunities One of the ways a User can capture opportunities is by exploiting new and emerging technology either through internal research and development, currently available in the marketplace, or by forging strategic partnerships with other organizations.

4.5.1.4 Location-Based Opportunities Location-based opportunities, as the name implies, relate to being in the right place at the right time. In the business world, success is often said to be driven by "Location! Location! Location!" Obviously, a good location alone does not make a business successful. However, the location positions the business for potential mission success.

4.5.2 Understanding the Problem Space

Operational Needs Principle

Principle 4.12 System analysis requires recognition and validation of three types of Stakeholder—User and End User—operational needs: *real*, *perceived*, or *projected*.

One of the first steps in SE is to understand *what* problem or issue the User is attempting to solve. The term Problem Space is relativistic. Consider competition in the commercial marketplace or military adversaries. An Enterprise may view a competitor or adversary and their operating domain as a Problem Space. Hypothetically, if you were to ask the competitor or adversary if they were a "problem" to the other Enterprise, they may state unequivocally "yes!" or emphatically "no!" Therefore, the context of a Problem Space resides in the eyes and minds of those who *perceive* the situation. Sometimes there is little doubt as evidenced by acts of aggression or hostility such as invasion of a country's air space or hostile business takeovers.

Recognize the Problem Space has two contexts:

1. Threat Context—Eliminate vulnerabilities in system, product, or service capabilities to external threats.
2. Opportunity Context—Seize the Opportunity Space to create a new system, problem, or service that will enable your Enterprise to increase market share in a highly competitive market.

4.5.2.1 Opportunity Space Versus Problem Space Semantics

In a highly competitive marketplace and adversarial, hostile world, survival for many organizations requires proactive minimization of system vulnerability. Enterprises that are proactive in recognizing opportunities initiate risk mitigation actions to prevent hazards from occurring and becoming problems—or tomorrow's corporate headlines. In contrast, procrastinators deal with problems by becoming reactionary "firefighters," assuming that they were aware of the potential hazard and did not mitigate it; they seem to never get ahead. Since the term Problem Space is commonly used and one aspect of SE is *problem-solving*, this text uses the term Problem Space.

4.5.2.2 Problem-Solving or Symptom Solving?

Principle 4.13

Problem–Symptom Solving Principle

There are two types of solution development activities: *problem-solving* and *symptom solving*. Recognize the difference.

Enterprises often convince themselves and their executive management they are problem-solving. In many cases, the so-called *problem-solving* is actually *symptom solving*. This question leads to critical question for the User, Acquirer, and System Developers: *Is this the right problem to solve or a downstream symptom of an unknown or larger problem*?

4.5.2.3 Dynamics of the Problem Space

For most organizational systems, Problem Spaces are *dynamic* and *evolutionary*. They evolve over time in a number of ways. Some occur as instantaneous, catastrophic events, while others emerge over several years—such as the hole in the ozone layer of Earth's atmosphere. The root causes for Problem Spaces in system capabilities and performance originates from several potential sources such as:

1. System neglect.
2. Improper oversight or maintenance.
3. System degradation through normal wear.
4. Ineffective training of the User Operators.
5. Improper use, abuse, or misapplication.
6. Product or technology obsolescence.
7. Budgetary constraints.

Organizationally, managers have an obligation to track Problem Spaces. The challenge is that some managers are politically reluctant to surface the Problem Spaces until it is too late. In other cases, management fails to provide the proper visibility and priority to seemingly trivial issues until they become full-fledged Problem Spaces—the proverbial "head in the sand." When this happens, four potential outcomes can occur:

1. Operationally, the source of the Problem Space goes away.
2. The Enterprises management becomes distracted or enamored by other priorities.
3. Enterprise objectives change.
4. Catastrophic (worst-case) events force corrective action.

In general, people tend to think of the Problem Space as *static*. In fact, the primary issue with the Problem Space is its *dynamics*, especially in trying to bound and specify it as illustrated in Figure 4.3. Dynamically, gaps may occur rapidly or evolve slowly over time.

4.5.2.4 Forecasting the Problem Space

The challenge for most organizations is: *How do we translate a forecast of a potential Problem Space in terms of system capabilities with some level of confidence*? The answer resides in the organizational and system level strategic and tactical plans, system missions, and objectives.

Enterprise Strategic and Tactical Plans establish the reference framework for evaluating potential organizational weaknesses—*current* capabilities versus *planned* capabilities. Using these objectives as the basis, situational assessments and gap analysis are employed as tools to compare the state of *existing* to *projected* system capabilities and performance against *projected* capabilities and performance of

competitors or adversaries. The results may indicate a potential or emerging "gap" in capabilities and/or levels of performance as shown in Figure 4.3. The identification of the gap establishes the basis for Enterprise action.

Sometimes you cannot forecast a Problem Space. At best, you may need to prepare for its occurrence … but that costs money that some may view as wasteful. Consider the following example.

Example 4.7 **Asteroid Problem Space Example**

Planet Earth has encountered several near misses by asteroids. Since mankind started exploring space, the international scientific community has discussed the need for a solution such as launching a device into space to destroy or divert the asteroid before it can hit Earth.

In this case, you will not know an asteroid is on a collision path until it is discovered by astronomers. Then, response time becomes a critical issue. This leads to the question: *what happens if you develop such as device and there are no asteroids*? The rhetorical question becomes: *What are the cost and risk of development versus the cost and risk of not developing the device*?

4.5.2.4.1 When Does a "Capability Gap" Become a Problem? Technically, a *problem* does not exist until a *hazard* that poses a potential risk occurs as an event such as an accident or incident (Chapter 24). Then you actually have a problem! The infamous "OK, Houston, we have a problem …" communicated by Apollo 13 Commander Jim Lovell is one of the best illustrations of the context used here.

This is perhaps the toughest question, especially from a forecasting perspective. Obviously, you know you have a problem when it occurs such as malfunction, emergency, or catastrophic events. One approach may be to determine whether a potential hazard with a level of risk has outcome-based consequences that are *unacceptable*. In effect, you need to establish levels or thresholds for assessing the degree of problem significance.

4.5.2.4.2 Establishing Problem Space Boundaries One of the challenges in SE is defining the Problem Space boundaries. It is easy for people to debate the concept of Problem Spaces, their dependencies, and their *dynamics*. However, Problem Spaces are often vague, imaginary concepts that are difficult to bound and articulate.

Metaphorically, Problem Spaces are like the fabled Sasquatch. Everyone talks about their existence, and some claim to have recorded various types of objective evidence, but no one seems to be able to capture one. Rowe (1998, p. 56) illustrates this point by introducing what he describes as the *Problem Space Problem*—defining and bounding the Problem Space.

Conceptually, we illustrate a Problem Space with solid lines to symbolically represent its boundary. For some systems such as a lawn, the property boundaries are clearly defined for ownership accountability. In other cases, the boundaries may be elusive and vague. Consider civil unrest and wars in countries where people take sides but look, dress, and communicate similarly. *How does one differentiate friend versus foe*? *On which day of the week*?

Lines drawn around abstractions such as ideology, politics, and religion are often blurry, vague, and ill defined. Consider the graphic shown in Figure 4.4. The left side of the figure illustrates Problem Space boundaries as gray, fuzzy edges. The "center of mass" is indicated by the dark area whose edges, however, are blurry and indistinct. In some cases the Problem Space has tentacles that connect to other Problem Spaces, each having an effect on the other. The blurriness may not be static but a continuum of dynamic, evolving changes as with clouds or a thunderstorm.

4.5.2.5 Eliminating/Controlling the Problem Space

Principle 4.14 **Eliminate/Control the Problem Space Principle**

If you cannot eliminate the Problem Space, try to *control* it until you can *resolve* and *eliminate* it.

Depending on the *source* or *root cause* of the Problem Space and the degree of risk to your system or its mission objectives, the natural tendency of most Enterprises is to eliminate the problem, assuming it is within their control, resources, or sphere of influence. However, the reality is that you may not be able to eliminate the Problem Space. At best, you may only be able to *manage* and *control* it.

Once the Emerging Capability Gap is recognized as a Problem Space, *how do we articulate the problem to others*? This brings us to our next topic, Defining the Problem Statement.

4.5.2.6 Defining the Problem Statement

Principle 4.15 **Problem Statement Principle**

Every Problem or Opportunity Space should be *clearly* and *concisely* bounded by a well-articulated Problem Statement that does not identify causes, assign blame, or propose solutions.

Principle 4.16 **Contributory Cause(s) Determination Principle**

Investigative teams determine probable causes and recommend solutions based on analysis of the Problem Statement.

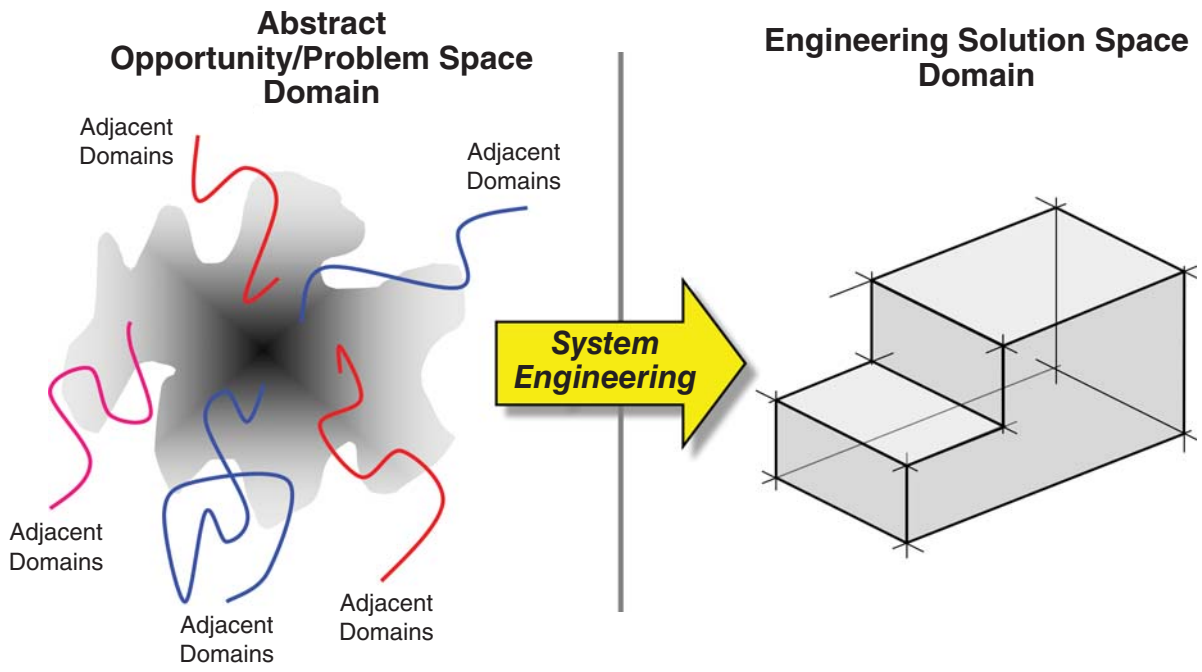


Figure 4.4 SE—Translating the Abstract Opportunity/Problem Space into an SE Solution Space

Author's Note 4.3

Root Cause Paradigms and Contributory Causes Realities

One of the sources of Problem Spaces is often incidents and accidents. Executives and news media boldly demand determination of the "root cause."

In Chapter 24, you will learn that most *incident* and *accident* Problem Spaces are not the result of a *single* root cause but are typically the result of a *series* (Figure 24.1) of contributory causes such as unsafe acts—operator, management, and EQUIPMENT latent defects—design errors, flaws, and deficiencies that occur under the same executive's purview. When a potential hazard *penetrates* Enterprise and Engineered System *safeguards*, *barriers*, or *defenses* intended to prevent such an occurrence, its trajectory culminates in an incident or accident.

Our discussion to this point focuses on the Problem Space in an abstract sense. The rhetorical question that requires specificity is: *What problem or issue is the User attempting to solve*? Before the Solution Space can be bounded and specified, it is crucial for you and your development team, preferably in collaboration with the User, to simply document what problem or issue the User is attempting to solve. You need to define a Problem Statement. Ultimately, this leads to the question: *How should a problem statement be written?* Although there are a number of ways to develop a problem statement, there are some general guidelines to apply. A Problem Statement should:

1. Clearly, concisely, and succinctly define the problem or issue in one sentence.
2. Avoid identifying the source or root cause of a problem.
3. Identify the operational scenario or operating conditions under which the problem occurs or leads to occurrence of the problem.
4. Avoid stating any explicit or implicit solutions.
5. Avoid assigning responsibility or blame.

Consider the following example.

Example 4.8

Simple Problem Statement

Viruses are corrupting computers connected to our Local Area Network (LAN).

Observe that the example does not specify: (1) *where* the viruses originate, (2) the *source* or *root cause* of the problem, (3) *what* the impact is, or (4) *how to solve* the problem.

Also observe that we said that a Problem Statement should be a single sentence. People have a tendency to write a paragraph about a problem leaving the reader to figure out what the true problem actually is. Write a single, *stand-alone* Problem Statements. If additional clarifying information is required, isolate it from the Problem Statement and label it as "Discussion" or "Clarifying information."

4.5.2.7 *Partitioning the Problem Space*

As your understanding of the Problem Space matures, the next step is to partition it into one or more Solution Spaces. Work with System Stakeholders to partition the complex Problem Space into more manageable Solution Spaces. The identification of one or more Solution Spaces requires *highly iterative* collaboration, analysis, and decision-making. Consider the challenge of attempting to partition the ambiguous Problem Space shown at the left side of Figure 4.4. Through partitioning, our objective is to isolate key attributes, properties, and characteristics of the problem to enable development of solutions.

Problem Complexity Reduction Principle

Principle 4.17 Partition—decompose - Problem or Issue Spaces into one or more manageable Solution Spaces as a means of reducing *complexity* and managing risk.

Humans have a tendency to believe that a Problem Space has one Solution Space. For some Problem Spaces, this may be true. This assumes all Problem Spaces are easily solvable by current hardware and software in the marketplace. The challenge is some problems—relatively speaking—are small, some complicated, and some simply complex.

Problem-solving requires establishing a conceptual solution as a starting point that may become several Solution Spaces. The Solution Space(s) may evolve throughout the process and may not be recognizable at completion from its starting point.

One approach is to gather the facts about a Problem Space and create *notional* or hypothesis-based Solution Space boundaries. Then, as the analysis progresses, adjust the boundaries until decisions about the Solution Space boundaries *mature* or *stabilize*. The key point is that some complex Problem Spaces are described as "wicked" (Chapter 2) due to their complex, dynamic nature and are effectively unsolvable. In general, you have a choice:

- Flounder in the abstractness.
- Make a decision, move on to the next decision, and then revisit and revise the original decision when necessary.

4.5.2.8 *Addressing Capability Gaps in Current Systems*

Systems, products, and services have *shelf lives* and ultimately outlive their usefulness, have diminishing utility to System Stakeholders over time, and ultimately become obsolete. Assuming the capability gaps are not significant and solution options are feasible, the User may decide to acquire capability upgrades and retrofit existing systems in the field. If the gaps are significant, the User needs to anticipate their size well in advance of their occurrence. In these cases, new system development may be the alternative.

Most systems, products, and services are *precedented* (Chapter 2). As a result, many of the required operational capabilities may already exist—the implementation may be different. In the case of *unprecedented* systems, the System Acquirer or User may have to issue several sequential contracts to develop prototypes that drive out key operational requirements. To better understand how system capabilities are derived, let's use the example illustrated in Figure 4.5.

During the System OM&S Phase of System #1, a decision event occurs to replace System #1. The acquisition strategy is to bring the new System #2 "on-line" or into active service as noted by the First Article Field Delivery event.

After a System Transition Period for checkout and integration of System #2 into the Higher Order System, an Existing System Deactivation Order is issued. At that time, System #2 becomes the primary and System #1 enters the System Disposal Phase of its life cycle. At some time period later, the disposal of System #1 is marked by the Existing System Disposal Complete event.

So, how do we initiate actions to get System #2 into active service by the planned new system's First Article Field Delivery event without disrupting organizational operations? Let's explore that aspect.

When the new system Operational Need Decision event is made, procurement actions are initiated to initiate System #2's life cycle. Thus, the System Definition Phase of System #2's life cycle begins. System #2's System Development Phase must be complete and ready for field integration by the New System's First Article Field Delivery event. System #2 then enters the System OM&S Phase of its life cycle.

By the time the Existing System Deactivation Order event is issued, System 2 should be "on-line" and in active service. As a result, System #1 completes its life cycle at the Existing System Disposal Complete event thereby completing the transition.

4.5.2.9 *Problem Space Degree of Urgency*

The level of risk and degree of urgency of the Problem Space as a whole or portions thereof may influence or drive Solution Space decisions, especially, where budgets or technology are constrained. The net result may be a decision to prioritize Solution Spaces and levels of capability within them.

We meet this challenge by establishing an IOC at system delivery and acceptance. Then, as budgets or technologies permit, IOC is followed by a series of incremental "builds" or upgrades (Figures 15.5 and 15.6) that enhance the overall capability. Finally, as the system matures with the integration of the "builds," overall capability referred to as a FOC is achieved.

Given the preceding Problem/Opportunity Space discussion, *how does a User resolve the issue*? The answer resides in our next topic, Solution Spaces.

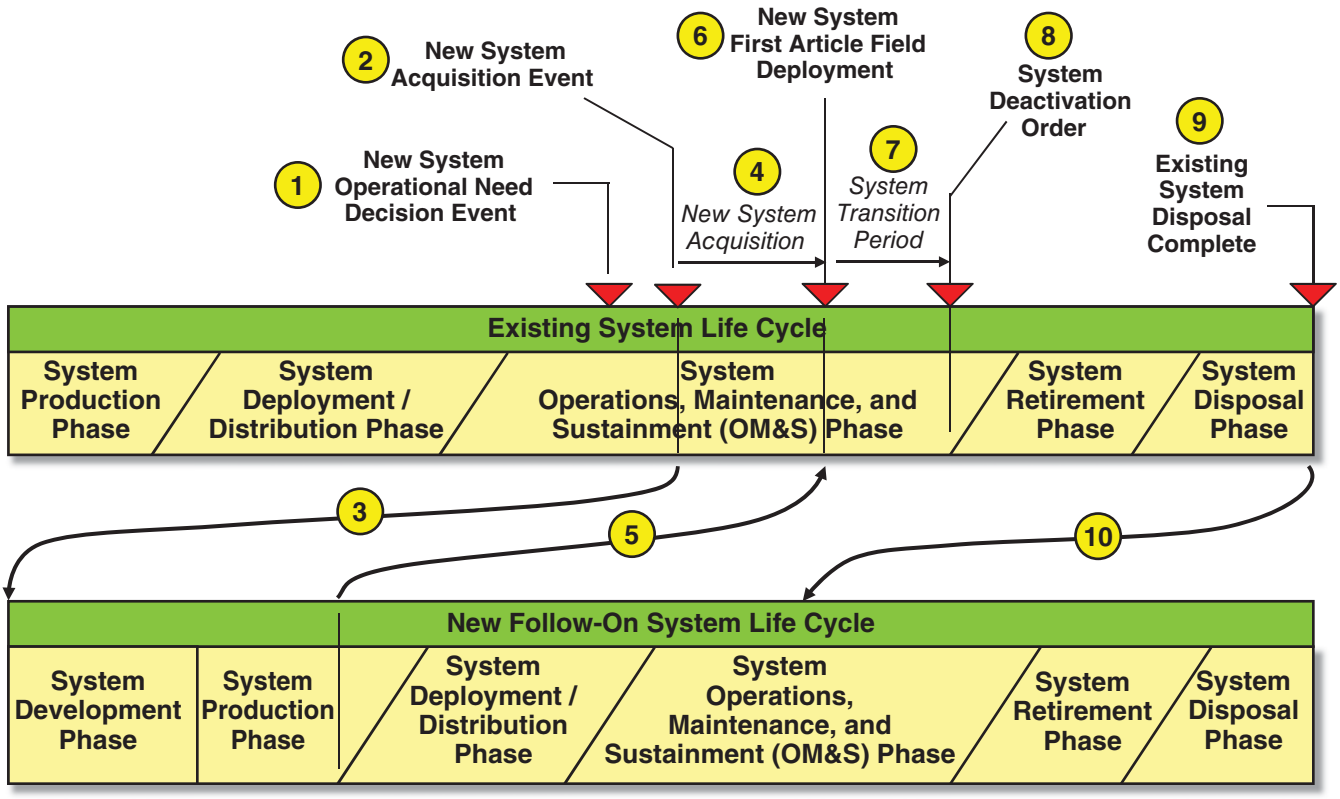


Figure 4.5 The Acquisition of a New System and Phase-Out of an Existing (Legacy) System

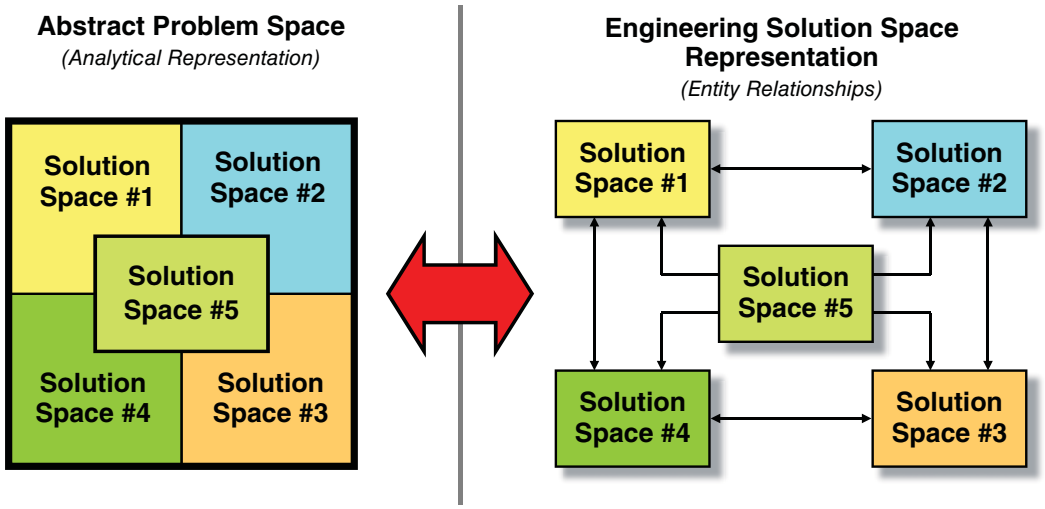


Figure 4.6 Partitioning the Problem Space into an SE Solution Space Representation

4.5.3 Understanding Solution Spaces

To illustrate the partitioning of the Problem Space into Solution Spaces, consider the graphic in Figure 4.6. Symbolically, we begin with a Problem Space represented by a large box. Next, we arbitrarily partition the box into five Solution Spaces, each focused on satisfying a set of Problem Space capability and performance requirements allocated to the Solution Space. Initially we could have started with four or even six solution spaces. Through analysis, we ultimately decide there should be five Solution Spaces; it could have been four or six.

So, *how does this relate to system development*? The large box symbolizes the total system solution. We partition the

complexity of the system solution into multiple levels of Solution Spaces via the System's multi-level architecture as a framework. Let's explore this point further.

4.5.3.1 Problem–Solution Space Partitioning and Decomposition

Problem Space Decomposition Principle

Principle 4.18 Partition or decompose each Problem Space into one or more Solution Spaces.

Problem Space complexity is resolved through a process of partitioning and multi-level refinement - decomposition - as shown in Figure 4.7. We partition the overall Problem Space into four Solution Spaces, 1.0 through 4.0. Solution Space 4.0 becomes Problem Space 4.0 for the next lower level and is partitioned into Solution Spaces 4.1 through 4.4. The partitioning and decomposition process continues to the lowest level. The net result of the narrowing process is shown in the upper right-hand corner of the figure.

In general, Solution Spaces are characterized by a variety of boundary conditions:

- Distinct, rigid boundaries.
- Fuzzy, blurry boundaries.
- Overlapping or conflicting boundaries.

The degree to which the Solution Space is filled is determined by the capabilities required, priorities assigned to, and resources allocated to the Problem Space "zone." HIGHER ORDER Systems (Chapter 9) contractually and organizationally impose resource and operating constraints that may ultimately limit the degree of solution coverage.

Referral Warwick and Norris (2010) introduce a topic for SE debate concerning an observation that *complicated systems*, aircraft and cars, are *decomposable* versus *complex systems*, social network systems and the Internet, cannot be easily decomposed due to their dynamic and evolving nature.

4.5.3.2 Eliminating the Problem Space When a capability gap is identified in an Enterprise or Engineered system, product capability, or service, it generally takes a finite amount of time to resolve, especially if system development is involved. If the gap is of a *defensive* nature, the system or

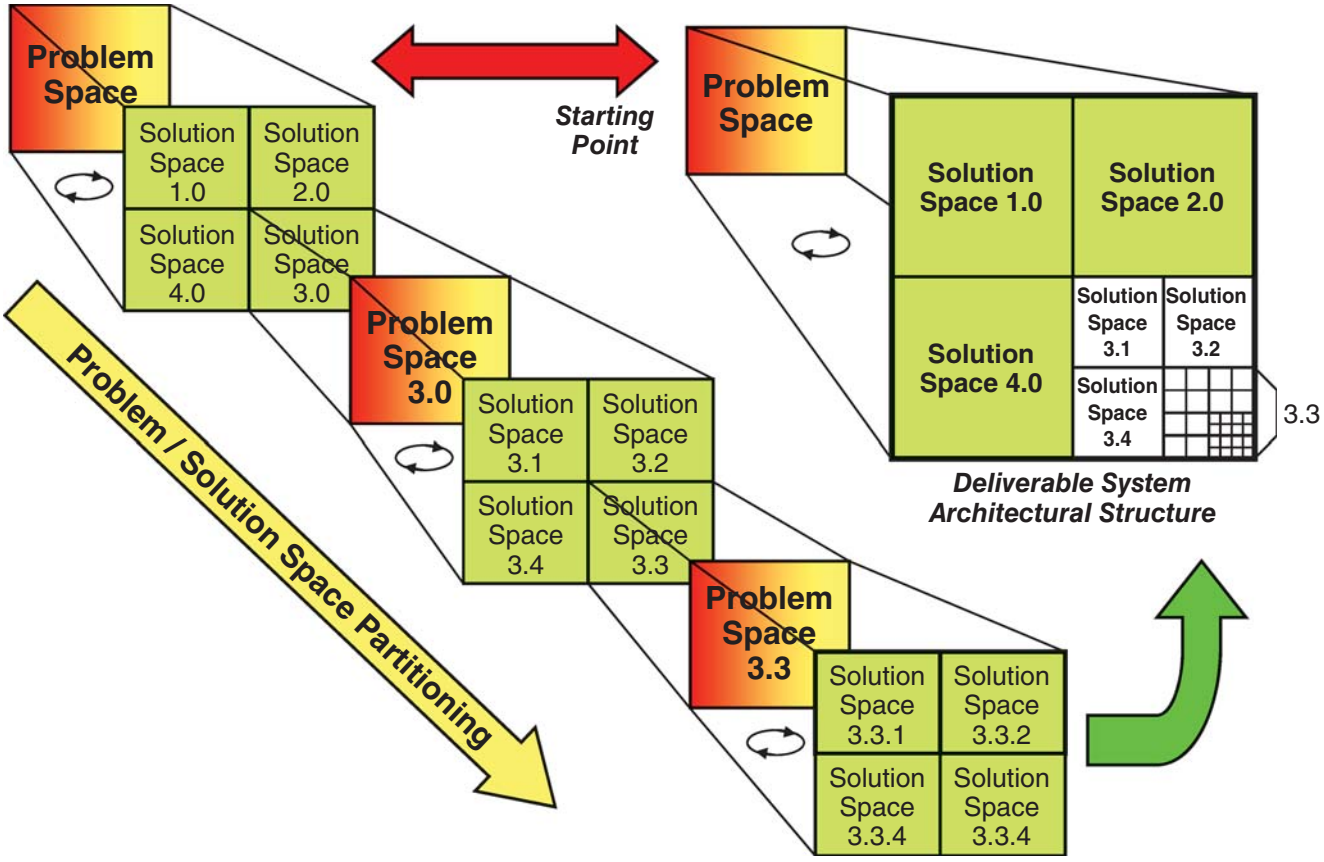


Figure 4.7 Partitioning (decomposing) the Problem Space into Manageable Pieces

product may be *vulnerable* or *susceptible* to acts of aggression and hostilities from competitors or adversaries. If the gap represents a *deficiency* in an offensive capability, work must be performed to eliminate the gap by upgrading system capabilities and performance.

Depending on the system or product's application, operational *tactics* such as decoys, camouflage, and operational patterns may be employed to supplement the gap until a new system, product, or service is available.

4.5.3.3 Solution Capability Force Multipliers

Solutions Spaces require capabilities that are *affordable* but may not completely fill all of the User's operational needs. *Capability-based solutions*, as viewed by the User, have a finite strength, capacity, and reliability required to accomplish mission objectives:

- Strength—Power to accept a specific type of mission challenge.
- Robustness—Capability to withstand threat attacks and accommodate internal failures.
- Capacity to Project—Multiply that power over a defined range.
- Reliability—A probability of completing a mission of a given duration in a specified Operating Environment

The question is: *How can the User expand the limited Solution Space capability of a system to fill the void*? The answer resides in strategic leveraging of capabilities. Consider the following example.

Example 4.9

Developing a high-performance aircraft with a given operating range and payload may be prohibitively expensive. *How do you solve the problem*? What you need to do is figure out how to build an affordably priced aircraft and be able to "project" that capability into a 10× square mile domain by leveraging other systems and their capabilities. One way of doing this is by leveraging the capabilities of air-to-air refueling with fuel tanker aircraft or logistical bases. As a result, the operating range is increased significantly.

4.5.3.4 Selecting Candidate Solutions for the Solution Space

4.5.3.4.1 Enterprise Each Solution Space is bounded by technical, technology, support, cost, schedule, and acceptable risk constraints. The challenge is to identify and evaluate several *viable* candidate solutions that satisfy the technical requirements and then recommend the *preferred* solution. The selection requires establishing pre-defined, objective criteria and then performing an Analysis of Alternatives (AoA) (Chapter 32) to select the recommended solution.

Recognize that Solution Spaces are simply *boundary* spaces for solution development *without* regard to *how* the system solution is to be developed. By default to reduce risk and minimize cost, SEs should always determine how they can leverage or modify existing User assets or find solutions already available in the marketplace. New development should only be investigated and performed as a *last resort after all other options have been exhausted* (Principle 16.7).

4.5.4 Problem Space Exposure to Facilitate Solution Space Definition

On-Site Visits Principle

Principle 4.19

Always send a qualified SE to accompany business development personnel during on-site visits to understand, analyze, and document the Opportunity/Problem and Solution Spaces.

Every person in your Enterprise should have firsthand exposure to and an understanding and the User's Problem Space as they relate to your Enterprise's Opportunity and Solution Spaces. Unfortunately, travel budgets and the User's desire and ability to accommodate throngs of people prevent firsthand observations of *how* the system they develop will be deployed, operated, and supported. As with any scientific field, *observation* is a critical *Systems Thinking* skill for SEs. Seeing, touching, feeling, operating, hearing, and studying existing systems in action have a profound influence on SE problem-solving and solution development throughout system development. Consider the following example.

Mini-Case Study 4.1

Understanding the Problem and Solution Spaces

A company is contracted to design a large piece of computer equipment. Prior to the trip to the User's site, Engineering personnel are denied the opportunity by Business Development to participate in a site survey of the facility. Since one of the challenges is always maneuvering equipment through doorways and hallways, Business Development personnel visit the site and take a few notes about the doorway and hallway leading to the room entrance. Convinced that they could develop a cabinet to fit through a narrow doorway, Engineering proceeded with the design.

When the system was delivered, the installation team encounters major problems. They discover that the cabinet

could be moved down the narrow hallway and through the door, except for one "minor detail"—a case of the *overlooked constraint*. The cabinet could not physically be maneuvered around a 90-degree turn between the hallway and the doorway.

Heuristic 4.1 On-Site Visits to the User

Always send a qualified SE to accompany Business Development personnel during on-site visits to understand, analyze, and document the Opportunity/Problem and Solution Spaces.

4.5.5 Final Thoughts

Principle 4.20

Counter Reactions Principle

When bounding a Solution Space, anticipate *short-term* and *long-term* competitor or adversarial *reactionary responses* and *countermeasures* to the Solution Space.

Understanding Problem–Solution Spaces requires continual assessments due to the *dynamics* of the OPERATING ENVIRONMENT. SEs often *erroneously* believe that filling the Solution Space with development of a new system, product, or service is an end-all answer. As in the case of Newton's Third Law of Motion, every action initiated by the User can be expected to have an equal and opposite counter-reaction by competitors and adversaries. So, when you bound the Solution Space, the bounding process must also consider:

- The potential reactions of competitors and adversaries.
- How the new system, product, or service minimizes susceptibility and vulnerability to those threats, at least for a reasonable period of time until reactionary capabilities can be fielded and retrofitted on existing systems.

4.6 CHAPTER SUMMARY

Chapter 4 introduced the concept of Problem, Opportunity, and Solution Spaces. Key points include:

1. Identify and bound the Problem or Opportunity Space and its Stakeholders—Users and End Users.
2. Recognize and appreciate the difference between *problem-solving* and *symptom solving*.
3. A Problem Statement *clearly* and *concisely* describes a Problem, Issue, or condition that must be resolved. *Avoid* identifying the root cause or probable cause. Do not assign responsibility, blame, or the root cause(s).
4. Manage Problem Space complexity by decomposing it into one or more levels consisting of one or more Solution Spaces
5. Decompose or partition an overall Problem Space into multiple levels of one or more Solution Spaces with each Solution Space serving as the Problem Space for the next lower level Solution Spaces.
6. Each Solution Space represents a set of performance-based outcomes and capabilities that are required to resolve the User's Problem Space.
7. Leverage existing assets, legacy system designs, and marketplace solutions before embarking on new system design.

4.7 CHAPTER EXERCISES

4.7.1 Level 1: Chapter Knowledge Exercises

1. What is a Mission System (Producer) role?

2. What is an Enabling System (Supplier) role?

3. What is a Problem Space?

4. What is an Opportunity Space?

5. What is the relationship between a Problem Space and an Opportunity Space?

6. What is a Solution Space?

7. What is the relationship between the Problem/Opportunity Space and a Solution Space?

8. How do you write a Problem Statement?

9. Identify three rules for writing Problem Statements.

10. How do you forecast the Problem Spaces?

11. How does an Enterprise resolve gaps between a Problem Space and its Solution Space(s)?

12. Where and how do Users obtain system requirements for development?

13. Cite various Solution Space tools that enable a homeowner to leverage their time, resources, and skills to maintain their lawn.

14. Cite two examples of Human Systems—organizational and engineered—that project or expand their sphere of influence by leveraging the capabilities of other systems.

4.7.2 Level 2: Chapter Knowledge Application Exercises

Refer to www.wiley.com/go/systemengineeringanalysis2e

4.8 REFERENCES

Carlzon, Jan (1989), *Moments of Truth*, New York, NY: Harper Business.

Carroll, Lewis (1865), *Alice's Adventures in Wonderland*, London: Macmillon.

Donne, John (1623), Meditation #17 - Devotions Upon Emergent Occasions.

Rowe, Peter G. (1998), *Design Thinking*, Cambridge, MA: The MIT Press.

Warwick, G. and Norris, G. (2010), *"Is It Time to Revamp Systems Engineering?" Aviation Week*, Washington, DC: Aviation Week & Space Technology.

5

USER NEEDS, MISSION ANALYSIS, USE CASES, AND SCENARIOS

The primary purpose of any system, product, or service is to accomplish consumer or organizational objectives with an expected Return on Investment (ROI)—*tangible* or *intangible*—as defined by its Stakeholders—Users and End Users. These objectives may range from the quality of life such as happiness, entertainment, education, and health to the basic necessities of life—organizational survival, profitability, food, and shelter. The act of *identifying, bounding*, and *defining* the set of capabilities to accomplish these objectives encompasses the System Definition Phase of the System/Product Life Cycle (Figure 3.3).

The *form, fit,* and *function* of a system, product, or service is *defined* by (1) its Stakeholders (Users and End Users), (2) mission applications, (3) performance-based objectives and outcomes to be achieved, and (4) Operating Environment conditions. For example, consider shape or form as a *distinguishing application characteristic* such as an automobile versus high-performance race car, aircraft versus spacecraft, desk phone versus smartphone, and desktop computer versus tablet computer.

Humans leverage systems, products, and services as Enabling Systems to supplement our own finite capabilities and limitations to accomplish feats greater than we can achieve individually or collectively. Examples include long-distance travel in a short period of time, space travel, Internet access to world knowledge, fresh foods from great distances, and telecommunications. Selection or acquisition of those Enabling System capabilities begins with understanding the *who*, *what*, *when*, *where*, *why*, and *how* system User(s) plan to accomplish the mission(s).

Contrary to the traditional Engineering viewpoint of developing "widgets" – Engineering the Box - from the Engineer's perspective for the User to figure out how to use, the reverse should occur. As Chapter 24 User-Centric System Design (UCSD) will address later, Engineering needs to *shift* its traditional paradigm to understanding a system, product, or service's Users—their capabilities and limitations. Then, design Equipment as Enabling Systems for their operators and maintainers to accomplish their assigned missions, not vice versa. This is a *critical staging point* in System Definition. Every System Definition decision from this point forward, including mission success, evolves from shifting this paradigm to a new one.

Chapter 5 introduces the key elements of the System Definition—User Needs Analysis and Mission Analysis. *Mission analysis* becomes the key tool for defining the analytical infrastructure and model of a system, product, or service based on its mission applications. Our discussions introduce a Mission Methodology that leverages concepts such as User Stories from Agile Development, Use Cases (UCs), and scenarios that enable SEs and Analysts to:

- Identify and understand the *values* and *priorities* Users place on Key Performance Parameters (KPPs) or quality attributes—performance, usability, and reliability—that drive *customer expectations* and *satisfaction*.

Employ UCSD methods (Chapter 24) to develop systems, products, or services that conform to User *capabilities* and *limitations, not vice versa,* and enable them to accomplish mission outcomes.

System Engineering Analysis, Design, and Development: Concepts, Principles, and Practices, Second Edition. Charles S. Wasson.

Companion Website: www.wiley.com/go/systemengineeringanalysis2e

5.1 DEFINITIONS OF KEY TERMS

- **Actor**—"An actor specifies a role played by a user or any other system that interacts with the subject. (The term 'role' is used informally here and does not necessarily imply the technical definition of that term found elsewhere in this specification.)" (OMG, 2006, p. 230).
- **Anthropometrics**—"Quantitative descriptions and measurements of the physical body variations in people. These are useful in human factors design" (MIL-HDBK-470A, p. G-2).
- **Compensating Provisions**—"Actions that are available or can be taken by an operator to negate or mitigate the effect of a failure on a system" (Mil-Std-1629A, p. 3).
- **Cost-Effectiveness**—"A measure of the operational capability added by a system as a function of its Life Cycle Cost (LCC)" (DAU, 2012, p. B-49).
- **Cost-as-an-Independent-Variable (CAIV)**—A concept that offers a range of viable alternatives to a decision-maker to trade-off operational capabilities versus cost for purposes of selecting the best value solution to meet organizational needs.
- **Critical Operational Issue (COI)**—A concern expressed by the User, System Acquirer, or System Developer regarding the deployment—Operation, Maintenance, and Sustainment (OM&S)—and retirement or disposal or operating constraint(s) of a system, product, or service.
- **Critical Technical Issue (CTI)**—A concern expressed by the User, System Acquirer, or System Developer regarding the achievement of a specification requirement; technology limitation, application, or implementation; or conflict with another specification requirement.
- **Effectiveness**—"The extent to which the goals of the system are attained, or the degree to which a system can be elected to achieve a set of specific mission requirements … " (DAU, 2012, p. B-75).
- **Engagement**—A single instance of a friendly, cooperative, benign, competitive, adversarial, or hostile interaction between two systems.
- **Key Performance Parameter (KPP)**—An attribute representing a *quality characteristic* with a *minimum acceptable value* the User has determined is critical to achieving mission success and subsequently customer satisfaction. Example *quality characteristics* include performance, usability, survivability, safety, reliability, and shelf life.
- **Measure of Effectiveness (MOE)**—A *quantitative* measure that represents the outcome and level of performance to be achieved by a system, product, or service and its level of attainment following a mission.
- **Measure of Performance (MOP)**—A *quantitative* measure to represent the level of performance to be achieved by a specified capability, typically in the form of a stated requirement statement in a specification.
- **Measure of Suitability (MOS)**—"Measure of an item's ability to be supported in its intended operational environment. MOSs typically relate to readiness or operational availability and, hence, reliability, maintainability, and the item's support structure" (DAU, 2012, pp. B-140). An MOS is a key contributor to operational suitability.
- **Mission**—A pre-planned exercise that integrates a series of sequential or concurrent operations or tasks with an expectation of achieving outcome-based success criteria with *quantifiable* performance objectives.
- **Mission Event Timeline (MET)**—A timeline that: (1) identifies key mission events and (2) when they must occur to ensure successful completion of a mission.
- **Mission Reliability**—"The probability that a system will perform its required mission-critical functions for the duration of a specified mission under conditions stated in the mission profile" (DAU, 2012, p. B-143).
- **Mission-Critical System**—"A system whose operational effectiveness and operational suitability are essential to successful completion or to aggregate residual (mission) capability. If this system fails, the mission likely will not be completed. Such a system can be an auxiliary or supporting system, as well as a primary mission system" (DAU, 2012, p. B-144).
- **Operational Effectiveness**—"Measure of the overall ability of a system to accomplish a mission when used by representative personnel in the environment planned or expected for operational employment of the system considering organization, doctrine, tactics, supportability, survivability, vulnerability, and threat (*Defense Acquisition Guidebook*)" (DAU, 2012, p. 156).
- **Operational Scenario**—An hypothesized narrative that describes system or entity interactions, assumptions, conditions, activities, and events that have a *likelihood* or *probability* of actually occurring under prescribed or "worse-case" conditions.
- **Operational Suitability**—"The degree to which a system can be satisfactorily placed in field use with consideration to reliability, availability, compatibility, transportability, interoperability, … usage rates, maintainability, safety, human factors, habitability, manpower supportability, logistics supportability, documentation, environmental effects and training requirements (*Defense Acquisition Guidebook*)" (DAU, 2012, p. B-156). Operational suitability is characterized by one or more MOSs.
- **Phase of Operation** A high-level, objective-based abstraction representing a collection of System of Interest (SOI) operations required to support accomplishment of a system's mission. For example, a system has Pre-Mission, Mission, and Post-Mission Phases.
- **Point of Delivery**—A waypoint or one of several waypoints designated for delivery of mission products, by-products, or services.

- **Point of Origination or Departure**—The initial starting point or location of a mission.
- **Point of Termination or Destination**—The final destination of a mission.
- **Scenario**—"A specific sequence of actions that illustrates behaviors. A scenario may be used to illustrate an interaction or the execution of a use case instance" (OMG, 2006, p. 244). Also see Use Case Scenario.
- **Sequence Diagram**—"A diagram that depicts an interaction by focusing on the sequence of messages that are exchanged, along with their corresponding event occurrences on the lifelines. Unlike a communication diagram, a sequence diagram includes time sequences but does not include object relationships. A sequence diagram can exist in a generic form (describes all possible scenarios) and in an instance form (describes one actual scenario). Sequence diagrams and communication diagrams express similar information, but show it in different ways" (OMG, 2006, p. 244).
- **System Effectiveness**—A *quantitative* measure of a system's capability to accomplish a specified mission outcome and performance-based objectives.
- **Task Order**—A document that: (1) serves as a triggering event to initiate a mission, (2) defines mission objectives, and (3) performance-based outcomes.
- **Time Requirements**—"Required functional capabilities dependent on accomplishing an action within an opportunity window (e.g., a target is vulnerable for a certain time period). Frequently defined for mission success, safety, system resource availability, and production and manufacturing capabilities" (MIL-STD-499B Draft, p. 41).
- **Timeline Analysis**—"An analytical task conducted to determine the time sequencing between two or more events and to define any resulting time requirements. Can include task/time line analysis. Examples include:
  - A schedule line showing key dates and planned events.
  - An engagement profile detailing time-based position changes between a weapon and its target.
  - The interaction of a crew member with one or more SUBSYSTEMS" (MIL-STD-499B Draft, p. 42).
- **Use Case (UC)**—A statement that expresses an outcome-based capability the User requires from a system, product, or service to enable their achievement of a specific mission or task objective.
  "A use case is the specification of a set of actions performed by a system, which yields an observable result that is, typically, of value for one or more actors or other stakeholders of the system" (OMG, 2006, p. 248).
- **Use Case (UC) Diagram**—"A diagram that shows the relationships among actors and the subject (system), and use cases" (OMG, 2006, p. 248).
- **Use Case (UC) Scenario**—A situation of probable set of conditions a MISSION SYSTEM UC may encounter in its OPERATING ENVIRONMENT that requires a unique set of capabilities to produce a desired result or outcome. Scenarios include considerations of how a User or threat might apply or misapply, abuse, or misuse a system, product, or service. Also see Scenario.
- **Waypoint**—A geographical or objective-based point of reference along a sequence of mission steps to mark progress and measure performance along a Mission Event Timeline (MET).

5.2 APPROACH TO THIS CHAPTER

Chapter 5 focuses on *missions* that provide the foundational concepts for conceptualizing, formulating, and developing systems, products, or services. Although the term has a *military* connotation due to its application, it applies to any type of HUMAN SYSTEM (Chapter 9).

- Enterprise Systems consist of organizational elements—Divisions, Departments, Branches—such as hospitals, universities, and the government. Engineered Systems such as Automotive, Transportation, Energy, Medical, Communications, Financial, Aerospace and Defense (A&D) perform missions.

Conceptually, System Engineering and Development (SE&D) concepts, principles, and practices are generally universal across all types of system, product, or service development with one key exception. Although similar, there are differences in *how* User operational needs are identified and analyzed in systems, and products are developed for the *Consumer* versus *Contract*-based System Development. Since the identification and analysis of User operational needs are a key first step in System definition, we begin there.

Given the distinction between *consumer* versus *contract*-based development and the identification of User operational needs, we shift our focus to establishing the system, product, or service's mission definition methodology. This methodology defines how the User expects conduct missions to accomplish Enterprise System missions. Within the methodology, we introduce the concepts of User Stories, UCs, and Scenarios that enable us to identify, derive, and elaborate system capabilities as the basis for system, product, or service specification requirements. We conclude our discussion with a focus on how to develop UCs and Scenarios.

Let's briefly discuss the differences in *Commercial versus Contract Development*.

5.3 COMMERCIAL/CONSUMER PRODUCT VERSUS CONTRACT SYSTEM DEVELOPMENT

In general, the System Development Workflow Strategy (Chapter 12) follows a basic User Operational Needs: Analysis → Specification Design → Build → Integrate and Test workflow progression over time. However, two distinct models emerge between two predominant development strategies

in the industry—consumer product development versus contract system development as shown in Figure 5.1.

Observe how Commercial/Consumer Product Development shown on the left side of Figure 5.1 assesses the consumer marketplace, identifies PRODUCT capabilities and features, and develops the SYSTEM/PRODUCT—for a speculative ROI—then iterates the *evolving* and *maturing* Developmental Configuration (Chapter 16) through a series of test marketing cycles to arrive at the final System Design Solution. This ultimately leads to *large-scale* or *mass* production which can range into millions of units such as smartphones.

In contrast, Contract System Development analyzes User operational needs and missions; identifies, bounds, and specifies capabilities; develops the SYSTEM/PRODUCT; and performs User acceptance. User acceptance based on contract criteria does not mean that the Developmental Configuration of the System/Product can be *cost-effectively* produced. As a result, a few units may be produced via Low Rate Initial Production (LRIP) for field use. After an initial phase of field usage, a contract may be awarded to refine and improve the design for cost-effective, *large-scale, or mass* production that may range from a few dozen units to several thousand units such as military systems.

In both cases, *marketplace feedback* for consumer products or future *User operational needs* for contract-developed systems may lead to the end for upgrades on the SYSTEM/PRODUCT. This typically occurs in the form of new models for consumer products or contract-based upgrades for retrofit into existing field units of contract-developed SYSTEMS/PRODUCTS. From an SE perspective, both employ the same System Development Processes Workflow (Figure 12.2); however, each has their own nuances based on their respective marketplaces.

Commercial/Consumer Product versus *Contract-Based* system, product, or service development both require innovation and creativity but two different applications of *Systems Thinking* (Chapter 1).

5.3.1 Commercial/Consumer SYSTEM or PRODUCT Development

Commercial/Consumer SYSTEM or PRODUCT Development invests in developing new systems, products, or services in anticipation of an ROI for its Stakeholders and market share. This requires:

- Understanding the global marketplaces, segments, and niches; Users and End Users—their emotional

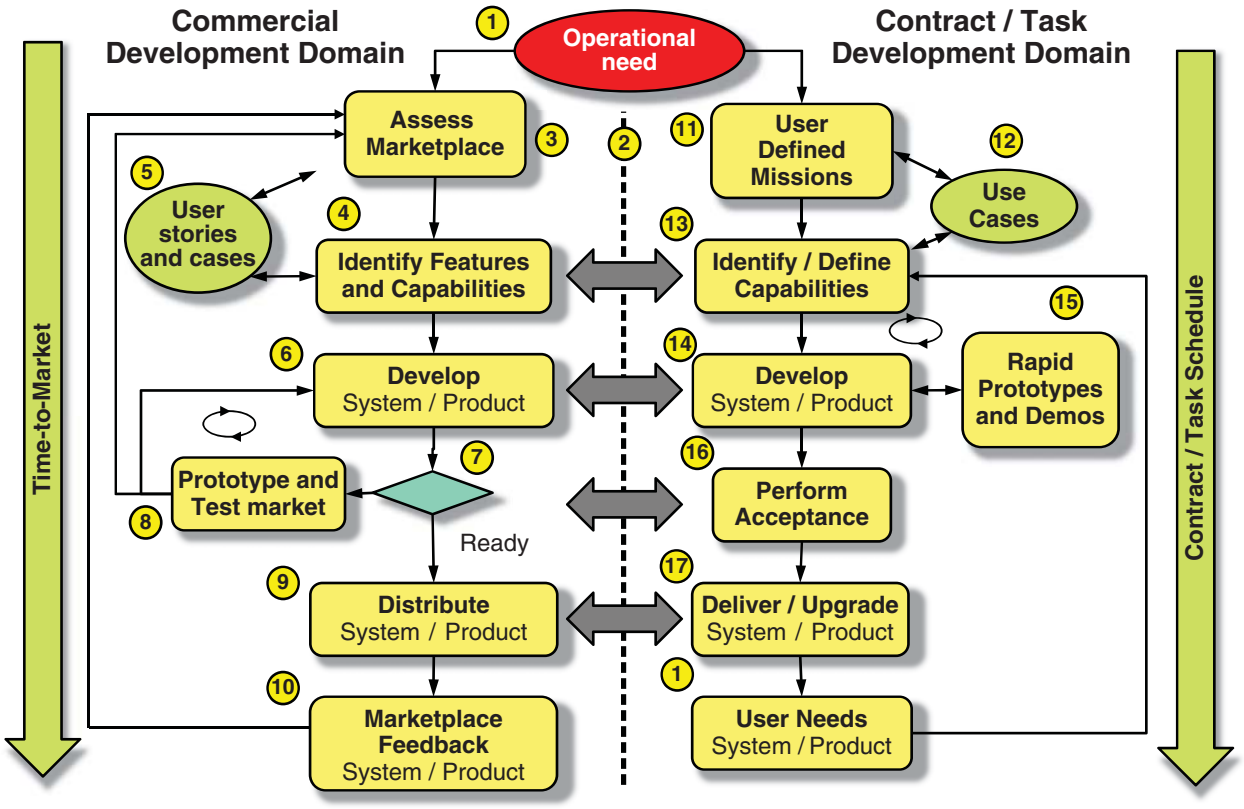


Figure 5.1 Generalized Comparison of Commercial versus Contract System Development.

tendencies that motivate them to purchase—contexts of usage; competition; and statutory/regulatory restrictions.

- Anticipating, innovating, developing, and applying *new* or *emerging* technologies to create new products in a *fast-paced, time-to-market* environment before the competition seizes on the Opportunity Space.

Chapter 4 introduced the concepts of Problem, Opportunity, and Solution Spaces. Commercial/Consumer System or Product development involves recognizing customer perceived or actual Problem Spaces as Opportunity Spaces and responding with affordable systems, products, or services to fulfill those operational needs.

Commercial/consumer systems, products, or services are often referred to as Commercial-Off-the-Shelf (COTS) vendor products that are available for ordering via catalogs, Web sites, or marketing organizations. If the vendor is willing to customize an existing COTS product for fee to respond to an external User's—customer's—specification requirement, the modified COTS product is referred to by governmental organizations as a Non-Developmental Item (NDI).

Referral For additional information about COTS and NDI, refer to Chapter 16 Configuration Identification and Component Selection Strategy.

Commercial/consumer systems, products, or services are outcome driven via metrics such as customer satisfaction, sales, and price points, performance, profitability, et al.

When commercial industry or government organizations require development and installation of specialized systems, products, or services such as buildings, machinery, vehicles, aircraft, et al that may not be readily available in the marketplace. Problem Space, the Solution Space is *Contract-Based System Development*.

5.3.2 Contract-Based System Development

In contrast, Contract-Based System Development originates from a single System Acquirer source that acquires a system, product, or service. The Acquirer may be the actual User, a procurement representative within the User's organization, or an external organization serving as the User's technical and procurement representative. Contract-Based System Development may occur within a commercial organization for procurement of large, capital investment items such as buildings, and machinery.

Contract-Based System Development is:

- Typically mission driven to develop an Engineered System such as facilities, machinery, manufacturing equipment, and networks, to support Enterprise System missions.
- Dependent on customer satisfaction concerning results of performance-based outcomes such as KPPs, MOEs and MOS introduced later in this chapter.
- Usability of the Equipment and handling characteristics.

Given these two backdrops concerning Commercial versus Contract System Development, let's begin our discussion of the System Development Workflow Strategy.

5.4 USER OPERATIONAL NEEDS IDENTIFICATION

The first critical step in system definition is to *identify* and *understand* who the key Stakeholders—Users and End Users (Chapter 3)—of a system, product, or service are, their System usage roles, decision-making roles, and relationships.

Once these are identified, the next step is to identify their specific operational needs. *Operational needs* are often thrown around to impress people. The so-called operational needs identified are actually symptoms of an underlying Problem or Issue Space. To exacerbate the challenge, Stakeholders many times specify a *need-based solution* such as new computer, new software, and new car, et al, instead of identifying the central problem or issue via a problem statement that may lead to one of those solutions. As a result, Operational Needs is an *abstract* term that has a context relative to the Stakeholder and requires further delineation. So, *what does the term mean*? Let's begin by differentiating some semantics:

Step 1—Stakeholders have *challenges* such as Problem, Issue, or Opportunity Spaces such as threats, and barriers that need to be addressed.

Step 2—SEs and other trained professionals perform a *Needs Analysis* in collaboration with the User community to understand the problem, issue, or opportunity to be addressed, root cause, and its boundaries.

Step 3—Results of the Needs Analysis express the true operational need in the form of a *Problem or Issue Statement* (Chapter 4) that Stakeholders agree by consensus to be the underlying motivation for their actions.

Step 4—The Problem Statement leads to a set of recommendations in the form of *Solution Spaces* with performance-based outcomes and objectives required to solve it.

Step 5—Some organizations create a Capabilities Development Document (CDD),formerly the Operational Requirements Document (ORD) (DAU, 2014), and Statement of Objectives (SOO) that express Solution Space capabilities to be developed or acquired.

The steps above illustrate *why* the *ad hoc* Specify-Design-Build-Test-Fix (SDBTF)–Design Process Model (DPM) Engineering Paradigm (Chapter 2) organizations that *amateurishly* create specifications containing "wish list" requirements often result in systems, products, or services that fail to meet the Stakeholders operational needs—*System Validation*.

In general, *operational needs* may represent performance-based outcomes, system/product characteristics, usability, and factors. The question is: *how do we translate those into a set of system, product, or service capabilities that form the basis for deriving specification requirements*? SEs elaborate those outcomes via tools such as Mission Analysis, UCs and Scenarios addressed later in this chapter and Model-Based SE (MBSE) addressed in Chapters 10 and 33. The central thread through these chapters is Mission-based Outcomes → User Stories → Use Cases and Scenarios → Operational Tasks → Operational Capabilities → Performance-Based Requirements → Specifications.

This discussion leads to a question: *what approach should an SE use to identify User operational needs*?

5.4.1 Operational Needs Identification Approaches

Approaches for identifying operational needs occur in a number of forms. These include anonymous surveys, personal interviews, and Nominal Grouping Techniques (NGTs), both formally and informally. Approaches that are handled amateurishly as "check the box" exercises may identify *symptoms*, not the actual underlying *problems* or *issues*. One of the advantages of having trained professionals conduct the Needs Analysis is the ability to narrow the focus on the driving needs or problems.

Chapter 4 introduced the concepts of Problem/Opportunity Spaces and Solution Spaces. An *operational need* does not necessarily mean there is a *problem* or *issue*. It could be as simple as "wonder why someone doesn't invent a (widget) that will ...." In that context, we are dealing with an Opportunity Space. Additionally, you will discover that some "needs" are actually user non-essential "wants" or "desires" (Figure 21.4)-"shopping lists" that vaporize when budget thresholds are reached.

5.4.1.1 User Stories Agile Development addressed later in Chapter 15 employs a concept called User Stories. User Stories enable Users to write their specific need in their own words on an index card using the following statement syntax:

> "As a <type of user>, I want <some goal> so that <some reason>."
>
> —(Cohn, 2008)

Then, write Conditions of Satisfaction (COS) (Cohn, 2008) in their own words on the back of the card concerning how they would verify if their need had been satisfied. On inspection, it is tempting to accept the User Story "as is" and begin identifying requirements. However, what a Stakeholder—User or End User—writes on the card *may not be* the central problem or issue to be solved. That takes us to our next topic, *The Five Whys Analysis*.

A Word of Caution 5.1

Qualify Actual Users vs Administrative Paper Pushers

Observe the operative term *User* in User Stories. One of the fallacies of Operational Needs Analysis is failure to qualify *actual Users*. Many times, managers and executives, who are Stakeholders but not Users, are the ones interviewed to identify needs; they may be fully qualified to address needs and should be interviewed. However, the consequences of interviewing the wrong people who claim to be button pushing Users when, in fact, they are not, can be devastating for the true Users that have to actually operate and maintain the system, product, or service when it is fielded or distributed.

5.4.1.2 Five Whys Analysis One method for driving to a real problem or issue is the *Five Whys Analysis*. If you start with some condition such as an incident, event, perceived need, and ask "why" after each response, you will ultimately get to the root problem or issue. Consider the following example:

Example 5.1

5 Whys Analysis

Using Cohn's (2008) User Story syntax, let's assume a User Story states: *As a <User>, I need an XYZ computer so that I can have a high tech office*.

- SE Question #1: *Why do you need an XYZ computer?*
- User Response #1: I need to create/review and exchange documents with my customers.
- SE Question #2: *Why not use the mail or a FAX machine?*
- User Response #2: Our office has gone paperless. We have no file cabinet storage.
- SE Question #3: *Why is there no file cabinet storage?*
- User Response #3: The cost of renting space and labor cost in maintaining paper copy files is becoming astronomical.
- SE Question #4: *Why is the cost increasing?*
- User Response #4: The volume of paper documents we create and receive is increasing.

- SE Question #5: *Why is the volume increasing?*
- User Response #5: We have new regulations that require us to keep more paperwork above and beyond the systems, products, and services we develop.

In this example, the User stated that they needed an XYZ computer. Upon inspection one could certainly interpret the User Story as pointing to an operational need. However, applying the 5 Whys Analysis revealed the actual problem—increasing paperwork due to regulatory requirements while needing to reduce paper document storage space—file cabinets. In turn, this discussion raises several questions:

- How will buying one individual a computer solve the overall office problem when there are XX people in the office?
- What if one individual wants an XYZ computer and everyone else wants an ABC computer?

In summary, understand the central problem, issue, or opportunity to be solved. User Stories are fine for identifying *wants/desires*, but should be validated through *every* User by a 5 Whys Analysis. Only then can the true operational need be identified.

As noted earlier, the term *operational need* is abstract and could have many contexts. For example:

- **Operational Needs Context #1**—Consumers purchase systems, products, or services (TVs, smartphones, computers, refrigerators, freezers) based on *key characteristics* they value as criteria for a buying decision.
- **Operational Needs Context #2**—Enterprises acquire systems, products, or services that require capital expenditures that align with their mission objectives. For example, develop a spacecraft for an XX (time) duration mission to fly astronauts to Mars for scientific studies and return them safely to Earth.

What tools can we use to drive out these outcomes and key characteristics? This brings us to our next topic, *Quality Functional Deployment (QFD)*.

5.4.1.3 QFD One tool that is useful in identifying User operational needs is QFD. *What is QFD?* Dr. Yoji Akao (1990, p. 5), one of the co-founders of QFD, defines it as:

> Converting the consumers' demands into quality characteristics and developing a design quality for the finished product by systematically deploying the relationships between the demands and the characteristic, starting with the quality of each functional component and extending the deployment to quality of each part and process.

QFD places special emphasis on the Voice of the Customer (VOC), priorities, and the *value*s customers place on the key characteristics of systems, products, or services they purchase and employ to perform their jobs or missions. *Quality* in the context earlier represents attributes of a system, product, or service as discussed in Chapter 3. Examples include performance, aesthetics, durability, shelf life, usability, reliability, and maintainability.

Unfortunately, people today often associate QFD with a rooftop matrix originated from the 1980s known as the House of Quality (HoQ) that employed terms such as Whats, Hows, How Much, and Whys, to depict customer value relationships. The paradigm is promulgated by engineering textbooks that continue to present the HoQ as the focal point for QFD. Mazur (2014) indicates that HoQ charts, now referred to as Classical QFD, are no longer used. Modern QFD employs *seven management and planning tools* such as spreadsheets to work with Users to identify key characteristics for system product or service development.

5.4.1.3.1 QFD and User Requirements When collaborating with the User community to identify potential needs, you can expect to receive a diverse range of User, not specification, requirements—*wants, desires, must-haves, and nice-to-haves*. Some are important to some Users and not to others. Ultimately, when we develop specification requirements for the system, product, or service, every User requirement may not be *realistically achievable* within the User's schedule, budget, technology, and risk constraints. (Figure 21.4) Ultimately, we need to reduce the final list of requirements down to *essential* requirements.

A Word of Caution 5.2

Needs versus Requirements

Observe the phrase "… collaborating with the User community to identify potential needs …" It says *needs*, not *requirements*. In general, needs are bounded and specified by requirements. This is a key point. If you ask a User for requirements and they provide them, they will expect to see those requirements stated in a specification.

Our objective here is to simply collect User needs via User Stories (Chapter 15) and methods for analysis and assimilation as inputs for deriving the true Operational Need(s)—Problem, Issue, or Opportunity Space. Then, using the need as the basis for identifying capabilities that will enable us to derive specification requirements for a system, product, or service.

Essential requirements are typically a subset of the overall data set of requirements Users identified. This process is typified by the illustration shown in Figure 21.4 and Principle 21.1—What the User Wants, Needs, Can Afford, and Is Willing to Pay.

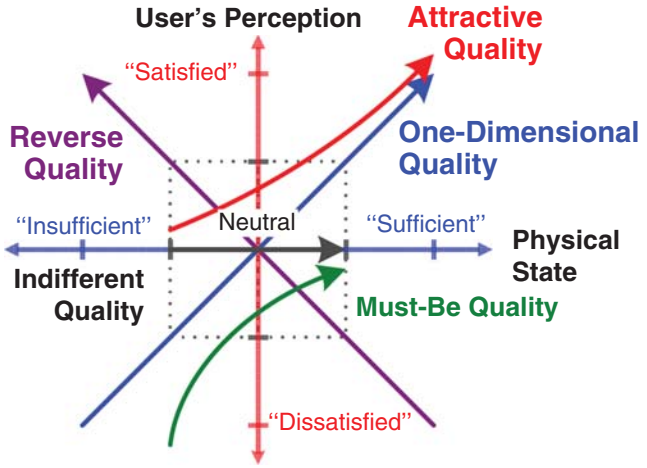


Figure 5.2 Kano's Model of Customer Satisfaction—Recent Developments (Source: Zultner and Mazur (2006, p. 3). Used by permission.)

A key question emerges from this discussion: *how do we distill the total set of User requirements down to essential requirements*? Traditionally, SE has often categorized User requirements using terms such as Mandatory, Desired, Nice-to-Have, Don't Care, and Not required categories. QFD offers a similar approach.

In the late 1970s and early 1980s, Dr. Noriaki Kano developed the Kano Model for Customer Satisfaction. Early QFD research focused on the concepts of *expected* and *exciting* requirements.

In 2006, Zultner and Mazur (2006) addressed recent developments in the Kano Model shown in Figure 5.2. Their graphic associated table value scales enable analysts to assess *customer values* for specific *quality attributes* of a system, product, or service. The graphic illustrates two intersecting axes representing the:

- **Physical State** (Horizontal Axis)—a Quality Attribute such as performance, usability and reliability, in accomplishing a mission or task, and the User's view of its sufficiency—Insufficient or Neutral.
- **User's Perception** (Vertical Axis) of the product for specified Physical State—Quality Attribute.

Several key points concerning the graphic that you should note:

- **Exciting Requirements** are:
  - "… generally *unknown* to customers and they generally will not mention them – but when they see them, they really like them" (Zultner and Mazur, 2006, p. 4).
  - "Sort of "out of the ordinary" functions or features of a product or service that cause "wow" reactions in customers. Exciting requirements are also usually *invisible* unless they become *visible* when they are fulfilled and result in customer satisfaction; they do not leave customers dissatisfied when left unfulfilled" (QFD Institute, FAQ, 2013).
- **Expected Requirements** are:
  - "Assumed by the customer, so they don't mention them – unless they have been recently disappointed. These are "deal breakers" for customers. Often they would not consider products lacking these requirements" (Zultner and Mazur, 2006, p. 4).
  - "Essentially basic functions or features that customers normally expect of a product or service. Expected requirements are usually *invisible* unless they become *visible* when they are *unfulfilled*" (QFD Institute, FAQ, 2013).
- **Must-Be Requirements** are mandatory.
- **Desired Requirements**:
  - Are those Users identified when you ask them what they want? "Customers are willing to trade-off performance on one for less on another" (Zultner and Mazur, 2006, p. 4).
  - "These requirements satisfy (or dissatisfy) in proportion to their presence (or absence) in the product or service. Fast Delivery would be a good example. The faster (or slower) the delivery, the more they like (or dislike) it" (Mazur, p. 5).
- **Indifferent Requirements**—Are don't cares from the customer's perspective.
- **One-Dimensional Quality**—As the quality of the Physical State increases, so does the User's Perception.
- **Reverse requirements** are:
  - "Those the customer would prefer not to have …"
  - "… dissatisfying and the absence is satisfying" (Zultner and Mazur, 2006, p. 4).

Referral The descriptions earlier are very brief and high level. Refer to technical papers by Mazur (2003) and Zultner and Mazur (2006) for more detailed discussions of QFD applications in these areas.

Author's Note 5.1

User Needs Analysis Application Contexts

The User Needs Analysis discussion earlier has two application contexts:

1. **Application Context #1**—A Commercial/Consumer Product Developer performs the Needs Analysis in-house or employs the services of a professional organization or consultant to help specify and bound marketplace needs for development in-house or by an external System Developer.

2. **Application Context #2**—A User employs the services of a System Acquirer or professional organization or consultant to help specify and bound *mission* and *system* needs that provide the basis for a System Requirements Document (SRD) or Capability Description Document (CDD) used in a Request for Proposal (RFP) for System Development.

In summary, employ QFD or other methods used to *distill* initial User requirements to only *essential* requirements that (1) represent a consensus of the Stakeholders—Users and End Users—and (2) are realistically achievable within their development schedules, budgets, existing technologies, and level of risk.

Referral Remember—The discussion earlier is just one example of the power of QFD methods and tools for providing quality and value to the User driven by the VoC. For additional information about QFD, refer to:

- The QFD Institute—www.qfdi.org
- QFD Case Studies—www.mazur.net

Heading 5.1 Our discussion to this point has focused on the similarities yet differences in Stakeholder Needs Analysis for Commercial/Consumer Product Development versus Contract-based System Development. The remaining sections of the chapter focus on System Definition that will ultimately lead to the identification of system, product, or service capabilities.

5.5 MISSION ANALYSIS

Principle 5.1

Mission Success Principle

Mission success requires five key elements: a purpose; timely, sustainable resources; a reasonably achievable outcome-based performance objective(s); a MET; and a willingness to perform. Where there is no *willingness to perform*, the system suffers neglect, becomes technologically obsolete, and falls into disrepair.

"Success always comes when preparation meets opportunity" (Hartman) is a quote attributed to Henry Hartman that exemplifies the cornerstone of mission success. Mission success requires insightful planning, and *solving* the *right* problem with the *right system* at the *right* time.

Every system, product or service exists as an ENABLING SYSTEM for the purpose of fulfilling the operational needs of its Users to accomplish their Enterprise System missions or tasks (Principle 3.7). *How* a system, product, or service's Stakeholders—Users and End Users—intend to deploy, operate, maintain, and sustain to accomplish missions or tasks establishes the analytical framework for System Definition.

As a result, we begin with Mission Analysis and its methodology.

Author's Note 5.2

Mission analysis often conveys a *military* connotation to many people. This is a paradigm; *do not* succumb to its connotations. Example - system missions include development of:

- Medical Devices—Intravenous drug delivery, Magnetic Resonance Imaging (MRI), and heart monitors
- Public Services—Police, fire, water, sewer, and refuse collection
- Energy—Exploration and extraction, Seismic equipment, oil rigs, and pipelines, et al
- Telecommunications
- Transportation—Inter-modal, subways, vehicles, and airlines
- Internet—Web sites
- Businesses, Restaurants, movies, and shopping centers

5.5.1 Mission Analysis Methodology

Enterprise and Engineered Systems missions range from simple tasks such as writing a letter to performing highly complex International Space Station (ISS) operations and managing a government. Regardless of application, Mission Analysis requires a methodology that enables us to *bound* and *specify* the Solution Space and its KPPs. The methodology consists of the following steps:

Step 1: Define the Enterprise or Engineered System Mission.
Step 2: Derive the Mission Operational Requirements.
Step 3: Develop the Mission Profile.
Step 4: Identify and Define the Mission Phases of Operation.
Step 5: Perform a Mission UC Analysis.
Step 6: Develop the MET.
Step 7: Identify MISSION RESOURCES.
Step 8: Identify System Quality Factors.
Step 9: Assess and Mitigate Mission and System Risk.
Step 10: Iterate Steps #1–9, as necessary.

Author's Note 5.3

Missions versus Systems

Observe usage of the terms *mission* versus *system* earlier. Remember—One or more missions may be required to eliminate an Enterprise Problem Space or seize an Opportunity Space. The physical implementation of a mission

may require one or more MISSION SYSTEMS and one or more ENABLE SYSTEMS. For example, if your mission is to keep your lawn beautiful, you (MISSION SYSTEM) will need tools such as a lawn mower (ENABLING SYSTEM), a lawn edger (ENABLING SYSTEM), and a blower (ENABLING SYSTEM).

Note that the Mission Analysis Methodology is based on system, product, or service development. If the system already exists, Steps #1–10 are still applicable; however, the decision process becomes one of selecting the *right* system, product, or service for the mission.

Let's explore each of these steps in more detail.

5.5.1.1 Step 1: Define the System, Product, or Service Mission

Principle 5.2

Mission Statement Principle

A *mission statement* should specify *one and only* one outcome to be achieved and supported by one or more performance-based objectives.

Principle 5.3

Mission Objectives Principle

Each *mission* should be *bounded* and *specified* by one or more performance-based objectives.

The first step in planning missions is to *eliminate* or *minimize* a Problem Space or *exploit* an Opportunity Space. Once the mission is identified, *mission objectives* for accomplishing the mission outcome need to be established.

Mission objectives represent User performance-based outcomes that contribute to accomplishment of a mission outcome within a specified timeframe for a prescribed Operating Environment. Mission objectives also serve as the foundation for selection or acquisition of systems, products, or services to support a special mission or different types of missions. Consider the following example:

Example 5.2

Family Vehicle Mission Objectives

A family requires a vehicle to support their many activities—different types of missions—that include general transportation, grocery shopping, biking, kayaking, or transporting family members to school in all types of road and weather conditions. As a result, the family establishes performance-based mission objectives—such as 6 passengers, storage space, and fuel economy—that will serve as the basis for acquiring a vehicle to meet their operational needs.

Observe that the family *did not* identify a specific type of vehicle, only mission objectives.

The term *mission objectives* has a human connotation. Since humans are the mechanisms for planning, orchestrating, and performing missions, *mission objectives* or *tasks* apply directly to the SOI and subsequently its PERSONNEL Element. Recall that an SOI, as a performing entity, is comprised of several types of System Elements—PERSONNEL, EQUIPMENT, MISSION RESOURCES, PROCEDURAL DATA, and SYSTEM RESPONSES. To illustrate this point, consider a heavy construction company Mini-Case Study 5.1.

Mini-Case Study 5.1

Heavy Construction Company

A heavy construction business consists of several SOIs such as bulldozers, heavy rock haulers, front-end loader tractors, and water trucks. Each of these SOIs—Bulldozer System, Heavy Rock Hauler System—as a *performing entity*, is assigned a mission consisting of a PERSONNEL Element to operate the EQUIPMENT Element that consumes and processes MISSION RESOURCES such as fuel in accordance with its PROCEDURAL DATA Element concerning how to safely operate the equipment and produces System Responses Element results to achieve performance-based outcomes.

As a *performing entity*, each SOI requires harmonious integration of its respective System Elements to accomplish the assigned mission. The "production line" Supply Chain of SOIs and their outcomes enable the construction company to extract rocks from a quarry and transport the rocks to a processing facility that crushes the rocks in various sizes for commercial sale and distribution to Users.

Government organizations such as the military develop a *Mission Needs Statement* (MNS) or a SOO for acquiring systems to fill Solutions Spaces related to one or more missions (Problem Space). NASA, for example, works with principal investigators to establish science objectives—mission objectives—for development of experiments and systems NASA manifests and deploys to space.

Principle 5.4

Mission Operating Constraints Principle

Each mission should bounded by *operating constraints* that limit its acceptable usage—safety, operating range, affordability, and environmental conditions.

Throughout a mission, an SOI—MISSION SYSTEM or ENABLING SYSTEM—interacts with external systems within its OPERATING ENVIRONMENT. These systems may include *friendly*, *benign*, *hostile*, *adversarial*, threats, encounters, and interactions and OPERATING ENVIRONMENT conditions

that may range from *benign* to *harsh*. Specify at least one or more *Operating Constraints* objectives that establish boundary conditions that restrict mission operations. For example, time of day, day of week, and *single-use* versus *multi-use* applications.

Mission Reliability Principle

Each mission should be bounded in terms of the *mission reliability* to be achieved.

Principle 5.5

HUMAN SYSTEMS, despite careful planning and execution, are not infallible. The question is: *Given resource constraints, what is the minimum level of success you are willing to accept to provide a specified ROI*? From an SE perspective, we refer to the level of success as *mission reliability*. Mission reliability is influenced by internal EQUIPMENT failures or over/under tolerance conditions, human operator performance (judgment, errors, fatigue) and interactions with OPERATING ENVIRONMENT entities and threats.

Mission reliability is defined as the conditional probability that a system with a given operating condition will successfully accomplish a mission of a specific duration in a prescribed OPERATING ENVIRONMENT without failure. Depending on the system application, 100% *mission reliability* may be prohibitively expensive, but 95% *mission reliability* may be affordable.

Operating Constraint Trade-Offs

Author's Note 5.4

Since increased reliability ultimately has a cost, establish an initial reliability estimate as simply a starting point and compute the cost. Some Acquirers may request a CAIV plot of cost as a function of capability to determine what level of capability or reliability is affordable within their budgetary constraints. Figure 32.4 introduced later provides an example.

You may be thinking that *mission reliability* refers to EQUIPMENT Element reliability—that is incorrect. Mission reliability provides the basis for deriving MISSION SYSTEM and ENABLING SYSTEM reliabilities that will ultimately be allocated to and specified in their respective System Performance Specifications (SPSs). The reliability for the EQUIPMENT Element within a MISSION SYSTEM and ENABLING SYSTEM is derived from the SPS reliability. Figure 5.3 illustrates these dependencies. Observe that overall *mission reliability* is a function of:

- MISSION SYSTEM Reliability, which is a function of its System Element reliabilities:
  - PERSONNEL Element Reliability
  - EQUIPMENT Element Reliability
  - MISSION RESOURCES Reliability
  - PROCEDURAL DATA Reliability
  - SYSTEM RESPONSES Reliability
- ENABLING SYSTEM Reliability, which is a function of its System Element reliabilities:
  - PERSONNEL Element Reliability
  - EQUIPMENT Element Reliability
  - MISSION RESOURCES Reliability
  - PROCEDURAL DATA Reliability
  - FACILITIES Element Reliability
  - SYSTEM RESPONSES Reliability

Each of the System Element reliabilities is dependent on other reliabilities such as training and information accuracy.

In summary, at a *minimum*, there should be at least:

- One or more performance-based objectives for each mission outcome to be accomplished and its expected level of performance to be achieved.
- One mission reliability objective.
- One operating constraints objective.

5.5.1.2 Step 2: Derive the Mission Operational Requirements

Once the mission outcomes and objectives are identified, the next step is to derive mission requirements. Each type of Enterprise mission is documented by a Strategic Plan and supporting Tactical Plans. These plans identify, define, and document specific mission requirements.

Mission requirements are sometimes referred to as *operational requirements*. In general Operational requirements express a User's perspective of how they intend to:

- Integrate the propose system, product, or service as an asset into their Enterprise System to accomplish its missions
- Deploy, operate, maintain, sustain, retire, and dispose of a system, product, or service for specific mission applications

Documents that capture operational requirements include what is explicitly referred to as an ORD and now a CDD and others types of documents.

5.5.1.3 Step 3: Define the Mission Profile

User Mission Profile Principle

Every system, product, or service should include a characterization of its User mission profiles.

Principle 5.6

A mission begins with a *point of origination* and terminates at a *point of destination*. As end-to-end boundary constraints, the challenge question is: How do we get from the *point of origination,* Point A, to the *point of destination,* Point B? We begin by establishing a strategy that leads to a *mission profile* such as the one shown in Figure 5.4 for a

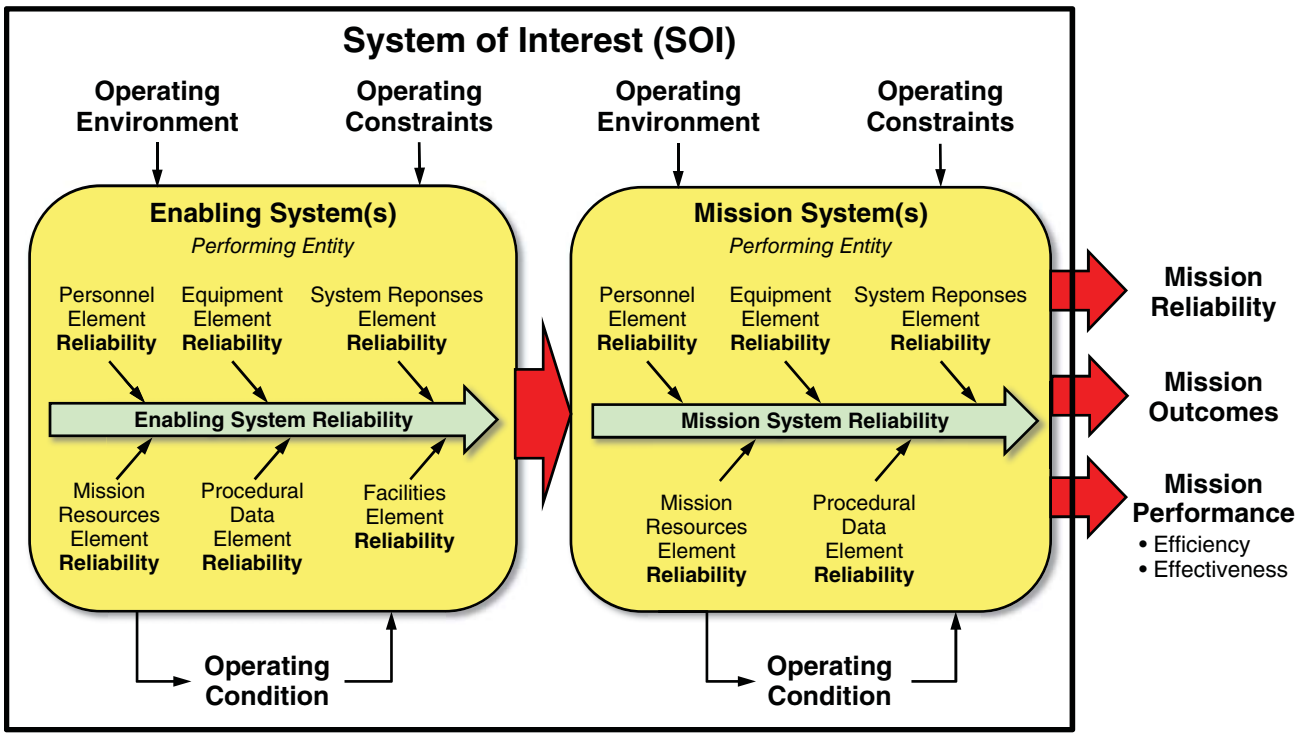


Figure 5.3 Understanding SOI Mission Reliability with MISSION SYSTEM and ENABLING SYSTEM Performance Effectors

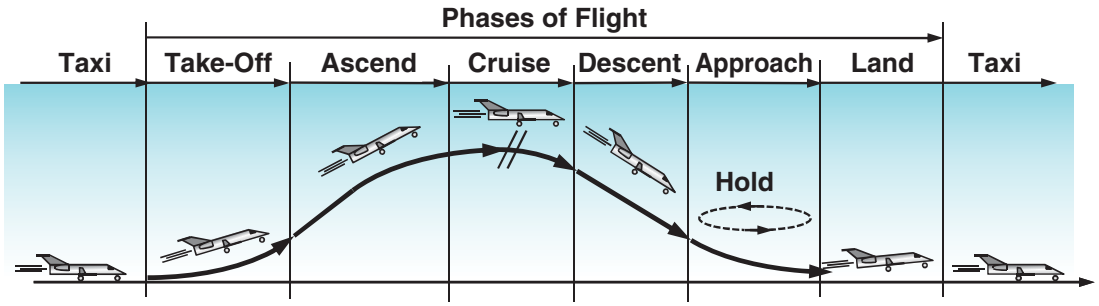


Figure 5.4 Commercial Aircraft Mission Profile Example

commercial aircraft. Observe that Take Off, Ascend, Cruise, Descend, Approach/Hold, and Land are considered Phases of Flight. In the context of System UCs introduced later, the Mission Profile is a graphical illustration of a System UC of an aircraft performing a mission. A UC is characterized by a *main success scenario* (Cockburn, 2001) when everything performs as planned—Take -Off, Ascend, Cruise, Descend, Approach/Hold, and Land.

5.5.1.4 Step 4: Identify and Define the Mission Phases of Operation

Principle 5.7

Mission Phases of Operation Principle

HUMAN SYSTEMS—Enterprise and Engineered—have at least three primary phases of operation: Pre-Mission, Mission, and Post-Mission. An interim phase, such as STORAGE, may be required for some systems between missions.

HUMAN SYSTEMS, especially *cyclical* systems, sequence through three sets of objective-based actions to accomplish a mission: (1) prepare for the Mission, (2) conduct or perform the Mission, and (3) perform Post-Mission actions and processing. We characterize these objectives as the Pre-Mission, Mission, and Post-Mission phases of operation. For those systems to be placed in storage following the Mission, an interim Storage Phase may be added.

A key question emerges during mission definition: *When do the* Pre-Mission, Mission, and Post-Mission *Phases of Operation begin and end*? To aid our discussion, consider Figure 5.4 representing the Mission Profile:

- Does an aircraft's Pre-Mission begin when it starts loading passengers? When it arrives from the previous flight?
- When does its Mission Phase of operations begin? When it leaves the terminal? At Takeoff?
- When does the Mission Phase end and its Post-Mission Phase begin and end? When it lands? When it arrives at the gate? When all passengers and baggage have been unloaded?

Now how do you decide which of the Phases of Flight indicated in Figure 5.4 is in the Pre-Mission, Mission, and Post-Mission Phases of operation. As a starting point, we could say that:

- The Pre-Mission Phase begins when the aircraft has been deemed ready to initiate operations for the next flight—baggage, fuel, and passenger loading.
- The Mission Phase begins when (1) the passengers and baggage are stored, (2) all maintenance actions have been completed, and (3) the aircraft has been released for flight.
- The Post-Mission Phase begins when the aircraft has arrived at the terminal gate and the gate agents and flight crew informed the passengers to safely leave the aircraft.

5.5.1.5 Step 5: Perform a Mission UC Analysis

Throughout the Pre-Mission, Mission, and Post-Mission phases, specific mission UCs and scenarios and their respective operations and tasks must be performed to accomplish the phase-based mission objectives. Therefore, the mission analysis should:

- Identify the high-level outcome-based mission tasks to be accomplished.
- Synchronize those tasks to the MET.
- Identify the performance-based task objectives.
- For each operational task, translate it into a required operational capability of the System or Product.

5.5.1.6 Step 6: Develop the MET

MET Principle

Principle 5.8 Every system, product, or service should include an MET that identifies time-dependent events that represent the *start* and *completion* of mission operations and interim waypoints required to be accomplished to achieve mission success.

Once the overall mission profile concept has been established, identify one or more *staging*, *control*, or *waypoints* along the MET to pace critical operations. A waypoint represents a geographical location or position, a point in time, or objective to be accomplished as an interim step toward the destination, as illustrated in Figure 5.5.

As an example of an MET, Figures 5.6 and 5.7 represent a NASA example of the Launch and Descent, Entry, and Landing Concepts for the NASA Mars Exploration Rovers. Observe how launch and landing operations are partitioned into key *control* or *staging events* with timeline milestones.

5.5.1.7 Step 7: Identify Mission Resources

Mission Efficiency and Effectiveness Principle

Principle 5.9 Every system, product, or service mission should be defined in terms of its *efficiency* and *effectiveness*.

Human Systems—Enterprise and Engineered—have finite resource capacities that require *efficient* and *effective* use as well as *replenishment* and *refurbishment*. Depending on the mission operating range of the system relative to its current mission application, mission analysis must consider how the system's *expendables* and *consumables* will be employed and resupplied, replaced, and replenished. Operationally, the question is: *How will the Enterprise maintain and sustain the mission from beginning to end?*

The solution begins with definition of a mission profile and development of the MET and its waypoints; however, that does not necessarily indicate that the mission performance is *efficient* or *effective*. Therefore, investigate how the mission performance can be improved. *"Why" and "what if" questions related to conducting simultaneous previous flight and next flight operations to save time and become more efficient*? As another example, consider the following commercial aircraft fuel efficiency and effectiveness example.

Aircraft Fuel Efficiency and Effectiveness Example

Example 5.3 *How could you improve aircraft fuel efficiency and effectiveness*? In general, if the aircraft is consuming fuel on the ground and not flying, it is *inefficient* and *ineffective* use of the asset. Performance improvements might include improving aircraft engine efficiency, reduction in terminal-to-takeoff time, reduction in holding pattern time, and reduction in fuel loading based on travel distance. In the case of aircraft fuel loading, every additional pound of fuel that is not required beyond safety limits to fly between airports is additional weight that must be transported thereby reducing fuel economy.

Now consider how mission strategy has an impact on Enterprise efficiency and profitably in Mini-Case Study 5.2.

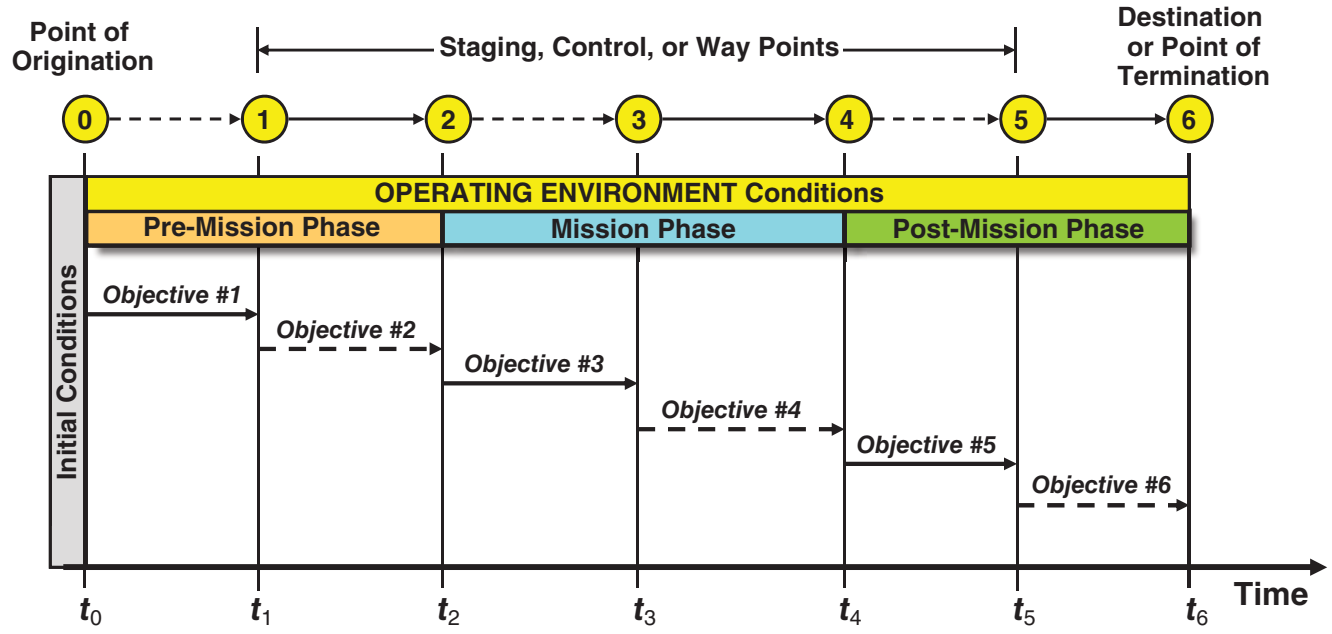


Figure 5.5 Mission Event Timeline (MET) Example

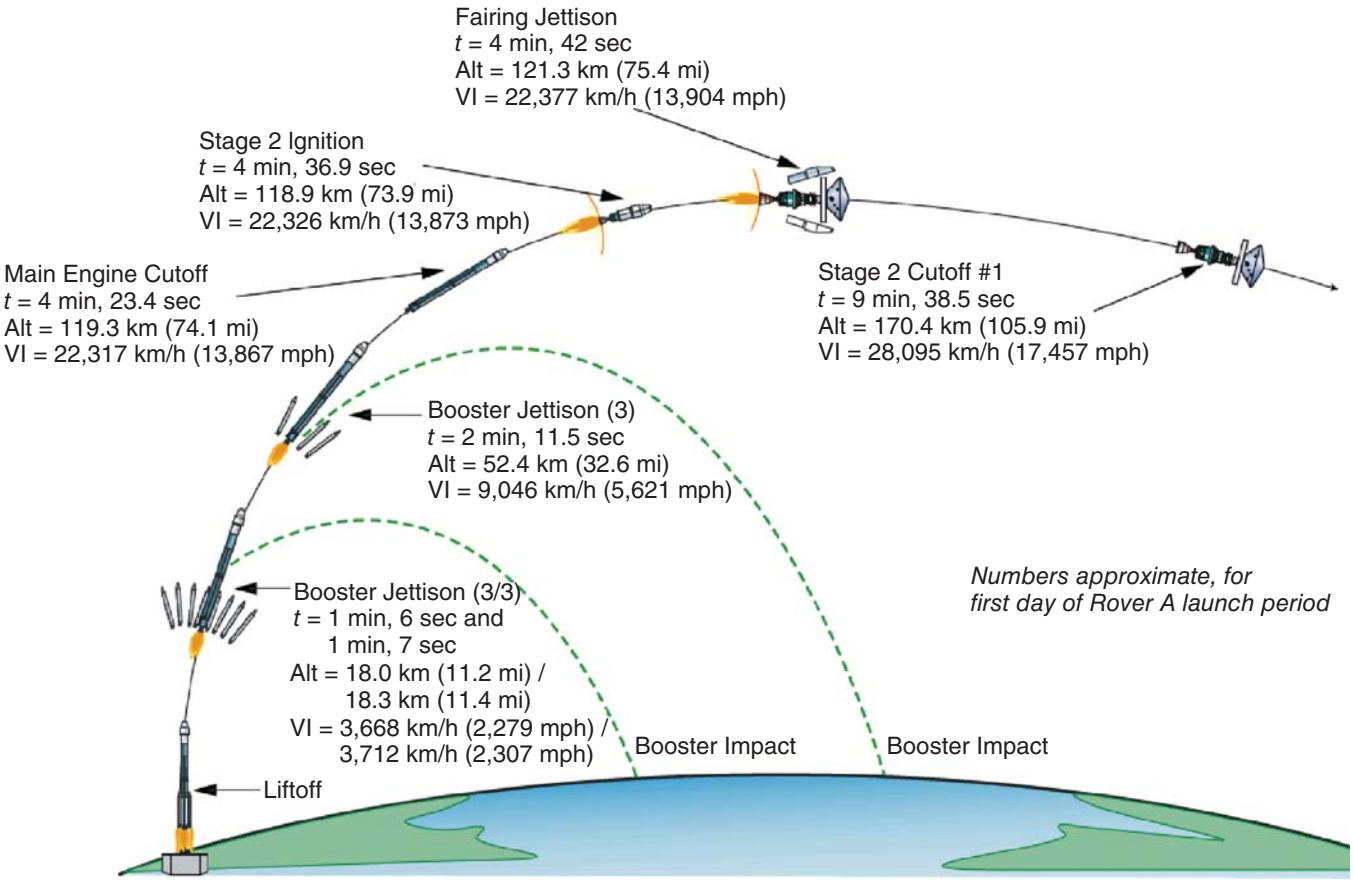


Figure 5.6 Example MET—NASA Mars Exploration Rover Launch Phases (Source: NASA (2003, p. 26).)

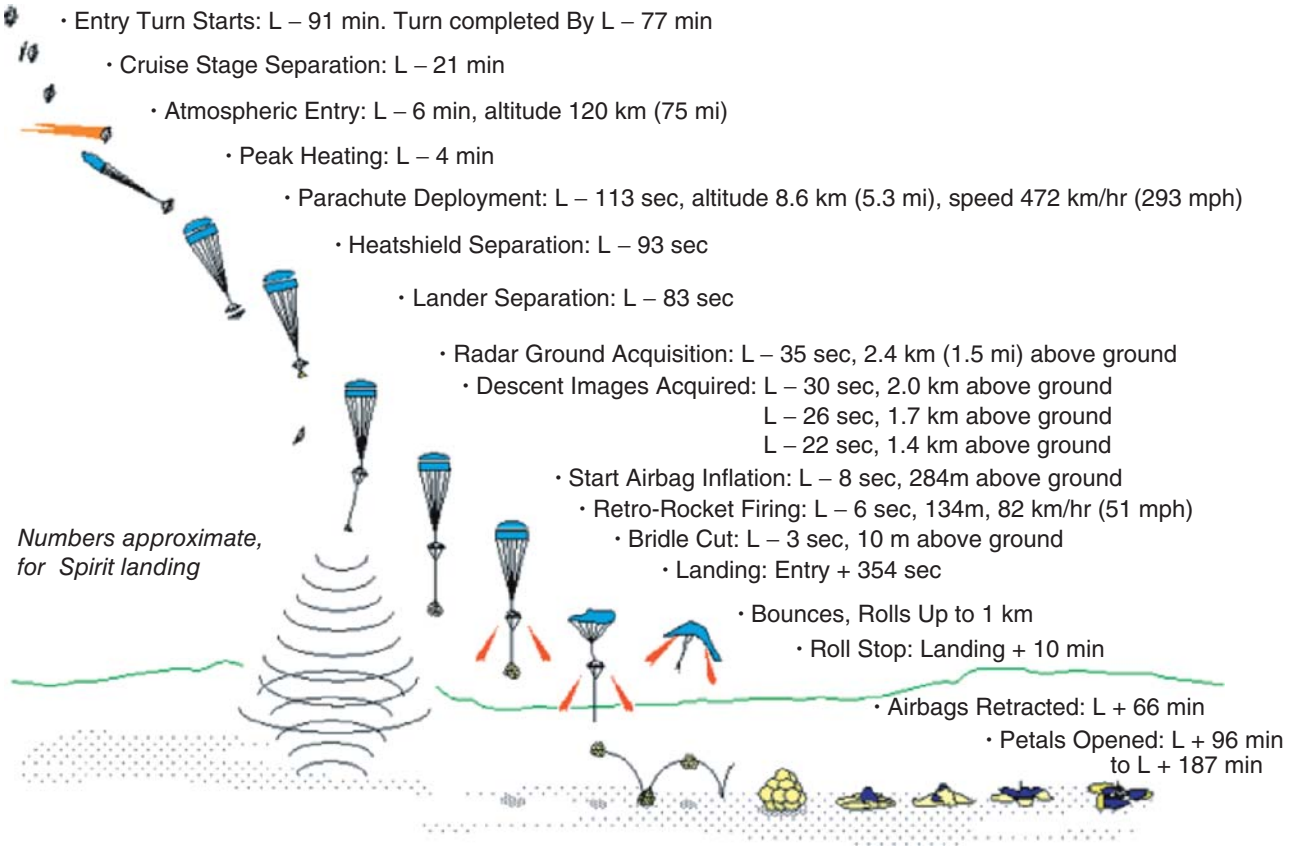


Figure 5.7 Example MET—NASA Mars Exploration Rover Entry, Descent, and Landing Phases (Source: NASA (2004, p. 33). http://marsrover.nasa.gov/newsroom/merlandings.pdf)

Mini-Case Study 5.2

Package Delivery Service Example

United Parcel Service (UPS) discovered that it could improve its delivery system performance and reduce fuel consumption. Waiting on traffic to make a left turn consumes valuable fuel and time—Problem Space—while the vehicle is idling, poses potential accident risks when crossing traffic lanes, and delays delivery performance. UPS discovered that creating a route strategy for each vehicle based on making right hand turns improved fuel consumption and delivery performance (UPS, 2012).

5.5.1.8 Step 8: Identify System Quality Factors Throughout all phases of the Mission, an SOI may be required to produce a series of behavioral responses, products, by-products, and services to satisfy *internal* and *external* requirements:

- Examples of Internal requirements include performance monitoring, resource consumption, and payload/cargo manifests.
- Examples of External requirements include establishment of rules of engagement, communications protocols, detection and avoidance strategies, and evasive tactics.

Based on the preceding discussion of mission outcomes, objectives, and requirements, we can derive outcomes, objectives, and requirements for each of the System Elements. Analytically, we establish each of the SYSTEM Elements as peers. However, the PERSONNEL Element has overall accountability for performing missions and achieving their outcomes.

Engineered Systems, as inanimate objects, are *incapable* of achieving mission objectives without some form of *reasoning* and *intervention* by their Users - operators or maintainers, either by remote control or by pre-programmed scripts and tasks to be performed. Robots, computers, and aircraft autopilots are examples.

As technology evolves, Engineered Systems will eventually possess increasing levels of the capability and reasoning. In any case, a SYSTEM must possess the capability to perform specific actions-UCs- that enable their human operators to achieve an SOI's mission objectives. That responsibility is assigned to the PERSONNEL ELEMENT to MC2 the EQUIPMENT Element to produce SYSTEM RESPONSES that accomplish mission outcomes. This leads to the question: *what objectives must be assigned to the* EQUIPMENT *Element to respond to Personnel Element MC2*?

EQUIPMENT Element objectives are derived from SOI MISSION SYSTEM or ENABLING SYSTEM objectives and synchronized with PERSONNEL MC2 objectives. For example, the mission may require the EQUIPMENT Element capabilities such as:

- Usability
- Single-Use/Multi-use Applications
- Comfort
- Interoperability
- Transportability
- Mobility
- Maneuverability
- Portability
- Growth and Expansion
- Reliability
- Availability
- Maintainability
- Producibility
- Mission Support
- Deployment
- Training
- Vulnerability
- Lethality
- Survivability
- Security and Protection
- Efficiency
- Effectiveness
- Reconfigurability
- Integration, test, and evaluation
- Verification
- Maintainability
- Disposal
- Safety

Capabilities such as these provide the basis for deriving SYSTEM or lower level ENTITY capabilities and translation into SPS requirements.

5.5.1.9 Step 9: Assess and Mitigate Mission and System Risk Some missions require the system to operate in harsh OPERATING ENVIRONMENTS that may place the system at risk to threats, not only in completing its mission but also in returning safely to its home base.

Mission assessments include considerations of system vulnerability, susceptibility, survivability, and maintainability. Most people tend to think in terms of external benign, adversarial, or hostile systems that may be threats to the system. However, since a system interacts with itself, it can also be a threat to itself as shown earlier in Figure 3.2. Consider the following examples.

Example 5.4

Risk Mitigation Example

Develop risk mitigation procedures and training to detect the conditions noted below and perform corrective actions:

- Failed automobile components, such as a blown tire, can cause a driver to lose control of the vehicle while driving.
- Failures in an aircraft's Flight Control System (FCS) or broken blades in a jet engine fan can force an emergency landing or have catastrophic consequences.

Internal failures and/or degraded performance also have *negative* impacts on system performance that ultimately translates into mission failure or degree of success. Perhaps one of the most notable examples is the Apollo 13 catastrophe. Mission analysis should identify system capabilities that are *mission critical* and may be *vulnerable* or *susceptible* to external or internal threats.

Author's Note 5.5

Failure Modes and Effects Analysis (FMEA) and Failure Modes, Effects, and Criticality Analysis (FMECA)s

Internal failure analysis is typically performed via FMEA. For mission-critical components, the FMEA may be expanded into a FMECA that assesses the degree of criticality (Chapter 34).

5.5.1.10 Step 10: Iterate Steps #1–9 as Necessary The order of Steps #1–9 is based on a set of sequence dependencies. Although they may appear to be linear, iterate the set of steps until the mission solution reaches maturity.

Heading 5.2

Our discussion to this point has focused on understanding, analyzing, and structuring a system's mission—for example, Problem Space—into Solution Spaces. This brings us to a key topic—effectiveness of the system to perform missions.

5.6 MISSION OPERATIONAL EFFECTIVENESS

A system, product, or service must be capable of supporting User missions to a level of performance that makes it *operationally effective* in terms of accomplishing Enterprise System goals and objectives, namely outcomes, cost, schedule, and risk.

Example 5.5

System Effectiveness Example

For a military system, *system effectiveness* depends on environmental factors such as operator training doctrine and tactics, system vulnerability and survivability, and threat characteristics.

If you analyze *operational effectiveness*, two key concerns emerge:

- **Concern #1**—How well the system accomplishes its mission objectives—*operational effectiveness.*
- **Concern #2**—How well the system integrates and performs missions within the User's Enterprise structure and Operating Environment—*operational suitability.*

Therefore, we need to establish metrics that enable us to analyze, predict, and measure mission operational outcomes. We do this with two key metrics: (1) MOEs and (2) MOSs. MOEs and MOSs are outcome-based measures of operational effectiveness. MOEs and MOSs "identify the most critical performance requirements to meet system-level mission objectives, and will reflect key operational needs in the operational requirements document" (DAU, 2001, p. 125).

5.6.1 MOEs

Principle 5.10

Measures of Effectiveness Principle

Mission and System success requires establishment of one or more MOEs that *quantify* mission objectives in terms of performance-based outcomes.

MOEs enable us to evaluate *how well* the system accomplished its mission objectives. MOEs are objective measures that represent the most critical measures—performance effectors—that contribute to overall mission success. An MOE characterizes the *operational effectiveness* of a system, product, or service to achieve a specific performance-based outcome. For example, Figure 5.8 provides an illustration of the variables that affect an automobile's Fuel Efficiency MOE.

Consider the following examples.

Example 5.6

MOE Example

Enterprises must continuously answer the following questions:

- Did the new vaccine reduce the incidence of XYZ in age groups under Y years of age?
- What is the airline's on-time performance for Flight XYZ?
- Did the system detect the target at the predicted range?

Example 5.7

Office Copier MOE Example

Let's assume that an Enterprise has decided to purchase a copier for its offices. A survey is conducted of the printing needs for each of the offices. Since the cost of labor is critically important, the Enterprise has decided to establish the following MOEs:

- MOE #1—Reproduce color copies at the rate of 30 ppm (pages per minute).
- MOE #2—XX cents per copy.
- MOE #3—Energy consumption—XX watts per hour.
- MOE #4—Maximum heat dissipation—YY BTUs per hour.
- MOS #5—Toner particulate size filtration.

Observe the context of the achieved MOE. The MOE represents the SOI Level mission performance for a specified outcome to be achieved. To attain this level of performance, the integrated set of SOI System Elements—Personnel, Mission Resources, Equipment, and Procedural Data—must perform together as a System. As contributory performance effectors, each System Element has MOEs that contribute to the achievement of each SOI Level MOE. We will discuss MOEs and MOPs in more detail later in Chapter 21 (Figures 20.1 and 20.2).

5.6.2 Mission MOEs versus System MOEs

Now, suppose you are a System Developer contracted to develop a system, product, or service such as the Equipment Element. You have no control over the driving habits of the User—Personnel Element within the SOI. Thus, the Equipment and Personnel Elements are primary contributory performance effectors to the SOI's Mission MOE, assuming the Mission Resources and Procedural Data Elements are accurate and adequate. Since we know that the Personnel Element interacts with the Equipment Element throughout the Mission, it has its own MOEs. This discussion brings us a key point: recognition of the difference between SOI *Mission* MOEs versus Equipment Element MOEs and Personnel Element MOEs—(Figures 10.13 – 10.16)

5.6.3 MOS

Principle 5.11

MOS

Mission and System success require establishment of one or more MOSs that quantify *how well* a system is required to perform its mission.

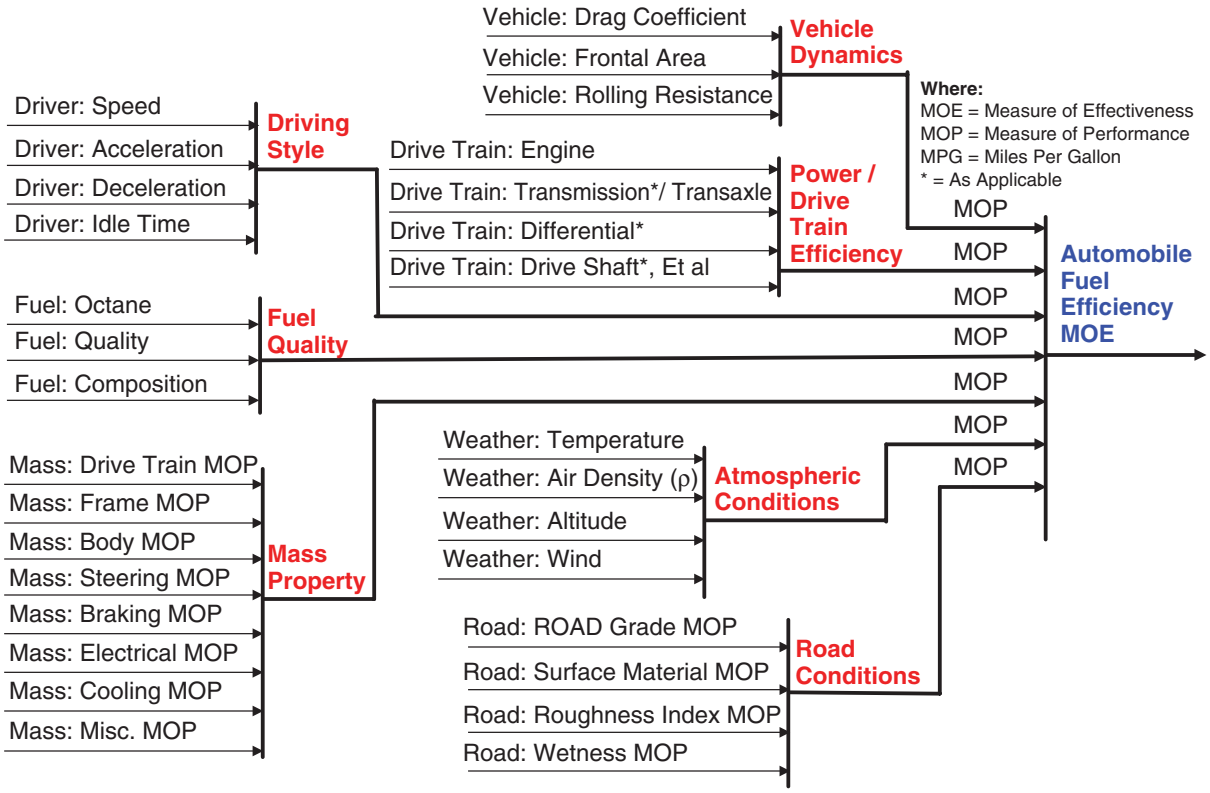


Figure 5.8 Automobile Fuel Efficiency MOE and Contributory MOPs (Source: Collaboration with Matthew Doude (2012)—Center for Advanced Vehicular Systems (CAVS), Mississippi State University.)

One of the challenges in developing systems, products, or services is developing the *right* system—validation—for specific mission applications and OPERATING ENVIRONMENT conditions. If a system, product, or service is not *suitable* for a mission application—reliable, dependable, and difficult to use or maintain—obviously, that impacts its ability to perform and achieve an outcome-based MOE. For example, special laptop computers and devices are made for harsh industrial or field environments such as dropping and spills on keyboards. We can say that these types of devices are made to accomplish specified MOEs in prescribed types of environments.

An MOS is metric that characterizes the *operational suitability* of a system, product, or service. The DAU (2001) states "… Operational suitability is the degree to which a system can be placed satisfactorily in field use considering availability, compatibility, transportability, interoperability, reliability, usage rates, maintainability, safety, human factors, documentation, training, manpower, supportability, logistics, and environmental impacts …" DAU (2012, p. 156). DAU (2001) also suggests that MOS metrics include:

> "… measurements that indicate improvement in the producibility, testability, degree of design simplicity, and design robustness. For example, tracking number of parts, number of like parts, and number of wearing parts provides indicators of producibility, maintainability, and design simplicity"
> —(DAU, 2011, p. 126).

Consider the following example.

Example 5.8

Based on the previous office copier example, the Enterprise recognizes that not only should the copier be *operationally effective* in achieving the specified MOEs, it should be *operationally suitable* for the planned office areas. This includes background noise, jam-free reliability, and energy savings on office utilities. As a result, the following MOSs were established as key metrics representing concerns to the officer workers:

- MOS #1—Maximum noise level— XX db (decibels)
- MOS #2—Number of copies reproduced without jamming
- MOS #3—Increase/decrease for Heating, Ventilation, and Air Conditioning (HVAC) costs

Example 5.8 illustrates the acquisition of a system, product, or service via consideration of its MOSs. Although

this was a simple example to introduce the concept of MOSs, *what types of decisions does a System Developer make to develop a contract-based system or a product for the commercial marketplace*? Again, this is a simple example, but consider the following MOS decisions related to PERSONNEL–EQUIPMENT Element interactions.

Example 5.9

PERSONNEL–EQUIPMENT Element Interactions Example

- What User skills, tools, and EQUIPMENT considerations impact system design concerning *how* to operate and maintain the system to achieve a specific level of performance?
- Do the control panels incorporate ergonomic designs and devices that minimize operator fatigue and errors (Chapter 24)?

One point we have not addressed is: *What is the relationship of MOS to MOEs*? Since an MOE is a metric that characterizes a system's *operational effectiveness* to accomplish mission performance-based outcomes, obviously, *operational suitability* has an impact on that achievement. Therefore, an MOS is a contributory performance effector to one or more MOEs.

5.6.4 System Effectiveness

As an objective factor, *system effectiveness* represents the physical reality of outcome-based performance and results. System effectiveness requires understanding the contributory factors such as reliability, maintainability, and performance.

Outcome-based performance and results occur in two basic forms: *planned* and *actual* performance. When system development is initiated, the System Developer is dependent on analyses, models, and simulations to provide technical insights that reveal how a system is projected to perform. The data is used to:

- Bound, specify, and model system performance or one of its items.
- Compare *actual* versus *planned* performance or *expected* results.

Various techniques, such as rapid prototypes, proof of concept prototypes, and technology demonstrations, are employed to validate models and effectiveness of simulations. The intent is to collect objective, empirical evidence "early on" to gain a level of confidence that the system, or portions thereof, will perform as expected. The net result is to V&V system effectiveness predictions.

When the actual SYSTEM or PRODUCT is ready for field-testing, actual performance data are collected to:

- Verify accomplishment of the requirements.
- Validate SYSTEM or PRODUCT models.

5.6.5 Cost-Effectiveness

The objective measure of system success ultimately depends on its *cost-effectiveness*. From an Enterprise perspective, *Will a system produce outcome-based performance and results that provide an ROI that justifies its continued use*? *Are there other alternative systems that produce similar or comparable results that are more cost-effective*?

Engineers, by virtue of their technical backgrounds, often have difficulty in relating to the concept of *cost-effectiveness*; they focus by virtue of experience, education, and training on technical—*system effectiveness*. The reality is that the Total Cost of Ownership (TCO) and the profits derived from system applications drive Enterprise decision-making. You can innovate the most elegant system, product, or service with outstanding system effectiveness. However, if the *recurring* operating and support costs are *unsustainable* for its Users, the system may be "dead on arrival (DOA)" at system delivery, especially in commercial environments.

Cost-effectiveness, as a metric, is computed from two elements: (1) LCC – TCO and (2) system effectiveness. It is important to note that a system, product, or service can be characterized by its system effectiveness. However, having *system effectiveness* does not mean that it is *cost-effective*. Examples include experimental medical drugs and engineering designs that are unaffordable.

5.7 DEFINING MISSION AND SYSTEM UCs AND SCENARIOS

The subject of UCs and Scenarios applies to Enterprise missions; SOI missions, as well as the EQUIPMENT System Element consisting of SUBSYSTEMS and ASSEMBLIES. Although there are debates concerning the association of the term *mission* to Enterprises, it can be argued that any Equipment components of a system, product, or service have their own *missions* to perform.

Author's Note 5.6

System Use Cases (UCs) and Agile Development User Stories

Agile Development employs the term *User Story* to capture a User's operational needs in their own words. Observe that we said "in their own words." In that context, one could say that User Stories serve as the basis for identifying UCs. We will address User Stories as part of our discussion of Agile Development later in Chapter 15, which includes their relationship to UCs. You may want to consider gaining insights about User Stories in Chapter 15 and their relationships to UCs.

From an SE perspective, *User Stories* help identify User operational needs; *UCs* elaborate sequences of operational tasks to be performed to accomplish a mission or task. The sequences of operational tasks enable us to model

system operations using methods such as MBSE. Operations, in turn, require performance-based capabilities to achieve the UC objectives and their outcomes for translation into specification requirements.

5.7.1 Underlying Philosophy of UCs

The introduction to this section highlighted the need to understand Stakeholder needs. System Acquirers and Developers respond by writing engineering specifications that include highly technical requirements statements and terms for the system, product, or service that will satisfy their needs. Since the Stakeholder community may or may not be engineers, the process involves a level of trust and risk due to interpretation. As a result, a void exists between Stakeholder abilities to express and articulate their needs. *How do we solve this problem*?

Recognizing the need to solve this problem, Ivar Jacobson introduced the concept of UCs to the public in 1987 with a paper accepted for OOPSLA '87. The paper, which described the development of functional requirements for software and modeling, was based on work that evolved over 16 years (Jacobson, 2003, pp. 1–2).

Jacobson (2003, p. 2) makes a very key point concerning "functions," the traditional focus of SE, versus "use cases". Beginning in the 1960s, he began to observe that functions have "no interfaces;" they are bounded entities with a set of inputs, a transfer function, and an output (Figure 3.1). As a *problem space*, the challenge is: since a system, product, or service outcome is dependent on "interfacing" unique sets - configurations - of functions (capabilities), *how do we accomplish this task?*

This challenge lead to Jacobson's evolution of Use Cases (UCs) beginning in 1967 (Jacobson, 2003, p. 1). As a solution space, UCs provided the linking mechanism for "interfacing" the set of "functions" (capabilities) into an integrated Input/Output (I/O) response thread. In that context, each UC represents a higher level system, product, or service "capability" to produce a performance-based outcome required by the User (Principle 5.14).

The concept of a UC is simple. Employ *conversational* English or other appropriate language to articulate individual, narrative descriptions of how a stakeholder envisions using a system, product, or service to perform a mission or task. Each description should be supported by other relevant information such as *who*, *what*, *when*, *where*, *how*, and *how often*. UCs started as an *informal* method to "bridge the gap" between stakeholders and specification developers. Over the years, they have been transformed into formal documents.

What is the relevance of Use Cases to SE? Use Cases fill the void to enable SEs to communicate and collaborate with Stakeholders, irrespective of their technical skills. In so doing, the method enables SEs to breakdown the abstractness into a sequence of steps to be performed by the User and the system, product, or service to accomplish the desired outcome.

5.7.2 What Are System UCs?

UCs are methods that allow a System Developer SEs and Analysts to collaborate with the Users to identify and document in *plain everyday language* the key mission activities and outcomes they would like to achieve with a system, product, or service.

UCs serve as a valuable tool for SE and analysis, especially for identifying system capabilities that will enable us to model the SYSTEM and develop specification requirements. The scope of our discussion will not be a treatise on UCs but rather key elements of the concept that lead to the identification of system, product, or service capabilities.

Referral For a more in-depth discussion of UCs, refer to Cockburn (2001).

5.7.3 UC Applications

UCs are applicable to User Enterprise System missions, SOI level missions, EQUIPMENT such as physical systems, SUBSYSTEMS, and hardware or software missions.

5.7.4 UC Documentation

UCs typically begin as *informal working papers*. Professionally, they should be formalized and controlled by the Lead SE (LSE) for a project. In general, UC Documents should consist of an outline structure with a section—Section 3.0, "Use Case Descriptions"—that identifies and describes each UC for the system, product, or service.

Section 3.0 should begin with the introduction of a UC Diagram. Each UC Description should include Sequence Diagrams that depict the operational sequences of time-dependent interactions expected between the System Users referred to as Actors and the SYSTEM ELEMENTS.

Referral Refer to Appendix B for a brief overview of Systems Modeling Language (SySML™) and its diagrams.

5.7.5 UC Representations

UC representations employ the SySML™ and its graphical tools based on standards established by the Object Management Group (OMG®). UCs for a system, product, or service are represented by a UC Diagram as shown in Figure 5.9. Key points are:

[1]The *System Modeling Language* (SysML™) is a registered trademark of the Object Management Group (OMG®)

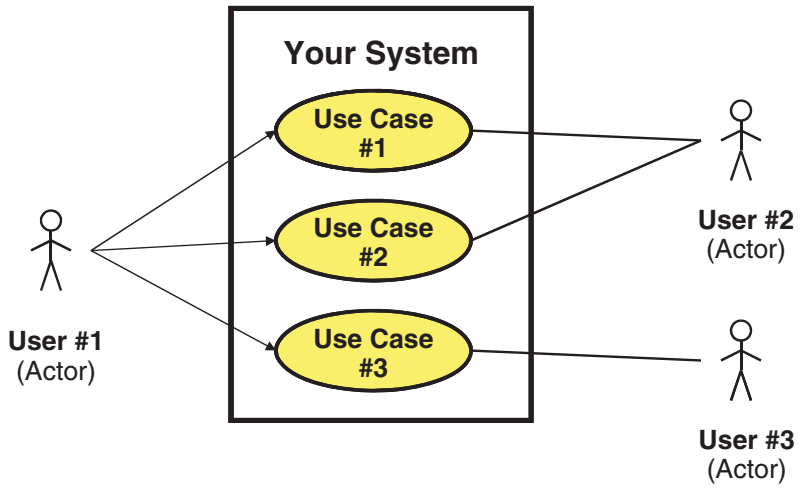


Figure 5.9 SysML™ Use Case Diagram Example[1]

- Each UC is unique within the system, product, or service and is symbolized by an oval.
- The syntax of each UC title consists of an *active verb* followed by an outcome-based action that the User (Actor) expects the system, product, or service to accomplish, not features of the system.
- Each UC may be employed by one or more stakeholders—Users and End Users—referred to as Actors and represented by *stick figures*.
- Each UC may have one or more Extensions that represent variations of the UC.

Given this basic understanding of UC fundamentals, let us establish some attributes that will enable us to develop UCs.

5.7.6 UC Descriptions

A UC is characterized by a UC Description that characterizes UC attributes that describe how the User might deploy, operate, support, sustain, retire, and dispose of a system, product, or service. The following is a customized list of UC attributes that are useful in developing SYSTEM Level UCs. If you are developing software UCs, you are encouraged to employ UC attributes that may be more appropriate for software development:

1. UC#_ Title
2. UC#_ Identifier
3. UC#_ Outcome and Performance-based Objectives
4. UC#_ Description
5. UC#_ Actors
6. UC#_ Assumptions
   a. Initial state
   b. Final state
   c. Environmental conditions
   d. Operating constraints
   e. Acceptable and Unacceptable Inputs
   f. Resources
   g. Event-based timeline
   h. UC Frequency of occurrence and utility priorities
7. UC#_ Pre-conditions
   a. Preceding Events
   b. System/Entity Operational Status and Health (OS & H)
8. UC#_ Trigger(s)
9. UC#_ Main Success Scenario
10. UC#_ Post-Conditions
11. UC#_ Extension Points
12. UC#_ Scenarios and consequences
    a. Probability of occurrence
    b. UC scenario actors
    c. Stimuli, excitations, and cues
    d. Scenario consequences
    e. Compensating/mitigating actions
13. UC#_ Artifacts

Given this list, let's briefly describe each one and its contribution to the characterization.

5.7.6.1 Attribute 1: UC#_ Title

Use Case Title Principle

Principle 5.12 Every Use Case consists of a title that *expresses* an outcome to be accomplished from the perspective of the User (Actor).

Each UC consists of a brief, two or three word title that expresses an outcome to be achieved. The syntax of the title requires an *active verb* that represents the action to be performed followed by a noun representing the outcome. One key point that is often confusing. The title captures *what* the Actor requires the system, product, or service to accomplish.

People may *erroneously misinterpret* a UC as being a *feature* or *capability* of the system, product, or service. In the end, SYSTEM or PRODUCT success is determined by meeting the operational needs of the User, not highlighting a feature someone within the System Developer's Enterprise thinks might be interesting. *These are very different concepts; recognize and appreciate the difference.*

5.7.6.2 Attribute 2: UC#_ Identifier

Use Case Identifier Principle

Principle 5.13 Every Use Case must be *unique* and consist of an identifier that: (1) differentiates it from other Use Cases and (2) facilitates referencing.

Each UC should have its own *unique* identity with no overlap, conflict, or duplicate other UCs for the system, product, or service. Therefore, each UC should be tagged with a unique identifier such as SYSTEM UC #1 and SYSTEM UC #2. Since there may be UCs at various levels of abstraction, attach the appropriate prefix such as SS#1 UC#1 for SUBSYSTEM #1 Use Case #1 accordingly or whatever naming convention works best for you and your team.

5.7.6.3 Attribute 3: UC#_ Outcome and Performance-Based Objectives

Use Case Outcome Principle

Principle 5.14 Every Use Case expresses what outcome the User(s) requires the system, product, or service to produce, not what the system does to perform the action.

UC success comes from producing timely, performance-based outcomes. Therefore, each UC should explicitly state the outcome and performance-based objectives to be accomplished.

5.7.6.4 Attribute 4: UC#_ Description

Use Case Description Principle

Principle 5.15 Every Use Case should consist of a brief synopsis that describes how the UC will accomplish the required outcome.

To help UC users understand the UC, include a brief narrative description that serves as an executive summary about how the UC will be employed by an Actor. Although the description could be several pages in length, best practice indicates this statement should be *brief* and *concise* with lengths ranging from one or two sentences to a full paragraph at most. The structure of the paragraph should reflect three phases of the UC: 1) preparation, 2) execution, and 3) completion.

5.7.6.5 Attribute 5: UC#_ Actors

Use Case Actors Principle

Principle 5.16 Each Use Case (UC) represents a system, product, or service capability required by one or more Users (Actors) in performing their assigned mission or task.

Each UC should identify the list of Actors that participate in and interact with the system, product, or service described by the UC. An *actor* can be a person, place, thing, or role and includes the entity for which the UC is written. This last statement is key to the implementation of a UC. Consider the following commercial aircraft UC example.

UC Actor Example Applications

Example 5.10

- Flight crew (Actor) interact with an aircraft during all phases of flight.
- The flight crew (Actor) consists of a Pilot (Actor), a Co-pilot or First Officer (Actor), and Flight Attendants (Actors).
- The Pilot (Actor) and Co-pilot (Pilot) perform the following roles: Communicator Role (Actor), Aviator Role (Actor), and Navigator Role (Actor).

Observe in the preceding example how an individual—Pilot or Co-pilot—can serve in multiple roles while performing their duties represented by UCs.

5.7.6.6 Attribute 6: UC#_ Assumptions The formulation and development of UCs requires that SEs make assumptions that characterize a UC. Assumptions include the following types of attributes.

5.7.6.6.1 Initial State The *initial state* of a UC represents the assumed physical configuration state (Chapter 7) of the system, product, or service when the UC is initiated.

5.7.6.6.2 Final State The *final state* represents the desired physical or operational state of the system when the desired outcome has been achieved.

5.7.6.6.3 Environmental Conditions The current environmental conditions specify and bound the OPERATING ENVIRONMENT conditions that exist when a system, product, or service UC is initiated.

5.7.6.6.4 Operating Constraints For some UCs, the system, product, or service may have operational constraints such as organizational policies, procedures, and task orders; local, federal, state, and international regulations or statutory laws; public opinion; or an MET. Therefore, operational constraints serve to bound or restrict the acceptable set of corporate, moral, ethical, or spiritual actions allowed for a UC.

5.7.6.6.5 Acceptable and Unacceptable Inputs Every system, product, or service processes external acceptable and unacceptable inputs to add value to achieve the specified outcome.

5.7.6.6.6 Resources

Principle 5.17

UC Event Timeline Principle

When applicable, every Use Case should be bounded by and synchronized to an event timeline.

Every system, product, or service requires MISSION RESOURCES to perform its mission. MISSION RESOURCES are typically finite and are therefore constrained. The resources attribute documents what types of resources such as *expendables* or *consumables* are required to sustain *System* or *Entity* operations.

5.7.6.6.7 Event-Based Timeline UCs may require an MET to synchronize the planned actions or intervention of System Users- operators or maintainers - or expected responses from the SYSTEM or PRODUCT.

5.7.6.6.8 Frequency of Occurrence and Utility Priorities

Principle 5.18

UC Frequency Principle

Every UC has a cost, risk, and time required to develop and assess its frequency of usage and prioritize based on level of criticality.

Every UC has a cost and schedule for development, training, implementation, and maintenance. The realities of budgetary cost and schedule limit the number of UCs that can be practically implemented. Therefore, prioritize UCs and implement those that *maximize* application and SYSTEM safety and utility to the User.

Observe that we said … to *maximize* application and System safety and utility. If you prioritize UCs, *emergency* capabilities and procedures should have a very remote frequency of occurrence. Nevertheless, they can be the most critical. You may need to analytically express *utility* in terms of a multiplicative factor. Instead of assigning a 1 (low) to 5 (high) weighting factor priority to each UC, multiply the factor by a *level of criticality* from 1 (low) to 5 (high) to ensure the proper visibility from a safety perspective.

Enterprise systems, products, and services must be safe for the Users to deploy, operate, maintain, sustain, and dispose. Hypothetically, you could focus all resources on safety features and produce a product that is so burdened with safety features that it has no application utility to the User.

Although our discussion focuses on the development of a product or service, remember that the system has other System Elements—PERSONNEL, MISSION RESOURCES, and PROCEDURAL DATA—than just EQUIPMENT. So, when confronted with increasing design costs, there may be equally *effective* alternatives for improving safety such as operator certification, training and periodic refresher training, *cautionary and warning labels*, and supervision required that might not require product implementation.

5.7.6.7 Attribute 7: UC#_ Pre-conditions

Principle 5.19

Use Case Preconditions Principle

Establish the pre-conditions context for each UC.

For some applications, the circumstance or sequence of events leading up to the initiation of a UC needs to be identified. *Pre-conditions* establish the contextual basis for documenting the UC. Pre-conditions such as Preceding Events and Operational Health and Status (OH&S) that are relevant to the UC should be documented.

5.7.6.7.1 Preceding Events Some UCs are dependent on what has been accomplished prior to the initiation of the UC. Therefore, list any preceding events such as power applied and switches set that enable accomplishment of the UC.

5.7.6.7.2 System/Entity OH&S Another pre-condition for a UC is the general OH and S of the SYSTEM or ENTITY for which the UC applies. The operational state or condition (Chapter 7) determines the UC's *Flow of Events* or the need for an *Alternative Flow*. For example, nominal operations may be required when "off".

5.7.6.8 Attribute 8: UC#_ Trigger(s)

UC Trigger Principle

Every UC requires a *triggering condition* or *event* to initiate its performance.

Principle 5.20

A UC requires one or more *triggers* as enablers to initiate performance of a UC. In general, HUMAN SYSTEMS—Enterprise and Engineered—require some form of human user intervention to *activate/deactivate* the system to perform its mission. There are several ways of doing this depending on the system design:

- **Manual intervention systems**—require a human to *manually* Start or Stop the system from performing operations. For example, a garden hose.
- **Semi-automatic systems**—perform operations that require human intervention to initiate a sequence of operations that continues until the system times-out, deletes resources, or completes processing of a single task. For example, an office copier.
- **Automated System**—automatically senses the need to initiate operations when specific conditions are met, performs the operation, completes the operation, and awaits conditions to repeat the process. For example, a facility badge security system that operates 24 hours per day, 7 days a week.

UC triggers ultimately depend on the system operations concept.

5.7.6.9 Attribute 9: UC#_ Main Success Scenario

UC Main Success Scenario Principle

Every UC has a *main success scenario* (Cockburn, 2001) that when everything works flawlessly the normal sequence of actions produces the required outcome.

Principle 5.21

The heart of a UC centers on the main stimulus/response processing scenario for a specified set of conditions to produce the desired or required outcome. Some domains refer to this as a *Transfer* or *Response* function.

Cockburn's (2001) *main success scenario* represents the sequence of steps and interactions required to initiate, execute, and complete the UC by the System's Actors. This series of steps provide the basis for developing SysML™ Activity Diagrams from which System or Entity capabilities will be extracted and derived.

When writing the *main success scenario*, it is important to *avoid* presuming that hardware or software exists. Remember, at the SYSTEM Level, the SYSTEM is simply a box that has inputs and provides outputs (Figure 3.2); its contents in terms of SUBSYSTEMS and ASSEMBLIES are abstract—unknown. So, all that exists is the SYSTEM. Consider the following example in which the Office Copier is treated as a box with inputs and outputs:

Office Copier UC Flow of Events Example

Example 5.11

Step 1. The Copier (Actor) displays readiness to copy.

Step 2. The User (Actor) places the document to be copied in the Copier's (Actor) hopper.

Step 3. The Copier senses paper in hopper.

Step 4. The Copier awaits User inputs.

Step 5. The Operator selects quantity of copies.

Step 6. The Copier reads User inputs—quantity of copies.

Observe that Example 5.11 Flow of Events does not specify how to design the Copier—displays, keyboards, and hopper—only how the Operator expects the Copier to respond to the user inputs.

An alternative to the linear sequence of steps is based on interactions between the Actors as shown in Table 5.1.

5.7.6.10 Attribute #10: UC#_ Postconditions

UC Completion Principle

Every UC requires definition of actions to be accomplished by the user or system to finalize the UC.

Principle 5.22

When a UC has achieved its prescribed result, postconditions describe any actions required by the Actors to perform any housekeeping tasks and place the system in a State of Readiness for the next UC. This includes storage of data, printouts, and notification of task or action completion.

5.7.6.11 Attribute 11: UC#_ Extension Points

UCs can be elaborated into Extension Points that represent unique instances of a UC. For example, a restaurant UC might be Take Order. The Take Order UC might be extended via Extension Points into: (1) Take Order - Beverage, (2) Take Order - Entrée, (3) Take Order - Dessert, and so forth.

5.7.6.12 Attribute 12: UC#_ Scenarios and Consequences

UC Scenarios Principle

Every UC should include considerations for the *most likely* or *probable* scenarios when things go wrong that require alternate flows from the *main success scenario* to enable recovery and avoid SYSTEM or PRODUCT failure.

Principle 5.23

TABLE 5.1 Table-Based Example of Office Copier UC Flow of Events and Interchanges between Actors

UC Step	User (Actor)	Copier (Actor)
Step 1		Indicates readiness to copy
Step 2	Loads original in Copier	
Step 3		Copier senses paper supply
Step 4		Copier awaits User inputs
Step 5	Selects quantity of copies	
Step 6		Copier reads User inputs
Step 7		Copier displays copy quantity selection
Step 8		Copier awaits User selection inputs
Step 9	Selects Color or B and W copies	
Step 10		Copier reads User inputs
Step 11		Copier displays B and W copy selection
Step 12		Copier awaits User selection inputs
Step 13	Selects Paper Size	
Step 14		Copier reads User inputs
Step 15		Copier displays Paper Size selection
Step 16		Copier awaits User selection inputs
Step 17	Selects Double-Sided copies	
Step 18		Copier reads User inputs
Step 19		Copier displays Double-Sided copy selection
Step 20		Copier awaits user selection inputs
Step 21	Initiates copy	
Step 22		Copier reads pages
Step 23		Copier prints copies
Step 24		Copier dispenses copies for user
Step 25		Copier displays "Ready to Copy" message
Step 26	Picks up copies	
Step 27		Copier awaits User inputs
Step 28		Copier initiates countdown timer
Step 29		Copier awaits User inputs
Step 30		Copier timer expires
Step 31		Copier switches to Energy Saver Mode

The preceding discussion addresses an ideal scenario in which everything works perfectly as planned. As humans, we tend to be optimistic and believe that everything will be successful. While this is true most of the time, uncertainties and *unknown-unknowns* do occur that create conditions we have not planned for operationally. Once a UC is identified, ask the question: *When the User employs this UC:*

- What can go wrong that we haven't anticipated?
- What are the consequences of failure and how do we mitigate them?

We refer to each of these instances as UC scenarios and consequences. Consider the following example:

Example 5.12

CD Player UC Scenarios Example

Suppose that we are designing a Compact Disk (CD) or Digital Video Disk (DVD) player. Ideally, the high-level Play CD/DVD UC describes a User inserting the CD/DVD into the player—and magic happens! The player produces the desired result, *positive* outcome of music or a movie.

Now, what happens if the User inserts the CD/DVD upside down? The User has a UC scenario with a *negative* outcome that has consequences—no music or video. This leads to the question: *When we design the CD/DVD device, how should the User be advised of this situation*? If we add an automatic notification capability via displays or alarms to the device, development costs increase. In contrast, the low-cost solution may be to simply inform the User via the product manual or user's guide about the convention of always inserting the CD/DVD with the title facing upward.

UC Scenarios include topics such as Probability of Occurrence; Scenario Actors; Stimuli, Excitations, and Cues; and Compensating/Mitigating Actions that are relevant. Let us briefly address each of these.

5.7.6.12.1 Probability of Occurrence Once UC scenarios are identified, we need to determine the *probability of occurrence* of each one. As in the earlier discussion of UC priorities, scenarios have a probability of occurrence. Since additional design features may increase the cost and risk, *prioritize* scenarios based on the *most likely* or probability of occurrence with User safety as a predominant consideration.

5.7.6.12.2 UC Scenario Actors Our discussion up to this point has focused on what is *most likely* or *probable* to occur: UCs or scenarios. The key question is: *Who or what are the interacting entities during UCs and scenarios?*

The Unified Modeling Language (UML®) and its subset SysML (Appendix C) characterize these entities as *Actors*. Actors can be persons, places, roles, real or virtual objects, or events. Actors are represented by a *human stick figure* icon and UCs as *ellipses* as shown in Figure 5.9:

- User 1 (Actor) such as a system administrator/maintainer interacts with UC #1 through UC #3.
- User 2 (Actor) interacts with UC #1 (Capability) and UC #2 (Capability).
- User 3 (Actor) interacts with UC #3 (Capability).

5.7.6.12.3 Stimuli, Excitations, and Cues UCs are initiated based on a set of actions triggered by system operator(s), external systems, or the system. Consider the following stimulus–response actions:

- The User or an external system initiates one or more UC-based Triggers that cause the SYSTEM to respond behaviorally within a specified time period.
- The SYSTEM notifies the User to perform an action to make a decision or input data.
- The User intervenes or interrupts ongoing actions by the SYSTEM.

Each of these examples represents instances whereby the User or System stimulates the other to action. Figure 5.10 illustrates such a sequence of actions using a UML Sequence Diagram.

5.7.6.12.4 Scenario Consequences Each UC and scenario produces an outcome that may have *positive* or *negative* consequences that impact SYSTEM or PRODUCT performance or mission success. Consider the following example:

Example 5.13 Scenario Consequences Example

If Scenario X occurs and the operator or system responds in a specified manner, *instabilities* and *perturbations* that may have *negative* consequences may be induced into the system. Therefore, each UC and scenario should identify the potential consequences of proper *use/misuse, application/misapplication,* or *abuse*.

5.7.6.12.5 Compensating Provisions/Mitigating Actions Given the set of consequences identified for UCs and UC scenarios, we need to identify what *compensating provisions/mitigating actions* should be incorporated into the system, product, or service to *eliminate* or *minimize* the effects of a *negative* outcome consequences.

Compensating provisions might include design changes and training operators in proper methods. Consider the

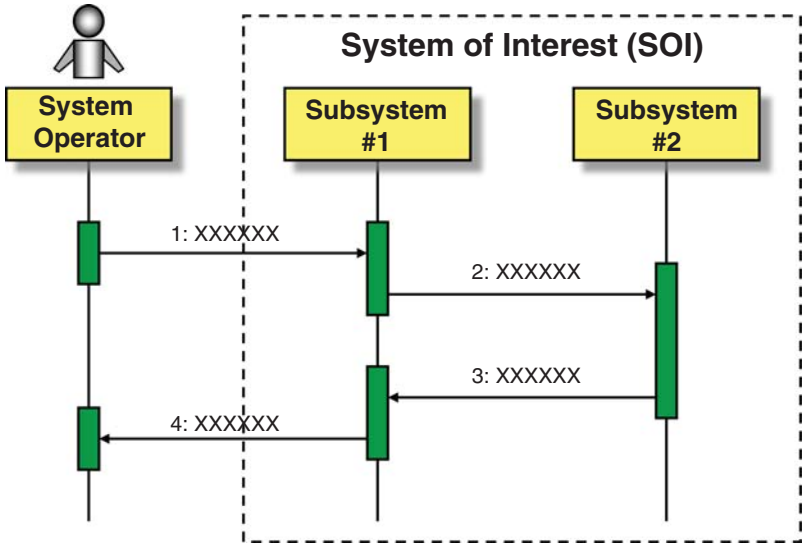


Figure 5.10 SysML™ Sequence Diagram Example

example below concerning what has occurred with automobile design over the past 100 years.

Example 5.14 **Automobile Evolution Mitigating Actions Example**

Suppose that we design a car. Since a car can collide with other vehicles, walls, or trees, a generalized interface solution of the car body-to-external system is insufficient. An analysis of UCs and UC scenarios suggests that passengers can lose their lives or sustain injuries in a collision. So a specialized interface consisting of a bumper is added to the car frame as a *compensating/mitigating* action. However, impact tests reveal that the bumper is inadequate and requires yet a more specialized solution including the following sequences of design actions:

- Design action 1: Specify proper vehicle operating procedures
- Design action 2: Incorporate shock absorbers into the vehicle's bumpers.
- Design action 3: Install and require use of seat belts.
- Design action 4: Install an airbag system.
- Design action 5: Install an Anti-lock Braking System (ABS).
- Design action 6: Increase driver awareness to drive safely and defensively.
- Design action 7: Install a collision avoidance system.

Compensating provisions also include actions the System operator or maintainer can take to avoid an outcome that may have negative or catastrophic consequence. Mini-Case Study 24.1 illustrates compensating provisions test pilot Chuck Yeager took in an aircraft that had been subject to several *unexplainable* accidents that resulted in loss of life.

5.7.6.13 Attribute 13: UC#_ Artifacts UCs are required to produce outcome-based results that are: (1) *specific*, *achievable*, *observable*, *measurable*, *testable*, and *verifiable* and (2) meet the Producer–Supplier "fitness for use" performance criteria for the next downstream customer as shown in Figure 4.1. This means that when the system, product, or service performs a UC, it must produce *objective evidence* in the form of an action report that documents results and completion of the UC. This section defines and specifies the objective evidence results that satisfy the observable, verifiable criteria.

5.7.7 UC Analysis

Each UC and its *most likely* or *probable* scenarios represent a series of anticipated interactions among the actors. Once the scenarios and actors are identified, SEs and System Analysts need to understand the most likely or probable interactions between: (1)the SYSTEM User and the Equipment, and (2) the SYSTEM or ENTITY of Interest and external systems within its OPERATING ENVIRONMENT.

UML tools are useful in understanding the stimuli, cues, and behavioral responses between interacting systems. UML *sequence diagrams*, which serve as a key tool for representing interactions with a UC, include:

- **Actors**—Consist of entities such as persons, places, things, roles, and other objectives that interact or provide information, energy, or other inputs to another Actor. Labels at the top of each Swim Lane identify actors.
- **Lifeline**—Consists of a vertical line to represent time relative processing. Activation boxes are placed along the lifeline to represent processing of external inputs, stimuli, or cues and behavioral responses to produce a specific output(s) for exchange with other downstream Actors.
- **Swim Lanes**—Consist of the regions between the Actor lifelines for illustrating sequential control flow of operations and tasks and data exchange interactions between each Actor.

Appendix C provides a brief overview of these and other SysML tools.

To illustrate how these are employed, consider the following example:

Example 5.15 **UC Swim Lanes Example**

Suppose a User (Actor) has a task to perform mathematical calculation and report the results. To perform the task, the User (Actor) interacts with a calculator (Actor) as shown in Figure 5.11. Observe that the figure is structurally similar to and expands the level of detail of Figure 5.10. Activation boxes (Figure 5.10) are transformed into Actor *activities* in Figure 5.11. To keep the example simple, assume the calculator consists of two SUBSYSTEMS—SUBSYSTEM #1 and SUBSYSTEM #2.

The User and each of the SUBSYSTEMS have an Initial State, Final State, and conditional loops that cycle until specific decision criteria are met to terminate operation. We assume each of the SUBSYSTEM activities include "wait states" for inputs. When inputs arrive, processing is performed, and control is passed to the next activity. Here is a potential UC scenario description:

- The System operator (Actor) turns on the calculator (Actor), which activates and initializes SUBSYSTEMS #1 and #2 from their Initial States.

- Each subsystem initializes and proceeds to Activities 20 and 30 to await SYSTEM Operator inputs.
- The SYSTEM Operator (Actor) enters data –Activity 10- into the calculator producing Output 10.
- Activity 20 accepts the SYSTEM Operator keyboard entries, processes the information, and sends Output 20 to Activity 30.
- Activity 30 processes the information and transfers control to Activity 31.
- Activity 31 performs the required computation and produces Output 31.
- In the interim, SUBSYSTEM #1 Activity 21 enters a "wait state" for the Output 31 results.
- On receipt of Output 31, Activity 21 converts the results into meaningful operator information and displays Output 21 to the System Operator.
- On receipt of Output 21, the SYSTEM Operator records the results – Activity 11 – and communicates the results – Output 11.
- SUBSYSTEMS #1 and #2 continue to cycle (Conditions 11, 21, and 31) awaiting inputs until the SYSTEM Operator decides to Power Down the calculator.
- On Power Down Conditions 10, 20, and 30), Subsystems #1 and #2 enter a Final State.

5.7.8 Relating UCs to Operational Tasks

A UC represents a capability traceable to a mission objective that enables a SYSTEM User (Actor) to Monitor, Command and Control (MC2) a system, product, or service. When you elaborate a UC's Flow of Events into a series of interaction steps such as Table 5.1 and Example 5.1, each step represents an *operational task* that enables the User (Actor) or Copier (Actor) to accomplish a mission objective—print copies of a document. In turn, each activity- capability - can be translated into an SPS capability requirement. Net result is:

UCs → Operational Tasks → Operational Capabilities → SPS Capability Requirements

5.7.9 How Many System UCs?

A key question people often ask is: *How many UCs are required for a system?* There are no magic answers; 10 through 30 UCs might be average. Some highly complex systems may just have 5 or 6; others, 10 to 20. It all depends on the individuals and organizations involved. Some want simplicity to keep the number small; others want detailed

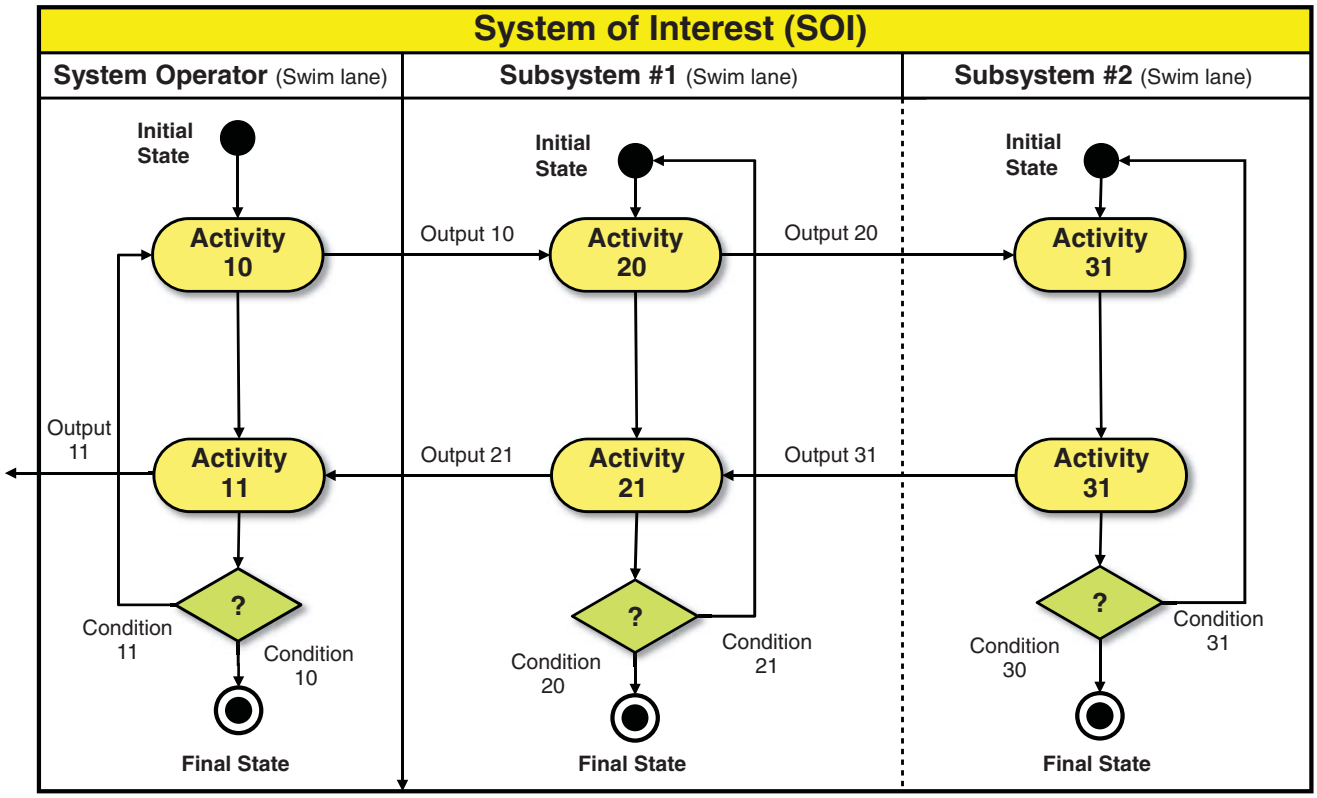


Figure 5.11 SysML™ Activity Diagram Example

lists. In general, a system may have 5 to 8 primary UCs; the remainder may be *secondary* or extension UCs of the *primary* UCs.

From an SE perspective, UC analysis should be a key tool of any System Development effort. However, Engineers often view this activity as non-value-added, bureaucratic paperwork to the User and product, and believe their time is better spent contemplating the creation of elegant designs. The reality is that *elegant* designs can be *useless*—unless the User can *easily* and *understandably* implement them with their current skill set. This is why "Just in Time" (JIT) training for SYSTEM Operators must take place prior to system acceptance and delivery.

People who promote the bureaucratic paperwork argument are the same people who, after a system fails during integration and test, capitulate and remark "… How was I to know what the User wanted? I'm only human … besides they couldn't decide what they wanted. I can't read their minds!"

Documenting UCs is a simple matter. It requires *professional discipline*, something that tends to get lost in modern-day *casual engineering* efforts. If you doubt this, ask yourself how many products have disappointed you and made you wonder did anyone within the System Developer's organization ever consult the Users and how they expected to use the system? If they had, they would have easily learned that this step is critical to the User's success and acceptance of the system, product, or service.

5.8 CHAPTER SUMMARY

This concludes our discussion of Mission Analysis, User Stories, Use Cases, and Scenarios. Our discussions highlighted the need to employ UCs as a means of avoiding *quantum* leaps (Figure 2.3) between User visionary requirements and System or Product design. We also noted that UCs and scenarios provide a powerful tool using plain language that Users, Acquirers, and System Developers can employ, to improve communications and understanding of how the system, product, or service is envisioned to be deployed, operated, maintained, sustained, retired, and disposed:

- Every mission must be founded on an operational strategy referred to as a *mission profile.*
- Each mission begins with a *point of origination* and concludes with a *destination* or *point of termination* with intervening *staging, control*, or *waypoints* based on specific objectives and MET events.
- Between the point of origination and point of termination, some missions may require interim *waypoints* or *delivery points* that satisfy specific mission objectives.
- Every mission is characterized by at least three mission phases of operation: (1) Pre-Mission, (2) Mission, and (3) Post-Mission.
- Every mission is characterized by outcomes and supporting performance-based objectives that represent UCs of the system, product, or service.
- Each mission requires consideration of mission scenarios that might occur during the performance of a UC.
- UCs provides a means of identifying key sequential or concurrent operational tasks that represent lower-level capabilities that will ultimately be translated into SPS requirements.
- UCs must be prioritized for development based on *most likely* or *probable* occurrences for development subject to program technical, cost, and schedule constraints.
- UC scenarios provide a basis for understanding: (1) how the User expects to use a system, product, or service. And (2) how the *misuse or abuse* might result in risks with *consequences* that require design *compensating* or *mitigating actions*.
- UC scenarios must be prioritized within UC technical, cost, and schedule constraints.
- UC attributes provide a standard framework to uniformly and consistently describe each use case.
- SysML™ *Sequence Diagrams* serve as a useful tool for understanding the sequencing of Actor interactions and behavioral responses.
- Each UC and its attributes should be captured in a *System XYZ UCs and Scenarios Document* and placed under baseline management control for decision-making.

5.9 CHAPTER EXERCISES

5.9.1 Level 1: Chapter Knowledge Exercises—Mission Analysis

1. How does *System Definition* vary between consumer product development and contract system development?
2. What is a *mission*?
3. Is the term "mission" restricted to the military applications? Explain why?
4. Do consumer products and services perform *missions*?
5. How do you plan a *mission*?
6. What is a Mission Event Timeline (MET), its key attributes, and how is it developed.
7. How is *mission* task analysis is performed?
8. What are the primary phases of operation of a system, product, or service? Can there be other phases of operation?

9. What system operations and decisions are performed during the PRE-MISSION, MISSION, and POST-MISSION Phases?
10. How do SEs bound each mission phase of operation and establish criteria for triggering the next phase? Why is it important to bound phases of operation? What occurs if you do not bound each phase of operation?
11. What is a User Story?
12. What is a Use Case (UC)?
13. How many UCs does a system, product, or service need?
14. Which types of systems employ UCs? Organizations, systems, products, or services; SUBSYSTEMS, or ASSEMBLIES?
15. What are the attributes of a UC?
16. What is UC analysis and how do you perform one?
17. What is an Actor and what is its relationship to a UC?
18. Is each UC restricted to one Actor?
19. What is the structure of a UC Description? What is the optimal length of a UC Description? Paragraph? Pages?
20. Where are UCs documented?
21. What is a UC scenario?
22. Why are UC scenarios important to defining systems, product, and services?
23. What is the relationship between UCs and system capability requirements?
24. List the key SE principles learned from this chapter.

5.9.2 Level 2: Knowledge Application Exercises

Refer to www.wiley.com/go/systemengineeringanalysis2e

5.10 REFERENCES

Akao Yoji ed. (1990), *Quality Function Deployment: Integrating Customer Requirements into Product Design*. (Translated by Glenn H. Mazur) Cambridge, MA: Productivity Press.

Cockburn, Alistair (2001), *Writing Effective Use Cases*, Boston, MA: Addison-Wesley.

Cohn, Mike (2008), *Advantages of the "As a user, I want" User Story Template*, Blog Post, Broomfield, CO: Mountain Goat Software. Retrieved on 9/12/13 from http://www.mountaingoatsoftware.com/blog/advantages-of-the-as-a-user-i-want-user-story-template.

DAU (2001). *Systems Engineering Fundamentals*, Ft. Belvoir, VA: Defense Acquisition University Press. Retrieved on 1/16/14 from http://www.dau.mil/publications/publicationsDocs/SEFGuide%2001-01.pdf.

DAU (2012). *Glossary: Defense Acquisition Acronyms and Terms*, 15th ed., Ft. Belvoir, VA: Defense Acquisition University (DAU) Press. Retrieved on 4/12/15 from http://www.dau.mil/publications/publicationsDocs/Glossary_15th_ed.pdf

DAU (2014), *Capabilities Development Document (CDD)* webpage, ACQuipedia, Ft. Belvoir, VA: Defense Acquisition University. Retrieved on 3/11/13 from https://dap.dau.mil/acquipedia/Pages/ArticleDetails.aspx?aid=99320318-1216-4566-9aea-e44966c5ee32.

Hartman, Henry, *Success*. Accessed on 5/20/14 from http://thinkexist.com/quotes/henry_hartman/.

Jacobson, Ivar, (2003), *Use Cases—Yesterday, Today, and Tomorrow, Rational Software*.

Mazur, Glenn (2003), *Voice of the Customer (Define): QFD to Define Value*, Kansas City, MO: Proceedings of the American Society for Quality (ASQ) Annual Quality Congress Retrieved on 5/24/14 from http://www.mazur.net/works/qfd_to_define_value.pdf.

MIL-HDBK-470A (1997), *DoD Handbook: Designing and Developing Maintainable Systems and Products*, Vol. 1, Washington, DC: Department of Defense (DoD).

MIL-STD-499B Draft (1994), *Military Standard: Systems Engineering*, Washington, DC: Department of Defense (DoD).

MIL-STD-1629A (1998), *Military Standard; Procedures for Performing a Failure Mode, Effects, and Criticality Analysis*. Washington, DC: Department of Defense (DoD).

NASA (2003), *Mars Exploration Rover Launch Press Kit*, Washington, DC: National Aeronautics and Space Administration (NASA). Retrieved on 1/16/14 http://www.nasa.gov/pdf/44804main_merlaunch.pdf.

NASA (2004), *Mars Exploration Rover Landing Press Kit*, Washington, DC: National Aeronautics and Space Administration (NASA). Retrieved on 1/16/14 from http://marsrovers.jpl.nasa.gov/newsroom/merlandings.pdf.

OMG (2006), *SysML™ Glossary (Draft), ad/2006-03-04*, Needham, MA: Object Management Group (OMG®). Retrieved on 3/16/13 from http://www.sysml.org/docs/specs/SysML-v1-Glossary-06-03-04.pdf.

QFD Institute (2013), Frequently Asked Questions (FAQ), Retrieved on 7/6/13 from http://www.qfdi.org/what_is_qfd/faqs_about_qfd.html.

Zultner, Richard E. and Mazur, Glenn H. (2006), *"New Kano Model and QFD," Proceedings of the Eighteenth Symposium on Quality Function Deployment*, Austin, TX. Retrieved on 5/24/14 from http://www.mazur.net/works/Zultner_Mazur_2006_Kano_Recent_Developments.pdf

UPS (2012). *"When in doubt: UPS avoids left turns: How a simple rule increased our drivers' efficiency," UPS Compass, July 2012*, Louisville, KY: United Parcel Service (UPS). Retrieved on 5/17/15 from http://compass.ups.com/UPS-driver-avoid-left-turns/

6

SYSTEM CONCEPTS FORMULATION AND DEVELOPMENT

Chapter 5 introduced concepts for the Mission Definition Methodology and its underlying foundation of mission and system Use Cases (UCs) and scenarios. These concepts provide the framework from which Systems Engineering (SE), Analysis, and Development will emerge. Once the organizational mission has been defined using the Mission Definition Methodology, Systems of Interest (SOIs) have been identified, and ENABLING SYSTEMS have been established, the next step is to formulate and develop the conceptualizations of how the system, product, or service is envisioned to be deployed, operated, maintained, sustained, retired, and disposed by the User.

We begin our discussions with the introductory overview of the System Operations Model. This model provides the "headwaters" analytical framework for structuring system operations that lead to the identification of system capabilities and subsequently System Performance Specification (SPS) capability requirements. This model illustrates the deficiencies in the traditional, ad hoc, Plug & Chug … Specify-Design-Build-Test-Fix (SDBTF) Paradigm that often focuses on "engineering the box" for normal mission operation. Commonly ignored are the Pre-Mission, Post-Mission, Storage, Sustainment, and other operations that emerge after the system, product, or service has been fielded.

Given a foundational understanding of the System Operations Model concept, we introduce the System Concept of Operations (ConOps) document that describes how a system, product, or service is envisioned to be deployed, operated, and maintained. Observe that we did not include *sustainment*, *retirement*, and *disposal*. Why?

In general, a ConOps is developed by the System Developer or Services Provider organization in collaboration with the User to serve as a *shared vision* for developing the system, product, or service. Once the system has been contractually accepted by the User and has been fielded, it belongs to the User and their ENABLING SYSTEMS. The Developer can and should include the concepts for sustainment, retirement, and disposal; however, some users will tell you very explicitly that is their business and is not within the scope of your duties as a System Developer. This does present a dilemma for "smart" SE that incorporates features related to disposal, especially for easy removal of toxic and hazardous materials such as heavy metals. Check with the System Acquirer and User regarding their position on this topic.

Our discussions provide an example ConOps outline and then provide information for developing the Deployment; Operations, Maintenance, and Sustainment (OM&S); Retirement; and Disposal concepts.

6.1 DEFINITIONS OF KEY TERMS

- **ConOps**—A project document that serves as the central focal point early in system development to communicate the vision for a system, product, or service's Operational Concept Descriptions (OCDs); system context and interfaces; operational architecture; System Operations Model; mission phases, modes, and states of operation; sequential and/or concurrent operations workflow; and others required to achieve mission performance objectives.

System Engineering Analysis, Design, and Development: Concepts, Principles, and Practices, Second Edition. Charles S. Wasson.

Companion Website: www.wiley.com/go/systemengineeringanalysis2e

The ConOps should be developed specifically by a key technical visionary leader for the project—Project Engineer, System Architect, or Lead Systems Engineer (LSE). Since the ConOps expresses the project leader's vision, its development should not be delegated. This is not a "guess what the project engineer's vision is today" exercise to be performed by a subordinate!

- **Control or Staging Point**—A major decision gate that limits advancement of workflow progress to the next set of objective-based operations until a set of go–no go decision criteria are accomplished.
- **Corrective Maintenance**—"All actions performed as a result of … failure to restore … an item to a specified condition. Corrective maintenance can include any or all of the following steps: localization, isolation, disassembly, interchange, reassembly, calibration, alignment, and checkout" (DAU, 2012, p. B-48).
- **Deployment Concept**—An ConOps OCD that expresses how a system, product, or service will be (1) deployed from the System Developer's facility to a designated User's site, facility, or distribution system or (2), if applicable, redeployed to a new site.
- **Disposal Concept**—A ConOps OCD that expresses how (1) a system, product, or service will be disposed such as sale, transfer of ownership, lease, or destruction, (2) key components will be salvaged and recycled, and (3), as applicable, environmental remediation and reclamation will be accomplished.
- **Entry Criteria**—One or more thresholds that must be met individually or collectively as a condition to perform the next life cycle stage, phase of operation, mode, state, task, or activity.
- **Exit Criteria**—A set of performance-based outcomes that must be completed individually or collectively to enable transition to the next life cycle stage, phase of operation, mode, state, task, or activity.
- **Maintenance Concept**—A ConOps OCD that expresses how a system, product, or service will be maintained via (1) preventive and corrective maintenance actions, (2) training, (3) upgrades and retrofits, etc. prior to, during, and after each mission for the remainder of its life cycle.
- **Operations Concept**—A ConOps OCD that expresses how a system, product, or service will conduct Pre-Mission, Mission, and Post-Mission operations.
- **Operational Concept description (OCD)**—A narrative that describes a unique aspect of a fielded system, product, or service's life cycle such as deployment, mission operations, mission support Operations, Maintenance, Sustainment (OM&S), retirement, or disposal. In general, a deployment, OM&S, and retirement/disposal OCD should be incorporated as a section within the System ConOps. However, if a given OCD is critical to technical decision-making prior to release of the System ConOps, the OCD is sometimes released in interim form.
- **Operational Task**—A workflow directive that includes an outcome-based objective and performance-based completion criteria.
  Operational tasks are implemented in accordance with Enterprise Organizational Standard Processes (OSPs), methods, and procedural activities.
- **Preventive Maintenance**—"All actions performed in an attempt to retain an item in specified condition by providing systematic inspection, detection, and prevention of incipient failures" (DAU, 2012, p. B-167).
- **Retirement Concept**—A ConOps OCD that describes (1) how a system, product, or service will be decommissioned and transitioned from *active* service to *inactive* service and (2) retraining and reassignment of Personnel.
- **Sustainment Concept**—A ConOps OCD that expresses how a system, product, or service will be sustained logistically by its Enabling System via (1) Mission Resources (consumables and expendables), (2) maintenance, and (3) Personnel training (basic, proficiency, remedial, and skills enhancement).
  Organizations sometimes have different views of Sustainment versus Maintenance. Some view the two activities as: separate, combined, Maintenance as part of Sustainment, or Sustainment as part of Maintenance. Typically, Maintenance and Sustainment are separate, peer level activities.
- **System Operations**—A unique set of multi-level, interdependent, value-added tasks and activities that collectively contribute to satisfying a Pre-Mission, Mission, or Post-Mission Phase for a given System/Product Life Cycle and Mode of Operation.
- **System Operations Dictionary**—A project document that serves as central focal point for scoping and defining activity-based tasks required to perform or support Pre-Mission, Mission, and Post-Mission Phases of Operation and their respective Modes of operation within the Deployment, OM&S, and Retirement/Disposal Life Cycle Phases.
- **System Operations Model**—A generalized template of system operations that can be employed as an initial framework for identifying and tailoring the operational workflows for most systems from the System Development Phase to the System Retirement/Disposal Phase of the System/Product Life Cycle.

6.2 CONCEPTUALIZATION OF SYSTEM OPERATIONS

One of the first critical steps in SE is creating an expression of how system, product, or service will be deployed, operated, maintained, sustained, retired, and disposed. Although this can be accomplished via text, humans instinctively gravitate to graphics such as: (1) block diagrams, timelines, artistic renderings, etc. (2) 3-D wooden or cardboard models, or (3) 3-D printing. The intent is to create a common focal point that expresses the look, feel, and emotion that immerses system development PERSONNEL into the "mental models" of the User (Chapter 24) and inspires them to action.

Building and landscape architects communicate these visions via artistic rendering. Engineers tend to employ sketches, Architecture Block Diagrams (ABDs), prototypes, models & simulations to represent a view of the system. Then, corporate publications artists to create artistic renderings of vehicles, device-based products, etc. applications within the User's intended OPERATING ENVIRONMENTS.

In the twenty-first century, creating visions that inspire is even more critical and leads to the need for "thinking outside the box" when employing ABDs. Walt Disney served an example for inspiration through its world-wide amusement parks. Perhaps one of the best examples of illustrating visions for new systems, products, and services is captured in a text entitled *Designing Disney* (Hench, 2009). The text employs storyboard graphics to express conceptualization views of the futuristic Disney World through the eyes of its patrons—Users and End Users.

- **Key point:** When developing system concepts, think about the various methods you can employ to conceptualize, communicate, and inspire in the least amount of space. As an old cliché, a "picture is worth a thousand words."

6.3 THE SYSTEM OPERATIONS MODEL

Graphical models provide an excellent way for SEs, engineers, analysts, etc. within the System Acquirer, User, and System Developer organizations to establish a shared vision for a system, product, or service. In this context, there are two perspectives:

- Enterprise Perspective—how the User intends to deploy, operate, maintain, and sustain the SOI.
- Engineering Perspective—how an SOI is envisioned to provide capabilities to support mission operations.

We will address the Engineering Perspective in Chapters 7–10. Our discussions here focus on the Enterprise Perspective that provides the basis for the Engineering Perspective.

Figure 6.1 provides a generalized System Operations Model as a construct that can be applied to Human Systems—enterprise and engineered—as a *starting point* template for characterizing its deployment, OM&S, retirement, and disposal. The System Operations Model provides a high-level operational workflow that represents "A Day in the Life of a Mission." It describes how a system, product, or service:

- Is configured for a mission (Pre-Mission)
- Conducts the mission (Mission)
- Is supported following a mission (Post-Mission)

The structure of the model consists of a series of sequential and concurrent operations and tasks that can be translated into specification capability requirements.

Principle 6.1 **System Operations Model Principle**

HUMAN SYSTEMS—Enterprise and Engineered—are characterized by a Generalized System Operations Model developed in collaboration with Users concerning how a system, product, or service will be deployed, operated, maintained, sustained, retired, and disposed.

Principle 6.2 **Operational Capability Principle**

Every system operation represents a required *operational capability* that must produce a specified performance-based outcome while coping with one or more *probable* or *most likely* OPERATING ENVIRONMENT scenarios.

First, a word about the contents of the graphic. Each box in Figure 6.1 represents an integrated, multi-level collection of stakeholder Use Case (UC) based operations required to achieve an overall mission objective. We will decompose or expand each of these UC operations into a series of sequential and concurrent tasks and activities—processes—to achieve the UC performance-based outcome. Ultimately, these tasks are translated into capabilities, and their respective levels of performance are allocated to one or more of the System Elements such as the PERSONNEL, EQUIPMENT, and FACILITIES (Chapter 8). Several key points:

1. Each block consists of a unique identifier—numbered circle—that serves as a navigational aid for narrative description references such as an OCD within the ConOps document.

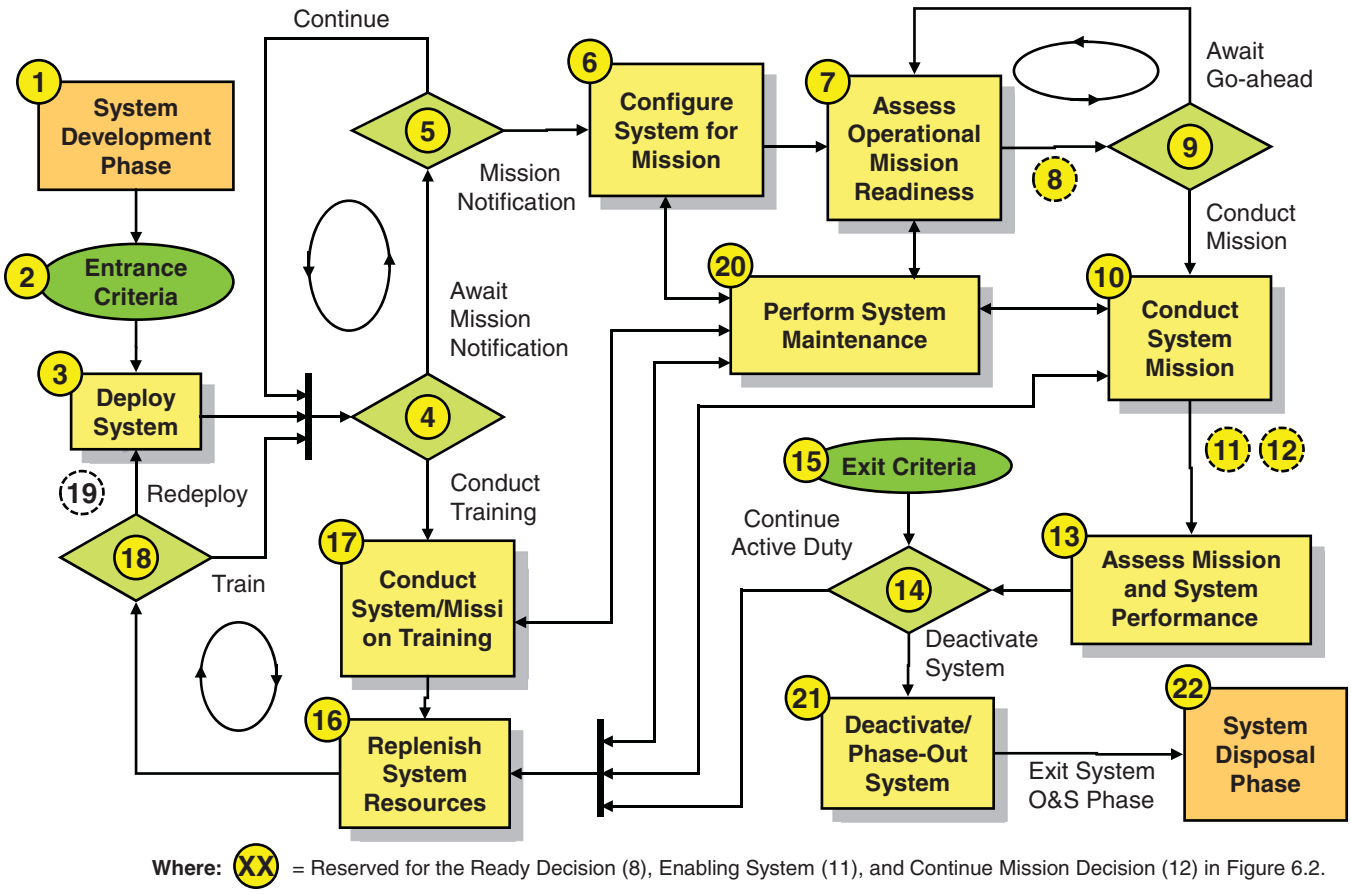


Figure 6.1 Generalized System Operations Model

2. Each decision block in the figure is referred to as a *control* or *staging gate* and requires a Go–No Go decision from a decision authority based on a pre-defined set of *exit* or *entry* criteria.
3. Operations 8.0, 11.0, 12.0, and 19.0 are reserved in Figure 6.1 for a follow-on discussion later in this chapter.

6.3.1 System Operations Model Description

Figure 6.1 depicts the System Operations Model that applies to most HUMAN SYSTEMS—enterprise and engineered. Entry into the model begins when a system completes and is transitioned from its System Development Phase of the System/Product Life Cycle. Entrance or *entry criteria* are evaluated to assess system readiness to begin active service. Let's explore each of the operations.

6.3.1.1 Operation 3.0: Deploy System Operation 3.0, Deploy the System, addresses system capabilities and activities required to deliver and install the system, product, or service at the User's required destination. As each system completes the System Development Phase or System Production Phase the system is packed and shipped for deployment or distribution to the User. Examples of operational activities include transportation; loading/unloading; crating/uncrating; initial setup, installation, and assembly; system checkout; verification; integration into higher-level systems; and verification of *interoperability* at that level. On completion of Operation 3.0, Operation 4.0, the Conduct System/Mission Training Decision is made.

6.3.1.2 Operation 4.0: Conduct System/Mission Training Decision Operation 4.0, Conduct System/Mission Training Decision, a decision control point, determines if the system is to be commissioned into *active service* or reserved for operator training or demonstrations:

- If the System/Mission Training Decision is Yes or True, workflow progresses to Operation 17.0, Conduct System Training.
- If the System/Mission Training Decision is No or False, workflow progresses to Operation 5.0, Await Mission Notification Decision.

6.3.1.3 Operation 5.0: Mission Notification Decision Operation 5.0, Mission Notification Decision, a decision

control point or gate, must await notification to initiate preparations to conduct a mission. Depending on the system and its mission application, Operations 4.0 and 5.0 are each effectively cyclical Do Until *wait states* that loop until a higher-level decision authority issues an order to perform the mission:

- If the Mission Notification Decision is Yes or True, workflow progresses to Operation 6.0, Configure System for Mission.
- If the Mission Notification Decision is No or False, workflow continues to cycle back to Operation 4.0, Conduct System/Mission Training Decision.

6.3.1.4 Operation 6.0: Configure System for Mission

Operation 6.0, Configure System for Mission, includes operational tasks and activities required to prepare and configure the system for the mission. On receipt of mission task orders, the system is configured for the mission. Depending on the type of system, Pre-Mission configurations may necessitate a standing configuration before mission tasking occurs, for example, a surgical suite in a hospital having a standing configuration before a crisis occurs versus the necessity to configure a commercial airline aircraft before a flight.

Operational activities include Pre-Mission analysis and planning; physical hardware and software updates, if required; PERSONNEL Element training; and replenishment of *consumable* and *expendable* MISSION RESOURCES. System configuration/reconfiguration activities include the *synchronized orchestration* of the System Elements such as:

- PERSONNEL Element—Operators, maintainer, administrators, etc.
- PROCEDURAL DATA Element—Operating procedures, media, etc.

On completion of Operation 6.0, System Verification is performed to ensure that the system is properly configured for the mission:

- If the verification is successful, workflow progresses to Operation 7.0, Assess Operational Mission Readiness.
- If system *latent defects* – design errors, flaws, or deficiencies -are discovered during system Pre-Mission inspections, workflow progresses to Operation 20.0, Perform System Maintenance.

6.3.1.5 Operation 7.0: Assess Operational Mission Readiness

Operation 7.0, Assess Operational Mission Readiness, includes system capabilities and activities required to review the overall readiness to conduct the assigned mission. After the system has been configured for the mission and all System Element resources are fully integrated and operational, mission operational readiness is assessed. The assessment evaluates the *readiness* posture of the integrated set of System Elements—such as EQUIPMENT, PERSONNEL, and FACILITIES—to perform their assigned mission on demand.

If the *readiness* assessment is No, the system is identified as *operationally deficient* with a color-coded tag or placard such as Red or Yellow. A mission impact risk assessment decision is made to determine if the deficiency warrants replacement of the SYSTEM/Entity with a backup system or postponement:

- If the system requires maintenance, workflow progresses to Operation 20.0, Perform System Maintenance.
- If the system is determined to provide the capabilities required to support the mission, workflow progress to Operation 9.0, Await Mission Go-Ahead Decision.

6.3.1.6 Operation 9.0: Mission Go-Ahead Decision

Operation 9.0, Mission Go-Ahead Decision, a decision control point, determines if tasking orders to conduct the mission have been issued:

- If Await Mission Go-Ahead Decision is Yes or True, workflow proceeds to Operation 10.0, Conduct Mission.
- If the Await Mission Go-Ahead Decision is No or False, system readiness is periodically checked by cycling back to Operation 7.0, Assess Operational Mission Readiness.

6.3.1.7 Operation 10.0: Conduct System Mission

Operation 10.0, Conduct System Mission, includes system operational tasks and activities required to conduct the system's primary and secondary mission(s). During the mission, the system may *encounter*, *engage*, and *interact* with external system threats and opportunities as it performs the mission objectives.

If the system requires maintenance during Operation 10.0, Operation 16.0, Replenish System Resources, or Operation 20.0, Perform System Maintenance, may be performed, where practical. Consider the following example.

Example 6.1 **Comparison of System Maintenance During a Mission**

- As a ground-based vehicle, most automobile repairs can be accomplished within a reasonable time period during its mission.
- As a space-based vehicle, maintenance of a satellite on Earth orbit may be impractical until a repair solution can be developed and manifested on an available flight in the future.

6.3.1.8 Operation 13.0: Assess Mission and System Performance Operation 13.0, Assess Mission and System Performance, includes system activities required to review the level of mission success based on mission primary and secondary objectives as well as MISSION SYSTEM and ENABLING SYSTEM performance contributions to that success. Activity examples include Post-Mission data reduction and analysis, target impact assessment, strengths, and weaknesses; threats; mission debrief observations and lessons learned; and mission success. These operations also provide the opportunity to review and assess the integrated PERSONNEL–EQUIPMENT Element interactions and performance; strengths, and weaknesses during the conduct of the mission; and corrective actions required.

6.3.1.9 Operation 14.0: Deactivate/Phase-Out System Decision Operation 14.0, Deactivate/Phase-Out System Decision, a decision control point, determines if the system is to continue current operations, be *upgraded*, or be *decommissioned* or *phased out* of active service. The decision is based on Operation 15.0 Exit Criteria that were established for the system:

- If the Deactivate/Phase-Out System Decision is Yes or True, workflow progresses to Operation 21.0, Deactivate/Phase-Out System.
- If the decision is No or False, workflow proceeds to Operation 16.0, Replenish System Resources.

6.3.1.10 Operation 16.0: Replenish System Resources Operation 16.0, Replenish System Resources, includes ENABLING SYSTEM operational tasks and activities required to *restock* or *replenish* system resources such as PERSONNEL and MISSION RESOURCES—*consumables* and *expendables*:

- If deficiencies are found in the system, the workflow returns to Operation 20.0, Perform System Maintenance.
- On completion of Operation 16.0, Replenish System Resources, workflow progresses to Operation 18.0, Redeploy System Decision.

6.3.1.11 Operation 17.0: Conduct System/Mission Training Operation 17.0, Conduct System Training, includes tasks and activities required to train Users - system operators, maintainers, and others - in how to properly operate the system. This includes classroom, simulator, and actual system usage. For larger, more complex systems, initial operator training is sometimes performed at the System Developer's factory prior to system deployment to the field. This includes normal operations as well as abnormal and emergency operations. *Remedial* and *skills enhancement* training occur after the system is already in field service.

During Operation 17.0, Conduct System Training, new system operators are instructed in the safe and proper use of the system to develop *basic* skills. Experienced operators may also receive *remedial*, *proficiency*, or *skills enhancement* training based on lessons learned from previous missions or new tactics employed by adversarial or competitive threats.

On completion of a training session, workflow progresses to Operation 16.0, Replenish System Resources. If the system requires maintenance during training, Operation 20.0, Perform System Maintenance, is activated.

6.3.1.12 Operation 18.0: Redeploy System Decision Operation 18.0 to redeploy the SYSTEM, a decision control point, determines if the physical system requires redeployment to a new deployment site to support organizational mission objectives:

- If Operation 18.0, Redeploy System Decision, is Yes or True, workflow progresses to Operation 3.0, which is to deploy the system.
- If Operation 18.0, Redeploy System Decision, is No or False, workflow proceeds to Operation 4.0, Conduct System/Mission Training Decision, and the cycle repeats back to Operation 18.0, Deploy System Decision.

6.3.1.13 Operation 20.0: Perform System Maintenance Operation 20.0, Perform System Maintenance, includes system capabilities and activities required to upgrade system capabilities or correct system deficiencies through *preventive* or *corrective* maintenance actions. SYSTEMS are tagged with easily recognizable color identifiers such as Red or Yellow to represent *corrective* or *preventive maintenance* actions (Chapter 34) required to correct any defects or deficiencies that may impact mission success.

On successful completion of Perform System Maintenance, the system is returned to *active service* via the next operation—be it Operation 6.0, Configure System Mission; Operation 7.0, Assess Operational Mission Readiness; Operation 10.0, Conduct Mission; Operation 16.0, Replenish System Resources; or Operation 17.0, Conduct System/Mission Training—of the requested need for maintenance.

6.3.1.14 Operation 21.0: Decommission, Deactivate, and Phase-Out System Operation 21.0, Decommission, Deactivate, and Phase-Out System, includes operational tasks and activities required to decommission, terminate, and remove the system from *active service*; store, warehouse, or disassemble the system; and properly dispose all its components and elements. Some systems may be placed in storage or "mothballed" until needed in the future to support surges in mission operations that cannot be supported by existing systems. On completion of the deactivation, the system proceeds to the System Disposal Phase of its System/Product Life Cycle.

6.3.2 System Operations Dictionary

Principle 6.3

System Operations Dictionary

Every project should consist of a *System Operations Dictionary* that clearly defines each system operation, its scope, and activities.

Obtaining team agreement on the graphical depiction of the ConOps is only the first step. When working with larger, complex systems and development teams, diagrams at this level require scoping definitions for each task and activity to ensure proper understanding among team members. For example, you and your team may define and scope a specific operational task or activity differently from a team operating in another business domain, depending on the system's application.

One solution is to create a *System Operations Dictionary*. The dictionary, which defines and scopes each capability similar to the previous System Operations Model descriptions, should be maintained throughout the life of the system.

In terms of accountability, the Systems Engineering and Integration Team (SEIT) under the leadership of the Project Engineer or LSE should lead plan and orchestrate development of the *System Operations Dictionary*. This effort should begin during the proposal phase of a system, product, or service prior to contract award and certainly on Day #1 After Contract Award (ACA).

6.3.3 Final Thoughts

The System Operations Model provides an initial, analytical framework for defining how systems, products, organizations, services, etc. will be deployed, operated, sustained, and retired/disposed. We can apply this model as an initial starting point for most, if not all, HUMAN SYSTEMS (Chapter 9)—enterprise or engineered—such as corporations, departments, projects, automobiles, airlines, hospitals, businesses, fire and ambulance services, etc. Collectively and individually, each of the model's operations represents a generalized construct that can be used as an initial starting point applicable to most systems.

- Your job as an SE is to collaborate with the Stakeholders—Users and End Users—to tailor the System Operations Model to reflect their needs within the constraints of contractual, statutory, and regulatory requirements. Each operation should be scoped and bounded via a *System Operations Dictionary* to ensure all members of the Acquirer, User, and System Developer teams clearly understand *what is/is not* included in specific operations—*with no surprises!*

Principle 6.4

Value-Added Operations Principle

Each System Operations Model operation or task and its performance by the System Elements either add value and contribute to achieving a mission-based performance objective or not; if not, eliminate it!

From an individual's perspective, the System Operations Model may appear to be very simple. However, on closer examination, even simple systems often require forethought to adequately define the operational sequences. If you challenge the validity of this statement, consider the following:

- Develop the System Operations Model for a car and driver.
- Conduct a similar exercise with each of three colleagues who are unfamiliar with the model. The diversity of colleague opinions may be enlightening.
- Repeat the exercise as a team focused on achieving a single, collaborative consensus for the final diagram.

Now, consider the case where the System Operations Model involves the definition of a more complex system with a larger stakeholder community. If you contemplated the previous car and driver exercise, you should appreciate the challenges of getting a diverse group of people from various disciplines, political factions, and organizations to arrive at a consensus on a System Operations Model for a specific system.

You will discover that the ad hoc, endless loop, Plug & Chug … SDBTF Engineering Paradigm (Wasson, 2012, p. 2) engineers often refer to the System Operations Model as "textbook stuff." Due to a lack of SE education and training, they:

- Do not recognize the need to spend time addressing this concept.
- Have a natural tendency to focus immediately on physical hardware and software design (Figure 2.3), such as resistors, capacitors, data rates, software languages, and operating systems.

A Word of Caution 6.1

If your project, customer, and User community has not agreed on some form of this top-level concept and its lower-level decomposition, System Development problems further downstream historically can be traced back to this fundamental concept. Even worse, fielding a system that *does not pass* customer validation for intended usage presents even greater challenges and risks, not only technically but also for your Enterprise's reputation.

Consider the following points:

- Obtain System Acquirer and User community consensus and "buy in" prior to committing resources for development of the system. Investigate how the User envisions operating the planned system to achieve organizational mission objectives. Avoid premature *ad hoc* "Plug & Chug … SDBTF" Engineering Paradigm hardware and software development efforts until these decisions are approved and flowed down and allocated to hardware and software specifications.
- Use the System Operations Model as an infrastructure to identify and specify UCs comprised of operational tasks. The tasks represent operational capabilities that can be translated into *SPS* requirements.
- When reviewing and analyzing specifications prepared by others, use the System Operations Model to assess top-level system performance requirements for completeness of system operations.

6.3.4 Developing a More Robust System Operations Model

The preceding System Operations Model provided a fundamental understanding of how a system might be employed by the User. As a high-level model, it serves as a useful instructional tutorial. The model, however, has some areas that need to be strengthened to accommodate a broader range of applications. Figure 6.2 provides an expanded System Operations Model. To maintain continuity with the previous model, we have preserved the original numbering convention and simply added the following operations:

- Operation 8.0: Mission Ready Decision
- Operation 11.0: Provide Mission Oversight and Support
- Operation 12.0: Mission Complete Decision
- Operation 19.0: Remediate and Restore Site

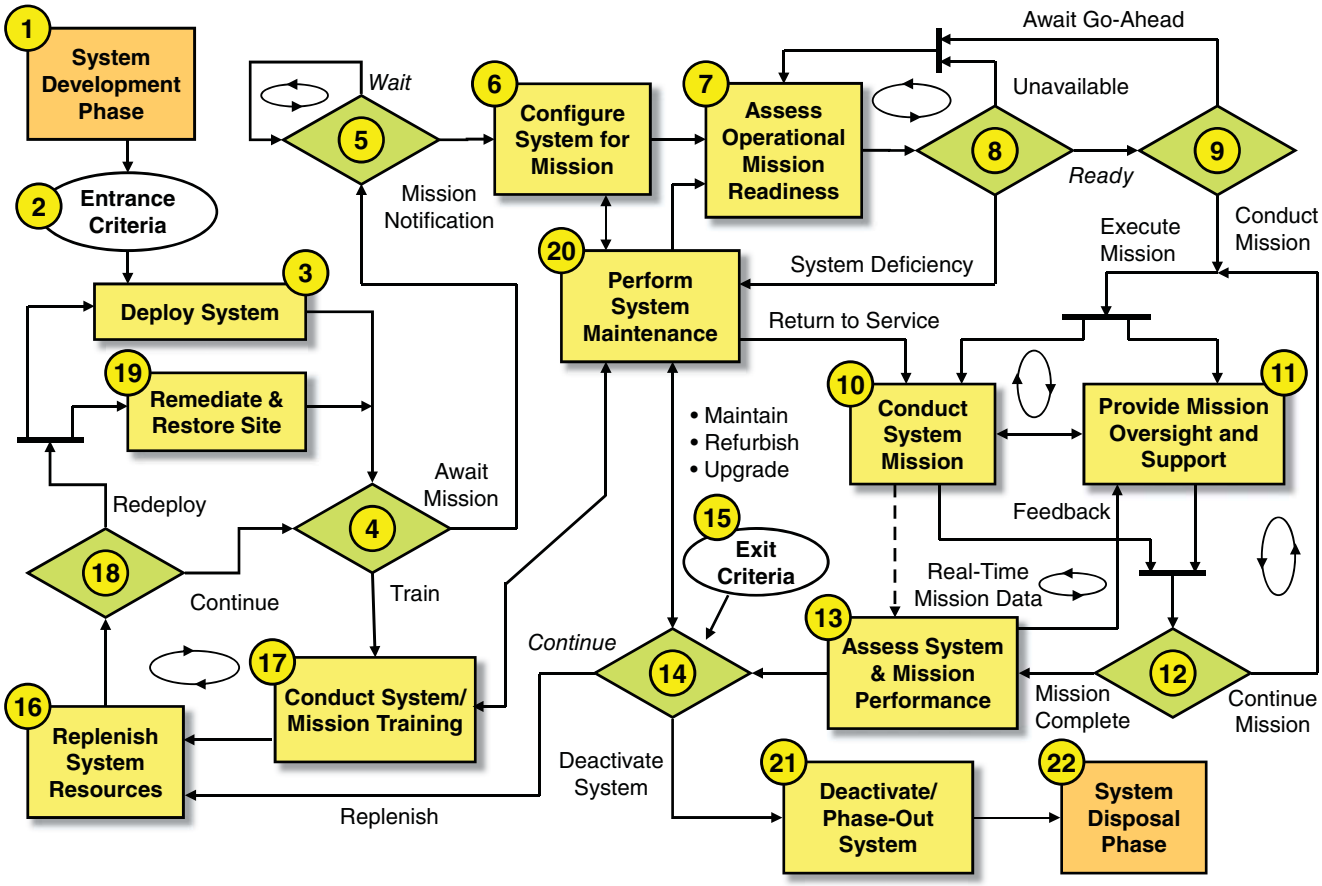


Figure 6.2 Robust System Operations Model

6.3.5 The Importance of the Generalized System Operations Model

Principle 6.5

Synchronized Operations Principle

Every System Operations Model activity should be synchronized with the system, product, or service's Mission Event Timeline (MET).

The System Operations Model (Figures 6.1 and 6.2) serves as a high-level framework that facilitates the orchestration of the totality of system synchronized to a time-based schedule such as a MET. Operations in the model represent UCs that require *capabilities* and time-based *interactions* between the SOI MISSION SYSTEM and one or more of its ENABLING SYSTEMS.

6.3.5.1 Specification Developer's Perspective From a specification developer's perspective, the System Operations Model construct provides the infrastructure for collaborating with Stakeholder Users and End Users to capture, organize, and create an analytical framework that enables us to identify operational tasks to be performed. Operational tasks, which represent Stakeholder UCs, provide the foundation for identifying MISSION SYSTEM and ENABLING SYSTEM capabilities. Capabilities will then be translated into specification capability requirements that are incorporated into the *SPS*.

6.3.5.2 Specification Analyst Perspective From a System Analyst's perspective, the System Operations Model construct can be used to correlate SPS requirements supplied by a third party with specific operations. If each System Operations Model operation is decomposed into hierarchical levels of sub-operations, the System Analyst can easily find the holes representing *missing or misplaced* requirements in the SPS or the need for clarification.

Author's Note 6.1

Many untrained specification writers focus exclusively on Operation 10.0, Conduct Mission. Even worse, they employ the *feature-based approach* in Chapter 20 APPROACHES by specifying *features* of the system for Operation 10.0.

As products of the Engineering education process, electrical, mechanical, and software disciplines, engineers immediately focus on their "comfort zone," physical system hardware and software requirements and solutions. Their specifications often fall short of complete system requirements coverage as noted by the absence of mission requirements for Operations 3.0 through 13.0 and 16.0 through 19.0. Even within Operation 10.0, Conduct Mission, specification writers focus only on specific physical features. As a result, critical requirements are *missed* or *misplaced* (Figure 20.1)

Despite the shortcoming noted in the previous points, standard system specification outlines such as the former MIL-STD-490A tend to guide the specification developers to at least partially consider these missing steps—Operations 3.0–13.0 and 16.0–19.0—in specification sections such as Design and Construction Constraints, Support, Training, etc.

Author's Note 6.2

Based on the author's experience, competent Systems Engineers (SEs) begin their systems analysis work with the System Operations Model or some version tailored specifically for their system application and User needs. This statement serves as a key indicator of the training and maturity level of Engineers and System Analysts claiming to be SEs. Application of the System Operations Model enables you to sort out the bona fide SEs from the ad hoc SEs and the level of risk associated with their positions on the program.

6.3.6 Assessing Coverage of Mission Operations

Finally, a key question you and your team will have to address is: *how do you know that all of the operations required for interactions between the* MISSION SYSTEM AND ENABLING SYSTEM(S) *have been properly addressed*?

One solution is to construct a simple matrix such as the one illustrated in Figure 6.3. Observe that the matrix lists the major operations from Figure 6.2 and links them to MISSION SYSTEM AND ENABLING SYSTEM operations as a function of Phase of Operation. Specifically, each bubble identifier represents specific operations that have to be accomplished by the MISSION SYSTEM AND ENABLING SYSTEM. MISSION SYSTEM AND ENABLING SYSTEM operations are paired—bubble IDs 1–2, 3–4, etc.—to represent interactions and outcomes as illustrated in Figure 7.4. This matrix can be tailored to meet the specific needs of the system. Gray fill areas represent operations that are not applicable. For example, Operation 3.0, Deploy System, obviously is not applicable during the Mission Phase of Operation.

A Word of Caution 6.2

When tailoring this matrix, avoid removing the bubble IDs. Simply gray out the background behind each identifier in the matrix. Why? Two reasons are:

1. It communicates to reviewers that you have assessed the applicability of an operation to your system and found it to be Not Applicable.

Where:

(#) = Reference to description

■ = Typically Not Applicable

Mission System Operations	System Phases of Operation					
	Pre-Mission		Mission		Post-Mission	
	Mission System Element	Enabling System Element	Mission System Element	Enabling System Element	Mission System Element	Enabling System Element
3.0 Deploy System	1	2	3	4	5	6
6.0 Configure System for Mission	7	8	9	10	11	12
7.0 Assess Operational Mission Readiness	13	14	15	16	17	18
10.0 Conduct Mission	19	20	21	22	23	24
11.0 Provide Mission Oversight and Support	25	26	27	28	29	30
13.0 Assess Mission and System Performance	31	32	33	34	35	36
16.0 Replenish System Resources	37	38	39	40	41	42
17.0 Conduct Mission/System Training	43	44	45	46	47	48
19.0 Remediate and Restore Site	49	50	51	52	53	54
20.0 Perform System Maintenance	55	56	57	58	59	60
21.0 Deactivate / Phase-Out System	61	62	63	64	65	66

Figure 6.3 Matrix for Mapping Mission System and Enabling System Operations to Phases of Operation

2. System requirements may change or reviewers may challenge your applicability assessment during a design review and determine that a bubble ID is applicable and should be reinstated, and described.

As a final point, meetings and reviews often turn into spirited discussions due to the participants using words that apply to different operations. The matrix enables the review leader or moderator to restrict focus to a specific bubble identifier, produce decisions, and target notes in meeting or review conference minutes. Figure 6.3 is actually a stepping-stone to a more comprehensive graphic introduced in Figure 10.21.

6.3.7 Summary: System Operations Model

The preceding discussions represent the embryonic, conceptual views of how the User intends to use the system. The General System Operations Model Construct:

1. Serves an initial starting point template for identifying high level operations applicable to most systems, products, or services. Since every system is unique, the intent of this discussion is to provide a basic orientation and awareness that will stimulate your "systems thinking" thought processes and enable you to plan, translate, and orchestrate these approaches into your own System Operations Model.
2. Shifts the Plug and Chug … SDBTF-DPM Engineering Paradigm (Chapters 2, 11, and 14) to focus on the need to understand all operations required to prepare a system for a mission, perform the mission, and maintain the system following a mission, not just mission operations.

6.4 FORMULATING AND DEVELOPING THE SYSTEM CONCEPTS

As the mission analysis identifies a system's UCs and scenarios, preferably in direct collaboration with the Stakeholders—Users and End Users—the next challenge is working with the User to *formulate* and *conceptualize* how they intend to deploy, operate, maintain, sustain, and retire/dispose of a system. One of the mechanisms for documenting the conceptualization is the System ConOps. This section introduces the System Operations Model that provides the structural framework for developing the ConOps.

Our discussions provide insights regarding how the model's operational tasks and activities are allocated and assigned to the System Elements such as Equipment, Personnel, and Facilities. As a result, these discussions provide the foundation for Chapter 7.

6.4.1 Developing the System ConOps

Once a system's *problem space* and *solution spaces* are fully understood and bounded, the next step is to understand *how* the User intends to employ a system, product, or service as an organizational solution space asset to conduct missions that resolve all or a portion of the Problem Space. Most systems are *precedented* (Chapter 1) and simply employ new technologies to build on the existing infrastructure of operations, FACILITIES, and skills. This does not mean, however, that *unprecedented* systems do not occur.

If we expand the Problem–Solution Space concepts (Figure 4.7), our analysis reveals that the Solution Space time-based interactions—namely, Entity Relationships (ERs)—can be generalized by a set of operations into a System Operations Model. In turn, the model provides a framework for developing the System ConOps, which describes the top-level sequential and concurrent operations required to deploy, operate, maintain, sustain, retire, and dispose of the system, product, or service.

Principle 6.6

ConOps Principle

Every project should include a *System ConOps* document that describes how the system, product, or service being developed is envisioned to be deployed, operated, maintained, sustained, retired, and disposed via its OCDs.

Example 6.2

A ConOps for a system such as NASA's Space Transportation System (STS)—Space Shuttle—describes the operational sequences and interactions of the MISSION SYSTEM and ENABLING SYSTEMS required to prepare for and conduct a mission to deliver a payload into outer space, deploy the payload and conduct experiments, and return the cargo and astronauts safely to Earth.

We can generalize a ConOps in terms of a common set of objectives that reflect how the User envisions employing the system, product, or service to fulfill its organizational missions. These objectives include:

- Deploy the system, product, or service.
- Configure the system for mission use.
- Assess the system's readiness to conduct Pre-Mission, Mission, and Post-Mission operations.
- Perform corrective actions to achieve mission readiness.
- Perform the System's Mission.
- Restore, replenish, refurbish, and/or store the system for the next mission.
- Decommission or dispose of the system, when appropriate.

There are numerous ways of developing a System ConOps that are organizational dependent. Table 6.1 provides an example outline listing of System ConOps topics.

Given this overview of the ConOps outlines, let's explore what would be included in descriptions of each of the System Concepts.

6.4.2 ConOps Accountability

In terms of accountability, the System Development Team (SDT) or Systems engineering and Integration Team (SEIT) under the leadership of the Project Engineer or LSE should lead plan and orchestrate development of the System ConOps. This effort should begin during the proposal phase of a system, product, or service prior to contract award and certainly on Day #1 ACA. Formulation and development accountability for each of the System Operational Concepts should be assigned to the Subject Matter Expert (SME) leads in each of the concept areas and then coordinated, integrated, and reviewed by the SEIT as well as other Stakeholders. On approval, baseline and release the document under formal configuration management control.

6.4.3 Integration of the System Concepts

The ConOps outline and the preceding discussions of formulating and developing the System Concepts may leave the impression that these exist as separate, static concepts. Avoid that logic!

The reality is these concepts are interrelated and transition from one to another according to the dynamics of deploying/redeploying the system, product, or service; conducting missions; performing maintenance; and placement in storage between missions as shown in Figure 6.4 derived from Figure 6.2.

In general, the System/Product Life Cycle Phases are too abstract to support complete analysis and require further decomposition and refinement. Each life cycle phase is further partitioned into Pre-XXXX Operations, XXXX Operations, and Post-XXXX Operations where XXXX represents a unique segment of mission operations. Table 6.2 represents implementation of the construct.

To illustrate the syntactical context of XXXX, consider the following examples.

Example 6.3

1. An aircraft's phases of operation are Pre-Flight, Flight, and Post-Flight operations.

TABLE 6.1 Example System ConOps Document Outline

Section	Section Title	Example Subsections
Section 1.0	Introduction	• Scope • System Purpose • General System Description • Definitions of Key Terms
Section 2.0	References	• User Documents • International Standards and Specifications • National Standards and Specifications • Interfacing System Documents • Project Documents
Section 3.0	System Missions	• System Roles and Missions • Mission Objectives • Mission Operations and Profile • Mission Event Timeline (MET) • System Stakeholders: Users and End Users • System UCs and Scenarios, etc. • System Objectives
Section 4.0	System Concepts	• System Deployment Concept • System Operations Concept • System Maintenance Concept • System Sustainment Concept (optional) • System Retirement Concept (optional) • System Disposal Concept (optional)
Section 5.0	System Architecture	• External Systems • System Context Diagram • Operational Architecture
Section 6.0	System Operations Model	Description of Model Operations
(Other topics)		
Appendices		As appropriate

2. A medical device's phases of operation are Pre-infusion, Infusion, and Post-infusion.
3. A football game's phases of operation consist of Pre-Game, Game, and Post-Game.

Let's establish the context of each type of operation.

6.4.3.1 Pre-XXXX Phase Objective The objective of the Pre-XXXX Phase of operations, at a *minimum*, is to ensure that an entity—SOI, MISSION SYSTEM and ENABLING SYSTEMS—is fully prepared, configured, operationally available, and ready to conduct its organizational mission when directed or tasked.

6.4.3.2 XXX Phase Objective The objective of the XXXX Phase of operations, at a *minimum*, is to conduct the entity's primary mission. Besides achieving the SOI's mission objectives, one must mitigate mission risks and ensure the system's safe operation and return.

6.4.3.3 Post-XXX Phase Objective The objective of the Post-XXXX Phase of operations, at a *minimum*, is to:

- Analyze an entity's mission outcome(s) and performance objective results
- Replenish system *consumables* and *expendables*, as applicable
- Refurbish the system
- Capture lessons learned
- Analyze and debrief mission results
- Improve future system and mission performance

To see how Phases of Operation may apply to a system, consider the following example of an automobile trip—mission.

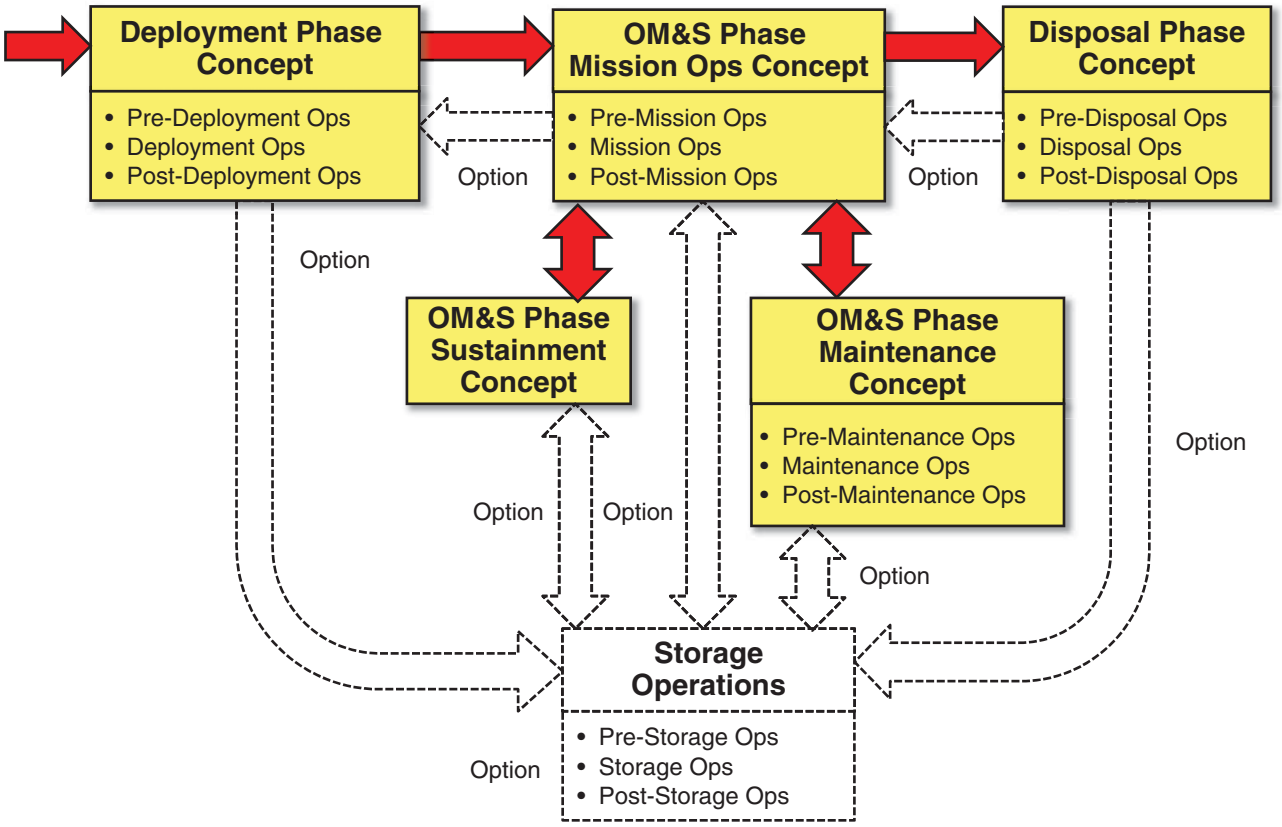


Figure 6.4 Fielded System/Product Life Cycle Concepts and Operations

TABLE 6.2 Applications of the Pre-XXXX Operations, XXXX Operations, and Post-XXXX Operations Construct

Life Cycle Phase	Pre-XXXX Operations	XXXX Operations	Post-XXX Operations
Deployment	Pre-Deployment Operations	Deployment Operations	Post-Deployment Operations
Operations (OM&S)	Pre-Mission Operations	Mission Operations	Post-Mission Operations
Maintenance (OM&S)	Pre-Maintenance Operations	Maintenance Operations	Post-Maintenance Operations
Storage (OM&S) Option	Pre-Storage Operations	Storage Operations	Post-Storage Operations
Retirement/Disposal	Pre-Disposal Operations	Disposal Operations	Post-Disposal Operations

Example 6.4

During the Pre-Mission Phase prior to driving an automobile on a trip, the driver:

- Services the vehicle (oil and filter change, new tires, repairs, etc.)
- Fills the tank with gasoline
- Checks the tire pressure
- Inspects the vehicle
- Loads the vehicle with personal effects (suitcases, coats, etc.)

During the Mission Phase, the driver:

- Departs on the trip from the point of origination
- Drives defensively in accordance with vehicle safe operating procedures
- Obeys vehicular laws
- Navigates to the destination
- Periodically checks and replenishes the fuel and coolant supply en route
- Arrives at the destination

During the Post-Mission Phase on arrival at the point of destination, the driver:

- Parks the vehicle in a permissible space
- Unloads the vehicle
- Safely secures the vehicle until it is needed again

TABLE 6.3 Examples of Deployment Phase Operations

Deployment Concept UCs		
Pre-deployment Operations UCs	Deployment Operations UCs	Post-deployment Setup Operations UCs
1. Teardown (option)	1. Transport	1. Unload
2. Disassemble (option)	2. Track Location	2. Uncrate
3. Inventory	3. Inspect	3. Unpack
4. Package	4. Storage (optional)	4. Inventory
5. Pack	5. Conceal	5. Assemble
6. Crate		6. Install
7. Load		7. Checkout
8. Store and Protect		8. Verify
		9. Storage (optional)

Observe in the example that each of the bulleted activities represents the Driver's UC of the automobile in the same manner as the Office Copier Example presented in Table 5.1. This is a key point in the discussions that follow. Remember that UC operational tasks are assigned, allocated to, and performed by the PERSONNEL and the EQUIPMENT Elements.

6.4.3.4 System Deployment Concept Development The System Deployment Concept is developed in collaboration with the User and describes the vision for fielding a system, product, or service. Deployment encompasses transport, storage, installation, checkout, and verification of system readiness to perform its missions. Deployment operations are performed by the User or a vendor contracted to perform the deployment. Table 6.3 provides an example listing of Deployment Concept UCs that require System User or Acquirer consideration when specifying the system, product, or service's capability requirements prior to the System Development Phase.

Author's Note 6.3

Observe that Table 6.3 and the ones that follow use an *active verb* applicable to the system to represent an *action* to be performed that may apply to numerous types of activities. For example, the Checkout UC may have UC extensions (Chapter 5) such as Checkout Sensor, Checkout Computer, Checkout System, and so forth.

To illustrate how Table 6.3 applies to MISSION SYSTEM and ENABLING SYSTEM development, consider the following example:

Example 6.5

Heavy Construction Crane Deployment Example

Assume we are developing a large crane used to erect tall buildings, specifically the transport of the crane to new job sites. To accomplish the Deployment Operations, Transport UC, the crane must be capable of being retracted into a small form factor to fit on some form of transporter vehicle; provide hooks and eyelet tie-downs for securing the crane to the vehicle; and travel on roadways that have restrictions in terms of size, weight, markers, etc.

Several key points to consider about Table 6.3 and those that follow are:

1. Observe that the list of process flow of tasks accommodates both large complex systems that require Teardown and Disassembly as well as commercial products that require packaging, packing, etc.
2. Whereas some commercial products are deployed and distributed to stores for sale as ready-to-use "out of the box," other products and larger, complex systems may require setup such as TVs, hospital EQUIPMENT, etc.
3. There may be some instances whereby the system, product, or service is placed in STORAGE or "On Hold" until requested for missions.
4. Operations represent what the SOI—MISSION SYSTEM and ENABLING SYSTEM(S)—must be capable of providing. Other operations may be unique to the MISSION SYSTEM or ENABLING SYSTEM(S). Some may require the MISSION SYSTEM to be operating; others do not.
5. Observe that even though each operation is sequentially numbered within a given cell, each one can be assigned

a unique code to be used as the basis for estimating SSE costs, correlating with the Work Breakdown Structure (WBS), and be used for collecting labor and other charges during System Development.

6.4.3.5 System Operations Concept Development The System Operations Concept is developed in collaboration with the User and describes the vision for operating a system, product, or service to accomplish organization-level mission objectives. These operations are typically performed by the System User or could be contracted to a Services Provider. Table 6.4 provides an example listing of System Operations Concept UCs that require System User or Acquirer consideration when specifying the system, product, or service's capability requirements prior to the System Development Phase.

Key points include:

1. Mission Operations such as Normal Ops, Degraded Ops, Emergency Ops represent the core capabilities required of the system, product, or service required to conduct its mission.
2. Whereas the concept of Model-Based Systems Engineering (MBSE) applies to the Deployment, OM&S, and Disposal Phase, the OM&S Phase Mission Operations should be a key focal point for modeling system performance.

6.4.3.6 System Maintenance Concept Development The System Maintenance Concepts is developed in collaboration with the User and describes the vision for how the system, product, or service will be maintained prior to, during, and after a mission. Maintenance operations are typically performed by the System User or by one or more vendors contracted to provide Mission Support Operations. Table 6.5 provides an example listing of System Maintenance Concept UCs that require System User or Acquirer consideration when specifying the system, product, or service's capability requirements for maintenance support.

TABLE 6.4 Examples of System Operations Concept UCs

System Operations UC Examples		
Pre-Mission Operations UCs	Mission Operations UCs	Post-Mission Operations UCs
1. Power Up	1. Normal Ops	1. Safe and Secure
2. Initialize	2. Degraded Ops	2. Retrieve
3. Configure	3. Emergency Ops	3. Analyze
4. Align	4. Power Down	4. Report
5. Calibrate		5. Refurbish
6. Train		6. Power Down
7. Replenish		
8. Power Down		

6.4.3.7 System Storage Concept Development (Optional) The System Storage Concept, which may or may not apply to your system, is developed in collaboration with the User and describes the vision for how the system, product, or service will be stored between missions. Storage Phase operations are typically performed by the System User or by one or more vendors contracted to provide Storage Operations. Table 6.6 provides an example listing of System Maintenance Concept UCs that require System User or Acquirer consideration when specifying the system, product, or service's capability requirements for maintenance support.

Key points include:

- The term *Storage* can have several different meanings. We typically think of storage as placement in an environmental controlled facility; however, storage of an aircraft might include parking on a tarmac. Since space at an airport is important, commercial aircraft might be relocated to a dry desert environment such as the Mojave Air and Space Port in Mojave, CA.

6.4.3.8 System Sustainment Concept Development The System Sustainment Concept is typically developed by the User and describes the vision concerning how the User or a third party will establish logistical supply chains to ensure a continual pipeline of MISSION RESOURCES (Chapter 8)—*consumables* and *expendables* including parts—to ensure that system, product, or service is sustainable in the field to perform its missions with minimal interruption. System Sustainment Concept UCs include:

- Analyze Failure Frequencies
- Maintain Inventory
- Order Parts
- Deliver Parts
- Deliver Consumables
- Deliver Expendables

6.4.3.9 System Retirement and Disposal Concept Development The System Retirement/Disposal Concept is typically developed by the Users and describes the vision for retiring and/or disposing of a system, product, or service that has been *decommissioned* or *deactivated* from active service. The User is organizationally accountable planning and orchestrating the retirement and disposal of an

TABLE 6.5 Examples of System Maintenance Concept Use Cases

System Maintenance Concept UC Examples		
Pre-Maintenance Operations UCs	Maintenance Operations UCs	Post-Maintenance Operations UCs
1. Relocate	1. Preventative Maintenance	1. Inspect
2. Inspect	2. Corrective Maintenance	2. Assess
3. Troubleshoot	3. Others	3. Verify
4. Analyze		4. Document
5. Document		5. Dispose/Recycle
6. Order Parts		6. Release
7. Receive Parts		

TABLE 6.6 Examples of System Storage Concept Use Cases

System Storage Concept UC Examples		
Pre-Storage Operations UCs	Storage Operations UCs	Post-Storage Operations UCs
1. Redeploy	1. Monitor	1. Deinstrument
2. Inspect	2. Inspect	2. Configure
3. Configure	3. Maintain	3. Replenish
4. Instrument	4. Replenish (optional)	4. Inspect
5. Protect		5. Redeploy
6. Secure and Protect		

asset. The actual retirement and disposal may be performed by the User, performed by another organization, or contracted out to a services provider organization that has the required capabilities. Table 6.7 provides an example listing of Retirement/Disposal UCs that require System User or Acquirer consideration when specifying the system, product, or service's MISSION SYSTEM and ENABLING SYSTEM capability requirements.

To illustrate how systems are retired and ultimately disposed, consider the following example.

Example 6.6 **NASA Space Shuttle Fleet Retirement Example**

At the completion of NASA's Space Shuttle Program, the fleet of shuttles—*Discovery*, *Atlantis*, and *Endeavor*—were retired between March and July 2011 (NASA, 2011). On completion of their final missions, all three were subsequently donated to museums. Each was ferried atop its 747 carrier from the landing site to airports in cities near the museums. Each was then transported over land, by air, or by water to their final resting places at the museums. Although the flight vehicles were not disposed, hardware components were donated, disposed of, or reused. One of the three Multi-purpose Logistics Modules (MPLM) underwent conversion for the International Space Station (ISS).

Figure 6.3 summarizes operations from all of these tables based on the sequencing options for system operations introduced in Figure 6.2.

6.5 CHAPTER SUMMARY

In summary, our discussions in this chapter addressed the formulation and development of the System Concepts that are documented as OCDs in the System ConOps document. Key points include:

- Develop and tailor a System Operations Model to serve as a guide for communicating the vision of how a system, product, or service will be deployed, operated, maintained, stored, sustained, retired, and disposed.
- Develop a ConOps document to capture the System Operations Model and the System Concepts—Deployment, OM&S, etc.

TABLE 6.7 Examples of System Retirement/Disposal Concept Use Cases

System Retirement / Disposal Concept UC Examples		
Pre-Disposal Operations UCs	Disposal Operations UCs	Post-Disposal Operations UCs
1. Redeploy	1. Destroy	1. Recover
2. Remove Toxic Hazards	2. Document	2. Donate
3. Configure		3. Recycle
4. Document		4. Reuse
		5. Reclaim
		6. Remediate
		7. Document

- The ConOps should:
  - Be lead and developed by the Project Engineer in collaboration with the Stakeholders—Users and End User.
  - Serve as a *shared vision* among the System Developer, System Acquirer, User, and End user communities to guide the development of the system, product, or service.
  - Focus on all aspects of system deployment, operation, maintenance, storage, sustainment, retirement, and disposal due to the implications of required operational capabilities in developing the system; however, the User may selectively exclude sustainment, retirement, and disposal and declare it outside the scope of the System Development contract.
  - Describe how the System Concepts are seamlessly integrated to accomplish smooth transitions.

6.6 CHAPTER EXERCISES

6.6.1 Level 1: Chapter Knowledge Exercises

1. What is the System Operations Model?
2. How do you graphically illustrate the System Operations Model?
3. How do you describe each of the model's operations?
4. How do you delineate the differences in the System Operations Model (Figure 6.1) from its robust version (Figure 6.2)?
5. What is a System Operations Dictionary?
6. What is a ConOps and how much does it facilitate System Development?
7. What Stakeholders have vested interests in a ConOps?
8. What System/Product Life Cycle Phases are covered by the ConOps?
9. Which System/Product Life Cycle Phases may not be covered by the ConOps and why?
10. What is the purpose of the System Operations Model and how does it aid development of the ConOps?
11. What topics should a ConOps, at a minimum, include?
12. What is a System Deployment Concept? Identify examples of its UCs.
13. What is a System Operations Concept? Identify examples of its UCs.
14. What is a System Maintenance Concept? Identify examples of its UCs.
15. What is a System Storage Concept? Identify examples of its UCs.
16. What is a System Sustainment Concept? Identify examples of its UCs.
17. What is a System Retirement/Disposal Concept? Identify examples of its UCs.

6.6.2 Level 2: Knowledge Application Exercises

Refer to www.wiley.com/go/systemengineeringanalysis2e

6.7 REFERENCES

DAU (2012), *Glossary: Defense Acquisition Acronyms and Terms*, 15th ed. Ft. Belvoir, VA: Defense Acquisition University (DAU) Press. Retrieved on 5/16/15 from http://www.dau.mil/publications/publicationsDocs/Glossary_15th_ed.pdf.

Hench, John (2009), *Designing Disney*, New York, NY: Disney Editions.

NASA (2011), *NASA Announces New Homes for Space Shuttle Orbiters After Retirement*, Washington, DC: National Aeronautics and Space Administration (NASA), April 12, 2011, Accessed 3/16/13 http://www.nasa.gov/topics/shuttle_station/features/shuttle_homes.html.

Wasson, Charles S. (2012), *System Engineering Competency: The Missing Course in Engineering Education*, *119th Annual ASEE Conference and Exposition*, San Antonio, TX: http://www.asee.org/public/conferences/8/papers/3389/view

7

SYSTEM COMMAND AND CONTROL (C2) - PHASES, MODES, AND STATES OF OPERATION

Chapter 4 highlighted the importance of partitioning *abstract system complexity* into lower levels of refinement that provide increasing clarity and manageable risk (Principle 4.17). The overarching theme is twofold:

1. Partition and refine a *problem space* into more or more *solutions spaces* to manage risk—*what* into *how*.
2. Assimilate—integrate—the refined "pieces" into higher levels of abstraction that represent *what* has to be accomplished and *why*.

This chapter represents a *solution space* for "how" we solve a problem. However, the title does not communicate the second point—what *problem space* they solve and "why" we need System Phases, Modes, and States of Operation. As a result, most Enterprises and Engineers approach the topic as "something we do" without ever understanding "why."

If you trace the evolution of System Development from the beginning of time, humans solved operational needs—*problem spaces*—by creating simple tools such as the lever, wheel, and spear, with specific capabilities. Early approaches to *problem-solving* and *solution development* focused on *developing* and *improving capabilities*.

As systems evolved, humans began to recognize the need to develop systems to not only accommodate a diverse range of usage and Operating Environment conditions but also *how to* configure and control system capabilities in certain situations. Specifically, deploy and redeploy systems to new sites, configure and prepare the system for battle, and perform maintenance after usage. Over time, the need for operator control became more evident, especially in situations related to hazardous conditions and usage related to safety. Examples included brakes on horse-drawn wagons, safety mechanisms on guns, and so forth. As a result, Engineering evolved into a build, test, fix paradigm for problem-solving–solution development.

As systems became more complex and powerful, the challenge for humans was how to Command and Control (C2) a system and the "forces of nature"—definition of Engineering (Chapter 1)—to accomplish their missions. Examples include: firearms, munitions, water turbines, dams, steam engines, and so forth. If there was a problem, the Engineering mind-set was to "add another capability" to prevent another capability or a scenario from occurring.

These capabilities, however, had its limitations and could cause a system to fail, especially if the mechanisms exceeded system boundary envelope performance thresholds. System performance monitoring became a function of the operator's instincts. Obviously, instincts varied from one operator to another and so did the system outcomes. So, the next challenge became: *how to we monitor the performance of a system to be able to predictably and repeatedly C2 its operation within its boundary performance envelopes*?

As technologies evolved, capabilities to "monitor" system or product performance evolved. Examples included: notches on trees as early depth gauges in rivers and streams, rotational frequency of windmills, turbines, ship speed based on trailing rope-based knots, sextants for navigation, and

System Engineering Analysis, Design, and Development: Concepts, Principles, and Practices, Second Edition. Charles S. Wasson.

Companion Website: www.wiley.com/go/systemengineeringanalysis2e

sundials. As a result, humans began to learn how to Monitor, Command, and Control (MC2) system performance.

With the evolving MC2 methods came the need for humans to *communicate* not only the configuration of a system but also its *current availability* and *readiness* to conduct missions, *operating condition*, and *operating status*. Since the laws of physics: (1) characterized physical interactions and (2) fluids changed *states*—solids, liquids, and gases, physics provided a vocabulary of terms for characterizing the *state* and *condition* of a system. Although the concept of *state machines* and *state transitions* are modern day terms, they characterize the early machines.

Observe the context of the *state* discussion. It focuses on the *physical configuration* of the system. However, they were soon confronted with the reality that they needed to redeploy the system to different geographic regions that required different methods—*modes*—of transportation that did not necessarily: (1) *impact the physical state* of the system or (2) require disassembly and reassembly for transport by other systems across geographical barriers such as oceans, mountains, rivers, and so forth.

Chapter 7 introduces the concept of system phases, modes, and states of operation that enable operators and maintainers to MC2 a System or Product to accomplish both mission and system objectives.

7.1 DEFINITIONS OF KEY TERMS

- **Affinity Analysis**—A data research and analysis method to discover similarities or common occurrences across sets of data.
- **Allowable Action**—A User- or system-selectable capability that is *available* or *enabled* for use when required for a given set of circumstances or conditions.
- **Mode of Operation**—An abstract label applied to a User selectable option that enables a set of UC-based capabilities to be employed in conjunction with Enterprise processes and procedures to MC2 an Enterprise or Engineered system, product, or service to achieve a specific mission outcome, objectives, and levels of performance.
- **Monitor, Command, and Control (MC2)**—A key mission task assigned and allocated to the Personnel, Equipment, and Procedural Data System Elements for monitoring system performance, issuing commands, and controlling performance of all system operations to ensure stability, safety, and successful completion of the mission.
- **Operational Health and Status (OH&S)**—The current operating condition and operational status of a System or Entity and its current state of use.
- **Phase of Operation**—Refer to the Chapter 5 Definitions of Key Terms.
- **Phase**—A label applied to segments of the System/Product Life Cycle or Mission Life Cycle of a System or Product—for example, Pre-Mission, Mission, and Post-Mission. Phases may consist of sub-phases. For example, an aircraft's Mission Phase of operations may be partitioned into Phases of Flight (sub-phases) such as takeoff, ascend, cruise, approach, or land.
- **Prohibited Action**—A system capability that is disallowed or inhibited for specific modes of operation and is not available as a User- or system-selectable option for a given set of circumstances or conditions. For example, car doors cannot be opened unless the automobile is in Park Mode.
- **State**—An observable and measurable physical attribute used to characterize the current configuration, status, or performance-based *condition* of a System or Entity. Based on principles of physics, *states* represent *conditions* that are *observable*, *measurable*, and *verifiable*. We can classify them in terms of four contexts: System States, Operational States, Configuration States, and Dynamic States (Wasson, 2014, p. 4).
  - **Configuration State**—"An attribute that characterizes the *physical* arrangement of components and connectivity of a system, product, or service's multi-level architecture required to support achievement of one or more Use Case (UC)-based objectives and levels of performance." (Adapted from Wasson, 2014, p. 4).
  - **Dynamic State**—"An attribute that characterizes a brief, time-dependent, response, instability, or perturbation—attitude, motion, or performance—induced by self-interactions or external interactions with specific types of Operating Environment conditions" (Wasson, 2014, p. 4). Dynamic States, as special conditions of an Operational State, have a present participle "ing" suffix such as: initializing, melting, landing, or accelerating.
  - **Operational State**—"An attribute that characterizes the operational *status*, *readiness*, *availability*, or *condition* of a system, product, or service at a specific instant in time to conduct or continue a mission. For example, a system or product is *active* or *inactive*; *operational/operating* (On), or *non-operating* (Off); failed; awaiting maintenance" (Adaption of Wasson, 2014, p. 3).
  - **System State**—"An attribute that represents the current logistical employment, availability, or performance-based condition of an Enterprise asset such as a system, product, or service. *System State* examples include: (in) Storage; (in) Deployment; (in) Operation, (in) Maintenance, (in) Retirement, (in) Disposal" (Adaption of Wasson, 2014, p. 3).

- **State Diagram**—"A diagram that depicts the states that a system or component can assume and shows the events or circumstances that cause or result from a change from one state to another" (Copyright © 2014 ISO/IEC/IEEE. Used by permission.). State Diagrams are also called State Transition Diagrams.
- **State of Operation**—The current OH&S status or operating condition of a System of Interest (SOI) required to safely conduct or to continue its mission.
- **Triggering Event**—An external OPERATING ENVIRONMENT stimuli, excitation, or cue such as a command or interrupt that causes a system, product, or service to shift from a current mode or state to the next mode or state.

7.2 APPROACH TO THIS CHAPTER

The concept of Modes and States of Operation is often very challenging due to the lack of SE courses in Engineering education (Chapter 2) and abstract references in professional standards. As a result, every Engineer, Enterprise, industry, and professional organization has its own perspectives, viewpoints, and applications concerning *modes* versus *states*.

Modes and states are often treated as "after thoughts" when a SYSTEM design is complete. Engineers scramble to develop operator manuals and user guides required as contract or consumer product deliverables. As a result, modes and states may implicitly exist in the System Design but are often "discovered" and "labeled" as such when a system, product, or service is nearing completion.

The untold reality is: *modes and states* serve as an analytical *decision aid framework* for conceptualizing *how* a User—operator, maintainer, trainer, et al—can MC2 a system, product, or service as illustrated in Figure 7.1.

When (1) Engineers apply the *ad hoc, endless loop,* Specify-Design-Build-Test-Fix (SDBTF)—Design Process Model (DPM) Engineering Paradigm (Chapter 2) and (2) take a "quantum leap" from requirements to physical design illustrated in Figure 2.3 to *modes* and *states*, the result can be devastating or even worse catastrophic. Imagine separate Product Development Teams (PDTs) designing, building, and verifying discrete automobile PARK, NEUTRAL, REVERSE, DRIVE, LOW 1, AND LOW 2 transmission components. Then, coming together to integrate them into he transmission assembly. On completion, they perform System Integration, Test, and Evaluation (SITE). During SITE, you and your team discover that the transmission will not shift between gears, and even worse … causes the parts to mechanically disintegrate when they engage. The result is a lack of *design integration* to achieve C2 of the vehicle in all aspects of User UCs operation.

In contrast, System Phases, Modes, and States of Operation enable SEs to conceptualize the *analytical framework* "up front" in System Development that will C2

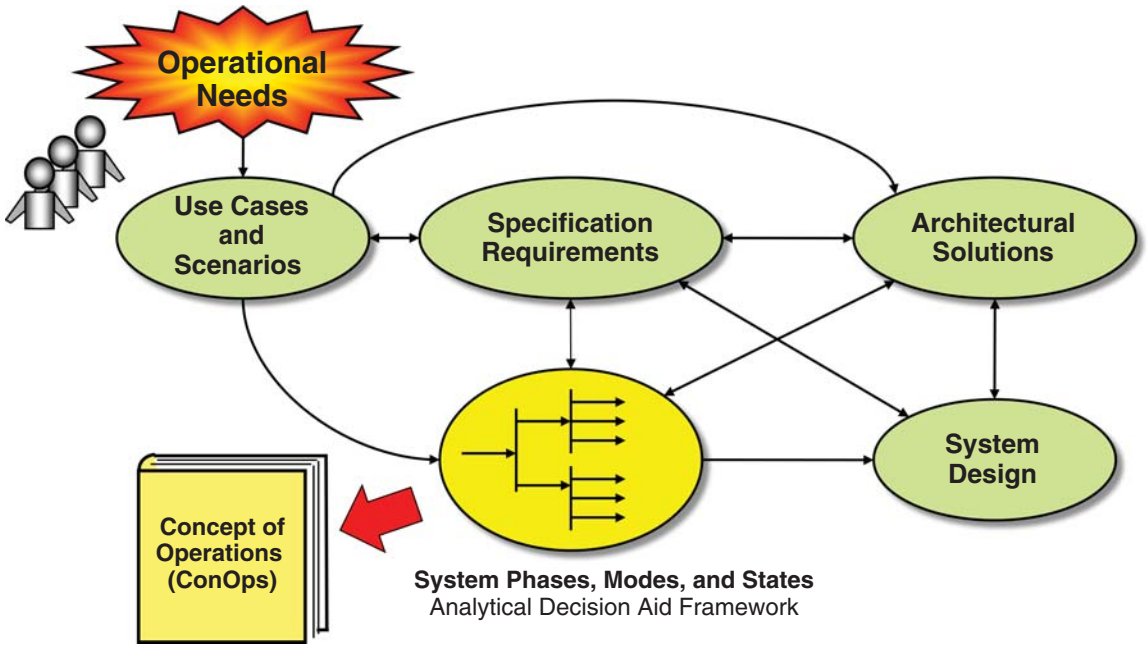


Figure 7.1 Phases, Modes, and States: Bridging UCs, Specification Requirements, Architectural Solutions, and System Design

the configuration and boundary conditions of the System Architecture and subsequently the System Design Solution. As an analytical framework, System Phases, Modes, and States of Operation serve as a linking mechanism for key technical decisions that influence development of specification requirements, the Concept of Operations (ConOps) Document, the system architecture, and subsequently the System Design. Within System Phases of Operation, the modes and states analytical framework provides the infrastructure for Model-Based Systems Engineering (MBSE) addressed in Chapters 10 and 33.

This chapter addresses four questions that are often the sources of controversial issues:

1. What are *modes* and *states* and how do they differ?
2. Do modes contain states or do states contain modes of operation?

Due to the complexities and controversies concerning system modes and states, our approach to this chapter will be to define the terms and do so based on their their Entity Relationships (ERs). Figure 7.2 provides an illustration of the ERs that will guide our discussion.

Let's begin with System Phases of Operation.

7.3 SYSTEM PHASES OF OPERATION

Since systems, products, or services are mission-oriented and are characterized at the highest level by the System/Product Life Cycle (Figure 3.3) consisting of System Phases of Operation that serve as the *analytical foundation* for analyzing and developing systems. Within the Operations Phase, we establish the Mission Life Cycle (Figure 5.5) consisting of the Pre-Mission, Mission, and Post-Mission Phases.

Our earlier discussion of the System Operations Model (Figures 6.1 and 6.2) introduced and described a *workflow* of operational tasks that represented *how* HUMAN SYSTEMS—Enterprise and Engineered—prepare for, conduct, and follow-up after missions. We later partitioned these operational tasks into Pre-Mission, Mission, and Post-Mission operations in Table 6.2. Tables 6.3–6.7 listed

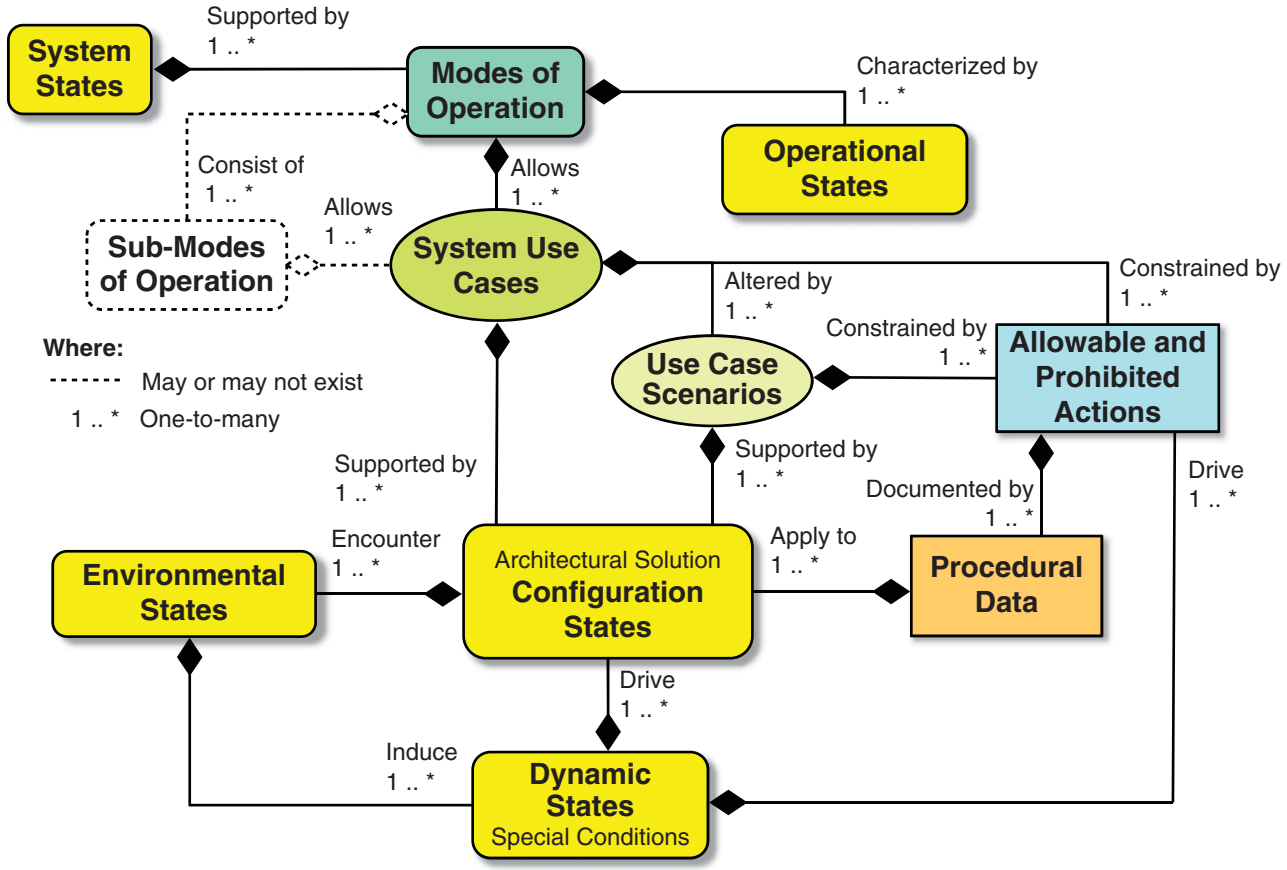


Figure 7.2 Illustration Depicting the Entity Relationships (ERs) of Modes and States

Use Cases (UCs) for the various System/Product Life Cycle Phases.

In general, the System/Product Life Cycle Phases and Mission Life Cycle—Pre-Mission, Mission, and Post-Mission—enable us to establish the foundational infrastructure for System Analysis. However, for more complex systems, these terms may be too abstract. This brings us to our next topic, Sub-Phases of Operation.

7.3.1 Sub-Phases of Operation

Sub-Phases of Operation Principle

Principle 7.1 When Enterprise or Engineered system, product, or service's Mission Life Cycle phases of operation become too abstract, partition them into lower level sub-phases of operation, outcomes, and objectives.

Mission Life Cycle phases of operation—Pre-Mission, Mission, and Post-Mission—are characterized by performance-based outcomes and objectives. When those outcomes and objectives are abstract, SEs need to partition them into lower level outcomes and objectives that contribute to the accomplishment of the higher level. To illustrate this point, consider aircraft operations.

Applying the naming convention introduced in Table 6.2, an aircraft's Mission Life Cycle consists of Pre-Flight, Flight, and Post-Flight phases of operation. The question is: *What do each of these terms mean*? We know that an airline transports passengers between airports. Therefore, the aircraft needs to be capable of supporting and performing several types of sequential operations—Load passengers, cargo, fuel, and food; Push Back from the terminal; Start the engines; Taxi to the runway; Taking Off; Climbing to altitude; Cruise and navigate to the destination; Descend for landing; Approach the airport; possibly Hold in a pattern until cleared to land; Land on the runway; Taxi to the terminal; Park at the terminal; Unload passengers, cargo, and refuse; and perform Maintenance (Figure 7.3). So, *how do we link the abstract Mission Life Cycle Pre-Mission, Mission, and Post-Mission phases to these sequential operations*?

Since aviation is based on Phases of Flight, we could construct a matrix that maps—links—the two data sets. As a result, we abstract:

- Load, Push Back, Start, and Taxi into Ground Departure Operations performed during the Pre-Flight Phase.
- Takeoff, Climb, Cruise, Descend, Approach, Hold, and Land into Flight Operations performed during the Mission Phase.
- Taxi, Park, Unload, and Maintenance into Ground Arrival Operations performed during the Post-Flight Phase.

Figure 7.3 illustrates the results. Note that due to space restrictions, the Start, Hold, and Maintenance operations are not shown in the figure.

Recognizing this was an aircraft example, *how would we analyze a device such as a medical infusion pump that administers meds to a patient*? Applying the naming convention from Table 6.2, we would designate the Mission Life Cycle Phases as Pre-Infusion, Infusion, and Post-Infusion Operations.

For simplicity, we use the terms Pre-XXXX, XXXX, and Post-XXXX Phases to simply categorize Stakeholder—User and End User—Mission Life Cycle operations. Tables 6.3–6.7 provided listings of Stakeholder UCs. Within a specific System/Product Life Cycle Phase, some SYSTEMS or PRODUCTS may have other phases of operation such as Storage or Refurbishment. For example, assume an airline has an excess of aircraft in inventory due to reduced traveler demand. Some are placed in a Storage Phase—mothballed—in desert OPERATING ENVIRONMENT conditions until marketplace demands motivate the need for return to active duty service.

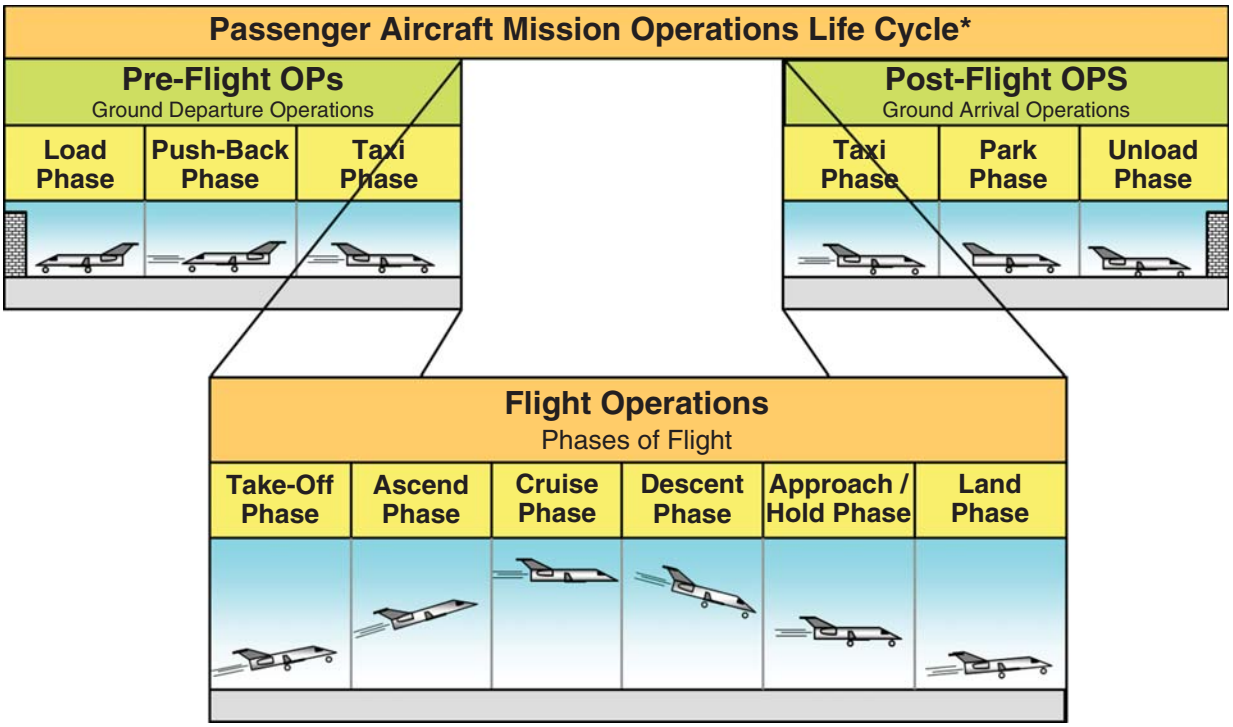
7.3.2 Phase-Based Mission System and Enabling System Operations and Tasks

Using Figure 7.3 as a reference, we can identify specific Stakeholders—Users and End Users, their UCs, and scenarios for each of the Pre-Flight, Flight, and Post-Flight Operations. I turn, each UC and its primary and alternate flows driven by various scenarios provide the basis for identifying operational tasks to be accomplished. Each task requires specific capabilities, which can then be translated into System Performance Specification (SPS) requirements.

Recognize that the preceding paragraph represents UCs and operational tasks organized by Mission Life Cycle Phase. To add meaning to the set, we need to use them to construct a System Operations Model such as the one shown in Figures 6.1 and 6.2. The context of here is the SOI comprised a MISSION SYSTEM(S) and ENABLING SYSTEM(S). Since each operational task may represent MISSION SYSTEM(S)—ENABLING SYSTEM(S) interactions, the description of each task would identify and describe those interactions. Figure 7.4 serves as an example.

7.4 INTRODUCTION TO SYSTEM MODES AND STATES

System modes and states are perhaps one of the most controversial topics in Engineering and SE. Every industry, profession, Enterprise, and Engineer has their own view as to what a *mode* and a *state* are. Some Enterprises exhibit an in-grained *state machine* paradigm and ignore modes. In their minds, *modes* are just another name for a state.



* Due to space restrictions, the Startup, Hold, and Maintenance Phases are not shown.

Figure 7.3 Example—Aircraft Mission Life Cycle Phases with Embedded Phases of Flight

- In general, states are *observable* and *measurable* physical attributes of a SYSTEM or ENTITY. For example, if an automobile dealer wants to know the *state* of the dealership's vehicle inventory, it is a simple counting, categorization, and summation exercise.
- *Modes*, however, are *abstract* labels applied by System Developers to User *selectable options* required to C2 an Enterprise or Engineered System. In the case of a computer word processor application, its modes might be: Create, Edit, Save, Open, and Print a document. Modes as abstract labels are not visible or measurable; they are a state of mind. You might argue that you can observe them being used but those are actually steps performed by the User. However, *modes* of operation enable you to accomplish objectives that produce results you can *observe* and *measure*. For example, you can Open, Create, Edit, Save, and Print a file.

Wasson (2014, p. 5) observes that if you research definitions of *modes* and *states*, two key points emerge:

- Dictionaries, standards, and authors use the terms to define the other—"circular" references. "Modes" are used to define "states" and vice versa.
- "Condition" is a common term to most *modes* and *states* definitions.

In general, the term *condition* is appropriate for defining *states*; however, *modes* have a different context of usage and application. For example, *modes* of transportation—land, sea, air, and space; Automatic/Manual Mode; Edit Mode; Print Mode; and so forth.

Modes represent User *selectable options* that enable accomplishment of performance-based outcomes and objectives. In contrast, *states* represent physical *conditions*, which characterize a "state of being or health." In general, *states* represent the Operational Health and Status (OH&S)—physical operating *condition*—of a system, product, or service.

How do modes and states influence Systems Engineering and Development (SE&D)?

When you listen to the discussions at Engineering meetings, Engineers and managers *intermix* usage of modes and states. Terms such as On/Off, Loading/Unloading, Open/Closed, Automatic/Manual Mode, Taking Off, Launching, Operational/Non-Operational, Enroute, and In-Maintenance flow around the room. Participants nod their heads in agreement. Then, discover after the meeting that everyone had a different perspective as to what was being communicated. *Is there any wonder why there is so much confusion among Engineers*?

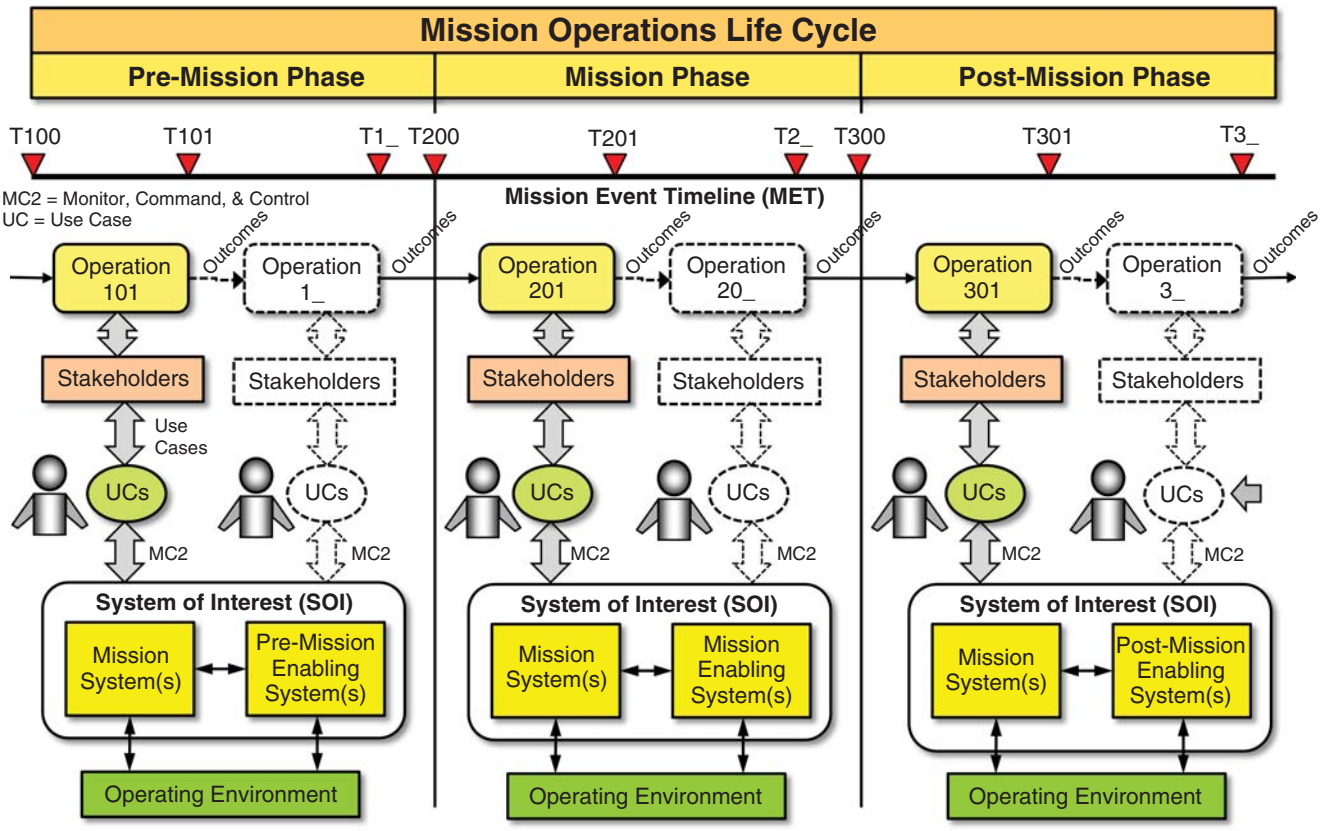


Figure 7.4 Mission Phases of Operation Concept illustrating Linkages between Operations, Stakeholders and Use Cases, and MISSION SYSTEM and ENABLING SYSTEM Interactions

Professional organizations and standards often treat *modes* and *states* as abstractions as if "everyone clearly understands their meaning—why doesn't the listener." The reality is: Engineers receive little or no education on this topic (Chapter 2). Where Engineering education and standards are lacking, *chaos* and *confusion* prevail. Since Enterprise management is typically unaware of the problem other than recognizing differing opinions within a project, they fail to exercise technical leadership and correct the problem.

If you collect these terms and analyze them, you soon discover that some are, by definition, *modes* and some are *states*. However, you soon discover that those identified as *states* are *contextual* and require further analysis. That analysis leads to further delineation that a system, product, or service has four types of *states*: System States, Operational States, Configuration States, and Dynamic States. Analysis of these four states reveals two types of usage: (1) an Enterprise perspective and (2) an Engineering perspective.

- From the Enterprise perspective, a system, product, or service has already been acquired. Their focus concerns the *employment* of a system, product, or service as an "organizational asset" to perform missions. In general, they have a need to know: (1) if the system is in use—System States—and (2) what is its current operating condition—Operational State—of OH&S.
- From an Engineering perspective, Users need to know how to (1) C2 the system—Modes and UCs, (2) configure its architecture—Configuration States, and (3) accommodate time and location-dependent perturbations—Dynamic States—resulting from interactions with its OPERATING ENVIRONMENT.

Dynamic States are induced by a *fifth* type of state—Environmental States, which characterize the external OPERATING ENVIRONMENT.

As a final point, Engineered systems, products, and services are *objects* developed by humans. As *inanimate* objects, systems, products, or services are *clueless* as to their *mode* or *state* other than "informed knowledge" programmed by their System Developer. As stimulus-behavioral response devices, they simply control the flow of electrons; absorb, direct, transfer mechanical forces; consume and convert energy; and communicate via optical paths. The

fact that humans refer to a *mode* is meaningless to these devices.

Modes are like a *virtual* switching networks that enable a User to C2 the system's architectural configuration and capabilities. Developers apply modal names to human interfaces such as dashboards, control panels, and procedure manuals … for their human Users, not the system. The system components are simply configured for a specific mode of operation to accomplish what is depicted in Figures 3.1 and 3.2.

In summary, the solution is not as simple as *modes* versus *states*; Engineers must consider not only *modes* of operation but also five types of *states* of operation. Given this foundation, let's begin with a discussion of the Enterprise perspective of System States. Then, introduce Modes of operation, which enable us to address the Engineering perspective of the Engineered System Operational, Configuration, Environmental, and Dynamic States.

7.5 ENTERPRISE PERSPECTIVE—ENGINEERED SYSTEM STATES

From an Enterprise perspective, Users need to answer two key questions:

1. *What is the state of deployment or employment - System State—of a system, product, or service as an organizational asset?*
2. *What is its current OH&S—Operational State?*

To answer these questions, let's begin with System States.

7.5.1 System States

Principle 7.2

System States Principle

Every Enterprise and Engineered system, product, or service is characterized by *system states* that represent its current logistical employment as an asset.

A *system state* is an attribute that represents the current *logistical* employment, status of an Enterprise asset such as a system, product, or service. The intent is to establish an understanding based on a line of questions:

- Has the system been fielded? Yes or no?
- Is it ready to perform missions? Yes or No? If not, why not?
- Is the SYSTEM undergoing maintenance? Yes or No? What is required to return it to active duty service? When will the SYSTEM return to service?

If we employ Domain Analysis to analyze Mission Life Cycle phase-based operations, we can identify System States that represent the "state of logistical employment of an organizational asset." Simply based on the System/Product Life Cycle Phases, System States are Deployment, Operations, Maintenance, Sustainment, Retirement, or Disposal. Sustainment, for example, supports the Operations System State and the Maintenance System State with *consumables* such as food, fuel, and lubricants and *expendables* such as replacement filters.

Within each of the System States, we further refine them into key Use Cases (UCs). Tables 6.3 to 6.7 list UCs that provide examples of System State Use Cases (UCs).

Example 7.1

System States Example

- Consumer products, in general, require TRANSPORT from their manufacturer to discounters and retailers for distribution to consumers. By virtue of their "out-of-the-box" capability, they typically do not require SETUP. In contrast, computers and TVs require both TRANSPORT and SETUP, either by the consumer or a service technician.
- Large, complex systems such as cranes, carnival amusement rides, and manufacturing machinery being relocated to new sites require TEARDOWN, TRANSPORT, and SETUP UCs.

If we depict System States graphically, Figure 7.5 emerges from Figure 6.4. Observe that System States are depicted as rectangular boxes with rounded corners. Some boxes include a 270° arc with an arrowhead. The symbology indicates that a system, product, or service remains in a *current state* until an external trigger initiates a transition to the *next state*.

Let's explore Figure 7.5 and its implementation.

7.5.1.1 Deployment Phase: System States Let's assume the System Deployment Phase consists of Disassemble, Transport, and Setup UCs. Several options are available to accommodate transition directly from a System Developer's or manufacturer's facility to a User's designated site:

- For Set-Up and transition to the Operations System State.

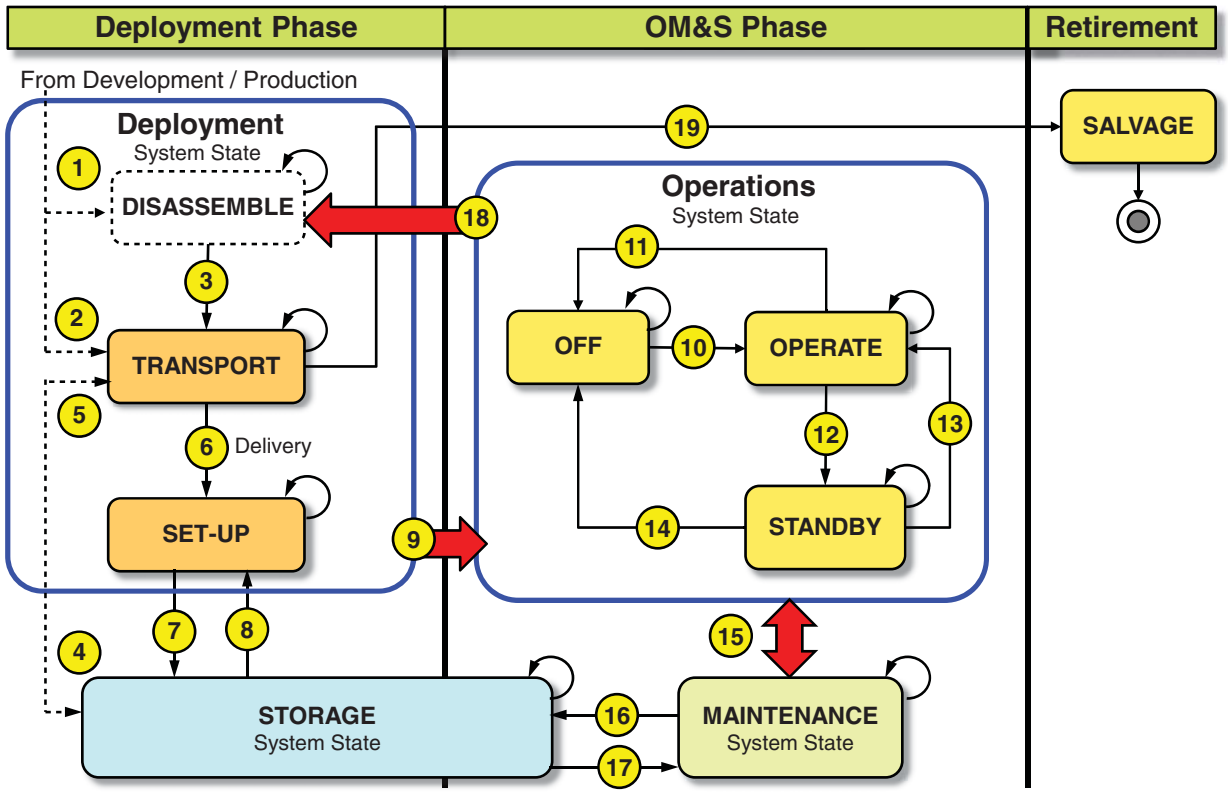


Figure 7.5 System States of Operation and Transitions as a Function System/Product of Life Cycle Phases

- For transition from the Storage System State to Set-Up.
- For transition from Set-Up to the Operations System State.

7.5.1.2 OM&S Phase: System States On completion of the Deployment Phase SETUP UC, a SYSTEM or PRODUCT transitions to its OM&S Phase. For example, assume the OM&S Phase consists of the OPERATIONS AND MAINTENANCE System States. The SYSTEM may be Parked or Off-Line and being configured for its mission. The Operations System State represents the reason for the SYSTEM's existence. At this point, the SYSTEM could transition to STANDBY to reduce energy consumption if it lacks a purposeful mission.

Office Copier System States Example

Example 7.2 An office copier, for example, transitions to STANDBY after a period of inactivity to conserve energy during normal business hours, after business hours, and during weekends. To resume normal operation, a User presses a button to transition the copier from STANDBY to OPERATE.

If a SYSTEM requires maintenance while in OPERATE or STANDBY ENABLING SYSTEM technicians transition it to the MAINTENANCE System State for repairs and upgrades. From the MAINTENANCE System State, there are several *next state* options are available:

- Option 1—Place the SYSTEM in OFF (16) and leave in place.
- Option 2—Transition (17) the SYSTEM to the STORAGE System State.
- Option 3—Transition the SYSTEM to STANDBY.
- Option 4—Transition the SYSTEM to OPERATE.

When the SYSTEM has completed its missions and has been tasked to relocate to another site, it transitions to the Deployment Phase TEARDOWN, TRANSPORT, and SETUP as applicable. If the system is no longer needed to conduct missions, it may be transitioned to TRANSPORT for relocation and transition to the SALVAGE State.

7.5.1.3 Retirement/Disposal: System State When the SYSTEM is ready for retirement/disposal, the SYSTEM or PRODUCT: (1) may be rented, leased, or sold; (2) enter a STORAGE State placed in a warehouse; (3) or be designated in SALVAGE for recycling of components, removal of toxic or hazardous materials, and destruction.

Heading 7.1

Our discussion of System States addressed the Enterprise perspective of having a "need to know" the deployment and employment status of a system, product, or service. Once that is determined, the next question is what is its OH&S? This brings us to our next topic, Operational states.

7.5.2 Operational States

Principle 7.3

Operational States Principle

Every Enterprise or Engineered system, product, or service is characterized by *operational states* that represent its current *condition, status, readiness, or availability* to perform or continue a mission or task.

Since an Enterprise employs systems, product, or services as organizational assets, they must be maintained and sustained to ensure they are operationally *ready* and *available to* conduct a new mission or continue their current missions. This requires understanding the SYSTEM or PRODUCT'S OH&S.

Enterprises and Users often have a "need to know and track" the *operational status*, *condition*, or *readiness* of an SOI, MISSION SYSTEM, ENABLING SYSTEM to perform or continue a mission. This means that the integrated PERSONNEL-EQUIPMENT SYSTEM (Figure 10.16) is:

- *Physically configured*—A system's architecture and interfaces have been configured to deliver *essential* mission capabilities that are considered *necessary* and *sufficient* to conduct specific types of mission(s).
- *Operationally ready*—The SYSTEM is in *satisfactory* operating condition with no Critical Operational or Technical Issues (COIs/CTIs) issues presently and meets the *minimum sufficiency* requirements to *safely* and *reliably* accomplish and complete the mission and its objectives.

From an Enterprise project perspective in which organizational assets are employed to perform missions, any revenue-generating asset that is not *operational* is consuming—wasting—valuable resources. For example, if an airline has aircraft that are unable to fly, revenue is not being generated. If passenger tickets have been sold for a specific flight and the aircraft is unavailable to leave on time, the airline has to scramble to either perform *corrective maintenance* actions (Chapter 34) while passengers wait or dispatch a replacement aircraft.

The airline's resource manager has to constantly juggle aircraft assets to ensure that operations remain seamless for the passengers. Every one of a passenger's interactions with the airline is a "moment of truth" (Principle 4.9) with *positive* or *negative* consequences. Therefore, to ensure *continuity* of operations, the resource manager has a "need to know" at all times what the OH&S—Operational State—is of any tail number aircraft in their inventory.

Based on the discussion, Users "need to know" what the OH&S of a system, product, or service is. Once the OH&S status is assessed, there is a follow-on need for a risk-based Situational Assessment (Chapter 24). For example, if a SYSTEM's OH&S indicates a COI/CTI, a Situational Assessment needs to inform the User concerning: (1) what the condition's level of significance is in terms of *threatening* the mission, User, EQUIPMENT, or the public and (2) if the condition—fault—is *contained* (Figure 26.8), *correctable*, and/or *recoverable* (Figure 10.17).

So, for a specific system, product, or service, *how do SEs assign labels to Operational States*? SEs should understand: (1) *what* the Users "need to know" about any system, product, or service in their inventory and (2) *how* they intend to track current status. Since every User and SYSTEM is different, a Domain Analysis should be performed to determine the correct set of terms. For example, terms such as operating, processing, navigating, and infusing.

Operational States, like System States, can be further refined into four contexts: *mission readiness and availability,* operating condition, and operational status. Consider the following examples:

- *Mission Readiness*—Examples include Configured, Loaded, Staffed Ready.
- *Mission Availability*—Examples include Awaiting Orders or Clearance
- *Operating Condition*—Examples include Upgraded, Degraded Ops, Loaded, Failed, Calibrated, Replenished, Refurbished, or Aligned.
- *Operational Status*—Awaiting Maintenance, On-Hold, Launching, Landing, Transmitting, Receiving, Processing, Storing, Converting, or Reporting,

One of its noteworthy characteristics of an operational state's *status* is its grammatical usage as a *present participle* with an "ing" suffix. For example, if a medical infusion device is delivering meds to a patient, its *operational status* is "infusing." Consider the following example.

To better understand the identification and development of Operational States, an office copier provides an example. Operational States concerning its *current operation* include OFF, WARMING UP, IDLING, COPYING, PRINTING, FAXING, SCANNING, COLLATING, STANDBY, RESETTING, POWERING DOWN, OR OFF.

Author's Note 7.1

One of the challenges of creating System State names is that PDT members assign their own "favorite" names without appropriate consideration of the context. Inevitably, they assign *mode* titles to *states* and vice versa.

For example, PDT members may try to introduce Copier Setting features such as paper size, double-sided copying, and black and white versus color copying, which are actually physical Configuration State attributes, not System States. Additionally, team members will try to apply terms such as warming up, copying, or collating, which are actually OPERATIONAL States that will be introduced later.

7.5.3 Enterprise Perspective Summary

In summary, the preceding discussion addressed the two key questions Enterprises need to answer concerning the deployment or employment of a system, product, or service as an organizational asset. We now shift our focus to understanding the Engineering perspective of *how* modes and states are applied to System Development to enable Users to C2 system capabilities and performance to accomplish Enterprise mission objectives.

7.6 ENGINEERING PERSPECTIVE—MODES AND STATES

7.6.1 Modes of Operation

Principle 7.4

Modes of Operation Principle

Every Enterprise or Engineered system, product, or service is characterized by User-selectable modes of operation that enable the User to Command & Control (C2) unique sets of UC-based architectural capabilities to accomplish a specific mission outcome and its objectives.

When a MISSION SYSTEM or ENABLING SYSTEM User operates or maintains a SYSTEM or PRODUCT, PERSONNEL-EQUIPMENT interactions occur as illustrated earlier in Figure 7.4 and later in Figures 10.15 and 10.16. Their tasking requires that the User perform MC2 of the EQUIPMENT Element, which responds and performs MC2 of its own components and performance (Figure 24.12). Modes of Operation provide a set of User *selectable* options based on UCs that allow a User—Operator or Maintainer—to safely MC2 SYSTEM or PRODUCT performance-based outcomes. Observe the phrase "User selectable options." This is a key attribute of *modes* of operation that delineate *modes* from *states*.

One of the simplest illustrations of modes of operation is an automobile. Modes enable the User to MC2 the automobile's performance to achieve performance-based objectives such as drive forward, drive backward, and idle. Physically, the automobile's transmission gear-shift capabilities such as Park, Neutral, Reverse, Drive, Low1, or Low2 represent User *selectable* modes that enable the User to accomplish the forward, reverse, or idle objectives.

Also observe that the mode transitions occur in a bi-directional serial sequence—PARK ⟷ NEUTRAL ⟷ REVERSE ⟷ DRIVE ⟷ LOW2 ⟷ LOW1—for safety as well as adjacency of related capabilities. This last point illustrates the need to seamlessly transition between a *current mode* and the *next mode*. In other words, these are not just shift from PARK ⟷ NEUTRAL ⟷ REVERSE ⟷ DRIVE ⟷ LOW2 ⟷ LOW1 transitions. These sequences are "interfaces" that require *control flow* operations and interactions that enable seamless transitions from a *current mode* to the *next mode*.

This leads to a key question: *How does a system, product, or service to transition from one mode to another during a mission*? This brings us to a discussion of Understanding Modal Transitions.

7.6.1.1 Understanding Modal Transitions

Modes of operation can be illustrated with a table such as Table 7.1 or graphically. The graphical approach is the preferred approach. Let's begin with a basic construct for depicting modes as shown in Figure 7.6. When creating Mode Diagrams, there are some basic rules and conventions to observe:

- **Rule 1**: Each mode configures the SYSTEM or PRODUCT's architectural configuration or interconnection of its components to enable the User to accomplish a specified outcome. For example, an automobile's Drive Mode enables the driver to perform the Drive–Forward UC.
- **Rule 2**: SYSTEMS are designed to remain in their *current mode* until a triggering event forces SYSTEM C2 operations to transition to the *next mode*. Graphically, we depict modes as an oval or circle icon with a 270° arc consisting of an arrow at one end around one part of the oval or circle.
- **Rule 3**: Modal Transitions Convention—The transition path from the *current mode* to the *next mode* is illustrated via a straight or curved arrow with the arrowhead touching the *next mode*.

We illustrate *modes* and their *control flow* (Figure 10.9) from one mode to another. Figure 7.6 illustrates a simple, two-mode system that transitions back and forth from Mode 1 to Mode 2. When Triggering Event 1 occurs, *control flow*

TABLE 7.1 State Transition Table for an Office Copier

Current State	Event	Transition Trigger	Next State
OFF	T10	Power-up command	OPERATE
OFF	T11	Redeploy system	TEAR DOWN
OPERATE	T12	Power-down command	OFF
OPERATE	T15	Maintenance required	MAINTENANCE
OPERATE	T13	Standby command	STANDBY
STANDBY	T14	Reactivate command	OPERATE
STANDBY	T20	Power-down command	OFF
STANDBY	T17	Perform Maintenance	MAINTENANCE
MAINTENANCE	T16	Standby command	STANDBY
MAINTENANCE	T18	Transfer to Storage	STORAGE
MAINTENANCE	T20	Power-down command	OPERATE
STORAGE	T19	Transfer from Storage	MAINTENANCE

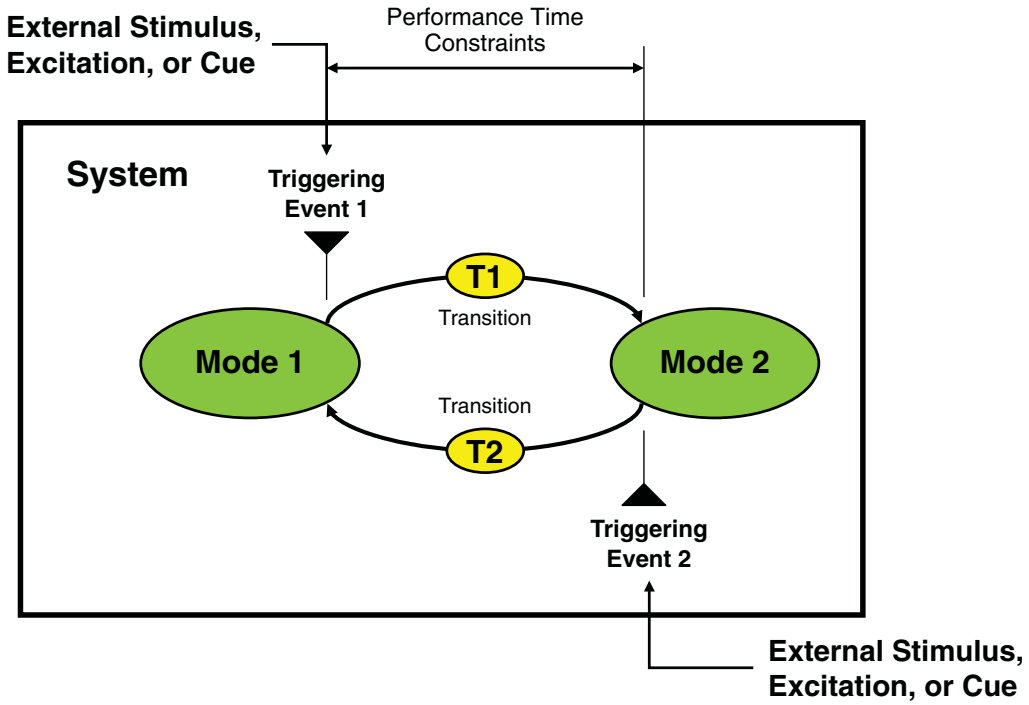


Figure 7.6 Modal Transition Loop Construct

transitions from Mode 1 to Mode 2. The SYSTEM idles in Mode 2 until Triggering Event 2 occurs, thereby initiating a control flow transition back to Mode 1. When making the transition from one mode to another, the User may impose specific time requirements and constraints on the transition.

Modal transitions represent interfaces in which *control flow* transitions from a *current mode* to the *next mode*. Although modes characterize SYSTEM operations, there may be instances whereby different PRODUCTS or SUBSYSTEMS drive the mode's primary capabilities.

Author's Note 7.2

If separate System or Product Development Teams (SDTs/PDTs) are developing products for use in a specific mode, make sure that both interfacing teams operate with the same set of assumptions, decisions, and transition criteria. Otherwise, *incompatibilities* will be created that will not emerge until SITE. If *latent defects*—design errors, flaws, and deficiencies—are left *undiscovered* and *untested*, a potential hazard can lead to SYSTEM failures (Figure 24.1), sometimes catastrophically.

The challenge for SEs is to ensure *compatibility* and *interoperability* (Chapter 27) between any two modes such that initializations and conditions established for the *current mode* are in place for the *next mode* processing. The intent is to ensure that modal transitions are seamless. *How do you ensure consistency*? Document the modes of operation, Mission Event Timeline (MET), and modal transitions in the ConOps document.

Author's Note 7.3

Author's Note

Our introduction of UCs in Chapter 5 noted several attributes including Pre-Conditions. When a system, product, or service shift receives an external *triggering event* to transition from the *current mode* to the *next mode*, the *current mode's* Exit Criteria serve as Pre-Conditions for the *next mode* and its UCs.

To better understand Triggering Event-Based Transitions, let's explore the topic further.

7.6.1.2 Triggering Event-Based Transitions

Principle 7.5

Triggering Event Criteria Principle

Every Enterprise or Engineered System mode of operation requires a pre-defined set of condition-based triggering event criteria to initiate a transition from one mode of operation to another mode.

Triggering events such as stimuli, excitations, or cues originate external to an SOI. Each triggering event should be characterized by a set of pre-defined criteria or condition that initiate the transition. In the case of a MISSION SYSTEM's or ENABLING SYSTEM's internal PERSONNEL-EQUIPMENT interactions, *triggering events* might originate from input devices such as keyboards, mice, touchscreen displays, switches, and buttons. External stimuli, excitations, or cues from external Systems such as interrupts, data messages, forces, energy application, optical movement, flashing lights, or hand signals.

As state machines, most SYSTEMS are designed to cycle in an infinite—Do Until—loop until an external stimulus, excitation, or cue initiates a triggering action to transition from a *current mode* to the *next mode*. The *occurrence* of the external trigger is referred to as a *triggering event*. For example, data-driven triggering events may be *synchronous*—periodic—or *asynchronous*—random—occurrences.

Author's Note 7.4

Importance of a Model Transition Convention

Enterprise Engineering should establish a modal transition convention. *Do you establish Entry criteria for the next mode or Exit criteria for the current mode*? Typically, by convention, a mode does not have *both* Entry and Exit criteria. The transition from the *current mode*, n, to the *next mode*, $n + 1$, is a single transition. You do not specify the Exit criteria for the *current mode* and then duplicate the same as Entry criteria for the *next mode* of the modal interface. Best practices suggest definition of Exit criteria for the *current mode* of operation.

7.6.1.3 Types of Modes of Operation

Modes of operations occur in at least two forms: (1) Outcome-Based Modes and (2) Mission Profile-Based Modes.

7.6.1.3.1 Outcome-Based Modes of Operation Engineered systems, products, or services, especially consumer goods, have modes of operation that produce outcomes based on the allowable UCs. For example, an office copier might consist of COPY, FAX, SCAN, STANDBY, RESET, AND POWER DOWN modes of operation. This allows the User at their own discretion to decide which mode of operation satisfies that User's requirements. The same is true for consumer products such as smartphones or computers.

7.6.1.3.2 Mission Phase-Based Modes Approach *Mission phase-based modes* are based on mission operations that must sequence through Mission Life Cycle Phases—Pre-Mission, Mission, and Post-Mission—to terminate. The difference in the Mission Phase-Based Approach is that Mission Life Cycle phase is started, safety considerations require completion of the mission or abortion of the mission. For example, aircraft, rocket, or missile launch in-flight preclude random switching out of a Mission Life Cycle phase of Operation. An exception might be a range safety self-destruct command to a wayward missile.

Example 7.3

Mission-Based Modes of Operation Example

Aviation partitions flight operations into Phases of Flight such as TAKEOFF, CLIMB, CRUISE, DESCEND, APPROACH, HOLD, and LAND. Despite being referred to as Phases of Flight, each represents a *mode* of operation that has unique performance-based outcome objectives.

7.6.1.4 Sub-Modes of Operation

Some modes of operation require further refinement into *sub-modes* based on the User's mission outcomes, objectives, and application of the

system, product, or service. Consider the example provided below.

Mission-Based Sub-Modes of Operation Example

Example 7.4 Let's assume we designate an automobile's modes of operation as PARK, NEUTRAL, REVERSE, and FORWARD. Note the FORWARD Mode. Analysis of typical OPERATING ENVIRONMENT driving conditions indicates the need to partition Forward operations into three Sub-Modes: (1) FORWARD–DRIVE; (2) FORWARD–LOW 1; and (3) FORWARD–LOW 2. Recall our discussion in Chapter 5 concerning UCs and UC extensions. Observe that each Sub-Mode is preceded by "Forward" to indicate traceability to the UC objective; DRIVE, LOW 1 and LOW 2 represent UC extensions.

Most automobile drivers are not Engineers and are not interested in lengthy Engineering terminology such as FORWARD–DRIVE. As a result, we simplify the User's mode options with simple terms and elevate the DRIVE, LOW 1, AND LOW 2 Sub-Modes to the mode level resulting in PARK, NEUTRAL, REVERSE, DRIVE, LOW 1, and LOW 2.

The preceding discussions provide an overview into what modes of operation are. The question is: *How do we identify modes of operation*? This brings us to our next topic, Approaches to Deriving Modes of Operation.

7.6.1.5 Approaches to Deriving Modes of Operation

There are two basic approaches Systems Engineers (SEs) and Analysts can employ for identifying modes of operation: (1) Abstracted Modes Approach and (2) Generalize Modes Construct Approach. Let's discuss each of these approaches.

7.6.1.5.1 Abstracted Use Cases-Based Modes Approach

Abstracted UC Modes Principle

Principle 7.6 Each Enterprise and Engineered System mode of operation represents a User selectable option to perform one or more UCs that share similar outcomes and objectives.

If we analyze the collection of Stakeholder UCs for all System/Product Life Cycle Phases and Sub-Phases, *similarities* and common *occurrences* emerge among the data sets. Using Affinity Analysis methods, we can mine and analyze the UC outcomes to group or cluster sets of UCs.

Further analysis reveals that the common link among these data sets is they represent refinements of a higher level abstraction that represents a mode of operation. You may discover that some modes, Sub-Modes, for example, can be further *abstracted* into higher level modes. Where this is the case, we establish a modal hierarchy and designate sub-modes within each mode of operation.

You may ask: *Are there guidelines for identifying modes of operation*? The answer is no, which contributes to the confusion and *ad hoc* implementation. Independent PDTs of equally capable SEs, analysts, and developers can hypothetically develop the same SYSTEM or PRODUCT to comply with a set of User capability and performance requirements, yet have variations in the naming of the system modes of operations. In both cases, the SYSTEM or PRODUCT will comply with the specification requirements and perform as expected by the User.

When we analyze UCs that have been aligned with specific Mission Life Cycle phases of operation—Pre-Mission, Mission, and Post-Mission, we soon discover that some UCs share or support a common objective or outcome. Where there is *sufficient* commonality in these sets or clusters of UCs, we abstract them into higher level modes of operation. This requires a two-step process.

Step 1: Align UCs by System/Product Life Cycle Phases

Based on the UCs collected from the System Operations Model Construct (Figures 6.1 and 6.2) and Tables 6.2–6.7, we align the UCs based on their usage during the Mission Life Cycle Pre-Mission, Mission, and Post-Mission phases of operation as shown in Table 7.2.

Step 2: Abstract UCs into Modes

Once we have aligned UCs with Mission Phases, we employ Affinity Analysis to abstract sets of UCs that share a common objective into higher level Modes. Figure 7.7 shows the results of the abstraction based on the UCs identified in Table 7.2.

To illustrate how modes might be abstracted, consider the following example.

UC-Based Modal Abstraction Example

Example 7.5 Assume that UC #1 represents an objective to power-up a SYSTEM. UC #2 represents an objective to initialize the SYSTEM and perform a diagnostic self-test prior to idling in a Desktop Mode to await User commands. Due to the commonality of UC #1 and UC #2 objectives derived from Affinity Analysis, we abstract the UCs into a higher level START-UP Mode, a User-selectable option, that has an overall objective of ensuring that the SYSTEM is operationally available and ready-to-commence User operations.

As a result, each abstracted mode is symbolized by an oval containing its related UCs. Some modes may have only one UC; others accommodate two or more UCs.

TABLE 7.2 Alignment of UCs to Mission Phases of Operation

Operational UC	UC Description	Pre-Mission Phase	Mission Phase	Post-Mission Phase
UC #1	XXXXXXXXXX	X		
UC #2	XXXXXXXXXX	X		
UC #3	XXXXXXXXXX	X		
UC #4	XXXXXXXXXX		X	
UC #5	XXXXXXXXXX		X	
UC #6	XXXXXXXXXX		X	
UC #7	XXXXXXXXXX		X	
UC #8	XXXXXXXXXX		X	
UC #9	XXXXXXXXXX		X	
UC #10	XXXXXXXXXX			X
UC #11	XXXXXXXXXX			X
UC #12	XXXXXXXXXX		X	
UC #13	XXXXXXXXXX		X	
UC #14	XXXXXXXXXX			X

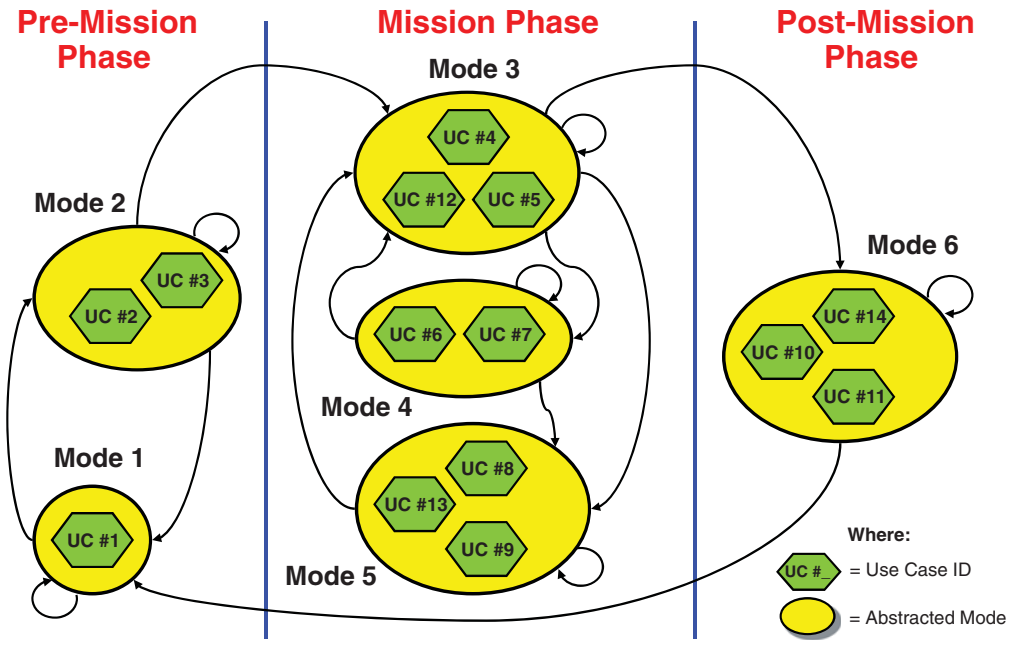


Figure 7.7 Illustration Depicting Abstraction of Use Cases into Higher Level Modes of Operation

Author's Note 7.5

Two key points:

1. In Figure 7.7, the hexagonal icons representing UCs embedded within each mode are unique to this illustration for instructional purposes. There are no hard rules requiring hexagons or other shapes. If it enhances communications and understanding of a mode, use the hexagonal symbols or simply include a bulleted list of applicable UCs.
2. In some cases, there are no specific guidelines for identifying modes of operation. Independent teams of equally capable SE analysts and developers can then hypothetically design and produce a SYSTEM OR PRODUCT that complies with a set of User capability and performance requirements. Each team nevertheless may have variations of SYSTEM modes of operation. The point is to learn to recognize, understand, and establish a team-based consensus concerning SYSTEM phases and modes of operation. Then, apply common sense to abstracting UCs into modes of operation.

Observe in Figure 7.7 the curved lines connecting Modes. These represent triggering events that initiate modal transitions from the *current mode* to the *next mode*.

The preceding discussion described how Modes can be abstracted from UCs. There is an easier method that serves

as an initial *starting point* for identification of modes. It is the Generalized Modes of Operation Construct, our next topic.

7.6.1.5.2 Generalized Modes of Operation Construct Approach Theoretically, we can spend a lot of time analyzing and abstracting UCs into modes of operation. Once you develop several types of systems, you soon discover that SYSTEM modes are similar across Engineered systems and products. You begin to see common patterns of modes emerge. This leads to the question: *What are these generalized modes*?

If we perform a Domain Analysis of potential modes for any type of system, Table 7.3 summarizes the results.

Further analysis of the modes identified in Table 7.3 reveals that these modes are more than simply discrete modes. They have sequential interdependency relationships as shown in Figure 7.8. *Is this construct applicable to every system as a starting point?* Generally so, especially Engineered Systems. Recognize that every system, product, or service is different and has its own nuances based on system application and customer preferences. This is simply a *starting point*, not an end result.

The construct is divided into Pre-Mission, Mission, and Post-Mission Phases of Operation to facilitate a general left-to-right *mission flow*. Other than a generalized left-to-right cyclical Pre-Mission to Mission to Post-Mission modes of operation may or may not be *time dependent*.

TABLE 7.3 Generalized Phases and Modes of Operation Construct

Phase of Operation	Example Mode of Operation
PRE-MISSION phase	START-UP mode CONFIGURE mode CALIBRATE/ALIGN mode TRAINING mode POWER-DOWN mode
MISSION phase	NORMAL Operations mode DEGRADED Operations mode
POST-MISSION phase	SAFE mode ANALYSIS mode REPORT mode MAINTENANCE mode
STORAGE phase	Optional—system dependent

MISSION Systems such as a commercial airline and military sorties establish METs that constrain (1) the Pre-Mission-to-Mission transition, (2) Mission-to-Post-Mission transition, and (3) Post-Mission back to Pre-Mission transition during the system turnaround such as a commercial aircraft. Within each mode of operation, the MET event constraints may be further subdivided.

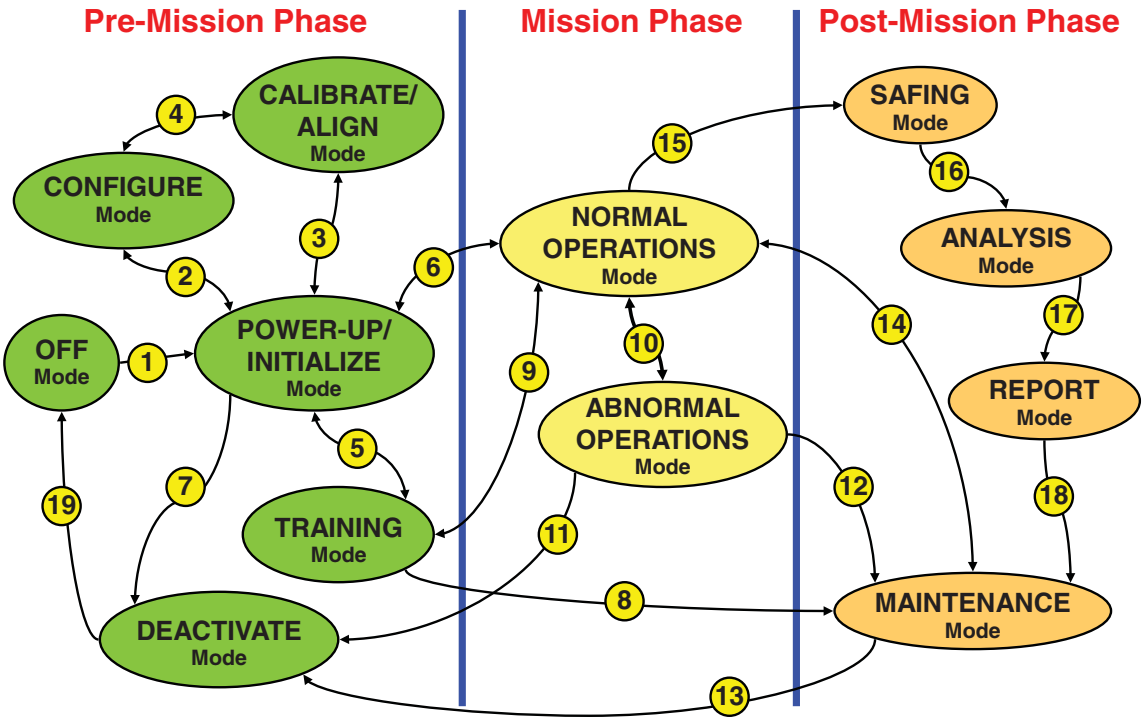


Figure 7.8 Generalized Modes of Operation Construct for Use as a Starting Point

7.6.1.5.3 Generalized Modes of Operation for Large, Complex Systems The preceding discussions addressed a simple product example as a means of introducing and illustrating system modes of operation. When you investigate the modes of operation for large, *complex*, *multi-purpose*, *reusable systems*, additional factors must be considered. For example, large, complex systems may require Reconfiguration, Recalibration, Realignment, Refurbishment, or Replenishment of *expendables* and *consumables* (Chapter 8).

7.6.1.5.4 Final Thoughts About Modal Transitions In our discussion of SYSTEM operations and applications in CHAPTER 6, we highlighted various types of system applications—*single use, reusable, and recyclable*. Most *reusable* systems such as an aircraft, automobiles, and smartphones are characterized by *cyclical mission operations*. Cyclical operations systems have a feedback loop that typically returns *workflow* or *control flow* (Figures 6.1 and 6.2) back to the Pre-Mission Phase of Operation, either powered down or refurbished and replenished for the next mission.

Now, consider a system such as the former Space Shuttle's External Tank (ET). From a mission perspective, the ET's fuel resource is a *consumable* item, and the ET is an *expendable* item. At a specific phase of flight and MET event, the ET is jettisoned from the Orbiter Vehicle (OV), tumbles back toward Earth, and burns up on reentry into the atmosphere.

In the case of an *expendable system* such as the ET, one might expect its modes of operation to be sequential without any loop backs to previous modes. However, from an SE design perspective, ET operations may require cycling back to an initial mode due to UC scenarios such as "scrubbed" launches. Therefore, *expendable, single-use* systems also require modes of operation that finally transition to a Termination mode—such as Reentry—to ET Impact.

7.6.1.6 Diagramming Modes of Operation The preceding discussion depicts modal transitions by segmenting Figure 7.8 into Mission Life Cycle Pre-Mission, Mission, and Post-Mission Phases of operation. While this method—ovals and transitions - communicates content, modal diagrams can be communicated in other ways.

Although the Space Shuttle Program ended in 2011, the program serves as an excellent example for SE. Figure 7.9 illustrates Space Shuttle Launch Modes (NSTS, 1988a). These include the:

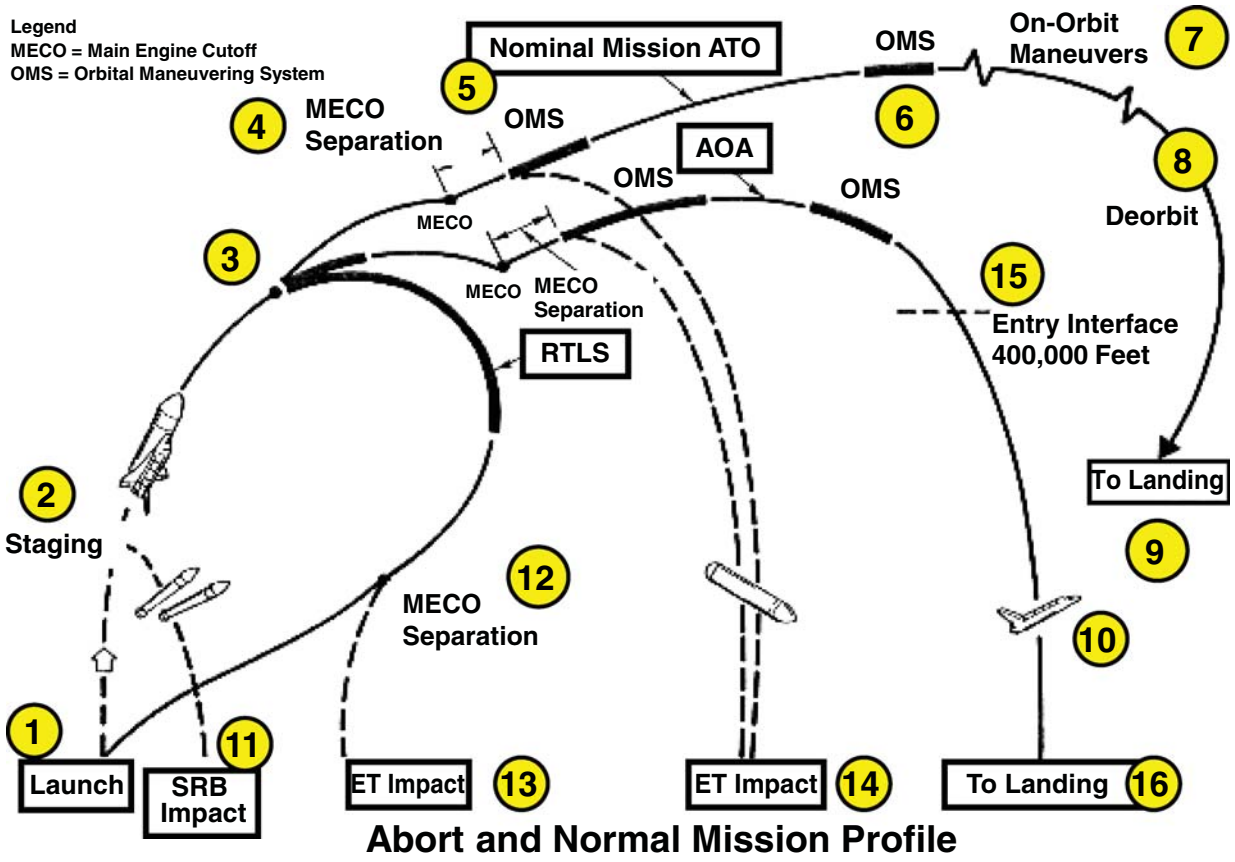


Figure 7.9 NASA Space Shuttle Flight Operations Modes (Source: NSTS (1988b))

- Redundant Set Launch Sequencer (RSLS) Abort
- Return to Launch Site (RTLS)
- Abort Once Around (AOA)
- Abort to Orbit (ATO)
- Transoceanic Abort Landing (ATL)

Figure 7.4 highlighted the importance of defining MISSION SYSTEM and ENABLING SYSTEM interactions that occur for each of the operational tasks in Figures 6.1 and 6.2. Figure 7.10 provides an illustrative example of the former NASA Space Shuttle's Post-Mission Phase SAFING Mode (Figure 7.8). In the case of a ConOps Document, the Maintenance Concept, for example, would describe the interactions between the Space Shuttle MISSION SYSTEM and the ENABLING SYSTEMS required to safely deplane the astronauts after removal of toxic gases at the end of a mission.

Heading 7.2

Now that we have established the an understanding of modes and operational states, the question is: *What do we apply them to C2 of a system, product, or service?* This brings us to our next topic, Configuration States.

7.6.2 Configuration States

Principle 7.7

Configuration States Principle

Each Enterprise or Engineered system, product, or service mode of operation Commands and Controls (C2) the Configuration State(s) of its architectural configurations.

A *configuration state* represents a unique physical arrangement—configuration or connectivity—of a system, product, or service's architectural components and external interfaces. For a given mode of operation, the intent is to configure sets of *essential* capabilities required to enable achievement of one or more UC-based performance objectives. Consider the following examples.

Example 7.6

Desktop Printer Configuration State Example

The PRINT Mode for a computer system requires that its Configuration State consist of:

- A computer that is accessible to the printer via cable or wirelessly via a network.
- A printer driver that enables a software application to MC2 the printer via the computer's Operating System (OS) services.

Example 7.7

Heavy Construction Crane Configuration State Example

A crane for erecting tall buildings requires Configuration States to support:

Figure 7.10 NASA Space Shuttle Post-Flight Operations/Safing Mode (Source: NASA (2012))

- Transport over public roads and highways.
- Off-loading and loading onto a transport carrier.
- Erecting itself to construct multi-story buildings.
- Counterbalanced lifting and turning to pick up and deliver construction materials.

The context of the term *configuration* refers to the state of physical components that impact overall performance. Consider the following example of a commercial aircraft:

Example 7.8

Aircraft Configuration States Example

An aircraft's:

- Landing gear is either: (1) DEPLOYED (wheels down) and LOCKED for landing or ground operations or (2) RETRACTED for flight.
- Flaps may be set in a number of positions such as 0°, 15°, and 22°.
- Doors are either (1) OPEN or (2) CLOSED and LOCKED.
- Fuel supply varies based on usage.
- Fuel remaining is XX pounds.
- Landing lights are ON or OFF.
- Take-Off weight is XX pounds; landing weight is YY pounds.

When User missions are performed, a specific mode of operation may require that the User dynamically C2 the system's Configuration States to accomplish the mode's objectives. The following aircraft example illustrates this point.

Example 7.9

Aircraft Landing Configuration States Example

During the LAND Mode Phase of Flight, an aircraft must reduce its airspeed, increase lift due to the reduced airspeed for the remainder of its landing profile, and be able to brake to safely stop before reaching the end of the runway.

To achieve a reduction in airspeed, the pilot reduces the engine thrust (Configuration State change) to slow the aircraft to achieve a specific landing speed on arrival at the runway. As a consequence of reduced airspeed, the aircraft loses altitude thereby necessitating the need to increase lift incrementally during final approach for a proper glide slope.

To sustain lift as a function of glide slope, the aircraft's control surfaces are adjusted (CONFIGURATION State change). Prior to landing, landing lights are activated (Configuration State change) and the landing gear are DEPLOYED and LOCKED (Configuration State change). When the aircraft touches down, the pilot deploys the engine thrust reversers and applies brakes (Configuration State change) until the aircraft comes to a full stop.

In summary, User-selectable modes of operation C2 a SYSTEM's architectural configuration to accomplish specific UCs and achieve mission outcomes and objectives for each mission phase of operation. Why we need to C2 the configuration States brings us to our next topic, Environmental States.

7.6.3 Environmental States

Principle 7.8

Environmental States Principle

Every Enterprise or Engineered system, product, or service must be capable of *operating* in and *interacting* with various types of time and location-dependent Environmental States that exist in its OPERATING ENVIRONMENT.

The preceding sections address how to design and establish User *selectable* modes to C2 a system, product, or service. These analytical frameworks are fine for Conceptual Design; however, the system, product, or service still has to *physically* perform missions in various types of OPERATING ENVIRONMENT conditions. These conditions are typically characterized in terms of (Chapter 8): (1) NATURAL ENVIRONMENT such as weather phenomena - clouds, wind, rain, snow, sleet, fog, and atmospheric, (2) PHYSICAL ENVIRONMENT conditions such as road and flight conditions—shock, vibration, velocity, and acceleration, and (3) INDUCED ENVIRONMENT conditions such as Electromagnetic Interference (EMI). To characterize these conditions, Engineering and Physics began to establish magnitude scales for gauging the level of significance. For example:

- Naval operations classify sea waves in terms of Sea States such as the Douglas Sea Scale.
- Geologists classify earthquake conditions in terms of the logarithmic Richter magnitude scale.
- Climatologists classify hurricane conditions in Levels 1–7.
- Security organizations classify threat conditions in terms of Threat Levels 1–5.

As a result, we introduce *Environmental States* characterizing Operating Environment conditions. Since the OPERATING ENVIRONMENT is comprised of the NATURAL, INDUCED, and HUMAN SYSTEMS Environments, (Chapter 9), Environmental States should be characterized in terms of these elements. Table 7.4 provides examples.

Heading 7.3

When a system, product, or service's Configuration States interact with the Environmental States, the results can range from benign to catastrophic. When we Engineer systems, the System Design Solution must: (1) provide the essential capabilities required to reliably

TABLE 7.4 Examples of Environmental States

Operating Environment States	Category	Example Attributes
Natural environment states	Atmospheric conditions	• Temperature, relative humidity, and pressure • Sun, rain, sleet, and snow • Fog and smog • Winds and wind gusts • Air density • Thunderstorms, tornados, and blizzards • Hurricanes and tropical depressions • Lightning • Blowing sand • Salt spray • Albedo • Occulting • Visibility
	Geophysical conditions	• Polar, desert, tropical, mountainous • Earthquakes, hurricanes, and floods • Landslides, and avalanches • Background radiation
	Sea conditions	• Wave heights • Tsunami levels
	Space conditions	• Van Allen belts • Solar storms and radiation • Asteroids, meteors
Induced environment states	Radiation conditions	• Radio frequencies (RF) and microwaves • Electromagnetic interference (EMI) • Nuclear radiation
	Pollution conditions	• Chemical spills
	Space conditions	• Satellites, space stations • Space junk and debris
Human Systems environment states	Traffic conditions	• Road construction • Accidents • Traffic flow
	Road conditions	• International roughness index (IRI) • Dirt, gravel, and pavement • Washouts • Icing, flooding • Flat, hilly, curvy, and mountainous

accomplish a mission and (2) be sufficiently robust to *survive* encounters with its Operating Environment during the mission. This brings us to our next topic, Dynamic or Transitory States.

7.6.4 Dynamic or Transitory States

Principle 7.9

Dynamic States Principle

Every Enterprise or Engineered system, product, or service must be capable of withstanding and surviving dynamic or transitory conditions created by interactions with its Operating Environment.

Dynamic or Transitory States are the result of a system, product, or service encountering and interacting with conditions in its Operating Environment. Specifically, during the course of normal mission operations, perturbations in the Operating Environment, conditions or events occur that may create System *instabilities*. Engineering's challenge is identifying and anticipating these conditions via modeling and simulation and prototypes to ensure *system stability* and *structural integrity* does not lead to an Abnormal or Emergency situation (Figure 19.5). Dynamic States are typically

externally induced. However, aircraft pilots and automobile drivers who are not paying attention can create uncontrollable system *instabilities* (Dynamic States) that may or may not be recoverable. Consider the following examples.

Example 7.10 **Dynamic or Transitory States**

Externally induced aircraft Dynamic States include Clear-Air Turbulence (CAT), wind shear, strong crosswind gusts or impacts with flocks of birds on TAKEOFF or LANDING. *Internally induced* aircraft Dynamic States include stalls and improper procedure application.

Externally induced spacecraft Dynamic States include:

1. Apollo 12 Launch lightning strikes (Mini-Case Study 11.1).
2. Space Shuttle Challenger SRB O-Ring failure (Mini-Case Study 26.2).
3. Space Shuttle Columbia wing leading edge thermal tile destruction during launch (Mini-Case Study 26.3).

In the case of computer systems Dynamic States, *compensating* and *mitigating design* actions include exception handling (Figure 10.17), hardware or software resets or automatic "watchdog" timer restarts, file recovery, and virus checkers.

As an illustration of Dynamic States, consider the NASA Space Shuttle example in Figure 7.11.

This mini-case study has two purposes: (1) to recognize and appreciate that Dynamic State events can create *instabilities* that must be *dampened* quickly and *controlled*, and (2) serve as a *problem space* for Engineers to solve analytically "up front" without having to discover during actual flight testing or missions:

- **Pre-launch Operations**—Prior to launch, the Space Shuttle's ET was fueled (Dynamic State) with liquid hydrogen propellant.
- **Launch Operations**
  - During Launch, the Space Shuttle Stack—Orbiter Vehicle (OV), ET, and Solid Rocket Boosters (SRBs)—executes a Roll Maneuver (Dynamic State) after it cleared the tower to rotate the stack so that the OV would be on its back relative to Earth (Figure 7.11).
  - During Launch, ice formation created by weather conditions and ET fueling presented risks of breaking OFF and damaging thermal tiles (Dynamic State)

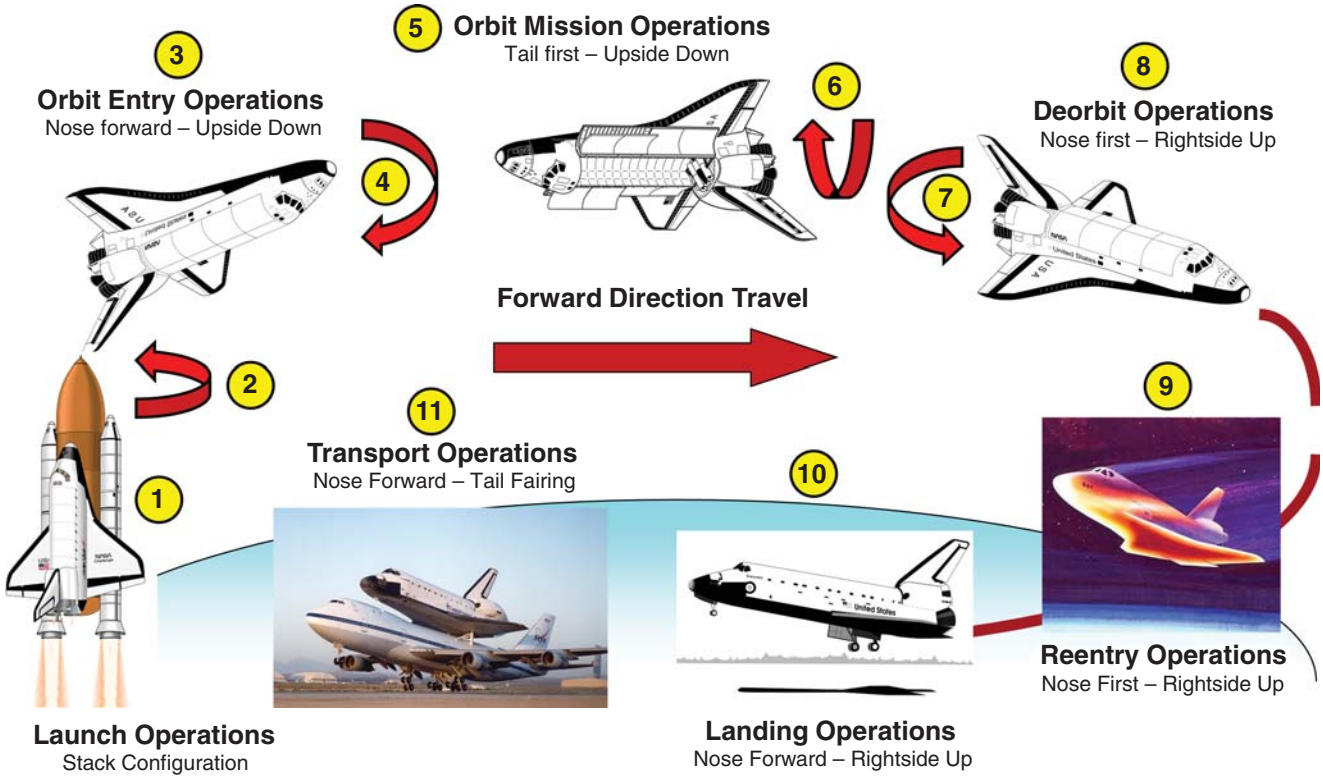


Figure 7.11 Dynamic States Example: NASA Space Shuttle as a Free Body in Space

that protected the wings. Unfortunately, Space Shuttle Columbia experienced a catastrophic event on reentry into the Earth's atmosphere as a result of this condition (Mini-Case Study 26.3).

- During its Ascent, the SRBs were jettisoned (Dynamic State) followed by the ET jettison (Dynamic State) in accordance with its MET.

- **On-Orbit Operations**
  - On reaching orbit, the OV's Cargo Bay doors were opened (Dynamic State) to release heat and perform other tasks such as launching and/or retrieving satellites and conducting experiments.
  - Due to the risk of being hit nose first by space debris left over from previous launches as well as meteorites, the OV executed a Yaw Maneuver to rotate 180° (Dynamic State) to fly backward upside down. This created a smaller, vehicular cross-sectional area for debris to impact and avoided nose-first impacts with leading edge thermal protective tile surfaces critical for Reentry Operations.
- **Deorbit Operations**—When the OV's pilot executed vehicle commands to return to Earth, the OV's main engines were fired to slow its travel velocity causing gravity to pull it toward Earth. Thrusters were fired to execute Yaw and Roll maneuvers that rotated the OV to a nose forward right-side up position in preparation for reentry.
- **Reentry Operations**—During reentry into the Earth's atmosphere, the OV experienced significant thermal stresses (Dynamic State) underside and leading edge surfaces causing a temporary communications blackout (Dynamic State).
- **Landing Operations**—During Landing, the OV had to maneuver through various atmospheric conditions to accomplish the load stresses (Dynamic State) on landing.
- **OV Transport Operations**—If the OV landed in California away from its Kennedy Space Center (KSC) home base, the OV had to be transported cross-country atop a special configured 747 aircraft back to the KSC. Although the OV was not operational at that time, its mounting onto the 747 carrier was *unconventional* and required special consideration of the effects of aerodynamic conditions as Dynamic States during transport including altitude, airspeed, and weather.

From a multi-disciplined SE perspective:

1. Identify scenarios and conditions that could potentially lead to system instabilities.
2. Mitigate them with *compensating design actions* (Chapters 24 and 34).
3. Highlight the conditions and appropriate actions in the Procedural Data Element.
4. Train System Users to recognize the conditions and how to *safely* and *properly* respond to them.

If the system, product, or service has been *properly* designed and Users have been trained well, Dynamic States, in general, should be *recoverable, non-event* operations.

7.6.5 Engineering Perspective Summary

In summary, we have addressed how *modes* and *states* are applied to System Development. Let's shift our focus to applying modes and states information to an aircraft example.

7.7 APPLYING PHASES, MODES, AND STATES OF OPERATION

To illustrate how we might apply phases, modes, and states of operation, let's continue with our commercial aircraft example. Figure 7.12 serves as a frame of reference for our discussion. Due to space limitations:

- Ground Departure Operations also include a Push Back Mode (not shown).
- Ground Arrival Operations also include a Park Mode (not shown).

Several key points are as follows:

1. **Mission Life Cycle Phases** have been partitioned into Pre-Flight, Flight, and Post-Flight Operations.
2. **Pre-flight Operations** are partitioned into Ground Departure Operations consisting of two Modes—Load and Taxi.
3. **Flight Operations** are partitioned into Phases of Flight (Modes)—Takeoff, Ascend, Cruise, Descend, and Land.
4. **Post-Flight Operations** are partitioned into Ground Arrival Operations consisting of two Modes—Taxi and Load.
5. **Operational States:**
   - Are derived from Stakeholder UCs unique to the Mode of Operation.
     a. Are partitioned into Flight Crew (Personnel Element) and Aircraft (Equipment Element) Operational States.
     b. Represent task-based operations and interactions required to fulfill accomplishment of UCs.

System / Product Lifecycle Phases									
	Deployment		Operations, Maintenance, & Sustainment					Disposal	
	Pre-Flight Ops		Flight Operations					Post-Flight Ops	
	Departure Ground Ops		Phases of Flight					Arrival Ground Ops	
Mode(s) UC Context-Based	LOAD	TAXI	TAKE-OFF	ASCEND	CRUISE	DESCEND	LAND	TAXI	UNLOAD
Configuration State(s)	• Config. A1 • Config. A_	• Config. B1 • Config. B_	• Config. C1 • Config. C_	• Config. D1 • Config. D_	• Config. E1 • Config. E_	• Config. F1 • Config. F_	• Config. G1 • Config. G_	• Config. B1 • Config. B_	• Config. A1 • Config. A_
Operational States Task-Based Conditions	Boarding Loading Fueling Inspecting Comm'ng	Taxiing Comm'ng Avoiding Deicing Steering	Taking-Off Commun'ing Aviating Avoiding	Climbing Aviating Navigating Commun'ing Avoiding	Cruising Aviating Navigating Avoiding Commun'ing Serving	Descending Aviating Navigating Commun'ing Avoiding	Landing Navigating Aviating Commun'ing Braking Stopping	Comm'ng Taxiing Avoiding Steering	Comm'ing Deplaning Unloading Cleaning Maintaining
Environmental States Encounters	Weather	Gnd. Traffic Weather Runway Icing	Waterfowl Gnd. Traffic Run'way Icing Rn'way Debris Weather	• Storms • Icing • Air Traffic		• Turbulence • Wind Shear • Cross Winds	• Runway Debris	Weather Gnd.Traffic	Weather
Dynamic States Motion-Based Conditions	Xmitting Receiving Push Back Initializing	Accelerating Moving Maneuver'g Decelerating Braking Stopping Xmitting Receiving	Accelerating Rotating Lighting Transmitting Receiving	Climbing Leveling Turning Accelerating Lighting Transmitting Receiving	Climbing Turning Leveling Accelerating Decelerating Lighting Transmitting Receiving	Decelerating Leveling Descending Lighting Transmitting Receiving	Decelerating Descending Circling Aligning Lighting Transmitting Receiving	Accelerating Moving Maneuver'g Decelerating Braking Stopping Lighting Xmitting Receiving Parking	Transmitting Receiving Power Down

Figure 7.12 Aircraft Example—Mission Life Cycle Phases, Modes, and States of Operation. (Flight Hold Phase is not shown due to space restrictions)

6. **Environmental States** represent types of conditions the aircraft may encounter on the ground and in flight due to the NATURAL ENVIRONMENT, INDUCED ENVIRONMENT, and entities in the HUMAN SYSTEMS ENVIRONMENT.
7. **Configuration States** represent the one-to-many aircraft architectural configurations the Flight Crew and Autopilot must MC2 to safely communicate, aviate, and navigate the aircraft.

Heading 7.4 The preceding section introduced the concepts of modes and the five types of states Engineers must consider. However, missing from these discussions are safety constraints. Engineers tend to think of technical constraints such as safety factors or design margins (Chapter 31) to ensure a margin of safety. Missing from these discussions is *operational safety* for the Users—Operators, Maintainers, and Trainers; the EQUIPMENT; the public; and the OPERATING ENVIRONMENT. These constraints bring us to our next topic, Allowable and Prohibited Actions.

7.8 MODES AND STATES CONSTRAINTS

Allowable and Prohibited Actions Principle

Principle 7.10 Every Enterprise or Engineered system, product, or service mode of operation is constrained by *allowable* and *prohibited actions* that serve as safeguards to ensure the safety of its Users and prevent damage to EQUIPMENT, the public, and external systems in its OPERATING ENVIRONMENT.

A key question arises: *If Modes of Operation are User selectable options, what do they permit the User to MC2*? They enable the User to MC2 UCs and scenarios to achieve specific mission outcomes and performance-based objectives. However, there are limitations and restrictions referred to as *Allowable* and *Prohibited Actions*.

- **Allowable Actions**—Represent User *discretionary* actions that are *allowed* or authorized to be performed without major disruptions to performing other operational tasks with a primary focus on safety.

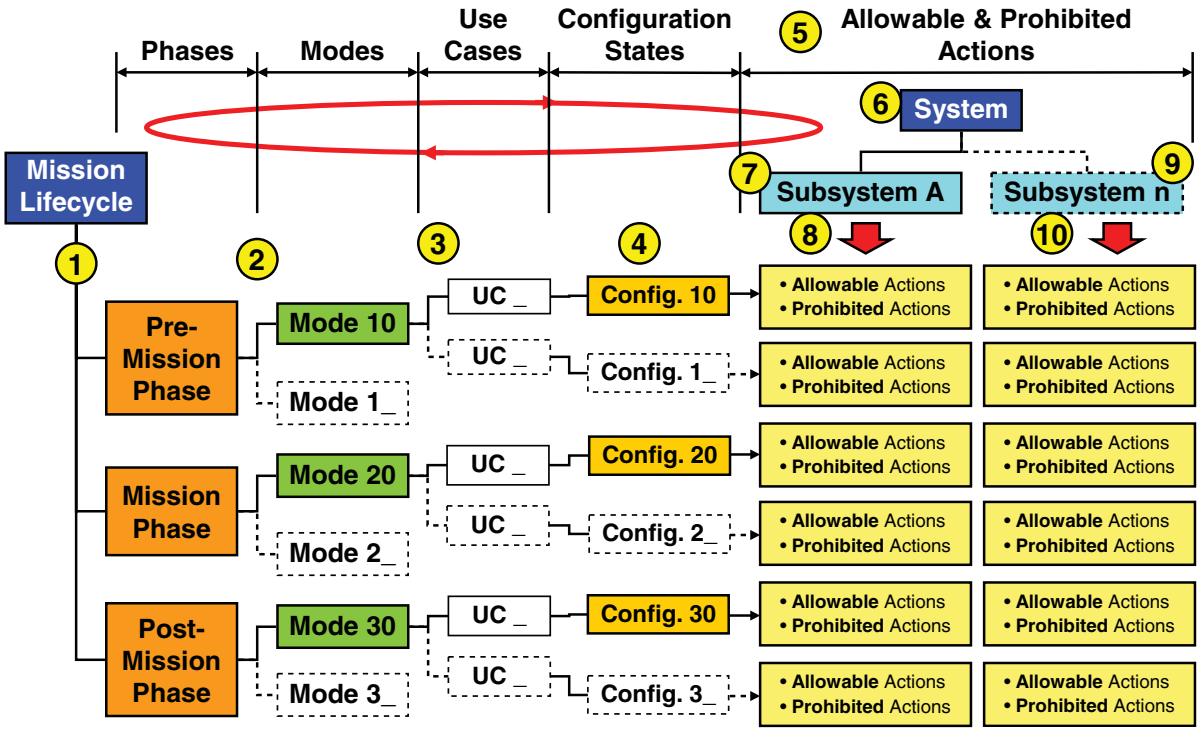


Figure 7.13 Illustration of the System's Command and Control (C2) Entity Relationships (ERs) Constrained by Allowable and Prohibited Actions

- **Prohibited Actions**—Represent User actions that are *disallowed* due to PERSONNEL Element safety-related consequences that may result in injury or death to the Users or End-Users, damage to the EQUIPMENT Element or disruption of performance, or damage or pollution to the OPERATING ENVIRONMENT entities.

To illustrate the concept, consider the graphic shown in Figure 7.13. Using a User's C2 of a system as a frame of reference, Mission Life Cycle phases employ one or more modes of operation.

- Each mode of operation accommodates sets of UCs.
- Each UC controls specific Configuration States.
- Each Configuration State is constrained by Allowable and Prohibited actions that are levied as constraints, allocated, and flowed down to various architectural entities such as SUBSYSTEMS AND ASSEMBLIES.

To illustrate the concept of Allowable and Prohibited Actions, consider the following examples.

Example 7.11

Automobile Example—Allowable and Prohibited Actions

Figure 7.14 provides an example of a hypothetical automobile system. In general, when the User drives an automobile, the vehicle has several modes of operation—PARK, NEUTRAL, REVERSE, DRIVE, and LOW. LOW may be implemented as LOW 1 and a lower gear, Low 2. Symbolically, the graphic represents the vehicle's modes—User-selectable options—via a switch with contacts. Each of these modes allows the User to C2 the motion of the vehicle and negotiate its travel through parking lots or traffic on highways to accomplish a mission—travel to work or vacation.

For safety of the driver and passengers as well as the public, the vehicle's physical design incorporates various types of mechanical and electrical safety interlocks that restrict driver and passenger actions, especially when the vehicle is in motion. We refer to these as *Allowable* and *Prohibited Actions*. Allowable and Prohibited Actions represent the *Acceptable* and *Unacceptable* Inputs illustrated in Figure 3.2. Several key points:

1. The vehicle's Modes of Operation are represented as rows in the matrix. Various types of UCs are represented by columns. Each matrix cell intersection is encoded as an Allowable Action, Not Advisable, or as a Prohibited Action.
2. Automobile System UCs are categorized as Engine UCs, Motion UCs, Braking UCs, Doors and Windows UCs, and Accessory UCs.

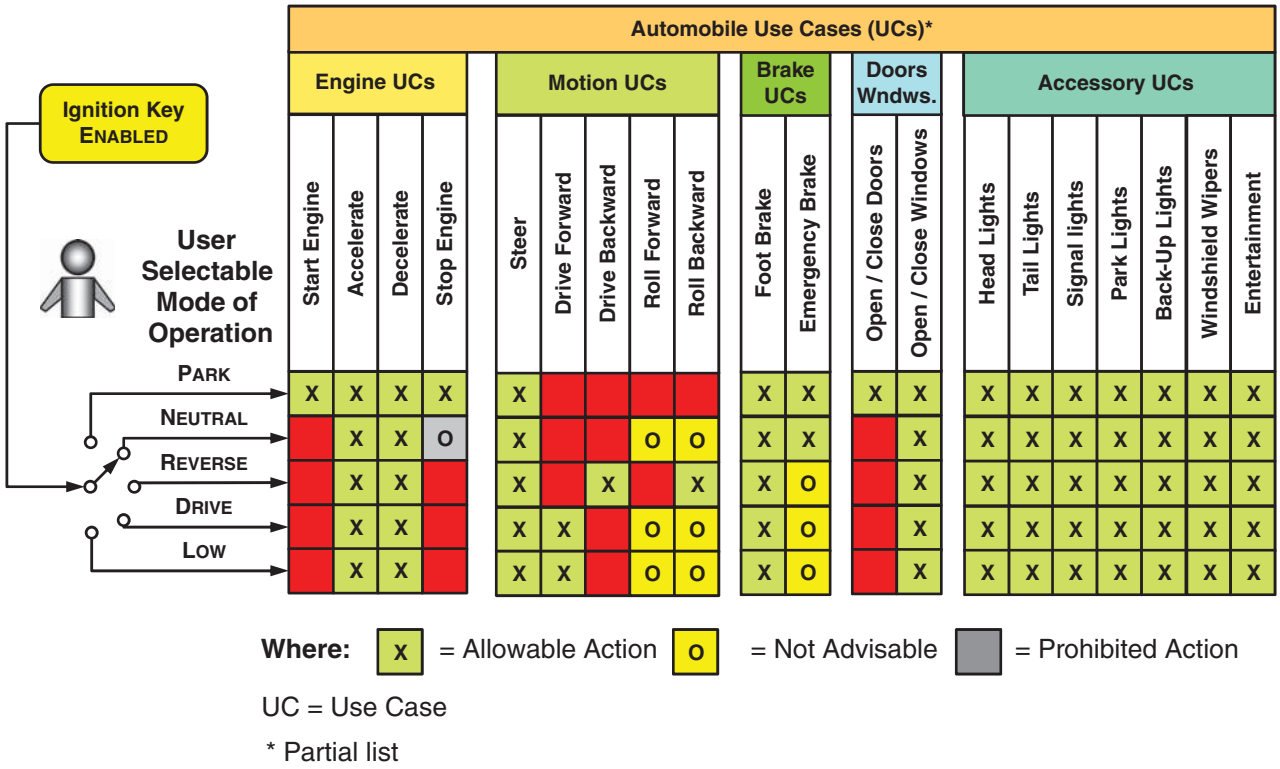


Figure 7.14 Automobile Example—Illustrating Command and Control (C2) Allowable and Prohibited Actions as a Function of Mode of Operation

3. As a result, the matrix maps vehicle modes of operation to its SYSTEM UCs and the applicability of UC usage—Allowable, Not Advisable, or Prohibited.

For example, the vehicle must be in the PARK Mode to perform the Start Engine UC (Allowable Action). The Open/Close doors UC is only permitted in the PARK Mode as an Allowable Action.

Given the matrix in Figure 7.14, *what does this mean in terms of SE and system design perspectives*? It means that each mode of operation needs to be able to *safely* C2 the vehicle's behavioral responses that are produced by its physical architecture configuration and component capabilities—engine, steering wheel, brakes, electrical system, and mechanical system. Modes of operation provide the *conceptual framework* that allows us to logically C2 the vehicle's physical architecture configuration. In turn, the physical configuration, referred to as Configuration States, control the automobile's performance-based behavioral responses and outcomes.

One of the challenges of SE is *protecting* humans from the consequences of their own actions—the Law of Unintended Consequences (Principle 3.5). Hypothetically, you can design a SYSTEM or PRODUCT to automatically inform the PERSONNEL Element or EQUIPMENT Element via safety mechanisms and software about *Allowable* and *Prohibited* Actions. However, these notifications can become prohibitively expensive, unaffordable, and not cost effective.

Unfortunately, *Allowable* and *Prohibited Actions* are often imposed as constraint requirements or determined "after the fact" based on restrictions of system applications and usage. Despite the objective to identify Allowable and Prohibited Actions "up front," inevitably, they may be discovered after the system has been designed or as *latent defects*—design flaws, errors, or deficiencies—emerge after the system has been fielded due to discovery of *unknown modes*. Discovery of the *latent defects* may result in corrective actions such as product recalls, retrofits, or upgrades via EQUIPMENT capabilities, PROCEDURAL DATA updates, PERSONNEL training, or combinations of these.

How do we establish Allowable and Prohibited Actions? Obviously, we can embedded capabilities in EQUIPMENT - HARDWARE and SOFTWARE—such as (1) access, (2) perform error and range checking and (2) provide notifications such as audible, vibratory, or visual cues. This could be very expensive. Alternatives include: notes in PROCEDURAL DATA—operator's manuals, training manuals, user's guides, and EQUIPMENT warning placards - as well as PERSONNEL training.

These actions must come from specification requirements such as Section 3.6 Design and Construction Constraints (Table 20.1). Additionally:

- CHAPTER 31 addresses the need to establish nominal design limit thresholds, cautionary limit thresholds, and warning thresholds (Figure 30.1).
- Chapter 34 notes that a Failure Modes and Effects Analysis (FMEA) (Figure 34.17) should be performed at all levels of System Design to identify any potential *design flaws* and recommend *compensating acti*ons for correcting the designs.

7.9 CHAPTER SUMMARY

In summary, we have introduced the concept of System Phases, Modes, and States of operation. We defined, described, and provided examples of each type as well as their ERs.

Finally, regarding the question *Do Modes contain States or do States contain Modes*? As illustrated in Figure 7.2, both questions are true; System States consist of Modes; Modes C2 Configuration States.

The question's phrasing leaves the impression that only one is correct. To summarize the answer's logic:

- System States characterize the deployment and employment of a system, product, or service as Enterprise organizational asset.
- Modes of operation:
  - Represent User selectable options for C2 of a system, product, or service.
  - Apply to one or more Mission Life Cycle Phase System States.
  - Control Configuration States of the system, product, or service's architecture.
- Configuration States:
  - Represent different arrangements of behavioral and physical architectural elements to provide the required capabilities to support mission operations.
  - Interact with Environmental States during the conduct of a mission.
  - Are constrained in use by *Allowable* and *Prohibited* Actions that limit what the PERSONNEL and EQUIPMENT Elements can exploit.
- Environmental States characterize conditions that may exist in a system, product, or service's OPERATING ENVIRONMENT.
- Dynamic States represent time and location-dependent *instabilities* created by:
  - *Externally induced* perturbations or disruptions resulting from a system's interactions with Environmental States that characterize its OPERATING ENVIRONMENT.
  - *Internally induced* by failure of the User(s) to perform in accordance with the system, product, or service's PROCEDURAL DATA.
- Collectively, identification and analysis of modes and states requires that the System Design Solution—EQUIPMENT, PERSONNEL, MISSION RESOURCES, PROCEDURAL DATA, and SYSTEM RESPONSES must be *sufficiently* robust *operationally*, *behaviorally*, and *physically* to *tolerate*, *survive*, and *recover* from their effects to successfully accomplish missions.

7.10 CHAPTER EXERCISES

7.10.1 Level 1: Chapter Knowledge Exercises

1. What is a Phase of Operation?
2. What is the objective of the Pre-Mission Phase of Operation? How do we bound its starting and ending points?
3. What is the objective of the Mission Phase of Operation? How do we bound its starting and ending points?
4. What is the objective of the Post-Mission Phase of Operation? How do we bound its starting and ending points?
5. What is a Mode of Operation?
6. What is a State of Operation?
7. What is the difference between a Mode and a State?
8. Do Modes contain States or do States contain Modes?
9. When are Modes and States defined in a project? Why?
10. How do we derive Modes and States?
11. What are the relationships among Phases, Modes, and States of Operation?
12. What is the relationship between UCs and Modes of Operations?
13. What is a modal Triggering Event? How are Triggering events characterized?
14. Why do Modes and States have a critical role in SE problem-solving–solution development?

7.10.2 Level 2: Chapter Knowledge Application Exercises

Refer to www.wiley.com/go/systemengineeringanalysis2e.

7.11 REFERENCES

NSTS (1988a), *1988 News Reference Manual*, *National Space Transportation System (NSTS)*, Kennedy Space Center (KSC), FL: National Aeronautics and Space Administration (NASA).

NSTS (1988b), *Mission Profile*, Information content from the NSTS Shuttle Reference Manual (1988), Washington, DC: National Aeronautics and Space Administration (NASA). Retrieved on 1/16/14 from http://science.ksc.nasa.gov/shuttle/technology/sts-newsref/mission_profile.html#mission_profile.

NASA (2012), *STS-114 Shuttle Mission Imagery*, Washington, DC: National Aeronautics and Space Administration (NASA). Retrieved on 1/16/14 from http://spaceflight.nasa.gov/gallery/images/shuttle/sts-114/html/sts114-s-047.html.

Wasson, Charles S. (2014), *System Phases, Modes, & States of Operation; Solutions to Controversial Issues*, 10/29/14 Rev. D, Originally presented at the INCOSE 2011 International Symposium, Denver, CO. www.wassonstrategics.com.

8

SYSTEM LEVELS OF ABSTRACTION, SEMANTICS, AND ELEMENTS

Engineers and others will often tell you that they have a full understanding of System Architecture Concepts as evidenced by their developing architectures for years. Yet, a review of their work products reveals a mixed venue of types and levels of information with no continuity. For most people, the concept of creating an architecture consists of employing presentation software to "drag and drop" boxes onto a slide and connect the boxes with lines. The net result is an *ad hoc* conglomeration of the slide developer's knowledge and experiences. Unfortunately, the audience many times does not recognize the *inconsistencies* that become obscured the presenter's presentation skills.

Today, Model-Based Systems Engineering (MBSE) methods and tools are evolving. The same people who claim to understand architectures are now embarking on new horizons of creating the same dysfunctional, ad hoc, "drag and drop" graphics into more complex MBSE tools, calling it an architecture. Even more challenging are the Plug and Chug … Specify–Build–Test–Fix (SBTF) Engineering Paradigm organizations that view SE as a bureaucratic paperwork. As a result, they refuse to learn SE methods and *erroneously* perceive that if they purchase and use an MBSE tool, by inference, they must be performing SE. When those efforts fail, their reasoning is that SE and MBSE are faulty. So, they revert to their traditional Plug and Chug … SBTF Engineering Paradigm, which they acknowledge is ineffective and inefficient.

This chapter serves a very important purpose: to introduce system architecture concepts that provide insights concerning *how to think about, conceptualize,* and *organize* architectural information into a pattern of behavior that provides analytical *continuity* and *consistency* of implementation. For those performing MBSE, these foundational concepts are absolutely essential to achieving MBSE success and avoiding the path to failure discussed above.

Author's Note 8.1

Chapters 8 and 9 introduce System Architecture Concepts and focus on the taxonomy of system architectural structures. This information serves as the foundational knowledge for Chapter 26 System and Entity Architecture Development. *Why separate system architecture chapters in Parts 1 and 2?*

- Part 1—represents requisite knowledge for understanding the missions, operations, behavior, and taxonomy - structure - of systems.
- Part 2—represents the application of that knowledge to the architecting systems, products, or services.

8.1 DEFINITIONS OF KEY TERMS

- **Entity Relationship (ER)**—A logical or physical association that exists between two or more entities and expressed in terms such as one-to-one, one-to-many, or many-to-one.

System Engineering Analysis, Design, and Development: Concepts, Principles, and Practices, Second Edition. Charles S. Wasson.

Companion Website: www.wiley.com/go/systemengineeringanalysis2e

- **Level of Abstraction**—Knowledge about an entity that suppresses lower-level information and details such as its attributes, properties, or characteristics. For example, the term "family" is a Level of Abstraction that suppresses information such as the quantity of family members – parents, children, and others; and their gender and ages.
- **Powered Ground Equipment (PGE)**—"An assembly of mechanical components including an internal combustion engine or motor, gas turbine, or steam turbine engine mounted as a single unit on an integral base or chassis.
  Equipment may pump gases, liquids, or solids; or produce compressed, cooled, refrigerated or heater air; or generate electricity and oxygen. Examples of this equipment: portable cleaners, filters, hydraulic test stands, pumps and welders, air compressors, air conditioners. Term applies primarily to aeronautical systems" (MIL-HDBK-1908B, 1999, p. 14).
- **Support Equipment**—"All equipment required to perform the support function, except that which is an integral part of the mission equipment. SE includes tools, test equipment, Automatic Test Equipment (ATE) (when the ATE is accomplishing a support function), organizational, intermediate, and related computer projects and software. It does not include any of the equipment required to perform mission operations functions" (MIL-HDBK-1908B, 1999, p. 30).

8.2 ESTABLISHING AND BOUNDING THE SYSTEM'S CONTEXT

One of the first steps in creating an architecture is to establish the System's context within the framework of systems. The key point here is to express *what is*/*is not* part of our SYSTEM. We do this via a *context diagram* such as the one shown in Figure 8.1.

In general, a *context diagram* resembles a wheel with a hub in the center representing the System of Interest (SOI)

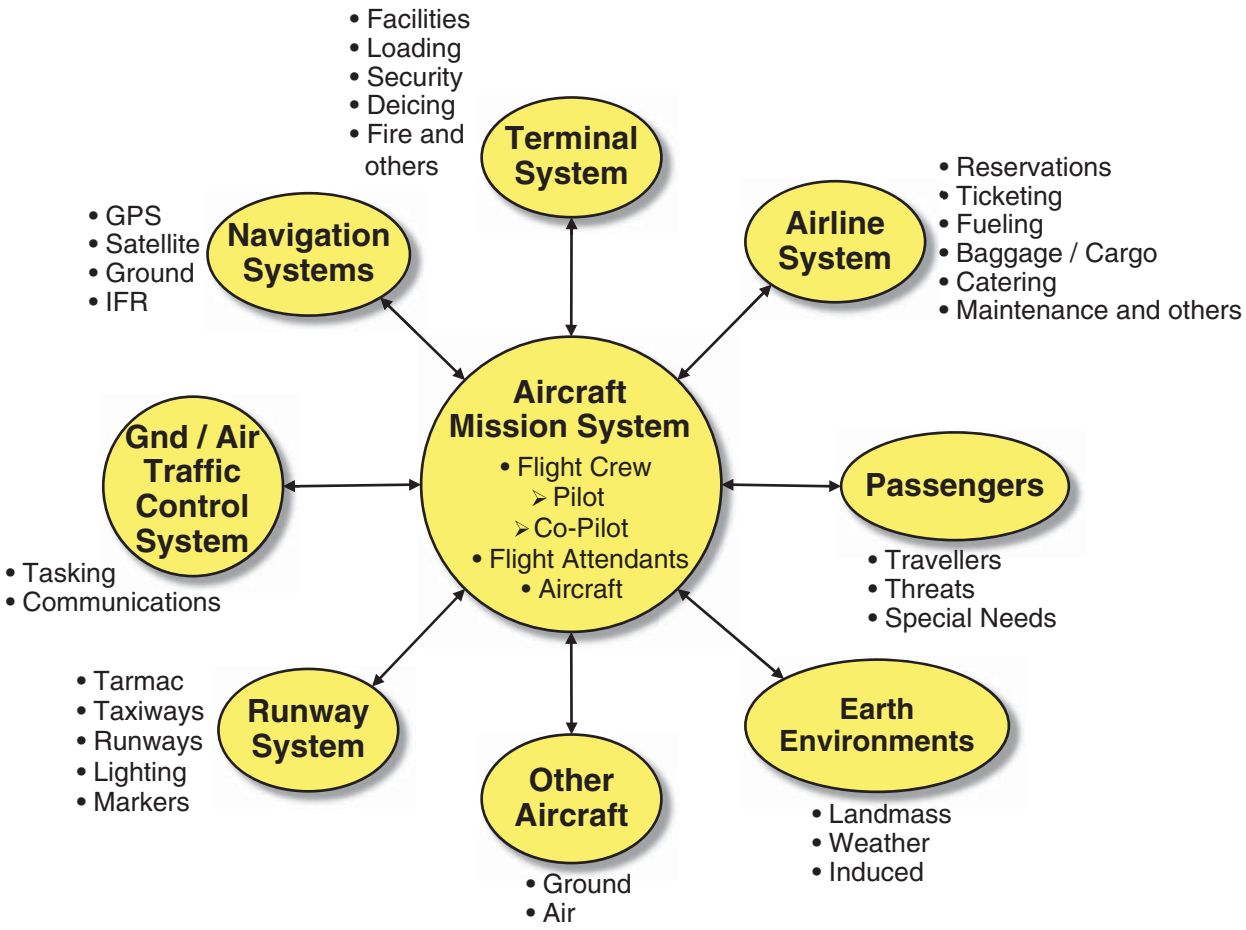


Figure 8.1 Context Diagram for an Aircraft MISSION SYSTEM

and spokes representing interfaces between the SOI and external systems represented by ovals. The external systems represent other SOIs in its OPERATING ENVIRONMENT that may encounter, engage, and interact with it during its Pre-Mission, Mission, Post-Mission, and other Phases of Operation. Interface arrows indicate directions of interchanges such as stimuli, excitations, or cues. Since each of the ovals may be abstract, bulleted lists are provided outside each oval to annotate additional clarifying information.

Author's Note 8.2

Context Diagram Annotations

Bulleted lists, which are *uncommon* in most context diagrams, are a preference based on the author's experience. *Context diagrams*, in general, lack this clarifying information and are typically presented in briefings only to have members of the audience ask more questions that detract from the continuity of the presentation. To avoid this situation, simply annotate each external system with bulleted words that preclude additional questions.

This point leads to a key heuristic.

Heuristic 8.1 Document Integrity

Every document – i.e., figure, table, report, presentation, et al:

- Has a context.
- Should clearly and succinctly communicate a message that precludes the need for further clarification.
- Explicitly answers the tasking that motivated the document's development.
- Identifies source material references and attribution.

If not, you may have failed.

To illustrate the application of a context diagram, Figure 8.1 provides an example of a commercial aircraft system. The Aircraft System and each of the external systems have been annotated with clarification bullets.

A context diagram is a useful tool for bounding the context for what is and is not part of the SOI, a MISSION SYSTEM role, or ENABLING SYSTEM roles. These are high-level systems that consist of multiple levels of lower-level entities such as SUBSYSTEMS, ASSEMBLIES, SUBASSEMBLIES, etc.

A Word of Caution 8.1

Discipline-Based Paradigms in a Multi-discipline Engineering Environment

Engineering education and training sometimes unwittingly create paradigms that have an application context. Such is the case for Software Engineering (SwE), which is data driven. SwEs sometimes perceive context diagrams to identify external systems that exchange *data* with the SOI and its software.

To illustrate this point, assume an SwE is promoted or assigned to lead an Equipment development team for a system that includes hardware. They recognize the need to create a *context diagram* based on their information/data driven paradigm. They challenge the team to create a context diagram for the new system to define and establish its boundaries.

Without recognizing that they are now in an Equipment-based, multi-discipline, hardware-software system, they direct that all *physical interactions* with any external system such as weather, et al be removed from the diagram. The rationale given is that physical interactions are not relevant to a context diagram because they do not exchange … data … with the SOI. *This is erroneous*!

If you believe that system interactions are restricted to data only exchanges, please refer to the NASA photo in Figure 8.2 capturing a lightning strike on Launch Pad 39A prior to a NASA Space Shuttle launch. Decide if this was a "data only" exchange or worthy of SE recognition as an interface in a *context diagram*.

Context diagrams encompass any type of physical interaction – energy, forces, data, et al. – with external systems. Recognize the context of your discipline and experience paradigms and adjust accordingly. If you are going to carry the title of SE, learn to "think outside your own disciplinary box!"

One of the challenges in bounding a system relates to the SYSTEM's Users and End Users. *Are they internal – part of - or external to the SOI*? This is a very critical point that a context diagram should reveal. For the Aircraft System shown in Figure 8.1, the aircraft's users—Flight Crew—that Monitor, Command, and Control (MC2) the aircraft are *internal* to the system. As End User's that benefit from using the system, the aircraft's passengers are shown *external* to the system.

The presence of multiple levels of entities within a MISSION SYSTEM or ENABLING SYSTEM brings us to our next concept concerning the need to establish levels of abstraction and a language of semantics to communicate about each level.

8.3 SYSTEM LEVELS OF ABSTRACTION AND SEMANTICS

Principle 8.1

Levels of Abstraction Principle

Every SOI, its MISSION SYSTEM(S), ENABLING SYSTEM(S), and EQUIPMENT – HARDWARE and SOFTWARE - consists of one or more *levels of abstraction* with standardized titles

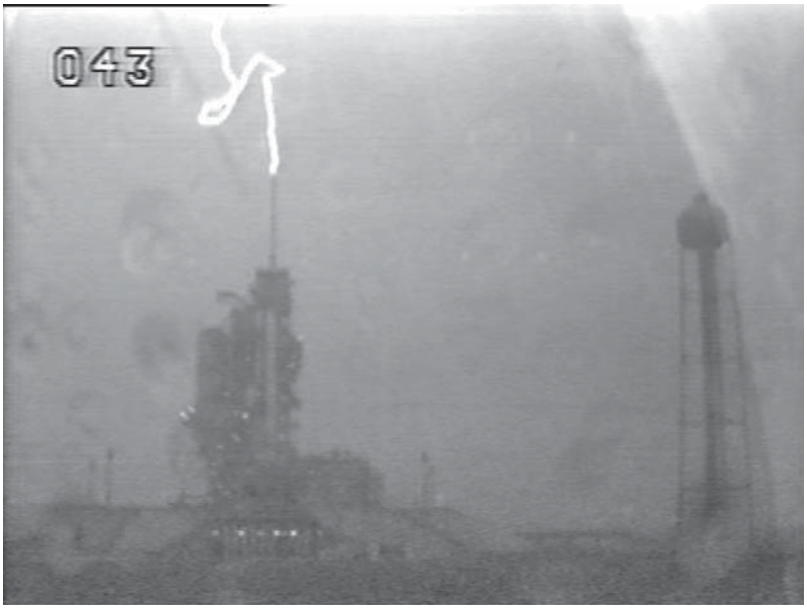


Figure 8.2 Lightning Strike on NASA Space Shuttle, Pad 39A—July 10, 2009. (Source: KSC (2009).)

indexed by levels or tiers relative to the System User's SOI frame of reference.

One of your first tasks as an SE or System Analyst is to establish a semantics frame of reference for your SOI. When most people refer to *systems*, their contextual view is based on their own perspective—an observer's *frame of reference*. When you listen to communications between Users, the Acquirer, and System Developers, you soon discover that one person's "System" equates to another person's "Subsystem." In a leadership role as a Systems Engineer, your job is to establish a consensus of levels of abstraction semantics that unifies team members with a common frame of reference for communications.

One of the ways to alleviate this problem is to establish a standard semantics convention that enables Engineers, System Analysts, and others to communicate intelligibly using a common contextual language. Once the convention is established, update the appropriate Enterprise command media—policies and procedures—for use in training Engineering personnel.

The best way to illustrate *Levels of Abstraction* is graphically. Figure 8.3 provides an illustration.

Beginning in the upper left corner, a system, product, or service consists of an initial set of loosely coupled entities such as ideas, objectives, concepts, and parts (i.e., items A through N). If we analyze these entities or objects, we may determine that various groupings share a common set of objectives, characteristics, outcomes, etc. as illustrated in the lower left portion of the figure. We identify several groupings of items:

- Entity 10 consists of Entities A and E.
- Entity 20 consists of Entities C, F, and I.
- Entity 30 consists of Entities D, J, H, and M.
- Entity 40 consists of Entities B, K, L, N, and 0.

On the right side of the figure, we establish a hierarchical framework consisting of three levels of abstraction. Observe that the title of each level of abstraction suppresses lower-level details. For example, the SYSTEM Level represents everything below it but does not reveal details concerning the number of levels of abstraction below it, quantities of entities at each level, or their contents. The same is true for Entities 10–40. As a result, we create an analytical framework that represents the hierarchical structure, or taxonomy, of a system and of its levels of abstraction.

The concept of generic levels of abstraction is useful information for simple systems. However, large, complex systems involve multiple levels of detail or abstraction. In fact, up to 10 or more levels, depending on the SOI. Where this is the case, *how do SEs and System delineate one level of abstraction from another*? They do this by establishing an observer's frame of reference convention.

8.3.1 Establishing an Observer's Frame of Reference Convention

When establishing any type of convention for an observer's frame of reference, one of the first steps is to decide what the origin is. For a consumer product or a contract System

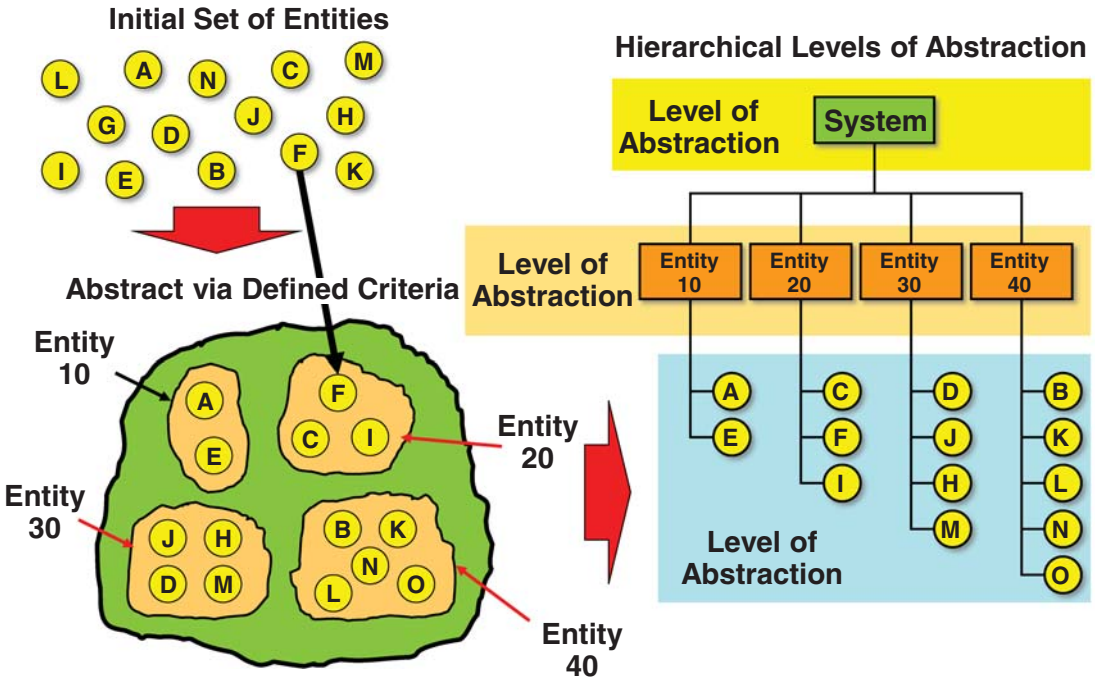


Figure 8.3 Abstracting Entities into levels of Abstraction

Development, the deliverable system – SOI – becomes the origin for the frame of reference.

Figure 8.4 illustrates two analytical conventions used to establish a convention for hierarchical levels of abstraction. One example convention employs Level 0, Level 1, Level 2, and so forth. Another convention employs Tier 0, Tier 1, and so forth semantics.

Observe that the highest level is Level 0 or Tier 0. By convention, Level 0 or Tier 0 represents the User's SYSTEM that will serve as the frame of reference for integrating the SOI—that is, your System, which is designated as a Level 1 or Tier 1 System. Recall from Principle 1.1 that SE *begins* and *ends* with the User(s) and End User(s). Therefore, the User's system becomes the Level 0 origin of the frame of reference. Since it does not make sense to communicate in negative numbers, we use positive integers to represent lower levels of abstraction. Remember, this is a reference convention for communications, not mathematics.

A Word of Caution 8.2

Levels of Abstraction Connotations

The SUBSYSTEM, ASSEMBLY, SUBASSEMBLY, AND PART Levels of Abstraction naming convention infers a hardware connotation. Since Engineers, in general, often think of systems in terms of *deliverable* hardware, the naming convention provides continuity with their mental models.

However, this text treats these terms as *generic, hierarchical descriptors*. Here's why. As you will discover later in the chapter, a SUBSYSTEM, ASSEMBLY and so on entity may consist of: personnel, equipment – hardware and software, mission resources, procedural data, and system responses. In this context, the entity performs a mission and is therefore considered to be a performing entity.

The establishment of a convention of hierarchical levels of abstraction is fine as a framework. However, we still have to solve the problem of communicating about entities at various levels of abstraction. This brings us to our next topic: establishing a semantics convention to express system nomenclature levels.

8.3.2 Establishing a System Nomenclature Semantics Convention

Continuing with Figure 8.4, observe the hierarchical structure on the right side. We begin with the highest-level system, the User's System, which consists of our SOI and Other Enterprise Systems. Below this level, we establish a naming convention of nomenclatures for each level of abstraction such as SYSTEM, SEGMENT, PRODUCT, SUBSYSTEM, ASSEMBLY, SUBASSEMBLY, and PART Levels. This is an 8-level convention that accommodates most types of systems. Your SOI, for example, may only have 3 levels or 5 levels.

Observe that the filled SySML™ diamond symbol (Appendix C), which represents Composition by Aggregation expresses an Entity Relationship (ER) between levels of

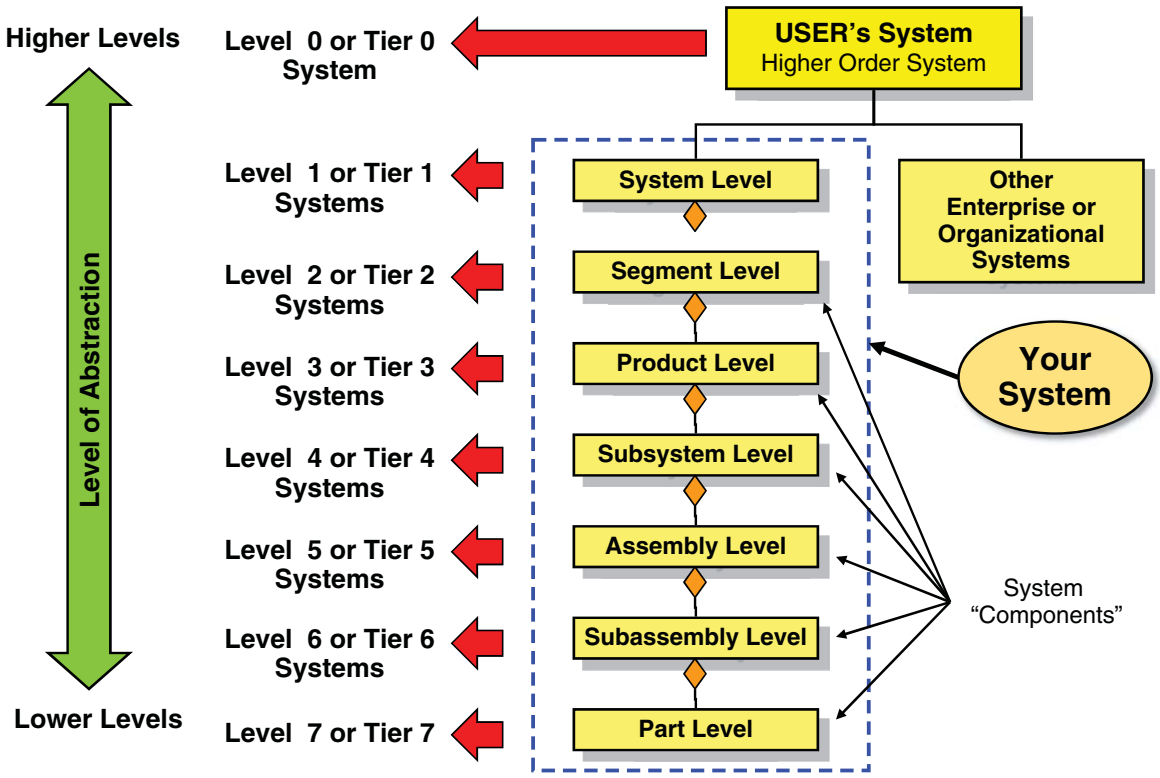


Figure 8.4 System Levels of Abstraction and Semantics Frame of Reference

abstraction. For example, each PRODUCT consists of one or more SUBSYSTEMS, each ASSEMBLY consists of one or more SUBASSEMBLIES, depending on tailoring discussed later in this section. To understand the origins and rationale for this naming convention, let's explore some of the background from industry.

8.3.2.1 Origins of the Semantics Nomenclature Convention In general, most Enterprises view their deliverable SYSTEM as consisting of SUBSYSTEMS. Interestingly, some commercial Enterprises sometimes refer to their Level 1 System as "product" that is comprised of lower-level "systems," not subsystems. Below the SYSTEM and SUBSYSTEM Levels, Enterprises may employ the terms ASSEMBLIES or SUBASSEMBLIES. Since organizations create parts lists, it is only natural to have the PART Level as the lowest level of abstraction.

8.3.2.1.1 Component Semantic Origins and Usage Observe that the term *component* is not used at any of the levels of abstraction in Figure 8.4. The term *component* is a generic term that has both explicit and ambiguous meanings: *explicit* in terms of being a physical entity and *ambiguous* in terms of being applicable to any level of abstraction. As a convention for this text, we adopt the context that a *component* refers to any entity at any level of abstraction – PRODUCT, SUBSYSTEM, ASSEMBLY, SUBASSEMBLY, or PART.

8.3.2.1.2 SEGMENT Level Origins and Usage Large, complex Enterprise Systems often have many levels of abstraction that represent land, sea, air, and space-based systems or combinations of these. Examples include NASA, military, and commercial organizations such as the United Parcel Service (UPS), Federal Express, and so on. To accommodate large, complex Enterprise Systems applications, we add a SEGMENT Level - Tier 2 - to accommodate land, sea, air, or space applications.

8.3.2.1.3 PRODUCT Level of Abstraction Origins and Usage As stated earlier, Enterprises often think of *systems* consisting of *subsystems*. As you will discover in Part 2, "SYSTEM DESIGN AND DEVELOPMENT PRACTICES," system design should be a last resort (Principle 16.6) after you have exhausted all means to find an internally developed or commercially available component that can be used directly or modified to meet a specific purpose at any level of abstraction. This being the case, a SYSTEM may be comprised of a combination of internally developed components and externally procured commercial products. To illustrate this point, consider the following example of a computer system.

Example 8.1 **Computer System Example**

In general, a computer system consists of a processor (tower), keyboard, monitor, printer, etc. The computer system developer may decide to develop these components internally or purchase separately from various commercial vendors specializing in development of a component. So, the computer system developer purchases "*products*" such as a keyboard, mouse, power supply, and so forth from a vendor's catalog, packages them in enclosure with their name on it, and then sells the item in their own catalog as replacement components.

Additionally, some systems such as a computer system are simply analytical abstractions that are *virtual* entities. Rhetorically speaking, unless all components are self-contained within a single enclosure, *can you actually "touch" a computer system*? The answer is no; the so-called computer system is a virtual, analytical abstraction as illustrated in Figure 8.5. You can only touch its PRODUCT or SUBSYSTEM Level components - keyboard, tower, monitor, printer, etc. From the computer system developer's frame of reference, *products* purchased from vendors will be designated as SUBSYSTEMS within their computer system. Given the virtual nature of some systems and commercial industry views of selling a *product*, we adopt the PRODUCT Level of abstraction – Tier 3 - below the SEGMENT Level – Tier 2.

8.3.3 Tailoring Levels of Abstraction for Your System's Application

The preceding discussion introduced a set of semantics for application to large, complex systems. You and your organization may or may not have an 8-Level System. Tailor the number of system levels of abstraction to match your system's application.

Figure 8.6 illustrates how a SYSTEM's levels of abstraction can be tailored for a specific Enterprise application. The left side of the figure represents the nomenclature naming convention used in this text for System Levels of Abstraction. The right side represents an Enterprise's tailoring of the standard system levels. In this case, a system development project has adopted the following semantics: User's Level 0 SYSTEM, SYSTEM Level, SUBSYSTEM Level, ASSEMBLY Level, and PART Level. Reference level numbers (Level 1, Level 2, etc.) have been sequentially applied to match the tailoring. As a result, the dashed boxes for the SEGMENT, PRODUCT, and SUBASSEMBLY Levels have been collapsed to form 5-Level System.

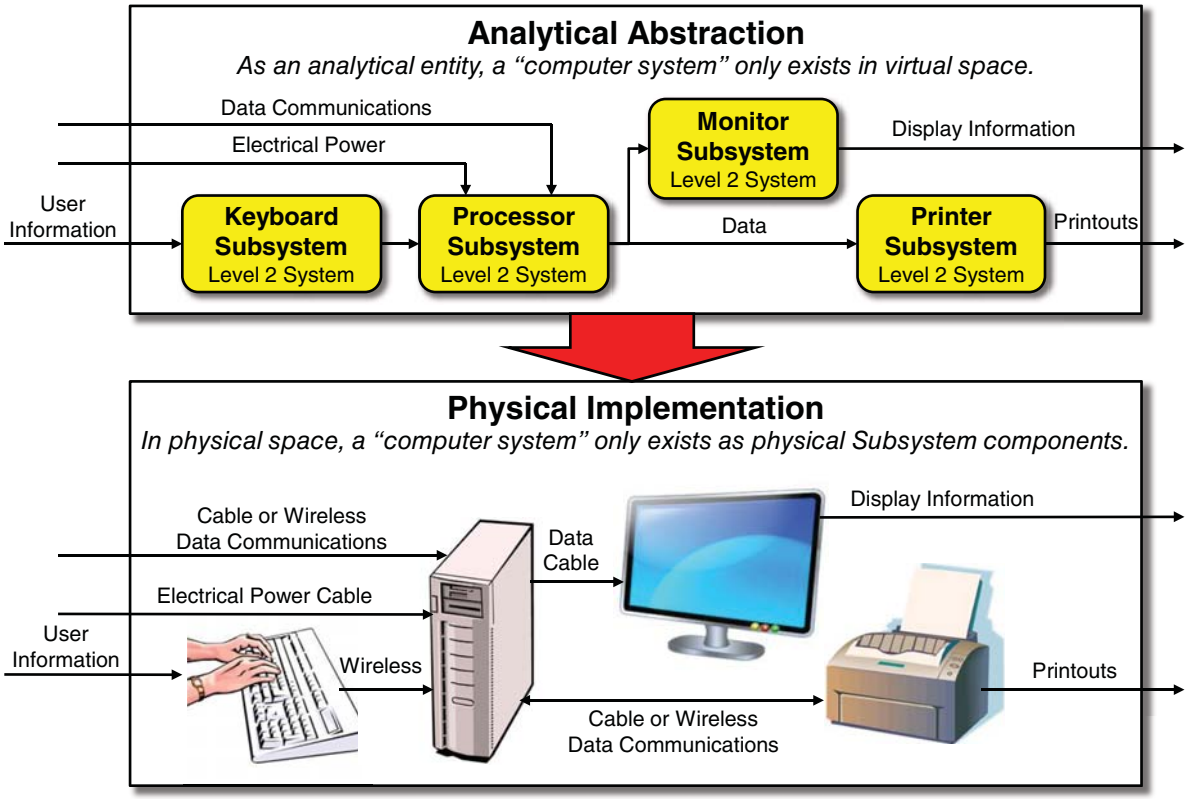


Figure 8.5 Desktop Computer System as a Virtual Analytical Abstraction

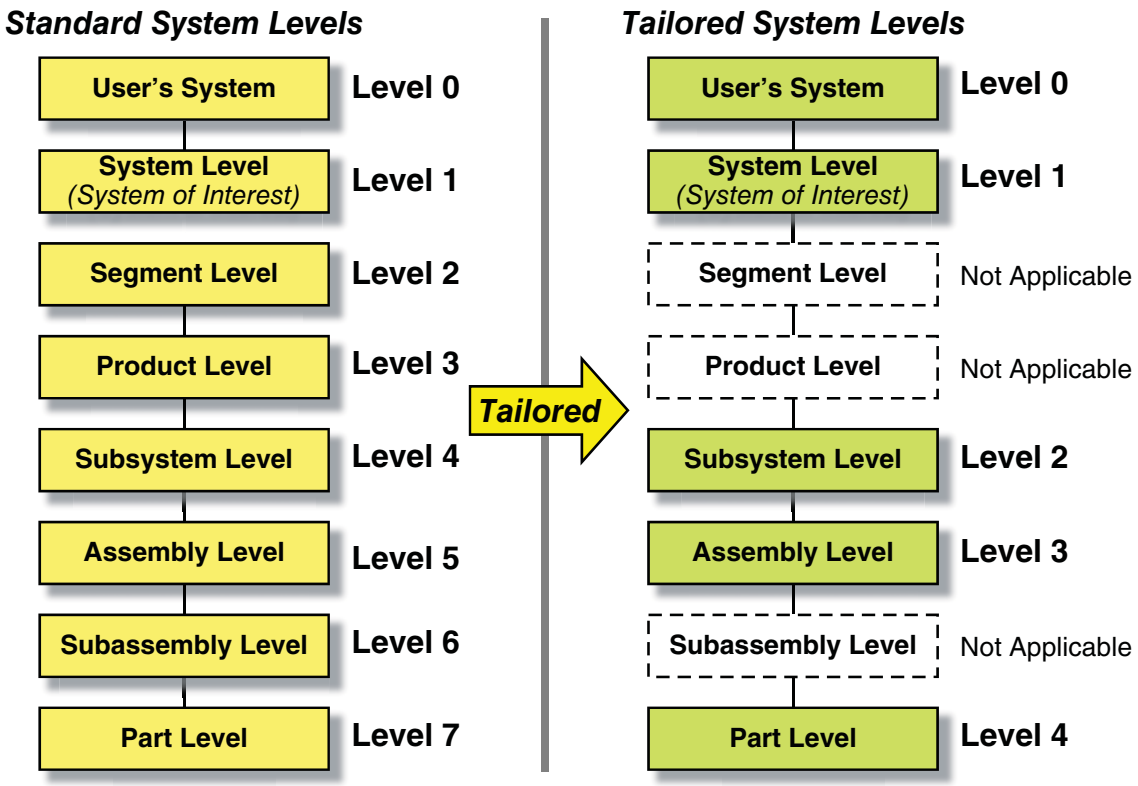


Figure 8.6 System Levels of Abstraction Tailoring Example

8.4 SYSTEM DECOMPOSITION VERSUS INTEGRATION ENTITY RELATIONSHIPS

The preceding discussion establishes a naming convention for *analytical* system decomposition. However, when we integrate two SUBSYSTEMS, for example, the integration may require an installation kit consisting of PART Level components such as brackets, nuts, and screws. to physically integrate the two SUBSYSTEMS.

From an analytical *decomposition* perspective, we expect our SYSTEM to be hierarchically composed of only SUBSYSTEMS, which is true. However, to accomplish integration of the SUBSYSTEMS into a SYSTEM, a SYSTEM Level parts list identifies the two SUBSYSTEMS and an installation kit composed of *physical* PART Level components. The same occurs for the integration of PARTS into SUBASSEMBLIES, SUBASSEMBLIES into ASSEMBLIES, and ASSEMBLIES into SUBSYSTEMS.

Observe two different concepts here: *analytical, hierarchical decomposition* (Top–Down) versus *multi-level physical integration* (Bottom–Up) of components into what is referred to as the System's physical Product Structure. The integration side of the issue typically emerges later during physical system design. *Why are lower-level* ASSEMBLIES, SUBASSEMBLIES, *or* PARTS *found in a system's analytical decomposition structure but located at different levels of integration in its physical product structure*?

To reconcile the differences between *analytical decomposition* versus *physical system integration*, we do so contextually via Figure 8.7.

In general, Figure 8.7 expresses:

- Top–Down multi-level *analytical decomposition* ERs
- Bottom–Up multi-level *physical system integration* ERs

Given these two points, observe that graphic consists of the analytical decomposition structure – Levels of Abstraction, semantics, and ERs - depicted earlier in Figure 8.4. However: (1) since the Levels of Abstraction can be tailored as shown in Figure 8.6 and (2) system integration may intermix components from different levels of abstraction, the SYSTEM Level entity *may or may not* be comprised of SEGMENTS, PRODUCTS, SUBSYSTEMS, ASSEMBLIES, or PARTS.

On inspection, Figure 8.6 may appear to be an academic exercise. However, it is real world example that Project Managers (PMs) and Project Engineers must address. Specifically, Project Work Breakdown Structures (PWBSs) typically depict the system's analytical product structure -

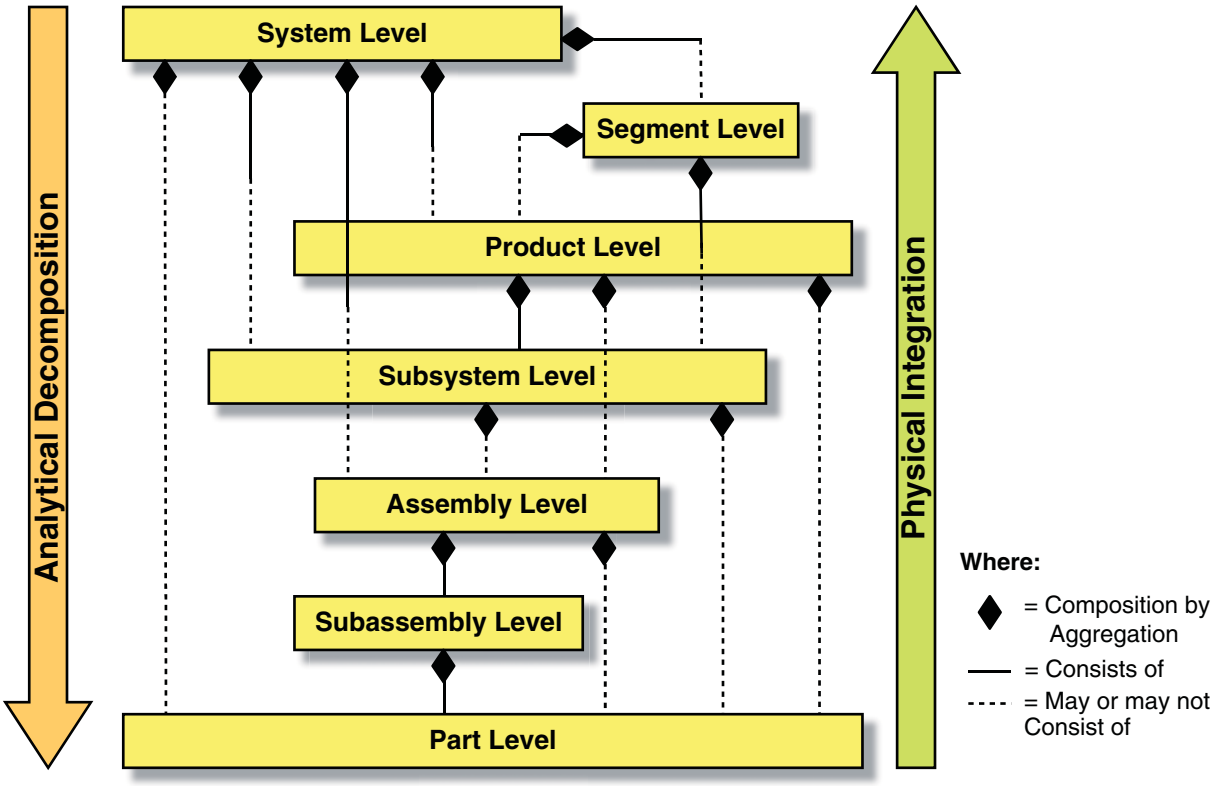


Figure 8.7 System Analytical Decomposition into Levels of Abstraction versus Physical Integration Entity Relationships (ERs) Defined in Table 8.2

SYSTEM comprised of SUBSYSTEMS and so forth – plus a System Integration & Test line item at each level of integration.

The analytical *product structure* enables project managers to collect weekly Project Work Order (PWO) labor charges. SEs, System Analysts, and others are assigned accountability via project memos for *developing* specific components within the structure. However, the PWBS analytical hierarchy product structure does not depict the reality that SUBASSEMBLY #1 such as a standard interface or software application that may be required for integration into multiple SUBSYSTEMS. As a result, creation of separate analytical decomposition and physical system integration structures similar to Figure 8.7 is critically important for planning projects.

Hierarchical decomposition employs the same rules as outlining a document. *Avoid* having a single item subordinated to a higher-level item. Best practice suggests that you should always have at least two or more entities at a subordinated level of indenture. In the case of systems, this does not mean that the subordinate entities must be at the same level of abstraction. For example, a SYSTEM may be composed of two SUBSYSTEMS and an installation kit of PARTS to mount of connect the two SUBSYSTEMS.

8.4.1 Problem Space–Solution Space Decomposition

The preceding discussion presents and describes an analytical framework of levels of abstraction and semantics that form the structure of a system architecture. The question is: *how do these levels and entities within these levels apply to the development of physical consumer systems such as smart phones or computers*? The answer resides in the Problem-Solving–Solution Development concept introduced in Figure 4.7 illustrating the Problem Spaces and Solution Spaces concept.

Beginning with a system's mission as an abstract and potentially complex Problem Space, we analytically partition it into one of more Solution Spaces (Principle 4.18). The boundaries on the Solution Spaces may be *fuzzy* or *notional* and shift back and forth as we analyze the entity; perform trade-offs such as cost, performance, risk, etc.; and make decisions until the entity reaches a level of maturity that enables us to establish firm boundaries.

In this raw form, we use a context diagram to establish its context relative to its sibling Solution Spaces and external systems. Our challenge is: *how do we deal with abstractness and complexity*? Analytically, the answer resides in analytically partitioning - decomposing - complexity (Principle 4.17) into successively lower levels that refine

solutions, enable us to manage risk, and ultimately lead to a physical solution. We will refer to this as the Problem Space–Solution Space Decomposition. As a result, we decompose the SYSTEM into SUBSYSTEMS, into ASSEMBLIES, and so forth.

8.4.2 Problem Space–Solution Space Decomposition Approach

The Problem Space–Solution Space Decomposition Approach is illustrated in Figure 4.7. Beginning in the upper left corner, the abstract System Problem Space is partitioned and decomposed into one or more lower-level Solution Spaces based on the set of mission-based capabilities to be provided by the system. The set of Solutions Spaces are conceptualized and evaluated using SE methods such as trade-off Analysis of Alternatives (AoA) (Chapter 32), Modeling and Simulation (M&S) (Chapter 33), and other methods. As decisions regarding each Solution Space mature, designations such SUBSYSTEM #1, SUBSYSTEM #2, and so forth emerge.

Now, we have a new challenge. SUBSYSTEM #1, SUBSYSTEM #2, and others may also be *abstract* and *complex* entities. So, SUBSYSTEM #1 Solution Space, for example, becomes SUBSYSTEM #1 Problem Space that must be decomposed into lower-level ASSEMBLY Level Solution Spaces—ASSEMBLY #1, ASSEMBLY #2, and others. We continue the process until we finally reach the PART Level of abstraction.

Several key points:

1. The problem-solving and solution development methodology required for conceptualization, formulation, trade-off, selection process described above is accomplished via the SE Process Model (Figure 14.1).
2. Observe that we did not use the term *functional decomposition*. Instead we *partitioned* capabilities into *notional* Solution Spaces to achieve what has been traditionally referred to as decomposition, another ambiguous term due to multiple connotations. Functional decomposition as a concept has served modern SE well for several decades. However, our SE knowledge has advanced to a new stage of understanding that reveals the fallacy of functional decomposition; its focus on *functions* – i.e., actions to be performed, not performance - *how well*. When we ultimately allocate specification requirements (Chapter 21) that bound this abstract Problem Space, we will do so based on capabilities, not functions.
3. The concept of *what* versus *how* emerges in Figure 8.7. Observe that Top-Down decomposition represents *how* each level of abstraction will be implemented. Conversely, Bottom-Up *integration* reveals *what* is to be accomplished at each higher level.

8.4.3 System Decomposition: Complicated versus Complex Systems

The concept of complex systems may be ad hoc, dynamic, autonomous, etc. Examples include social networks, political systems, healthcare systems, complex EQUIPMENT Element Systems, and others raises issues related to system decomposition. The central issue centers on the view that *complex systems* cannot be decomposed, while *complicated system*s are decomposable. Warwick and Norris (2010) provide a discussion of this issue.

8.4.4 Summary: Levels of Abstraction

In summary, our discussion in this section introduced the concept of system levels of abstraction and semantics. This hierarchical framework enables SEs to standardize analysis and communications about their SOI. The intent of a *semantics convention* is to establish a common terminology frame of reference across members of a System Developer's team, the Acquirer, and the User to facilitate communicating complex hierarchies.

Given an understanding of a system's taxonomy, we are now ready to introduce System Elements Concept.

8.5 LOGICAL–PHYSICAL ENTITY RELATIONSHIP (ER) CONCEPTS

Earlier in Chapter 1, we highlighted shortcomings in the *ad hoc, endless loop*, Plug and Chug … SDBTF-DPM Engineering Paradigm (Chapter 2). We noted that Enterprises that foster the SDBTF-DPM Paradigm often take *quantum leaps* from specification requirements to a physical, point design solution without due consideration of selecting the architecture from a set of viable candidates via an AoA (Chapter 32).

Shifting the SDBTF–DPM Paradigm requires understanding *what* has to be accomplished—missions, objectives, ConOps, Operational Concept Descriptions, and so on—before defining *how* the system will be designed - physical design solution. One of the key steps in this process is recognition that an ER or associative relationship exists between two entities—SYSTEM to SYSTEM, SUBSYSTEM to SUBSYSTEM, etc. We refer to this as a *logical ER*. The concept is that if we can determine *who* interacts with *whom*, we can employ SE methods to translate that relationship into a physical ER. Let's explore these two concepts further.

8.5.1 Logical Entity Relationships (ERs)

The first step in identifying logical ERs is to simply *recognize* and *acknowledge* that some form of association exists between two entities through deductive reasoning. You may

not know the physical details of the relationship—that is, how they link up—but you know a relationship does or will exist. Graphically, we depict these relationships as simply a line between the two entities.

The second step is to characterize the logical relationship in terms of logical functions—*what interaction* occurs between them—that must be provided to enable the two entities to associate with one another. When we assemble the logical entities into a framework that graphically depicts their relationships, we refer to the diagram as *logical architecture*. To illustrate, let's assume we have a simple room lighting situation as shown in Figure 8.8.

Example 8.2 **Room Lighting—Logical Architecture Entity Relationships**

The top portion of Figure 8.8 depicts a simple Room Lighting System consisting of a User (SysML™ Actor – Appendix C) desiring to control a room Light Source (Actor). As a logical representation, we draw a line between the User (Actor) and the Light Source (Actor) to acknowledge the relationship. Thus, we state that the User (Actor) has a *logical association* or ER with the Light Source.

Next, we need a control mechanism for the Light Source (Actor), which derives its energy from a Power Source (Actor). We complete the representation by connecting the User (Actor) with the Lighting Control. The Lighting Control enables the flow of current from the Power Source to the Light Source. When energized, the Light Source illuminates the room enabling the User to move around or perform tasks.

From this description, you should note that we purposely avoided specifying *how* the:

- User interfaced with the Lighting Control.
- Lighting Control controlled the Power Source.
- Power Source provided current to the Light Source.
- Light Source illuminates the User.

The diagram simply documents *associative* relationships. Additionally, we *avoided* specifying *what* physical mechanisms such as light switches, lighting fixtures, quantity lights and wattages, etc. will be used to implement the Lighting Control, Power Source, or Light Source. These decisions will be deferred to our next topic.

Based on this *logical* representation, let's investigate its *physical* implementation of the Room Lighting System.

8.5.2 Physical Entity Relationships (ERs)

The physical implementation of SYSTEM interfaces requires more in-depth analysis and decision making. *Why*? Typically, cost, schedule, technology, support, and risk become key drivers that must be "in balance" for the actual implementation. Since there should be a number of viable candidate options available for implementing an interaction, trade studies Analysis of Alternatives (AoA) (Chapter 32) may be required to select the best selection and configuration of physical components. Graphically, we refer to the physical implementation of an interface as a *physical representation*.

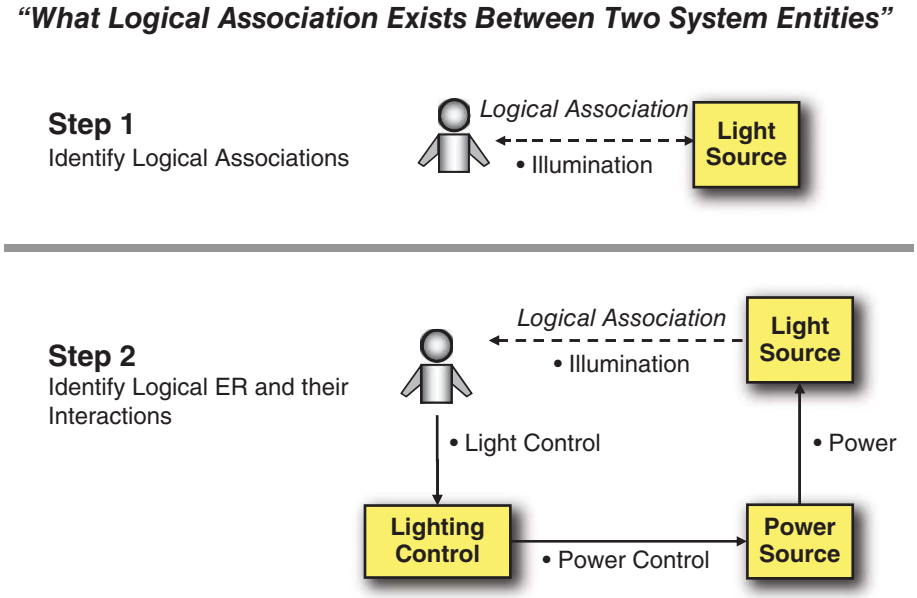


Figure 8.8 Logical Entity Relationships (ERs) Example

As Engineering designers select components such as copper wire types and sizes, light switches, lighting fixtures, etc., we configure them into a System Block Diagram (SBD) and electrical schematics that depict the physical ERs. These diagrams become the basis for the Physical System Architecture continuing with the logical architecture. Consider the following example that expands on the preceding logical ERs example:

Example 8.3 **Room Lighting—Physical Architecture Entity Relationships**

After some analysis, we develop a physical representation or physical system architecture of the Room Lighting System. As indicated by Figure 8.9, the Physical Lighting System consists of the following physical entities: a Power Source, Wire #1, Wire #2, Light Switch, a Building Structure, a User, and a Light Receptacle containing a Light Bulb. The solid black lines represent electrical interfaces; the dashed lines represent mechanical interfaces. In physical terms, the Building Structure provides mechanical support for the Light Switch, Wire #1, Wire #2, and Light Fixture that holds the Light Bulb.

When the User (physical entity) places the Light Switch (physical entity) in the ON position, AC current (physical entity) flows from the Power Source (physical entity) through Wire #1 (physical entity) to the Light Switch (physical entity). The AC current (physical entity) flows from the Light Switch (physical entity) through Wire #2 (physical entity) to the Light Receptacle (physical entity) and into the Light Bulb (physical entity). Visible light is then transmitted to the User until the Light Switch is placed in the Off position, the Light Bulb burns out, or the Power Source is disconnected.

8.5.3 Logical - Physical Architecture Approach

The partitioning and sequencing of these discussions provide a fundamental portion of the methodology for developing systems, products, or services. If you observe and analyze human behavior, you will discover that humans characteristically have difficulty deciding *what* decisions to make and the strategic steps required to make those decisions. Humans often desire lots of information but are often unable to synthesize all of the data at the individual or team levels to arrive at an encompassing, multi-level design solution in a single decision. As a result, the ramifications of the decision-making process increase exponentially with the size and complexity of the system.

Given this characteristic, SEs, System Analysts, Engineers, and others need to *incrementally* progress down a decision path from simple, high-level decisions to lower-level detail decisions based on the higher-level decisions. The flow from logical-to-physical ERs enables us to incrementally partition - decompose - complexity (Principle 4.17).

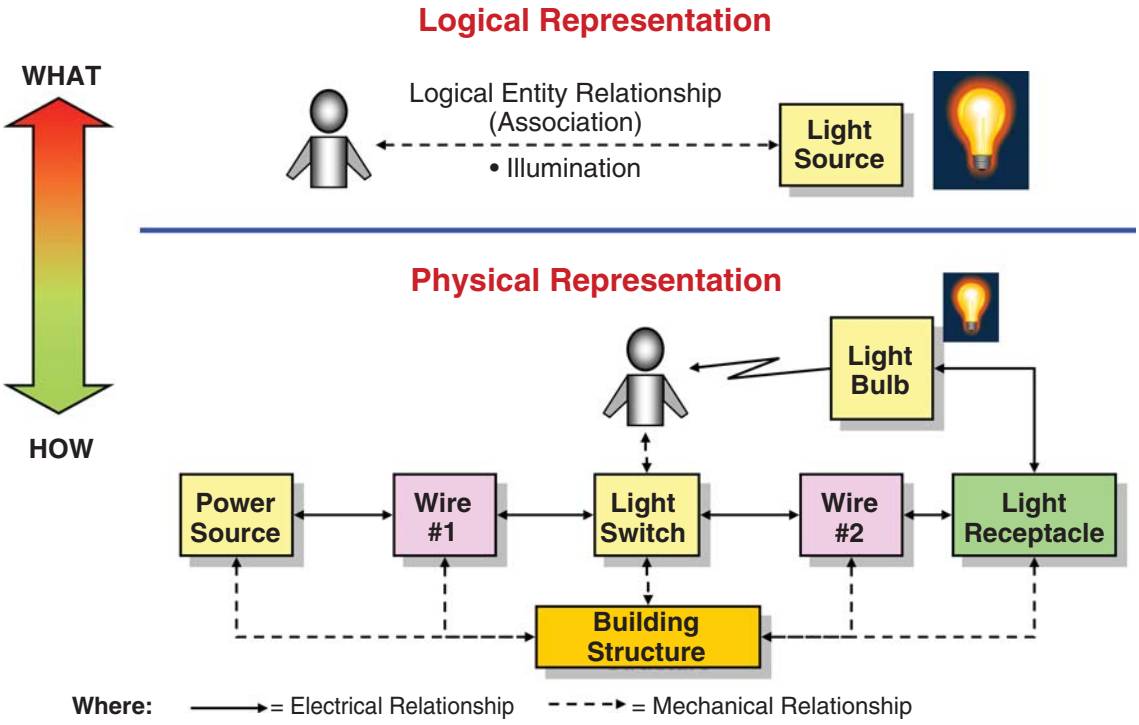


Figure 8.9 Translation of Logical ER into Physical ERs Example

Illustrations enable us to progress from simply acknowledging the existence of a relationship to detailed decisions regarding how the logical ER can be physically implemented.

In summary, we evolved the *logical* architecture representation of the Room Lighting System at the top of Figure 8.9 from the abstract – *what* - to the detailed *physical* architecture representation shown in the lower portion of Figure 8.9 – *how*.

8.6 ARCHITECTURAL SYSTEM ELEMENT CONCEPTS

Every HUMAN SYSTEM—Enterprise and Engineered Systems—and natural systems are characterized by a structure and framework that supports and/or enables their compositional elements to provide capabilities to perform missions and survive. We refer to this integrated framework as the SYSTEM or ENTITY's Architecture.

8.6.1 Introduction to the System Elements

As abstractions, an SOI can be a MISSION SYSTEM (e.g., system, product, or service) integrated with its ENABLING SYSTEM(S) to interact with HIGHER-ORDER Systems - authoritative Command & Control (C2), etc. - and its OPERATING ENVIRONMENT. Each of these abstractions is comprised of analytical building blocks referred to as the System Elements. The System Elements, when integrated into an architectural framework (Figure 10.14), form the System Architecture that serves as a key construct for SE, analysis, and development.

If you simply observe systems and their interactions, you soon discover that successful system outcomes and performance require the integrated efforts of HUMAN SYSTEMS—Enterprises and Engineered Systems. The attributes consist of:

- A mission such as an Enterprise task with sufficient funding from HIGHER-ORDER Systems that serves as an enabler to authorize, perform, and accomplish work.
- User – operator or maintainer - to perform the mission or task by leveraging organizational tools such as system, product, or service assets or personnel to produce performance-based outcome results.
- The *right* tool available at the *right* time—Enterprise system, product, or service asset—required to perform the mission or task.
- Processes and methods that enable the User to *safely*, efficiently, and effectively employ the tool to accomplish the task without adverse or catastrophic consequences and lasting effects such as injury to the User, damage to the tool, impacts to public safety, or the damage to the environment.
- A facility or location that is conducive to performing the mission or task.

If we perform a Domain Analysis of these attributes, accomplishment of an Enterprise mission or task requires the following: mission resources, personnel, equipment, procedural data, system responses, and facilities. We designate these categories as *System Elements* and use SMALL CAPS to represent them. To facilitate understanding of System Elements, consider the following example.

Example 8.4

System Element Examples

Applying the System Elements of the Car-Driver System here are some examples of each:

- PERSONNEL Element – Driver and Passengers
- EQUIPMENT Element – Automobile
- MISSION RESOURCES Element – Mission, fuel, fluids, maps, radio, etc.
- PROCEDURAL DATA Element – Automobile instruction manual
- SYSTEM RESPONSES Element – Proactive travel and defensive driving
- FACILITIES Element – Home garage, automobile dealership, repair shops.

Author's Note 8.3

Observe the presence of the SYSTEM RESPONSES Element. Unfortunately, Engineers think of *system responses* as something they measure with a digital voltmeter, oscilloscope, or other type of instrumentation. That is true; traditional Engineering outcomes should be observable, measurable, verifiable, etc. However, *how do you measure an external system response such as a human that has no intention of producing a response*? The reality is: HUMAN SYSTEMS *may or may not* produce an output to avoid detection, survival, and so forth. Since the scope of Systems Engineering decision making encompasses more than traditional Engineering (Figure 1.2), SYSTEM RESPONSES are purposely elevated as a System Element to ensure the proper level of consideration.

The previous System Elements example reveals a need to refine the application of the System Elements to delineate the following: (1) The Car–Driver System traveling to work as a MISSION SYSTEM versus (2) the car requiring periodic *preventive maintenance* by an ENABLING SYSTEM such as a car dealership. In this context, the Car–Driver system does not require the Facilities Element to travel to work. As a result, we refine the application of the System Elements to make the distinction shown in Table 8.1.

TABLE 8.1 System Elements Comparison: Mission System versus Enabling System

Mission System—System Elements	Enabling System—System Elements
Personnel Element	Personnel Element
Equipment Element	Equipment Element
• Hardware	• Hardware
• Software	• Software
Mission Resources Element	Mission Resources Element
Procedural Data Element	Procedural Data Element
System Responses Element	System Responses Element
	Facilities Element

From the perspective of a Mission System, the Facilities Element is unique to Enabling System roles. For example, an aircraft performing as a Mission System does not require a Facility to fly through the air from one city to another. In general, we tend to think of facilities as buildings. However, as we shall define in detail later, Facilities include not only buildings but also vehicles and structures such as platforms or frameworks that may or may not be exposed to the weather conditions. To illustrate this point, consider the following example.

Example 8.5

Airport Hanger as an Enabling System Facility for an Aircraft

An airline may designate aircraft hangars in certain cities as an Enabling System Facilities Element to perform major inspection and maintenance actions on Mission System aircraft. The Facility provides protection for maintenance operations during all types of weather conditions.

Likewise, an airport terminal serves as an Enabling System Facilities Element to service an aircraft between flights, load/unload passengers and cargo, and so forth.

Example 8.6

Tanker Aircraft as n Enabling System Facility for Mid-Air Refueling

In performing its Mission System role, a military jet fighter requires mid-air refueling from a fuel tanker aircraft that serves as an Enabling System Facilities Element.

Example 8.7

Aircraft Wing as an Enabling System Facility for Mounting Components

An aircraft's wing structure serves as an Enabling System Facilities Element for an aircraft Flight Control System (FCS) actuator developed by a vendor installation on the aircraft.

8.6.2 System Element Descriptions

Based on the preceding introduction to the System Elements for an SOI's Mission System(s) and Enabling System's, let's scope what is included in each one.

8.6.2.1 Personnel System Element The Personnel System Element (1) consists of all human roles required to perform the Mission System operations in accordance with Standard Operating Practices and Procedures (SOPPs) and (2) has overall accountability for accomplishing mission objectives assigned by Higher-Order Systems:

- **Mission System Personnel Roles** include all personnel directly required to operate the Mission System and accomplish its mission objectives. In general, these personnel are typically referred to as System Operators such as a pilot/copilot, driver, etc.
- **Enabling System Personnel Roles** include personnel such as System Maintainers who perform tasks such as maintenance, supply support, training, publications, security, and other activities.

8.6.2.2 Equipment System Element The Equipment System Element consists of any physical, multi-level, electromechanical, optical, or other types of physical systems. Equipment examples include Products, Subsystems, Assemblies, Subassemblies, and Parts shown earlier in Figure 8.4. Observe that the inference here is physical hardware, not necessarily true. The Equipment Element does (1) consist of hardware and (2) *may or may not* consist of software. For example, a simple garden shovel - hardware – does not require software for use.

The ultimate success of the Mission System requires that the Equipment Element be operationally *available* (Chapter 34) and fully capable of supporting the system missions and the safety of its Personnel to ensure a level of success. As a result, Specialty Engineering disciplines such as reliability, availability, maintainability, vulnerability, survivability, safety, and human become a key focus of the Equipment Element. Depending on the application, the physical requirements of the Equipment Element may require (1) a *fixed* or *permanent* structure such as buildings, (2) *transportability* such as heavy construction equipment, (3) *maneuverability* such as an automobile or aircraft, (4) *mobility* such as a trailer or cart, or (5) *portability* such as a smartphone.

To better understand the composition of the Equipment Element, let's explore its constituent hardware and software components.

8.6.2.3 The Hardware System Element The Hardware Element consists of the integrated set of

physical, multi-level components - mechanical, electrical/electronic, or optical—configured in accordance with the System Architecture. Whereas the HARDWARE Element is common to both MISSION SYSTEMS and ENABLING SYSTEMS, there are differences in the classes of its components:

- **MISSION SYSTEM Hardware Components** are physically integrated to provide the operational capabilities required to accomplish mission objectives.
- **ENABLING SYSTEM Hardware Components** consist of tools – systems and devices - required to deploy, support, and retire or dispose of the MISSION SYSTEM. Tools, for example, are categorized by military organizations as:
  - Common Support Equipment (CSE)
  - Peculiar Support Equipment (PSE)

8.6.2.3.1 CSE CSE consists of the items required to deploy, support, maintain, and retire the system or portions of the system while not directly engaged in the performance of its mission. CSE items, which are typically commercially available or owned by the User, consists of items such as hammers, screwdrivers, diagnostic equipment, data loggers and analyzers. CSE excludes overall planning, management, and task analysis functions inherent in the Work Breakdown Structure (WBS) element, SE/Program Management.

8.6.2.3.2 PSE MIL-HDBK-881 characterizes PSE as "… the design, development, and production of those deliverable items and associated software required to support and maintain the system or portions of the system while the system is not directly engaged in the performance of its mission, and which are not CSE" (MIL-STD-881C, p. 240).

PSE includes:

- "Vehicles, equipment, tools, etc., used to fuel, service, transport, hoist, repair, overhaul, assemble, disassemble, test, inspect, or otherwise maintain mission equipment.
- Any production of duplicate or modified factory test or tooling equipment delivered to the (Acquirer) for use in maintaining the system. (Factory test and tooling equipment initially used by the contractor in the production process but subsequently delivered to the (Acquirer) will be included as cost of the item produced.)
- Any additional equipment or software required to maintain or modify the software portions of the system" (MIL-STD-881C, p. 240).

8.6.2.4 Components Common to CSE and PSE CSE and PSE each employ two categories of EQUIPMENT that are common to both types: (1) Test and Measurement and Diagnostics Equipment (TMDE) and (2) Support and Handling Equipment.

8.6.2.4.1 TMDE In MIL-HDBK-881's characterization, Test, Measurement, and Diagnostics Equipment (TMDE) "… consists of the peculiar or unique testing and measurement equipment which allows an operator or maintenance function to evaluate operational conditions of a system or equipment by performing specific diagnostics, screening or quality assurance effort at an organizational, intermediate, or depot level of equipment support" (MIL-STD-881C, p. 228).

TME, for example, includes:

- "Test Measurement and Diagnostic Equipment (TMDE), precision measuring equipment, Automatic test equipment, manual test equipment, automatic test systems, test program sets, appropriate interconnect devices, automated load modules, taps, and related software, firmware and support hardware (power supply equipment, etc.) used at all levels of maintenance.
- Packages which enable line or shop replaceable units, printed circuit boards, or similar items to be diagnosed using automatic test equipment" (MIL-STD-881C, p. 228).

8.6.2.4.2 Support and Handling Equipment Support and Handling Equipment consists of the deliverable tools and handling equipment used for support of the MISSION SYSTEM. This includes "... Ground Support Equipment (GSE), vehicular support equipment, powered support equipment, unpowered support equipment, munitions material handling equipment, material-handling equipment, and software support equipment (hardware and software)" (MIL-STD-881C, p. 228).

8.6.2.5 The SOFTWARE System Element The SOFTWARE Element consists of all software code (source, object, executable, etc.) and documentation required for installation, operation, and maintenance of the EQUIPMENT Element. You may ask why some Enterprises separate the SOFTWARE Element from its EQUIPMENT Element. There are several reasons:

- EQUIPMENT Element HARDWARE and SOFTWARE may be developed separately or procured from different vendors.
- SOFTWARE may provide the flexibility to alter system capabilities and performance (decision making, behavior, etc.) without having to physically modify the EQUIPMENT Element HARDWARE, assuming the current design is adequate.

Author's Note 8.4

Two key points:

1. This text defines the EQUIPMENT Element as consisting of integrated Hardware and Software as subordinated, supporting elements. Some Enterprises treat the SOFTWARE Element as a peer to the EQUIPMENT Element, which is assumed to be hardware. That is technically incorrect. In such cases, the HARDWARE is useless without the SOFTWARE and vice versa. Their integrated capabilities form the EQUIPMENT Element.
2. Engineers typically become prematurely focused with hardware and software details (Figure 2.3) long before higher level SOI decisions have been made—namely EQUIPMENT Element requirements. EQUIPMENT Element decisions lead to lower level HARDWARE and SOFTWARE use cases and decisions. These decisions subsequently lead to a key system trade-off: *What capabilities should be implemented in Hardware versus those implemented in Software?*

EQUIPMENT Element HARDWARE and SOFTWARE may be separately procurable items. The underlying philosophy is Software, as a System Element, should be isolated to accommodate modification without necessarily having to modify the HARDWARE.

Application-specific SOFTWARE can be procured as a separate item, regardless of its position within the system structure as long as a *Software Requirements Specification (SRS)* requirements have been allocated and derived from the higher level EQUIPMENT Elements specification requirements. As new versions of application-specific SOFTWARE are released, the User can procure the item without modifying the EQUIPMENT. There may be exceptions, however, where externally driven Enterprise performance objectives, product obsolescence, new technologies, and priorities inevitably force it to upgrade the computer Hardware capabilities and performance to meet those requirements.

8.6.2.6 The PROCEDURAL DATA System Element The PROCEDURAL DATA System Element consists of documentation that specifies how to safely operate, maintain, deploy, and store the EQUIPMENT Element. In general, the PROCEDURAL DATA Element represents procedures or a manufacturer's instructions such as User Guides or Operator Manuals that specify the safe operation of the EQUIPMENT to achieve its intended level of performance for a prescribed OPERATING ENVIRONMENT.

The PROCEDURAL DATA Element includes items such as Enterprise roles and missions, operating constraints, reference manuals, operator guides, Standard Operating Practices and Procedures (SOPPs), and checklists.

Author's Note 8.5

The Value of Checklists

Unfortunately, people often view checklists as bureaucratic nonsense, especially for Enterprise processes. Remember, checklists incorporate lessons learned and best practices that keep you out of trouble. Checklists are a state of mind - you can view them as "memory joggers" to perform an action or as a "reminder" to "think about what you may have overlooked." As a colleague notes, when landing an aircraft, if the checklist says to "place the landing gear in the deployed and locked position" on landing, you may want to at least consider putting the landing gear down *before* you land! Bureaucratic or not, the consequences for a lack of compliance can be catastrophic!

8.6.2.7 MISSION RESOURCES System Element The MISSION RESOURCES System Element encompasses *real-time* and *nonreal-time* mission data in various media, fluids, lubricants, materials, energy, and so on. Examples include those that are necessary for performing a mission with a specified level of success. MISSION RESOURCES are categorized in terms of *consumables* and *expendables* required to support the MISSION SYSTEM or ENABLING SYSTEM during its missions.

8.6.2.7.1 Consumable Mission Resources *Consumable mission resources* consist of physical entities that are (1) ingested, converted, or input into a processor such as a combustible engine, solar converter, and so forth to transform these into energy or (2) used to *replenish* existing resources. Examples include items such as:

- Food, water, medicines for the PERSONNEL Element
- Fuel, water, lubricants, and fluids for the EQUIPMENT Element

8.6.2.7.2 Expendable Mission Resources *Expendable mission resources* consist of physical entities that are used, read, and disposed of during the course of mission operations or after a mission. Examples include:

- Automobile—air, oil, and transmission filters; windshield wiper blades; light bulbs; tires.
- Military aircraft—missiles.
- Data resources such as hardcopy or electronic mission information that enable PERSONNEL and EQUIPMENT to successfully plan and conduct the mission based on "informed" decisions. Examples include mission tasks, tactical plans, Mission Event Timelines (METs), cargo manifests, intelligence, previously recorded mission data, commands, data messages, navigational data,

weather conditions and forecasts, situational assessments, telecommunications, telemetry, and synchronized time.

8.6.2.8 **SYSTEM RESPONSE** ***System Element*** Every NATURAL and HUMAN SYSTEM, as a stimulus–response mechanism, responds *internally* or *externally* to stimuli, excitations, or cues originating from external systems in its OPERATING ENVIRONMENT. The responses may be:

- *Explicit* such as reports, communications, and altered behavior
- *Implicit* such as mental thought strategies, lessons learned, and behavioral patterns

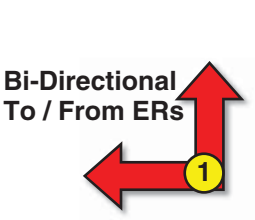
SYSTEM RESPONSES occur in a variety of forms that we characterize as behavioral patterns, products, services, and by-products throughout the system's Pre-Mission, Mission, and Post-Mission Phases of Operation. So, *what do we mean by system behavior, products, services, and by-products*?

- **System behavior** consists of system responses based on a plan of action or physical stimuli and audiovisual cues such as threats or opportunities. The stimuli, excitations, and cues invoke system behavioral patterns or actions that may be categorized as aggressive, benign, defensive, and everywhere in between. Behavioral actions include strategic and tactical tactics and countermeasures.
- **System products** include any type of physical outputs, characteristics, or behavioral responses to planned and unplanned events, external cues, or stimuli.
- **System by-products** include any type of physical system output such as heat, exhaust emissions, and thermal signatures or behavior that is not deemed to be a system, product, or service.
- **System services** consists of any type of system behavior, excluding physical products, that assist another entity in the conduct of its MISSION SYSTEM role.

8.6.3 Conceptualizing System Element Interactions

One approach to identifying System Element interactions is to create a simple matrix such as the one shown in Figure 8.10. For illustration purposes, each cell of the matrix represents *interactions* between the row and column System Elements. For your system, employ such a scheme and document the interactions in the *System Architecture Description*. Then, baseline and release this document to promote communications among team members developing and making System Element decisions.

The matrix in Figure 8.10 simply enables us to establish *who interacts with whom* within the set of System Elements. With skill and experience, one can jump to creating an Architecture Block Diagram (ABD) to illustrate the interactions. The challenge is that valuable time is wasted pushing boxes



System Element	Personnel	Equipment	Procedural Data	Mission Resources	System Responses	Facilities
Personnel	1	2	3	4	5	6
Equipment	7	8	9	10	11	12
Procedural Data	13	14	15	16	17	18
Mission Resources	19	20	21	22	23	24
System Responses	25	26	27	28	29	30
Facilities	31	32	33	34	35	36

Where: (X) = Entity Relationships (ERs) and associated capabilities

Figure 8.10 Simple Matrix Approach to Identifying ERs Between System Elements

and lines around a page instead of focusing on the *substantive content* and getting agreement among your colleagues.

One method for overcoming this situation requires the introduction of a tool referred to as the N2 Diagram as shown in Figure 8.11. The N2 Diagram is a very powerful but simple graphical representation of an N × N matrix of a set of entities. Lano (1977, p. 244–271) introduced the tool in a paper written in 1977. The tool consists of a virtual row–column matrix with the entities distributed in a diagonal from left to right. The diagonal easily accommodates external inputs – *acceptable* and *unacceptable* (Figure 3.2) to enter any of the entities from the left side of the graphic and outputs – *acceptable* and *unacceptable* (Figure 3.2) to exit out the right side.

Some people expand concept based on the *Integrated Definition for Function Modeling (IDEF0)* construct shown in Figure 8.12. The construct (IDEF0, 1993, Figure 3, p. 12) illustrates constraints entering downward from the top of each entity and resources entering from the bottom of the entity.

The N2 Diagram can be created as part of a presentation using a spreadsheet or using tool that has the capability. Engineers often create gigantic N2 Diagrams for highly complex system that have small fonts, require an entire office wall to post, and are extremely difficult to read. To mange complexity, a rule of thumb based on managerial "span of control" concepts is to limit the quantity of entities in an architecture for a given level of abstraction to a maximum of 6–8. This is more manageable and keeps everyone focused.

The office wall approach, which is sometimes justified, becomes so overwhelming with the amount of information that it results in a case of proverbial "can't see the forest for the trees." This violates the intent of being able to focus on a substantive review of the ERs and architecture.

An N2 Diagram is generally used as an analytical working paper that provides the basis that leads to the creation of the System Element Architecture (SEA) Construct for an entity shown in Figure 8.13.

8.6.4 Importance of the System Elements Concept

The System Elements concept taxonomy (Tables 8.1 and 8.2) is important for three reasons:

- First, the System Elements enable us to organize, classify, and bound SYSTEM and ENTITY abstractions and their interactions. That is, it is a way to differentiate *what is* versus *what is not* included in the system.
- Second, the SEA establishes a common framework for developing the *logical* and *physical* system architectures of each entity within the system hierarchy.
- Third, the System Elements serve as an initial starting point construct for allocations of multi-level performance specification requirements.

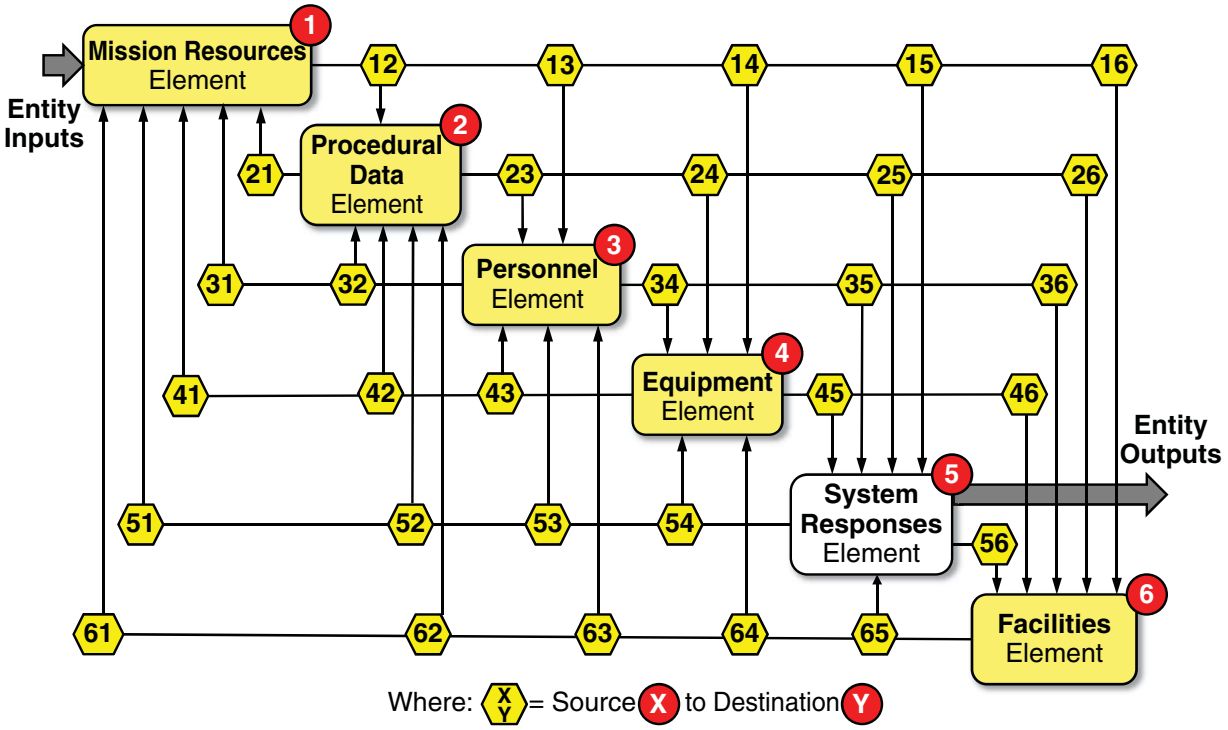


Figure 8.11 N x N (N2) Diagram Illustrating System Element ERs and Interface Identifiers

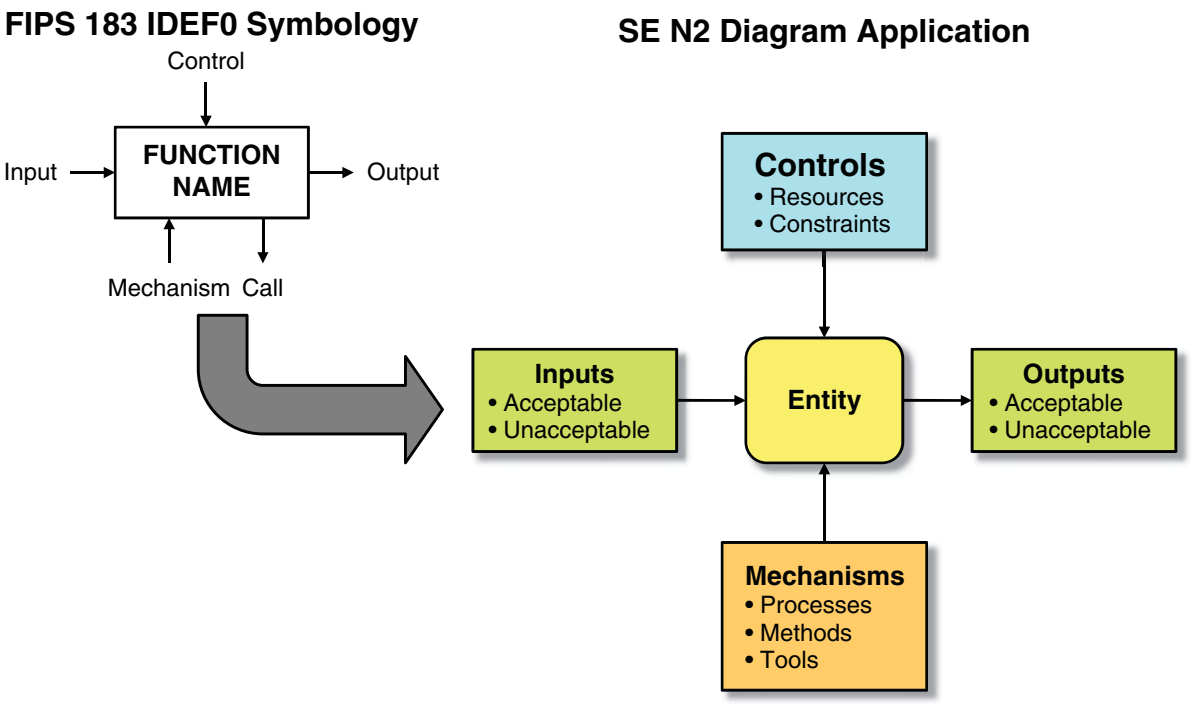


Figure 8.12 Integrated Definition for Function Modeling (IDEF0) Construct. (Source: IDEF0 (1993), Figure 3, p. 12.)

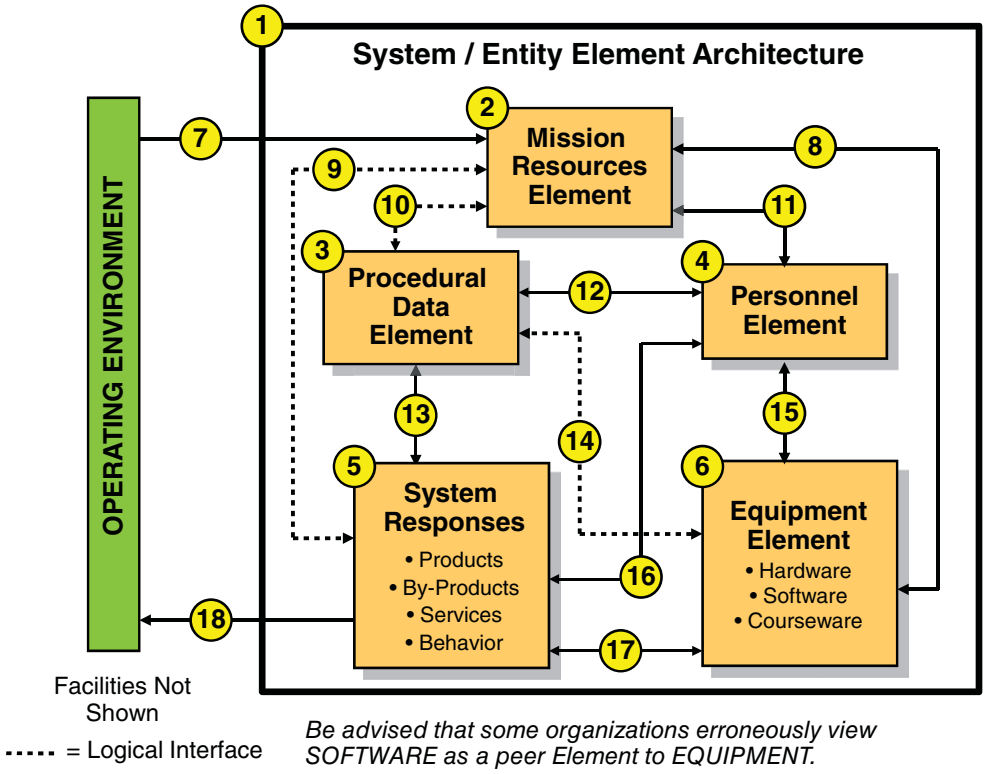


Figure 8.13 System Element Architecture (SEA) Construct

TABLE 8.2 System Entity Decomposition and Integration Guidelines

Level or Tier	Nomenclature	Entity Decomposition/Integration Guidelines (Figure 8.7)
0	User's SYSTEM Level	The User's System is bounded by its Enterprise or *organizational missions* and consists of one or more SOIs required to accomplish that mission within its OPERATING ENVIRONMENT.
1	SYSTEM Level	Each instance of a SYSTEM consists of at least two or more instances of SEGMENT, PRODUCT, SUBSYSTEM, ASSEMBLY, SUBASSEMBLY, or PART Level entities or combinations thereof.
2	SEGMENT Level Entity	If the SEGMENT Level of *abstraction* or *class* is applicable, each SEGMENT Level entity consists of at least two or more instances of PRODUCT, SUBSYSTEM, ASSEMBLY, SUBASSEMBLY, or PART Level entities or combinations thereof.
3	PRODUCT Level Entity	If the PRODUCT Level of *abstraction* or *class* is applicable, each instance of a PRODUCT Level entity consists of at least two or more instances of SUBSYSTEM, ASSEMBLY, SUBASSEMBLY, or PART Level entities or combinations thereof.
4	SUBSYSTEM Level Entity	If the SUBSYSTEM Level of *abstraction* or *class* is applicable, each instance of a SUBSYSTEM Level entity consists of at least two or more instances of ASSEMBLY, SUBASSEMBLY, or PART Level entities or combinations thereof.
5	ASSEMBLY Level Entity	If the ASSEMBLY Level of *abstraction* or *class* is applicable, each instance of an ASSEMBLY Level entity consists of at least two or more instances of SUBASSEMBLY or PART Level entities or combinations thereof.
6	SUBASSEMBLY Level Entity	If the SUBASSEMBLY Level of *abstraction* or *class* is applicable, each instance of a SUBASSEMBLY Level entity must consist of at least two or more instances of PART Level entities.
7	PART Level Entity	The PART Level is the lowest decompositional element of a system.

Despite strong technical and analytical skills, Engineers are sometimes poor organizers of information. Therein lies a fundamental problem for the "Engineering of systems." Being able to understand, frame, and structure the problem is 50% of the solution. The framework of the SEA Construct in Figure 8.13 provides the framework for defining the system and its boundaries.

The challenge in analyzing and solving System Development and Engineering problems is being able to identify, organize, define, and articulate the *relevant* elements of a problem (objectives, initial conditions, assumptions, etc.) in an easy-to-understand, intelligible manner that enables us to conceptualize and formulate the solution strategy. Establishing a standard analytical framework and its interfaces followed by decomposition and refinement of the architecture enables traditional Engineering to apply "plug and chug" mathematical and scientific principles, the core strength of Engineering education and training.

8.6.5 Developing the SEA

Based on an analysis of the interactions among the System Elements from the matrix in Figure 8.10, we can create the SEA construct shown in Figure 8.13. This construct serves as a "headwaters" template for developing systems, product, or services. Whereas most ad hoc architectures reflect personal experiences, you can begin with this simple template. Chapter 10 will discuss application of the SEA to the SOI and external systems within its *O*PERATING *E*NVIRONMENT.

8.6.6 Integrating the System Levels of Abstraction and System Elements Concepts

The preceding discussion of the System Elements implies that an SOI's MISSION SYSTEM and ENABLING SYSTEM(S) are comprised of PERSONNEL, EQUIPMENT, MISSION RESOURCES, PROCEDURAL DATA, SYSTEM RESPONSES, and FACILITIES System Elements. From the perspective of *what is required* for a system, this is true. However, some systems may not require PERSONNEL. Consider the following examples.

Example 8.8

Some systems such as a commercial aircraft performing its MISSION SYSTEM role require an on-board PERSONNEL Element - flight crew consisting of pilots, flight attendants, et al. Other systems such as a sending a planetary space probe and rover to Mars as a MISSION SYSTEM require a PERSONNEL Element, not on-board the spacecraft, but on Earth as part of the rover's ENABLING SYSTEM.

When the PERSONNEL Element is required, it may occur at the higher levels of abstraction such as SYSTEM, PRODUCT, or SUBSYSTEM Levels. For example, an Enterprise – SYSTEM Level - such as a TV station consists of a Control Room—SUBSYSTEM Level— that consists of numerous integrated PERSONNEL–EQUIPMENT Element workstations - ASSEMBLY Level, each as a *performing entity*, working to achieve their respective missions.

Observe that we said Control Room SUBSYSTEM and ASSEMBLY Level PERSONNEL-EQUIPMENT workstations. However, *avoid* being *lulled* into believing that the SUBSYSTEM Level or the ASSEMBLY Level by virtue of their connotations are exclusively HARDWARE. Not true! EQUIPMENT and its constituent HARDWARE and Software when integrated and operated by a human – PERSONNEL Element – serve as a *performing entity* that may be designated as a SUBSYSTEM, ASSEMBLY.

8.6.6.1 *Performing Entities Concept*

Principle 8.2

System *levels of abstraction*—System, Product, Subsystem, etc.—within each represent integrated sets of *performing entities* that (1) include Equipment—Hardware and/or Software, Resources, Procedural Data, System Responses and (2) *may or may not* include the Personnel or Facilities Elements

The challenge for SEs is *how do you know that the PERSONNEL Element is required*? The answer resides in understanding the concept of a *performing entity* that requires (1) EQUIPMENT Element only, PERSONNEL Element only, or the integration of PERSONNEL–EQUIPMENT Elements. As we will discuss in Chapter 24, a key trade-off occurs for decisions concerning system implementation—what the PERSONNEL Element does best versus what the EQUIPMENT does best (Figure 24.14)—as well as operational cost considerations. The key decision is: *what is the appropriate mix of PERSONNEL–EQUIPMENT tasks that results in achievement of the required performance for the least cost for development and life cycle Total Cost of Ownership (TCO)*?

To illustrate how performing entities can occur at any Level of Abstraction, Figure 8.14 illustrates the relationship of System Levels of Abstraction and the System Elements. On the left side, we have the SYSTEM Level that might represent hierarchical decomposition of a MISSION SYSTEM or an ENABLING SYSTEM. Each entity at each Level of Abstraction may include one or more of the System Elements.

Referring to Figure 8.14, entities at each Level of Abstraction might be comprised of some or all of the System Elements. However, in general, the PERSONNEL Element is typically limited to the PRODUCT or SUBSYSTEM Levels, especially in control center applications depending on complexity.

Author's Note 8.6

Recognition of the *performing entities* concept at various levels of abstraction is a key discussion point later for Chapter 21. For example, there are different ways of filling a *solution space*

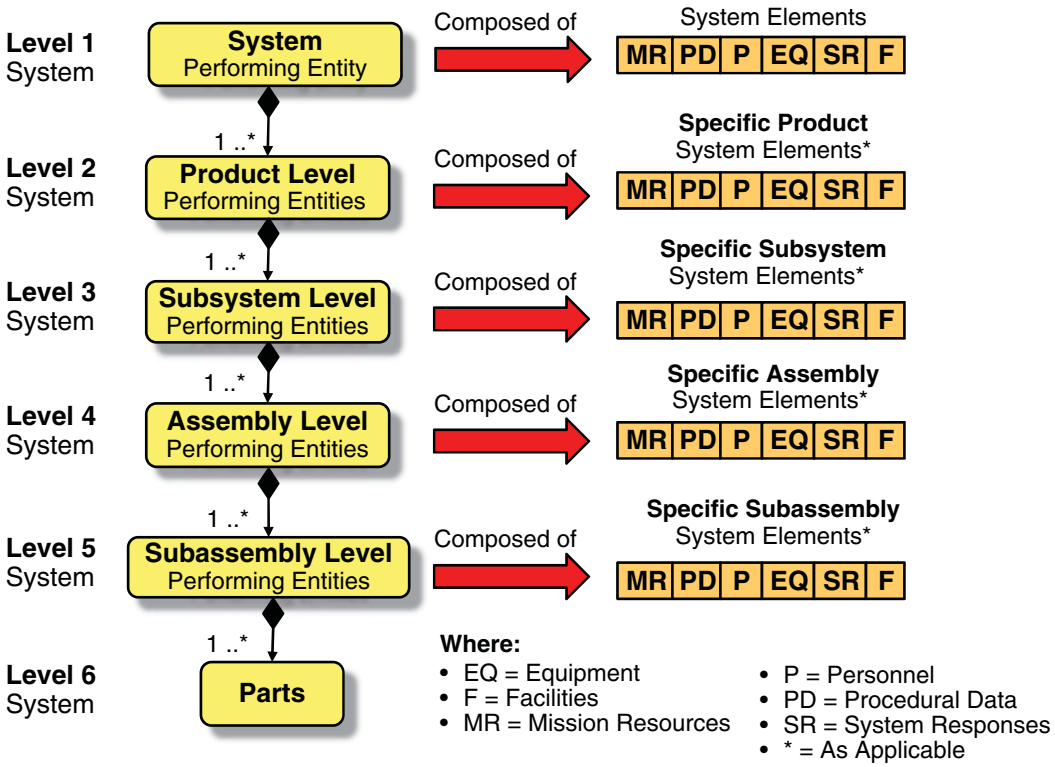


Figure 8.14 System Levels of Abstraction Depicting Composition of System Elements at Each Level

such as (1) automated Equipment only, (2) Personnel only, (3) Personnel manually operating the Equipment, or (4) Personnel operating automated or semi-automated Equipment. When you specify a System Performance Specification (SPS) or Entity Development Specification (EDS), you treat the *performing* System/Entity as an *object* (Figures 3.2 and 20.4) and avoid specifying how the *solution space* is to be implemented physically as noted in list items 1 — 4 above.

Based on the preceding paragraph, Principle 8.2 may appear to be obscure. Let's clarify:

- **Enterprise Systems**
  - Require all of the System Elements—Mission Resources, Procedural Data, Personnel, Equipment - Hardware and possibly Software, System Responses, and Facilities. The Equipment Element—owned, rented, or leased—could be as simple as office desks and chairs, office copier, FAX, telephone, desktop computers, etc.
- **Engineered Systems**
  - Require Mission Resources, Procedural Data, Equipment, and System Responses. Equipment such as machined or mass-produced hand tools *do not* require Software; however, computers require Software.
  - Typically depend on Enabling Systems to provide the Personnel and Facilities Elements. Recall our earlier discussion of Figure 8.1 and the importance of delineating whether the User – operator or maintainer – is considered part of the system.

Race Car-Driver Mission System and Pit Crew Enabling System as Integrated Performing Entities

Example 8.9 To better illustrate the performing entities concept, consider the illustration shown in Figure 8.15 (Christie, 2014). The figure illustrates two high level performing entities: (1) a Race Car-Driver Mission System and (1) its Pit Crew Enabling System. The Mission Event Timeline (MET) for servicing the Race Car-Driver System is 3.5 seconds (Kolwell, 2013). Think about it! This includes jacking the vehicle, changing tires, checking and replenishing lubricants and coolants, fueling the vehicle, cleaning windows, et al in 3.5 seconds or less. This is not simply a case of training the Pit Crew to work faster. The racecar's Equipment design will ultimately limit how fast the Pit Crew can perform *preventive* and *corrective maintenance* actions (Chapter 34).

SE of the vehicle must factor in this MET in the design of the race car and its components – wheels, lift points and devices, et al – to ensure completion of the maintenance the tasks in 3.5 seconds. Each interaction between the Personnel – tire changers, jack men, et al; Equipment - jacks, air wrenches, et al; Mission Resources – ties, air, et al; Procedural Data – racing rules; Facilities – bounded work safety areas; and System Responses – completed maintenance actions on time - are critical for success. Figure 8.15 illustrates and exemplifies the concept of *performing entities* depicted in Figure 8.14.

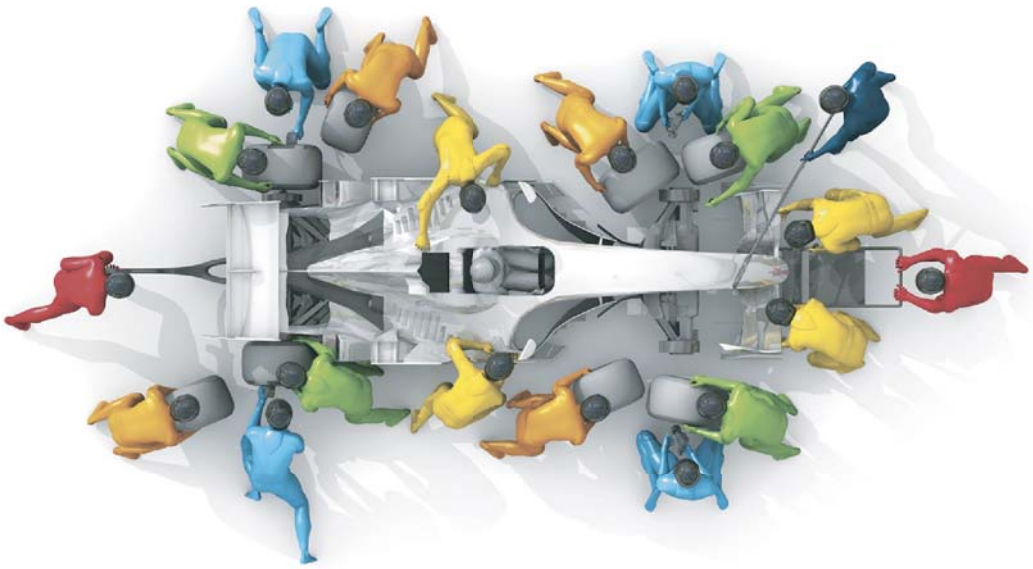


Figure 8.15 Race Car Pit Crew – Performing Entities Example. (Source: Bryan (2014).)

8.7 CHAPTER SUMMARY

In summary, this chapter provided the details of the System Elements introduced in Chapter 3. Our discussions:

- Defined, scoped, and provided details for each of the System Elements—Mission Resources, Procedural Data, Personnel, Equipment, System Responses, and Facilities.
- Introduced the concept of *logical* and *physical* relationships as a basis for overcoming the *ad hoc, endless* loop Plug and Chug … SBTF-DPM Engineering Paradigm (Chapter 2).
- Introduced a matrix and N x N (N2) Diagram approach to facilitate definition of logical and physical relationships among the System Elements.
- Introduced the concept of an SEA Construct that serves starting point for architecting an SOI, Mission System(s), Enabling System(s), or the Equipment Element.
- Introduced the concept of *performing entities* as an abstraction without regard to their implementation via Personnel, Equipment, or integrated Personnel-Equipment Systems.

Given this foundation, we are now ready to shift our discussions to higher level systems – SOI Mission Systems and Enabling Systems integration and interactions with external system in their Operating Environment.

8.8 CHAPTER EXERCISES

8.8.1 Level 1: Chapter Knowledge Exercises

Answer each of the questions listed in "What You Should Learn from This Chapter" section:

1. What graphical tool enables us to delineate what is within the scope of a system, product, or service's boundaries?
2. What graphical tool enables us to investigate the interrelationships among a set of system entities?
3. What is a Context Diagram? What is its purpose and how is it used?
4. Assume you have been tasked to describe a context diagram to someone unfamiliar with it. Write a brief description of its elements, how to construct it, and what information should be depicted.
5. What is a System Element?
6. What are the key components of the Equipment Element? Give examples.
7. What are the key components of the Mission Resources Element? Give examples.
8. What are the key components of the Procedural Data Element? Give examples.
9. What are the key components of the Personnel Element? Give examples.
10. What are the key components of the Facilities Element? Give examples.
11. What is CSE? Give examples.
12. What is PSE? Give examples.
13. What is TME? Give examples.
14. What is Support and Handling Equipment (CSE)? Give examples.
15. How do the System Elements relate to the System Architecture?
16. What is the System Element Architecture (SEA) Construct? What purpose does it serve? Where is it applied?
17. Should the Software Element be separate from the Equipment Element?
18. What types of systems comprised an SOI?
19. What are the key System Elements of an SOI's Mission System?
20. What are the key System Elements of the SOI's Enabling System?
21. What is an N2 Diagram? What is its purpose and how is it used? What problem(s) does it solve for an SE?
22. Create an N2 Diagram for each of the following Mission Systems using Figure 8.11 as a template create identifiers for each interface. Annotate each box with bulleted contents.
    a. Fast food restaurant
    b. Desktop computer system
    c. Enterprise organization
23. Using information from Exercise 22, create an alphabetized listing by Interface ID that describes the interactions that occur across each interface by identifier.
24. Using information from Exercise 22, graphically create each system's architecture using the System Element Architecture Construct shown in Figure 8.13. Cross check interfaces and IDs of N2 Diagram with the SEA.
25. Using Figure 8.15 as a reference, develop a description of the Race Car-Driver Mission System and Pit Crew Enabling System based on *performing entity* teams.

Decompose the two systems into a graphical hierarchy of teams using Levels of Abstraction, identify the System Elements for each team including PERSONNEL roles, assign specific mission objectives - maintenance actions - to each team, and describe sequences of actions, and a notional MET required to prepare the race car to return to the race in 3.5 seconds.

8.8.2 Level 2: Knowledge Application Exercises

Refer to www.wiley.com/go/systemengineeringanalysis2e

8.9 REFERENCES

Christie, Bryan (2014), *Illustration: Formula One (F1) Pit Stop*, New York: Bryan Christie Designs.

IDEF0, (1993), *Integrated Definition for Function Modeling (IDEF0). Draft Federal Information Processing (FIPS) Standards Publication 183*, 1993 December 21, Figure 3, p 12.

KSC (2009), *Lightning Strike on Launch Pad 39A*, KSC-2008-3940, NASA Photo, July 10, 2009, Kennedy Space Center (KSC), FL: NASA. http://mediaarchive.ksc.nasa.gov/detail.cfm?mediaid=42077. Accessed 3/19/13.

Kolwell, K. C. (2013), *"Anatomy of an F1 Pit Stop: 0.03 Is the Magic Number," Car & Driver*, Ann Arbor, MI: Hearst Corporation http://blog.caranddriver.com/anatomy-of-an-f1-pit-stop-003-is-the-magic-number/. Access 3/17/14.

Lano, Robert J. (1977). *"The N2 Chart," TRW-SS-77-04, Reprinted:* System and Software Requirements Engineering. New York, NY: IEEE Computer Society Press, 1990.

MIL-HDBK-1908B (1999), *DoD Definitions of Human Factors Terms,* Washington, DC: Department of Defense (DOD).

Warwick, Graham and Norris, Guy (2010), "*Designs for Success: Calls Escalate for Revamp of Systems Engineering Process*", Aviation Week, Nov. 1, 2010, Vol. 172, Issue 40, Washington, DC: McGraw-Hill.

9

ARCHITECTURAL FRAMEWORKS OF THE SOI AND ITS OPERATING ENVIRONMENT

Engineers, especially SEs, often sit at a computer and create *ad hoc, incoherent* SYSTEM level architectures comprised of a *random* conglomeration of SUBSYSTEM, ASSEMBLY, SUBASSEMBLY, and PART Level components. Yes, PART Level components!

- You know you are in trouble when … someone creates a complex SYSTEM Level architecture and the diagram contains an Analog-to-Digital (A-D) integrated circuit device with a part number and pinouts for wire connections.

Unfortunately, this reality reflects a deficiency in Engineering education, not necessarily the Engineers. When Engineers lack bona fide SE & Development (SE&D) courses, they naturally gravitate to mimicking the so-called architectures of others that are perceived to know what they are doing. Attempts to create *coherent* system architectures often turn into brainstorming sessions that "reinvent the wheel" despite similarities with other projects. The results are often *unpredictable* driven by different views in a meeting to achieve a consensus based on "opinion polls" of the *informed* and *uninformed*.

The System Element Architecture (SEA) Construct introduced in Chapter 8 provides a simple "headwaters" template for creating an architecture that applies to any ENTITY at any Level of Abstraction. Since the construct reflects the primary System Elements that comprise many different types of architectures, it serves as an informational starting point to be tailored for a specific system.

Chapter 9 employs the SEA Construct as a "building block" that can be integrated into levels thereby providing a more coherent architecture. As a result, we can create multi-level system architectures frameworks for (1) the SOI, Mission System, Enabling Systems, and constituent entity Levels of Abstraction and (2) external systems within the SOI's OPERATING ENVIRONMENT. The resulting multi-level architectures enable us to identify, define, and control interfaces and assign project teams such as Integrated Product Teams, Product Development Teams.

9.1 DEFINITIONS OF KEY TERMS

- **Entity Relationship (ER)**—Refer to Chapter 8 Definition of Key Terms.
- **Hierarchical Interactions**—Actions between natural and authoritarian Command and Control (C2) systems that lead, direct, influence, or constrain MISSION SYSTEM and ENABLING SYSTEM actions, behavior, and performance. For analytical purposes, we aggregate these C2 systems into a single entity abstraction referred to as the HIGHER-ORDER SYSTEMS Domain.
- **HIGHER-ORDER SYSTEMS**—Systems that exhibit C2 authority over HUMAN SYSTEMS or NATURAL SYSTEMS that are governed by natural and physical forces and laws related to the sciences.
- **HUMAN SYSTEMS**—Systems created by humans such as Enterprises, organizations, or Engineered Systems – robots, remote control devices, an so forth -

System Engineering Analysis, Design, and Development: Concepts, Principles, and Practices, Second Edition. Charles S. Wasson.

Companion Website: www.wiley.com/go/systemengineeringanalysis2e

that perform chartered missions to accomplish specific outcomes and performance objectives. Enterprise or Enterprise systems include those accountable for an SOI and its missions or *external* systems – cooperative, benign, aggressive, or hostile - within the SOI's OPERATING ENVIRONMENT. These systems exercise hierarchical C2 over lower tier systems via "chain of command" authoritative structures, policies, and procedures and mission tasking; constitutions, laws, and regulations; public acceptance and opinion; and so on.

- **INDUCED ENVIRONMENT Systems**—Discontinuities, perturbations, or disturbances created when natural phenomenon and events occur or HUMAN SYSTEMS interact with the NATURAL ENVIRONMENT. Examples include thunderstorms, oil spills, Electromagnetic Interference (EMI), and so forth.
- **NATURAL ENVIRONMENT Systems**—All natural, non-human, living, atmospheric, and geophysical entities that comprise the Earth and celestial bodies.

9.2 APPROACH TO THIS CHAPTER

Our discussions in Chapter 9 focus on creating an all-encompassing, multi-level system architecture that enables us to analytically represent the SOI and its interactions with its OE. Figure 9.1 depicts a graphical model that serves as the analytical framework of our discussions. Observe that the graphic consists of:

- An Enterprise System consisting of HIGHER-ORDER SYSTEMS and the SOI, which consists of MISSION SYSTEM(s) and ENABLING SYSTEMS.
- An OE comprised of a HIGHER-ORDER SYSTEMS Domain and a Physical Environment Domain comprised of NATURAL SYSTEMS, Induced, and HUMAN SYSTEMS Environments.

Our discussion begins with the SOI and its architecture. Since the SOI has various types of ERs with external systems in its OE, we will introduce the concept of logical and physical ERs. We will conclude the chapter with a discussion of the OE's architecture.

9.3 INTRODUCTION TO THE SOI ARCHITECTURE

Systems, products, and services exist within the boundaries of its User's Level 0 System. The context of the User's Level 0 System could be a division, department, program, or project level.

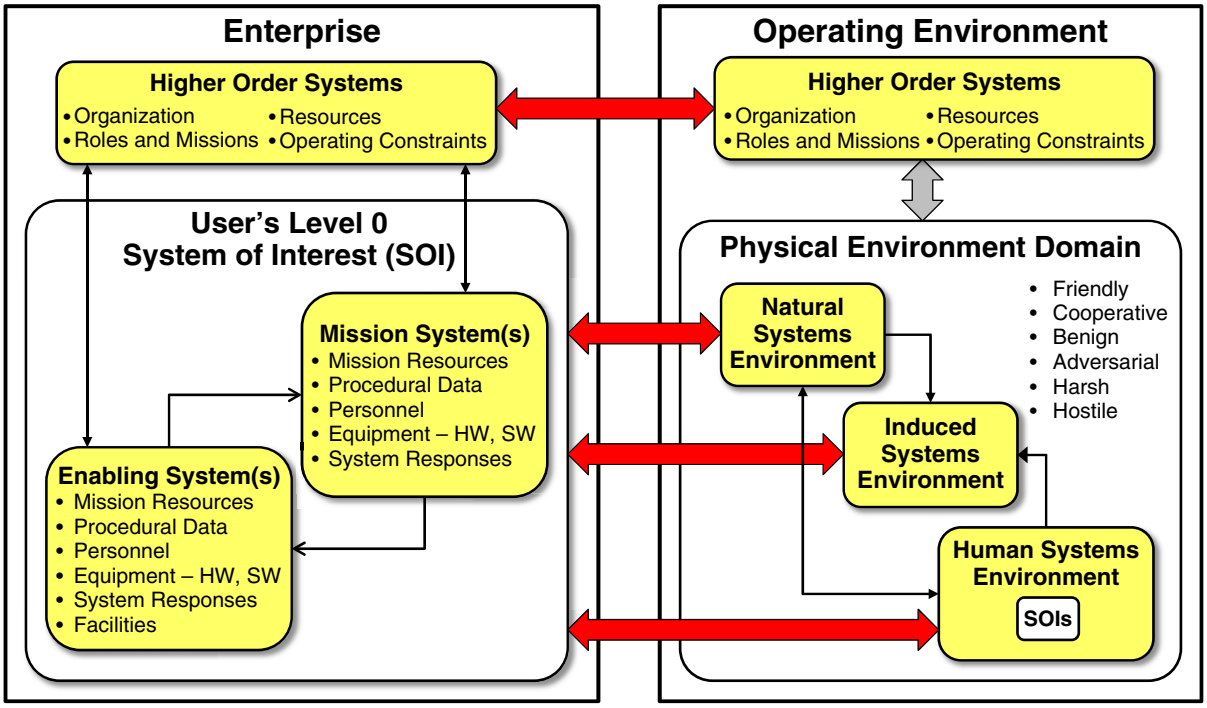


Figure 9.1 An Analytical Perspective of a System of Interest's (SOI) MISSION SYSTEM – ENABLING SYSTEM Interactions with External Systems in their OPERATING ENVIRONMENT (OE)

Within the User's Enterprise, there are HIGHER-ORDER SYSTEMS that exercise C2 over the various SOI assets. Each SOI consists of at least one or more MISSION SYSTEMS supported by one or more ENABLING SYSTEMS. Given this overview, let's begin by establishing the architecture of the SOI's MISSION SYSTEM and ENABLING SYSTEMS.

9.3.1 Developing the SOI Architecture

To create an SOI's architecture, the context of an SOI might be (1) a MISSION SYSTEM and ENABLING SYSTEM(S) that interact with each other, (2) one of the System Elements – PERSONNEL EQUIPMENT, and so forth, or (3) an ENTITY at any Level of Abstraction within the System Elements. Recall that every system performs two contextual roles (Figure 4.1): a MISSION SYSTEM (Producer) Role and an ENABLING SYSTEM (Supplier) Role to serve as a "Supply Chain" of contextual ENABLING SYSTEMS to perform missions.

We also know that:

- A system, product, or service consists of an integrated set of System Elements that collectively enable it to operate as a *performing entity*.
- The SEA construct provided in Figure 8.13 graphically integrates those System Elements into an Architecture Block Diagram (ABD) that serves as an architectural "building block" for any type of HUMAN SYSTEM—Enterprise or Engineered.

Therefore, we leverage the SEA's construct as an architectural building block to represent the architecture of the MISSION SYSTEMS and its ENABLING SYSTEM as shown in Figure 9.2.

The SOI's ENABLING SYSTEM architecture, like the MISSION SYSTEM, also employs the SEA as its architectural starting point. Due to restricted space, we have abstracted the ENABLING SYSTEM architecture in Figure 9.2 into a listing of these System Elements.

In summary, the SOI MISSION SYSTEM and ENABLING SYSTEMS exploit the SEA construct to create an analytical framework that provides *consistency* and *continuity*, which is much easier to understand, provides a simple way of developing architectures, and provides more complete coverage than the *ad hoc* Plug and Chug … SBTF Engineering Paradigm.

During Pre-Mission, Mission, and Post Mission operations, the MISSION SYSTEM and ENABLING SYSTEM are integrated into the SOI shown on the left side of Figure 9.1, interact with each other, and interact with the NATURAL SYSTEMS, INDUCED, and HUMAN SYSTEMS Environments that comprise the SOI's OPERATING ENVIRONMENT (OE).

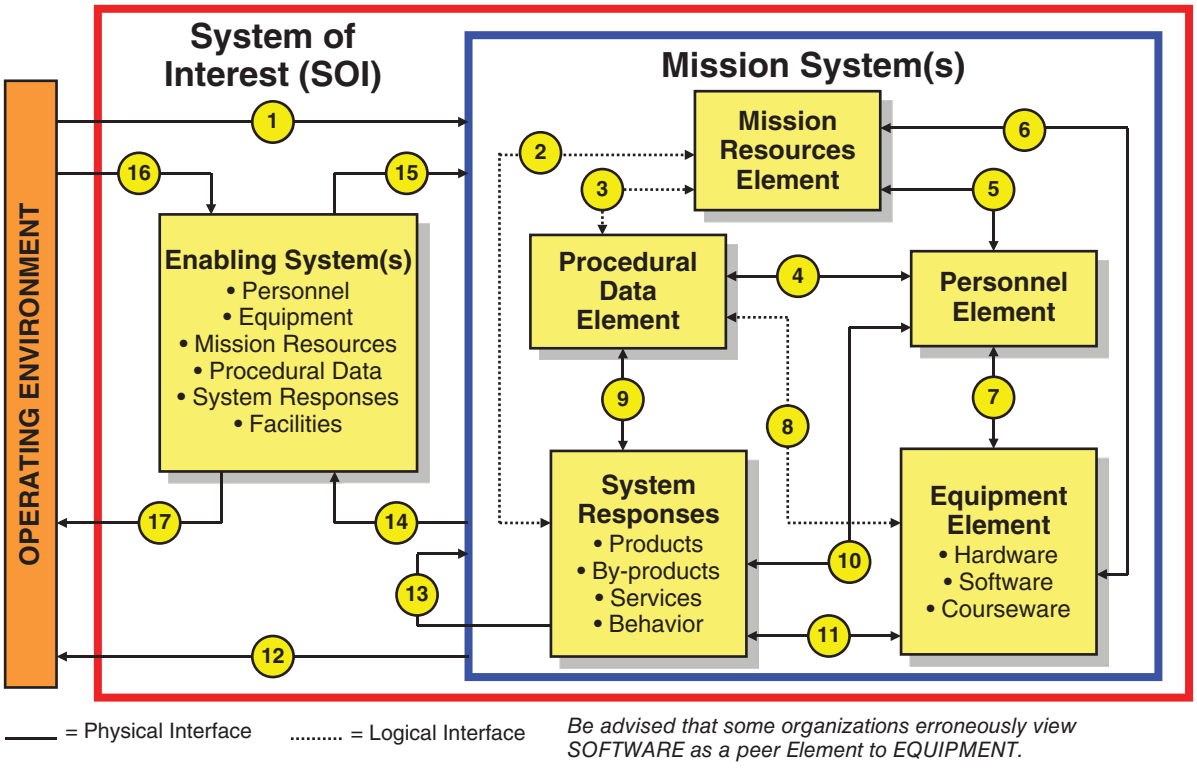


Figure 9.2 An Analytical Perspective an SOI's Architecture Depicting Interactions Between the MISSION SYSTEM, Its ENABLING SYSTEM(S), and their OPERATING ENVIRONMENT (OE).

9.3.2 OE Architecture Concepts

A significant aspect of Systems Engineering (SE) is recognizing and appreciating the need to fully understand, analyze, and bound relevant portions of a MISSION SYSTEM's or ENABLING SYSTEM's OE. *Why*? SEs and System Analysts must be capable of:

1. Fully understanding the User's Enterprise system mission, objectives, and *most probable* or *likely* Use Case (UC) applications.
2. Analytically organizing, extracting, and decomposing the mission relevant portions of the OE into manageable Problem Space(s) and Solution Space(s).
3. Effectively developing a system solution that ensures mission success within the constraints of cost, schedule, technology, support, and risk factors.

Engineers, scientists, and analysts can academically analyze a system's OE's Problem Space forever—a condition referred to as "analysis paralysis." However, success ultimately depends on making *informed* decisions based on the key facts, working within the reality of limited resources, drawing on seasoned system design experience, and exercising good judgment. The challenge is that most large complex problems require large numbers of disciplinary specialists to solve these problems. This set of people often has *diverging* rather than *converging* viewpoints of the OE.

As an SE or System Analyst, your job is to facilitate a *convergence* and *consensus* of viewpoints concerning the definition of the SYSTEM's OE. Convergence and consensus must occur in three key areas:

1. What *is/is not* relevant to the mission in the OE?
2. What is the degree of importance, significance, or influence of the OE's characteristics on missions?
3. What is the probability of occurrence of those items of significance?

So, how do you facilitate a convergence of viewpoints to arrive at a consensus decision?

As leaders, the SE and the System Analyst must have a strategy and approach for quickly organizing and leading the key stakeholders to a *convergence* and *consensus* about the OE. Without a strategy, chaos and indecision can prevail. You need skills that enable you to establish an analytical framework for OE definition decision-making.

As a professional, you have a moral and ethical obligation to yourself, your Enterprise, and society to ensure the safety, health, and well-being of the User and the public when it comes to system operations. If you and your team *overlook* or choose to *ignore* a key attribute of the OE that impacts human life and property, there may be *severe* consequences and penalties. Therefore, establish common analytical models for you and your team to use in your business applications to ensure you have thoroughly identified and considered all OE entities and elements that impact system capabilities and performance.

9.4 UNDERSTANDING THE OE ARCHITECTURE

Principle 9.1

OE Domains Principle

A system's OE consists of two classes of domains: a HIGHER-ORDER SYSTEMS Domain and a PHYSICAL ENVIRONMENT Domain.

Analytically, the OE that influences and impacts a system's missions can be abstracted several different ways. For discussion purposes, the OE can be considered as consisting of two high-level domains: (1) the HIGHER-ORDER SYSTEMS and (2) the PHYSICAL ENVIRONMENT as shown in Figure 9.3.

9.4.1 Introduction to the OE Domains and System Elements

When we think of a system's OE, people naturally focus on weather conditions. The reality is the OE consists of anything outside a system's boundaries. This leads to the question: *Where do we draw system's boundaries*?

- External to the system, product, or service?
- External to the Enterprise employing the system?

The answer is contextual depending on the frame of reference of the system. This includes (1) the NATURAL SYSTEMS Environment such as weather conditions and atmospheric conditions, (2) Engineered Systems under the C2 of other humans (e.g., internal and external to the Enterprise), and (3) the consequential effects of Engineered System interactions with the NATURAL SYSTEMS Environment that produce effects such as pollution and Electromagnetic Interference (EMI).

Enterprise systems are hierarchical and serve at the pleasure of HIGHER-ORDER SYSTEMS such as executive management and shareholders. For an Enterprise system such as a project, its OE includes:

- Multi-level organizations within an Enterprise such as marketing, accounting, and manufacturing, as well as HIGHER-ORDER SYSTEMS such as executive management, shareholders, industry, and government.
- External organizations such as customers, users, and suppliers.

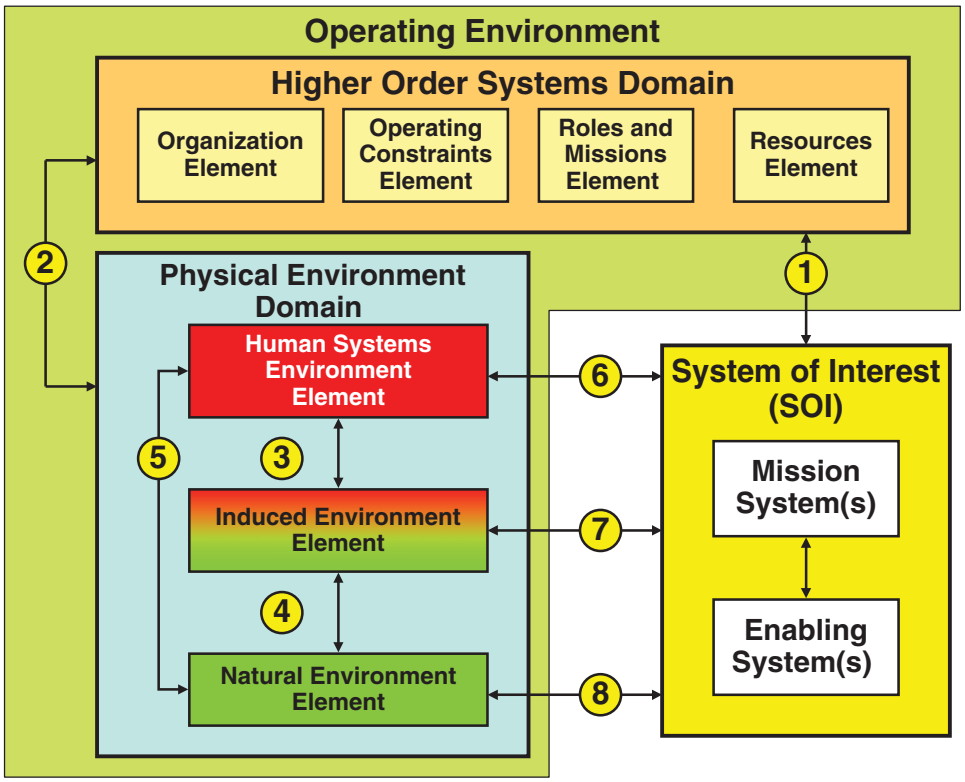


Figure 9.3 An Analytical Perspective of an SOI's Operating Environment (OE) Architecture

Human Systems—Enterprise and Engineered systems—also must be capable of not only supporting achievement of the required mission but also *surviving* encounters and interactions with Natural Systems, Induced, and Human Systems Environments that comprise the Physical Environment Domain.

As shown in Figure 9.3, the OE is partitioned analytically into a Higher-Order Systems Domain and a Physical Environment Domain. Let's explore, define, and scope each of these domains.

9.4.2 Higher-Order Systems Domain Architecture

Principle 9.2

Higher-Order Systems Principle

Every Human System exists and performs missions under the C2 of Higher-Order Systems Domain.

Human Systems—Enterprise and Engineered—perform as individual SOIs within a hierarchical System of Systems (SoS). Each higher-level abstraction serves as a Higher-Order System within the SoS hierarchy that has its own scope of authority and operational boundaries. Higher-Order Systems are characterized by:

- Enterprise purpose or mission.
- Enterprise objectives.
- An Enterprise organizational structure.
- Command media such as rules, policies, and procedures of operation.
- Resource allocations.
- Operating constraints imposed on embedded system entities.
- Accountability and objective evidence of value-added tasks performed.
- Delivery of systems, products, and services.

For most Human Systems, we refer to the vertical Higher-Order System to SOI interaction as authoritative C2. Military systems refer to it as C4I—Command, Control, Computers, Communications, and Intelligence.

Principle 9.3

Higher-Order System Elements Principle

Higher-Order Systems are composed of four analytical System Elements: (1) an Organization Element, (2) a Roles and Missions Element, (3) an Operating Constraints Element, and (4) a Resources Element.

If we observe the behavior of HIGHER-ORDER SYSTEMS and analyze their interactions, we can derive four types of System Elements: (1) ORGANIZATION, (2) ROLES AND MISSIONS, (3) OPERATING CONSTRAINTS, and (4) RESOURCES. Let's define each of these system element classes:

- **ORGANIZATION System Element**—The hierarchical C2 reporting structure, authority, and its assigned accountability for Enterprise roles, missions, and objectives.
- **ROLES AND MISSIONS System Element**—The various roles allocated to and performed by HIGHER-ORDER SYSTEMS and the missions associated with these roles and objectives to fulfill the Enterprise or organization's vision. Examples include strategic and tactical plans, roles, and mission goals and objectives.
- **OPERATING CONSTRAINTS System Element**—International, federal, state, and local statutory, regulatory, policies, and procedures as well as physical laws and principles that govern and constrain SOI actions and behavior. Examples include assets; capabilities; consumables and expendables; weather conditions; doctrine, ethical, social, and cultural considerations; and so forth.
- **RESOURCES System Element**—The natural and physical raw materials, investments, and assets such as time, money, and expertise that are allocated to the PHYSICAL Environment and SOI to sustain missions—namely, deployment, operations, support, and disposal.

9.4.2.1 Contexts of Higher-Order Systems

HIGHER-ORDER SYSTEMS have two application contexts: (1) HUMAN SYSTEMS, such as authoritative C2 or social structure moirés, and (2) physical or natural laws of science.

- **HUMAN SYSTEMS Context**—Enterprises and governments exercise hierarchical authority C2 over lower tier systems via "chain of command" structures, policies, and procedures and mission tasking; constitutions, laws, and regulations; public acceptance and opinion; and so on.
- **Physical or Natural Forces and Laws Context**—Physical and natural forces that influence, interact with, and exhibit levels of control over HUMAN SYSTEMS – Enterprise and Engineered – and their missions.

Given this structural framework of the HIGHER-ORDER SYSTEMS domain, let's define its counterpart, the PHYSICAL ENVIRONMENT Domain.

9.4.3 The Physical Environment Domain Architecture

HUMAN SYSTEMS generally require some level of interaction with external systems within the PHYSICAL ENVIRONMENT Domain. In general, we characterize these interactions with terms such as friendly, cooperative, benign, adversarial, or hostile.

PHYSICAL ENVIRONMENT Domain Composition Principle

Principle 9.4 A system's PHYSICAL ENVIRONMENT Domain is composed of three classes of analytical System Elements: a NATURAL SYSTEMS Environment, a HUMAN SYSTEMS Environment, and an INDUCED Environment.

NATURAL and HUMAN SYSTEMS Environments Systems interact; the INDUCED ENVIRONMENT represents the time-dependent result of that interaction.

If we observe the PHYSICAL ENVIRONMENT Domain and analyze its interactions with the SOI, we can identify three classes of System Elements: (1) NATURAL SYSTEMS Environment, (2) HUMAN SYSTEMS, and (3) the INDUCED Environment as shown in Figure 9.3. Let's briefly define each of these:

- **NATURAL ENVIRONMENT System Element**—All living, atmospheric, aquatic, and geophysical entities that comprise the Earth with varying amounts of these in celestial bodies.
- **HUMAN SYSTEMS Element**—External Enterprise or Engineered systems created by humans that interact with the SOI, its MISSION SYSTEM(s), and ENABLING SYSTEMS at all times, during Pre-Mission, Mission, and Post-Mission.
- **INDUCED ENVIRONMENT System Element**—Discontinuities, perturbations, or disturbances created when natural phenomenon and events occur or HUMAN SYSTEMS interact with the NATURAL ENVIRONMENT. Examples include thunderstorms, wars, and oil spills.

Based on this high-level introduction and identification of the OE System Elements definitions, we are now ready to address the structure of the OE architecture. Let's begin with the PHYSICAL ENVIRONMENT Domain.

9.4.3.1 Physical Environment Domain Levels of Abstraction

The PHYSICAL ENVIRONMENT Domain consists of three types of System Elements, each with three Levels of Abstraction as illustrated in Figure 9.4. For example, the HUMAN SYSTEMS Environment consists of three levels of analytical abstractions: (1) a contextual Local Environment, (2) a Global Environment, and (3) a Cosmospheric Environment.

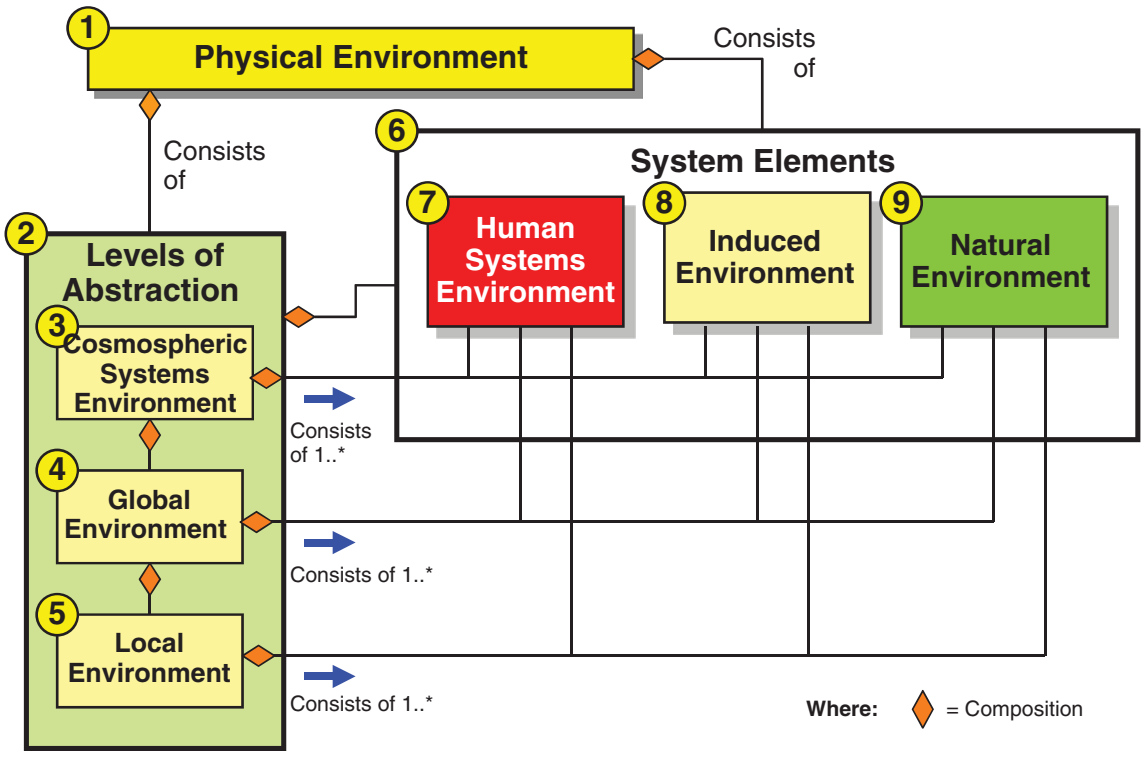


Figure 9.4 Analytical Framework Depicting the PHYSICAL ENVIRONMENT Domain, Its System Elements, Levels of Abstraction, and their Entity Relationships (ERs)

- HUMAN SYSTEMS (Earth LOCAL ENVIRONMENT Level of Abstraction) include anyone or anything that influences or interacts with an SOI such as cars, roads, animals, and so forth.
- HUMAN SYSTEMS (Earth GLOBAL ENVIRONMENT Level of Abstraction) include any one or anything that influences or interacts with an SOI such as carbon dioxide emissions, space debris, and so forth.
- HUMAN SYSTEMS (Cosmospheric ENVIRONMENT Level of Abstraction) include spacecraft, space debris, so forth.

Author's Note 9.1

Analytical Representation of the PHYSICAL SYSTEMS ENVIRONMENT

You may decide that some other quantity of system levels of abstraction is more applicable to your line of business. That's okay. What is *important* is that you and your team have a simple approach for abstracting the complexity of the PHYSICAL ENVIRONMENT Domain into manageable pieces. The "pieces" must support meaningful analysis and ensure coverage of all relevant aspects that relate to your problem and solution spaces. Remember, bounding abstractions for analysis is analogous to cutting a pie into 6, 8, or 10 pieces. As long as you account for the totality, you can create as many abstractions as are *reasonable* and *practical*; however, *keep it simple!*

9.4.3.1.1 The NATURAL ENVIRONMENT System Element

The NATURAL ENVIRONMENT System Element includes all naturally occurring entities that are not created by humans. These entities are actually environmental "systems" that coexist within a precarious balance of power. In general, these systems represent geophysical and life form classes of objects.

9.4.3.1.2 Cosmospheric Environment Level of Abstraction

The *Cosmospheric Environment* is an analytical abstraction that represents the totality of the Cosmos—the universe—as humans understand it. Analytically, the Cosmospheric Environment consists of an infinite number of Global Environments—suns, stars, planets, moons, and asteroids.

You may ask why the Cosmospheric Environment is relevant to SE. Consider space probes that have flown beyond the boundaries of our solar system. The SEs and physicists who bounded the OE had to identify the global entities that had a potential impact on the space probe's mission. Obviously, these global entities did not move out of the way of the probe's mission path. The SEs and physicists had to understand:

1. What known entities they might encounter during the missions.

2. What each global entity's performance characteristics are.
3. How to navigate and maneuver among those entities throughout the mission without an adverse or catastrophic impact.

9.4.3.1.3 Global Environment Level of Abstraction The *Global Environment* is an analytical abstraction that represents the physical environment surrounding a heavenly body. This includes entities such as stars, planets, moons, etc. Analytically, we state that the Global Environment of any heavenly body consists of an infinite number of Local Environments, each relative to an SOI observer's frame of reference. For example, a Local Environment – sphere - surrounds your automobile as it travels down a highway. The challenge is, how do you bound the Local environment for analysis purposes in terms of boundary conditions – weather, size, and so forth. If you operate an airline, each aircraft is surrounded by a Local Environment sphere within the Earth's Global Environment. In contrast, NASA launches interplanetary space probes experience an infinite number of Local Environments throughout the mission and global environments—namely, Earth, planets, and the planetary moons. Although Global Environments may share a common set of physical attributes, such as gravity and some level of atmosphere, their physical magnitudes can vary significantly. As a result, OE requirements may require SEs to bound ranges of *worst-case* parameters across the spectrum of global entities encountered.

As humans, our observer's frame of reference is planet Earth. Therefore, we typically relate to the Global Environment as that of the Earth and its gaseous atmosphere. Some simply refer to it as the Earth's environment. Applying the same convention to Mars, we could refer to it as the Mars's Global Environment.

The challenge question SEs need to answer is:

- *How do we bound and model a varying continuum between celestial bodies such as the electromagnetic field, the atmosphere, or the gravity*?

From an SE perspective, one approach is to delineate the two environments by asking the question, *What characteristics below or above some arbitrary threshold are unique or "native" to the "global" SOI*? If the boundary conditions of two or more entities overlap, the objective would be to determine criteria for the boundary conditions relative to the mission.

As an SE or Systems Analyst, you represent the technical side of the program technical boundary for the discussions that follow. Thus, you have a *professional*, *technical obligation* to ensure that the *integrity* of these decisions is supported by:

1. Objective, factual data to the extent practical and available.
2. Valid assumptions that withstand peer and stakeholder scrutiny.
3. *Most likely* or *probable* operational scenarios and conditions.

Remember, the analytical relevance of the decisions you make here may have a major impact on your system's design, cost, reliability, maintainability, vulnerability, survivability, safety, and risk considerations. These decisions may have adverse impacts on human life, property, and environment.

Analytically, we can partition the NATURAL ENVIRONMENT's Global and Local levels of abstraction into four sub-elements:

- Atmospheric Systems Environment
- Geospheric Systems Environment
- Hydrospheric Systems Environment
- Biospheric Systems Environment

Author's Note 9.2

Analytical Partitioning of the NATURAL ENVIRONMENT

The point of our discussion here is not to present a view of the physical science. Our intent is to illustrate how SEs might approach analysis of the NATURAL ENVIRONMENT. Ultimately, you will have to bound and specify applicable portions of this environment that are applicable to your SOI. The approach you and your team choose to use should be *accurately* and *precisely* representative of your system's operating domain and the relevant factors that drive MISSION SYSTEM capabilities and levels of performance. However, you should always consult Subject Matter Experts (SMEs), who can assist you and your team in abstracting the correct environment.

9.4.3.1.4 Atmospheric Systems Environment The Atmospheric Systems Environment is an abstraction that represents the gaseous layer continuum of varying density that extends from the surface of a planetary body outward into space.

9.4.3.1.5 Geospheric Systems Environment The Geospheric Systems Environment is an analytical abstraction that represents the physical landmass of a star, moon, or planet. From an Earth sciences perspective, the Earth's Geospheric Systems Environment includes the Lithospheric Systems Environment, the rigid or outer crust layer of the Earth. In general, the Lithosphere includes the continents, islands, mountains, and hills that appear predominantly at the top layer of the Earth.

9.4.3.1.6 Hydrospheric Systems Environment The Hydrospheric Systems Environment is an analytical abstraction that represents all water systems, such as lakes, ponds, rivers, streams, waterfalls, underground aquifers, oceans, tidal pools, and ice packs, that are not part of the Atmospheric Systems Environment. In general, the compositional entities within the Hydrospheric Systems Environment include rainwater, soil waters, seawater, surface brines, subsurface waters, and ice.

9.4.3.1.7 Biospheric System Environment The Biospheric Systems Environment is an analytical abstraction that represents the environment comprising all living organisms on the surface of the Earth or celestial body. In general, the Earth's Biosphere consists of all environments that are capable of supporting life above, on, and beneath the Earth's surface as well as the oceans. Thus, the biosphere overlaps a portion of the Atmosphere, a large amount of the Hydrosphere, and portions of the Lithosphere. Examples include botanical, entomological, ornithological, amphibian, and mammalian systems. In general, biospheric systems function in terms of two metabolic processes: photosynthesis and respiration.

9.4.3.1.8 Local Environment Level of Abstraction The Local Environment is an analytical abstraction that represents the PHYSICAL Environment encompassing a system's current geophysical location. For example, if you are driving your car, the Local Environment consists of the OE conditions surrounding the vehicle. Therefore, the Local Environment includes other vehicles and drivers, road hazards, weather, and any other conditions on the roads that surround you and your vehicle at any instant in time. As an SE, your challenge is leading system developers to consensus on:

1. What is the Local Environment's SOI observer's frame of reference?
2. What are the Local Environment's bounds (initial conditions, entities, etc.) at any point in time?

Author's Note 9.3

Atomic Level Environments

If you are a chemist or physicist, you may want to consider adding a fourth Atomic Level environment assuming it is germane to your SOI.

Author's Note 9.4

Discussion Summary

Your "take-away" from the preceding discussion is not to create five types of NATURAL ENVIRONMENT abstractions and document each in detail for every system you analyze. Instead, you should view these elements as a checklist to prompt mental consideration for *relevance* and *significance* and identify those PHYSICAL ENVIRONMENT entities that have relevance to your system. Then, bound and specify those entities.

9.4.3.1.9 The HUMAN SYSTEMS Element Our discussions of the NATURAL ENVIRONMENT omitted a key entity, Humans. Since Engineering, in general, focuses on benefiting society through the development of systems, products, or services, we analytically *isolate* and *abstract* humans into a category referred to as the HUMAN SYSTEMS Environment element. HUMAN SYSTEMS include sub-elements that influence and control human decision-making and actions that affect the balance of power on the planet.

If we observe HUMAN SYSTEMS and analyze how these systems are organized, we can identify seven types of sub-elements: (1) historical or heritage systems, (2) cultural systems, (3) urban systems, (4) business systems, (5) educational systems, (6) transportation systems, and (7) governmental systems. Let's explore each of these further.

9.4.3.1.10 Historical or Heritage Systems *Historical* or *heritage systems* as an abstraction include all artifacts, relics, traditions, and locations relevant to past human existence such as folklore and historical records.

You may ask why the historical or heritage systems are relevant to an SE. Consider the following example:

Example 9.1

Let's assume that we are developing a system such as a building that has a placement or impact on a land use area that has historical significance. The building construction may disturb artifacts and relics from the legacy culture, and this may influence technical decisions relating to the physical location of the system. One perspective may be to provide physical space or a buffer area between the historical area and our system. Conversely, if our system is a museum relevant to an archeological discovery, the location may be an integral element of the building design.

From another perspective, military tactical planners may be confronted with planning a mission and have an objective to avoid an area that has historical, cultural, or religious significance. Systems that have the potential to impact environmental resources such as water aquifers, rivers, streams, and various life forms are often required to assess the environmental impact. In these cases, an Environmental Impact Statement (EIS) is required as well as other supporting information.

A Word of Caution 9.1

Consult your Contracts and Legal organizations concerning laws and regulations that may impose specific environmental compliance constraints.

9.4.3.1.11 Urban Systems Urban systems include all entities that relate to *how* humans cluster or group themselves into communities and interact at various levels of organization—neighborhood, city, state, national, and international.

Example 9.2

Urban System Infrastructures

Urban systems include the infrastructure that supports the business systems environment, such as transportation systems, public utilities, city services, shopping, recreational, medical, telecommunications, and educational institutions.

SE, from an Urban Systems perspective, raises key issues that require some form of *dependence* on *predictive* assumptions about future operational needs. How do SEs plan and design a road system within resource constraints that provide some level of insights into the future to facilitate growth and expansion?

9.4.3.1.12 Cultural Systems *Cultural systems* are an analytical abstraction that represents multi-faceted entities that consist of humans and communal traits and how humans interact, consume, reproduce, and survive. Examples include the performing arts of music and other entertainment, civic endeavors, and patterns of behavior. As the commercial marketplace can attest, cultural systems have a major impact on society's acceptance of systems.

9.4.3.1.13 Business Systems *Business systems* are an analytical abstraction that represents entities related to how humans organize into economic-based enterprises and commerce to produce products and services to sustain a livelihood. These include research and development, manufacturing, products, and services for use in the marketplace.

9.4.3.1.14 Energy Systems Energy systems are an analytical abstraction that represents entities involved in the exploration, drilling, recovery, and delivery of energy products such as oil, natural gas, solar, and wind. These include research and development, manufacturing, products, and services for use in the marketplace.

9.4.3.1.15 Communications Systems *Communications systems* are an analytical abstraction that represents entities involved in the preparation, broadcast, and transmission of information to the marketplace. These include research and development, manufacturing, products, and services for use in the marketplace.

9.4.3.1.16 Educational Systems *Educational systems* are an analytical abstraction that represents an institution as an SOI dedicated to educating and improving society through formal and informal institutions of learning.

9.4.3.1.17 Financial Systems *Financial systems* are an analytical abstraction that represents institutions such as banks and investment entities that support personal, commercial, and government financial transactions.

9.4.3.1.18 Medical Systems *Medical systems* are an analytical abstraction that represents hospitals, doctors, and therapeutic entities that administer to the healthcare needs of the public.

9.4.3.1.19 Transportation Systems *Transportation systems* are an analytical abstraction that represents land, sea, air, and space transportation systems that enable humans to travel safely, economically, and efficiently from one destination to another.

9.4.3.1.20 Government Systems *Government systems* are an analytical abstraction that represents legal entities related to governing humans as a society—international, federal, state, county, and municipality.

9.4.3.1.21 INDUCED ENVIRONMENT *Element* The preceding discussions focused on the NATURAL ENVIRONMENT and HUMAN SYSTEMS as abstractions. While these two elements enable us to analytically organize OE entities, they are *dynamic* and physically *interactive*. In fact, HUMAN and NATURAL ENVIRONMENT Systems each create intrusions, disruptions, perturbations, and discontinuities in the other.

Analysis of these interactions can become very complex. We can alleviate some of the complexity by creating the third PHYSICAL ENVIRONMENT Domain element, the INDUCED ENVIRONMENT Element. The INDUCED ENVIRONMENT enables us to *isolate* entities that represent the existence of time-dependent intrusions, disruptions, perturbations, and discontinuities until they diminish or are no longer significant or relevant to the system's mission or operation. The degree of significance of INDUCED ENVIRONMENT entities may be temporary, permanent, or dampen as a function of time. For example, when a just aircraft lands, it creates vortices, thermal gradients, and so forth that dampen over time. Depending on their physical size, air controllers space aircraft distances to ensure a landing aircraft does not create disturbances that impact the follow-on aircraft's landing.

9.4.3.1.22 OTHER ARCHITECTURAL FRAMEWORKS The OE imposes various factors and constraints on the capabilities and levels of performance of an SOI, thereby impacting missions and survival over the planned service life. As an SE or System Analyst, your responsibility is to:

1. Identify and delineate all of the critical OE conditions.

2. Bound and describe technical parameters that characterize the OE.
3. Ensure that those descriptions are incorporated into the System Performance Specification (SPS) used to procure the SOI.

The process of identifying the SOI's OE requirements employs a simple methodology as depicted in Figure 9.5. In general, the methodology implements the logic reflected in the PHYSICAL ENVIRONMENT levels of abstraction and classes of environments previously described.

The methodology consists of three iterative loops:

- Loop 1: Cosmospheric level requirements.
- Loop 2: Global entity level requirements.
- Loop 3: Local level requirements.

When each of these iterations is applicable to an SOI, the HUMAN SYSTEMS logic branches out to a fourth loop that investigates which of the three classes of environments—HUMAN SYSTEMS, NATURAL ENVIRONMENT, or INDUCED ENVIRONMENT—is relevant to the SOI. For each type of environment that is applicable, requirements associated with that type are identified—HUMAN, NATURAL, and INDUCED. When the third loop completes its types of environment decision-making process, it returns to the appropriate level of abstraction and finishes with the local level requirements.

9.4.3.1.23 Entity OE Frame of Reference The preceding discussions address the OE from an SOI's frame of reference. However, lower levels of abstraction within a System are, by definition, self-contained systems of integrated entities. The OE contextually must be established relative to the ENTITY's frame of reference as illustrated in Figure 9.6. For example, *what constitutes an* ASSEMBLY *Level Entity's OE?* The OE is anything external to the ASSEMBLY's boundary, such as other SUBSYSTEMS. Consider the following example:

Example 9.3

A processor board within a desktop computer chassis has an OE that consists of the motherboard, other boards it interfaces directly with, Electromagnetic Interference (EMI) from power supplies, switching devices, and so on.

9.4.3.1.24 Concluding Point You may ask: *Isn't the* HIGHER-ORDER SYSTEMS *Domain part of the* PHYSICAL

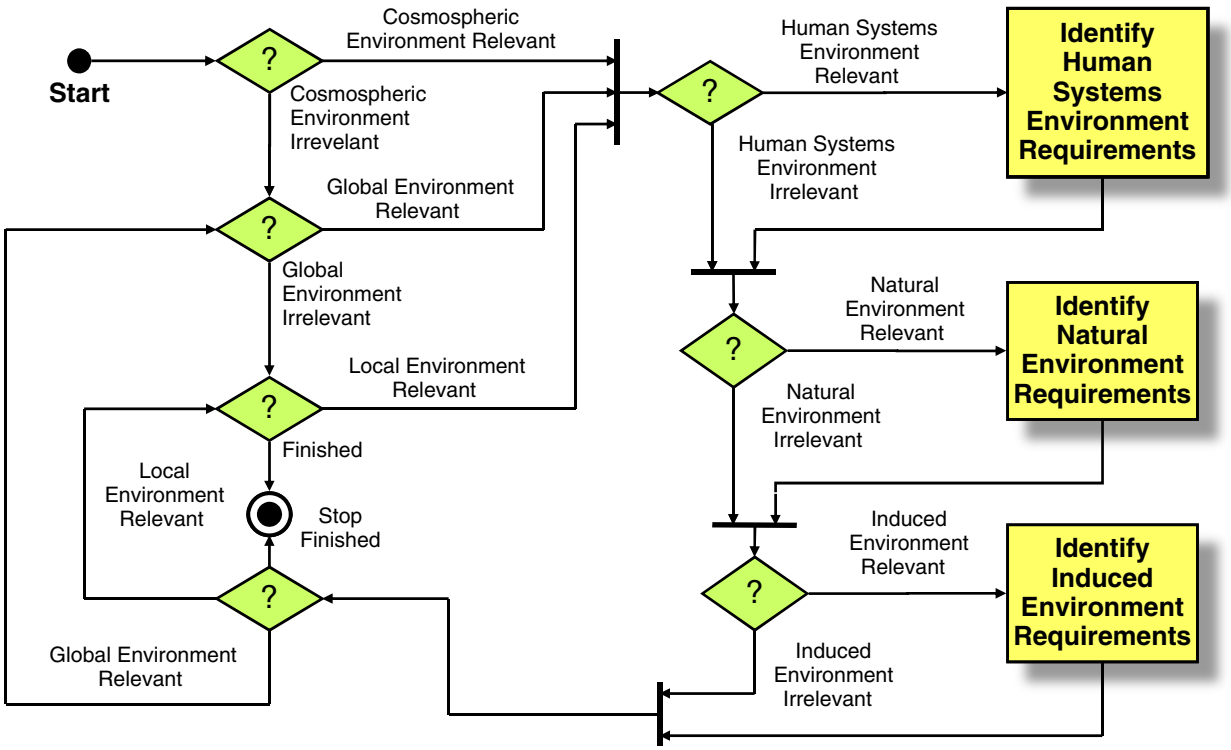


Figure 9.5 Example Methodology for Identifying PHYSICAL ENVIRONMENT Domain Requirements

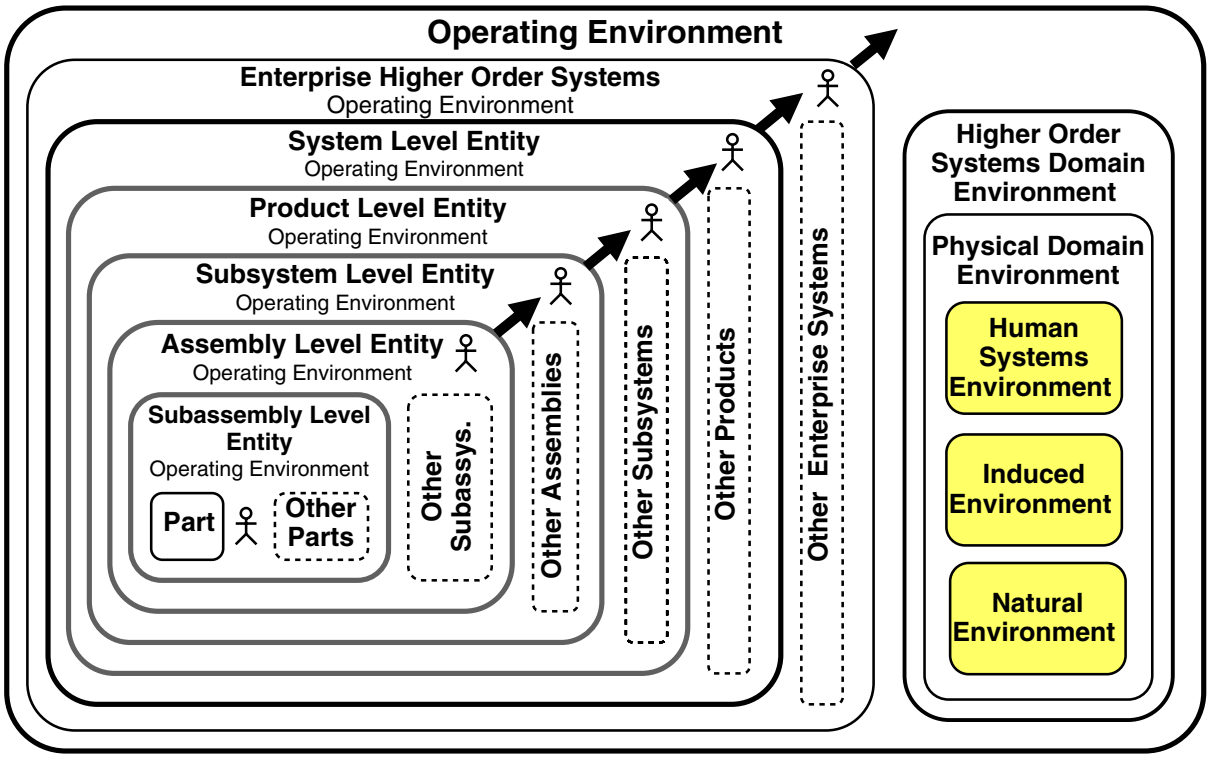


Figure 9.6 Understanding the Context of an SOI's Operating Environment Relative to the Observer's Frame of Reference

Environment *Domain*? You could argue this point. However, Figure 9.1 represents an *analytical* perspective with a key focus on direct, peer-to-peer, and C2 interactions. You can choose to analytically partition a system any way … as long as you cover all aspects. Analytically, an SOI is chartered, resourced, and responds to Enterprise or organizational authority. Therefore, we depict the Higher-Order Systems in terms of human authoritative or supervisory C2 the SOI.

Are Higher-Order Systems *(human) above the Physical Environment?* No, in fact, one can argue that humans are physically subject subject to its whims. Yet, as we see in phenomena such as debates about global warming, our collective actions can have an adverse impact on it long term and in turn on our lives.

9.4.3.1.25 Summary: OE Concept Our discussion in this chapter provides an orientation of the OE, its levels of abstraction, and classes of environments. Based on identification of these OE elements, we introduced Figure 9.4 to depict analytical ERs among Physical Environment Domain levels of abstraction and its System Elements. Next, we introduced the concept of the OE architecture as a framework for linking the OE System Elements.

9.5 OTHER ARCHITECTURAL FRAMEWORKS

This chapter provides the foundational knowledge in understanding the architectural framework of systems. For additional information concerning domain-specific architectural frameworks, Refer to ISO-Architecture (2014).

9.6 UNDERSTANDING THE SYSTEM THREAT ENVIRONMENT

The exploitation of opportunities may be viewed by some organizations as threatening to the sustainment and survival of an Enterprise or organization. Whether the scenario involves increasing market share, defending national borders, or developing a secure Internet Web site, you must ensure that your system is capable of sustaining itself and its long-term survival.

Long-term survival hinges on having a thorough and complete understanding of the potential threat environment and having the system capabilities to counter the threats that serve as obstacles to mission success. So, *how does this relate to SE*? When you specify requirements for your system, system requirements must include considerations of what capabilities and levels of performance are required to counter threat actions.

9.6.1 Threats Sources

System threats range from the *known* to the *unknown*; some people refer to the *unknown-unknowns*. One approach to identifying potential system threats can be derived from the PHYSICAL ENVIRONMENT Domain—HUMAN, INDUCED, and NATURAL SYSTEMS Environments.

9.6.1.1 NATURAL SYSTEMS Environment Threat Sources

NATURAL SYSTEMS ENVIRONMENT threat sources, depending on perspective, include lightning, hail, wind, rodents, and disease. For example, squirrels often chew into unprotected electrical cables, woodpeckers destroy some types of wooden siding on homes, and so forth.

9.6.1.2 HUMAN SYSTEMS Threat Sources

External HUMAN SYSTEMS—Enterprise and Engineered systems—threat sources include primarily PERSONNEL and EQUIPMENT Elements. The motives and actions of the external systems delineate friendly, competitive, adversarial, or hostile intent.

9.6.1.3 INDUCED ENVIRONMENT Threat Sources

INDUCED ENVIRONMENT threat sources include examples such as contaminated landfills leaking into underground water aquifers, EMI, space debris high velocity impacts on spacecraft, ship wakes on smaller vessels, and aircraft vortices on other aircraft.

9.6.2 Types of System Threats

System threats occur in a number of forms, depending on the environment—HUMAN SYSTEMS, INDUCED, and NATURAL SYSTEM ENVIRONMENTS.

Generally, most HUMAN and NATURAL SYSTEMS ENVIRONMENT aggressor threats fall into the categories of strategic threats or tactical threats related to the balance of power, motives, and objectives.

Other natural threats are attributes of the NATURAL SYSTEMS ENVIRONMENT that impact a system's inherent capabilities and performance. Examples include temperature, humidity, wind, salt spray, lightning, light rays, and rodents. Although these entities do not reflect premeditated aggressor characteristics, their mere existence in the environment, seemingly benign or otherwise, can adversely impact system capabilities and performance.

9.6.3 Threat Behavioral Characteristics, Actions, and Reactions

Threats exhibit characteristics and actions that may be described as adversarial, competitive, hostile, and benign. Some threats may be viewed as *aggressors*. In other cases, threats are generally *benign* and only take action when someone "gets into their space." Here's an example:

Example 9.4 **Unauthorized Intrusion of Sovereign Airspace**

An unauthorized aircraft *purposefully* or *unintentionally* intrudes on another country's airspace creating a provocation of tensions. The defending system response course of action may be based on protocol or a measured retaliatory strike or verbal warning.

Threats, in general, typically exhibit three types of behavior patterns or combinations thereof: aggressive, concealed, and benign. Threat patterns often depend on the circumstances. Here's an example:

Example 9.5 **Threat Behavior Patterns**

Aggressors exhibit acts of aggression.

Benign threats may "tend to their business" unless provoked.

Concealed or sleeper threats may appear to be friendly or benign or disguised and strike Targets of Opportunity (TOO) unexpectedly.

The threat environment should be characterized by attributes such as ideology, doctrine, and training are often key factors in threat actions.

9.6.4 Threat Encounters

When systems interact with *known* threats, the interactions—such as *encounters* or *engagements*—should be documented and characterized for later use by other systems in similar encounters. Threat encounters intended to probe another system's defenses can be described with a number of descriptors: aggressive, hostile, cooperative, inquisitive, investigative, bump and run, and cat and mouse.

When threat encounters turn *hostile* provoking defensive action, systems resort to various tactics and *countermeasures* to ensure their survival.

9.6.4.1 System Tactics

When systems interact with their OE, they often engage threats or opportunities. Systems and system threats often employ or exhibit a series of *evasive* actions intended to conceal, deceive, or camouflage the TOO. Generally, when evasive tactics do not work, systems deploy countermeasures to *disrupt* or *distract* hostile actions. Let's examine this topic further.

9.6.4.2 Threat Countermeasures

To counter the impact or effects of threats on a system, systems often employ threat countermeasures. Threat countermeasures are any physical action performed by a system to deter a threatening action or counter the impact of a threat (i.e., survivability). Sometimes adversarial systems acquire or develop the technology to counter the TOO's system countermeasures.

9.6.4.3 Threat Counter-Countermeasures (CCM)

Sometimes, system threats compromise the established security mechanisms by deploying CCM to offset the effects of a TOO's countermeasures.

9.6.5 Concluding Thoughts

This concludes our overview of system threats and opportunities. You should emerge from this discussion with an awareness of how the system you are using or developing must be capable of interacting with threats and opportunities in the OE. More explicitly, you will be expected to develop a level of technical knowledge and understanding to enable your team to specify or oversee the specification of system capabilities and levels of performance related to threats and opportunities.

Now that we have an understanding of the opportunistic and potentially hostile entities, we are now ready to investigate how a system interacts with external HUMAN SYSTEMS in its OE and the balance of power to perform missions.

9.7 SOI INTERFACES

One of the crucial factors of system success is determined by what happens at its *internal* and *external* interfaces. You can engineer the most elegant algorithms, equations, and decision logic, but if the system does not perform at its interfaces, the elegance is of no value. System interface characterizations range from cooperative *interoperability* with external friendly systems to layers of protection to minimize the system or entity's *susceptibility* and *vulnerability* to external threats (environment, hostile adversary actions, etc.) and structural integrity to ensure survivability.

This section provides an introduction to system interfaces, their purpose, objectives, attributes, and how they are implemented. Our discussions explore the various types of interfaces and factors that delineate *success* from *failure*.

9.7.1 What Is an Interface?

In general, most system development efforts often focus on the following as their Systems Engineering primary activities:

- Creation of a physical system architecture and its components.
- Development of physical system interfaces with external systems.

An *interface* represents a *constraint* based on the *logical* any *physical* boundary conditions between two or more entities within a level of abstraction, between System Elements, between the SOIs MISSION SYSTEM(S) and ENABLING SYSTEM(S), or between the SOI and its OE.

9.7.1.1 Interface Purpose

The purpose of an interface is to establish an ER, *logical* association, and *physical* connection between entities—for example, SYSTEM, PRODUCT, SUBSYSTEM, ASSEMBLY, SUBASSEMBLY, or PART Levels of abstraction and other external entities within its OE. An entity may have *logical* and *physical* associations (Figures 8.8 and 8.9) with several entities; however, each linkage represents a single interface. If an entity has multiple interfaces, the performance of that interface may have an influence or impact on the others due to an effect referred to *loading* that may *alias*, *bias*, or *degrade* system or entity performance. The question is: *How does an interface accomplish this?*

An interface has at least one or more objectives, depending on the component's application. Typical interface objectives include the following:

- Objective 1: Physically link or bind two or more System Elements or entities.
- Objective 2: Adapt one or more incompatible System Elements or entities.
- Objective 3: Buffer the effects of incompatible System Elements or entities.
- Objective 4: Leverage PERSONNEL or EQUIPMENT capabilities.
- Objective 5: Restrain an Entity's usage.

Let's explore each of the objectives further.

9.7.1.1.1 Objective 1: Physically Link or Bind Two or More System Elements or Entities

Some systems link or bind two or more *compatible* System Elements or entities to anchor, extend, support, or connect the adjoining interface. Here's an example:

Example 9.6

A TV or radio station communications tower includes cables at critical attach points to anchor the tower vertically and horizontally to the ground for stability.

9.7.1.1.2 Objective 2: Adapt One or More Incompatible System Elements or Entities

Some System Elements such as EQUIPMENT and PERSONNEL or entities may not have compatible or interoperable interfaces. However, they can be adapted to become *compatible*. Software applications that employ reusable models may create a "wrapper" around the model to enable the model to communicate with an external application, and vice versa.

9.7.1.1.3 Objective 3: Buffer the Effects of Incompatible System Elements or Entities

Some systems such as automobiles are generally not intended to interact with each other. Where the *unintended* interactions occur, the effects of the

interaction must be minimized in the interest of the safety and health of the Users. Consider the following two cases:

Example 9.7

1. An automobile's impact on another can be lessened with a shock-absorbing bumper and body crumple zones.
2. SYSTEM A is required to transmit data to SYSTEM B. Due to the limited speed of the interfacing components, SYSTEM A includes a buffer area for storing data for communication to free up its processor to perform other tasks.

On the other side of the interface, SYSTEM B may be unable to process all of the incoming data immediately. To avoid this scenario, a buffer area is created in SYSTEM B to store the incoming data until its processor can process the data.

For this objective, SEs analyze the interface and take reasonable measures to create a "boundary layer" or buffer between System Elements or its entities. Thus, each is buffered to minimize the effects of the impact to the system or environment and, if applicable, safety to the operators or the public.

9.7.1.1.4 Objective 4: Leverage PERSONNEL or EQUIPMENT Element Capabilities Humans employ interface capabilities to leverage our own skills and capabilities. Early humans recognized that simple machines could serve as interface devices to expand or leverage our own physical capabilities in accomplishing difficult or complex tasks. Examples include the lever and fulcrum, the wheel, bow and arrow, and spear.

9.7.1.1.5 Objective 5: Restrain an Entity's Usage Some interfaces serve as restraints to ensure a level of safety for System Elements. Here's an example:

Example 9.8

Interface Safety Constraints Example

1. A safety chain is required by law for travel on public roads and highways as a back-up safety mechanism to prevent a trailer from detaching from a vehicle if its trailer hitch fails.
2. A lock is added to an electrical, high-voltage, power distribution box to prevent opening and tampering by unauthorized individuals.
3. A machine fails in a manufacturing facility. For maintenance safety, power is switched OFF, the power breaker box is LOCKED, and Safety Tag is attached to the power breaker box until maintenance on the machine is complete.

Each of these objectives illustrates how an interface is implemented to achieve a purposeful action. Depending on a systems application, other objectives may be required. So, for each of the interfaces in the SOI, *what is its purpose and expected performance-based outcome*? You should know!

9.7.2 Interoperability: The Ultimate Interface Challenge

The ultimate success of any interface resides in its capability to interact with a diverse range of systems from friendly to hostile in its intended OE as envisioned by the User, specified by the System Acquirer, and designed by the System Developer. In general, the *interoperability* (Chapter 3) refers to the ability of a systems or entities to mutually exchange, understand, and process what is being communicated and similarly provide responses where appropriate. This brings another question: *What is meant by understanding what is being communicated*? To be *interoperable*, an entity must be able to accept, decode, interpret, process, act on, and encode data for retransmission.

9.7.3 Interface Classes

Interfaces exhibit three types of operation: *active*, *passive*, or *active/passive*.

9.7.3.1 Active Interfaces *Active* interfaces interact with external systems or components in a friendly, benign, or cooperative manner. Here's an example:

Example 9.9

Active Interfaces Example

Radio stations, as active "on the air" systems, radiate signals at a designated carrier frequency via patterns to specific areas for coverage.

Active interfaces can also have *negative* consequences. For example, when a radar is energized to sweep an area or "paint" a target, those actions may be detectable by the threats is seeks to illuminate and identify.

9.7.3.2 Passive Interfaces *Passive* interface interactions with external components simply receive or accept data without responding. Here's an example:

Example 9.10

Passive Interfaces Example

A car radio, when powered On, *passively* receives signals over a tuned frequency. The radio provides an *active* audio interface by transmitting signals on a designated frequency for occupants in the car tuned to the station.

9.7.3.3 Active/Passive Interfaces *Active/passive* interfaces perform under the control of a manual or automated C2 authority such as humans and/or computer hardware and software. Here's an example:

Example 9.11 **Active/Passive Interfaces**

A two-way, hand-held radio has an *active* interface when the User presses the "Push to Talk" button to transmit audio information to others listening on the same frequency within a specified transmission range and operating conditions. When the "Push to Talk" button is OFF, the device has a *passive* interface that monitors incoming radio signals for audio processing and amplification as controlled by the User.

9.7.3.4 Inactive or Dormant Interfaces Some interfaces are *dormant* or *inactive* until required for a specific Phase or Mode of Operation. Here's an example:

Example 9.12 **Inactive or Dormant Interfaces**

A space probe to Mars may consist of a mother ship and a lander or rover. During the flight from Earth to Mars, the mother ship provides a communications link capable of *passively receiving* mission data and commands, software updates, etc. and actively transmitting mission data back to ground controllers on Earth.

On arrival at Mars, the mother ship might land and deploy the rover or circle Mars and deploy a lander. During the flight, the lander or rover might be designed to be powered down resulting in *inactive* or *dormant* communications, camera, and other interfaces until activated by Earth-based commands relayed via the mother ship.

9.7.4 Physical Interface Types

If we analyze how interfaces are implemented in the PHYSICAL ENVIRONMENT Domain, our analysis will reveal that interfaces occur in mechanical, electrical, optical, acoustical, nuclear, chemical, and natural forms, and as combinations of these forms. For all these types of physical interfaces, there are specialized solutions that must be documented in some form of Interface Control Document (ICD). To further understand the specialized nature of these interface solutions, let's explore each one.

9.7.4.1 Mechanical Interfaces *Mechanical interfaces* consist of boundaries that exist between two physical objects and include characterizations such as *form*, *fit*, and *function*. Characterizations include properties such as:

- *Dimensional properties* include parameters such as length, width, and depth.
- *Mass properties* include parameters such as materials and composition, weight, density, weights and balances, moments of inertia, and Center of Gravity (CG).
- *Structural properties* include parameters such as ductility, hardness, shear strength, and tensile strength.
- *Aerodynamic properties* such as drag and fluid flow.
- *Thermal properties* such as conductivity, insulation, and coefficient of expansion/contraction.
- *Thermodynamic properties* include parameters related to Pressure–Volume–Temperature (PVT) such as specific heat capacity.

9.7.4.2 Electrical Interfaces *Electrical interfaces* consist of direct electrical or electronic connections as well as electromagnetic transmission in free air space. Attributes and properties include parameters such as voltages, current, resistance, conductivity, inductance, capacitance, dielectric constant, grounding, shielding, attenuation, and transmission delays.

9.7.4.3 Optical Interfaces *Optical interfaces* consist of the transmission and/or receipt of visible and invisible wavelengths of light. Attributes and properties include parameters such as intensity, frequency, special ranges, resolution, contrast, reflectivity, refractive index, scattering, transmittance, filtering, modulation, attenuation, polarization, albedo, and insolation.

9.7.4.4 Acoustical Interfaces *Acoustical interfaces* consist of the creation, transmission, and receipt of frequencies that may be audible or inaudible to humans. Attributes and properties include parameters such as volume, frequency, modulation, attenuation, acoustical absorption, and air or aquatic density.

9.7.4.5 Natural Systems Environment Interfaces *NATURAL SYSTEMS Environment* interfaces consist of those elements that are natural occurrences of nature. Attributes and properties include temperature, humidity, barometric pressure, altitude, wind, rain, snow, and ice.

9.7.4.6 Chemical Interfaces *Chemical interfaces* consist of interactions that occur when chemical substances are purposefully or inadvertently introduced or mixed with other chemicals or other types of interfaces. Attributes and properties include parameters such as corrosion resistance, viscosity, pH, and toxicity.

9.7.4.7 Biological Interfaces *Biological interfaces* consist of those interfaces between living organisms or other types of interfaces. Attributes and properties such as the Five Senses - touch, feel, smell, hearing, and sight.

9.7.4.8 Nuclear Interfaces *Nuclear interfaces* consist of those interfaces between radioactive material sources, your system, humans, and the environment concerning issues such as containment, protection, vulnerability, and survivability.

9.7.5 Standard versus Dedicated Interfaces

Interfaces allow us to establish logical or physical relationships between System Elements via a common, compatible, and interoperable boundary. If you analyze the most common types of interfaces, you will discover two basic categories: (1) standard, modular interfaces and (2) unique, dedicated interfaces. Let's define the context of each type.

Standard, Modular Interfaces—System developers typically agree to employ a modular, interchangeable interface approach that complies with a "standard" such as RS-232, Mil-Std-1553, Ethernet, and Universal Serial Bus (USB).

Unique, Dedicated Interfaces—Where standard interfaces may not be available or adequate due to the uniqueness of the interface, SE designers may elect to create a unique, dedicated interface design for the sole purpose of limiting *compatibility* with other System Elements or entities. Examples include special form factors and data encryption that make the interface unique.

9.7.6 Electronic Data Interfaces

When the User's *logical* interfaces are identified in the SYSTEM or ENTITY architecture, one of the first decisions is to determine how the interface is to be implemented. Key questions include:

- Does each interface require discrete inputs and outputs?
- What function does the interface perform (data entry/output, event-driven interrupt, etc.)?
- Are the data periodic (i.e., synchronous or asynchronous)?
- What is the quantity of data to be transmitted or received?
- What are the time constraints for transmitting or receiving the data?

Electronic data communications mechanisms employ *analog* or *digital* techniques to communicate information.

9.7.6.1 Analog Data Communications *Analog mechanisms* include Amplitude-Modulated (AM) microphone and speaker-based I/O devices such as telephones and modems.

9.7.6.2 Digital Data Communications *Digital mechanisms* include *synchronous* and *asynchronous* signals that employ specific transmission protocols to encapsulate encoded information content. Digital data communications consists of three basic types of data formats: discrete, serial, or parallel.

9.7.6.2.1 Discrete Data Communications *Discrete data* consist of dedicated, independent instances state-based ON or OFF data that enable a device such as a computer to monitor the state or status condition(s) or initiate actions by remote devices. Digital discrete data represent electronic representations for various conditions or physical configuration states such as ON/OFF, initiated, complete, and OPEN/CLOSED.

Discrete data communications also include event-driven interrupts. In these applications, a unique, dedicated signal line is connected to a switch such as a hardware reset or an external device that senses threshold conditions. When the condition is detected, the device sets, toggles, or switches the discrete signal line to notify the receiving device that a conditional event such as an interrupt has occurred. In other cases, a data message may be encoded with a specific command that has a high priority that causes processing to immediately transition to satisfying a specific objective.

9.7.6.2.2 Parallel Data Communications Some systems require high-speed data communications between electronic devices. Where this is the case, parallel data communications mechanisms may be employed to improve System performance by simultaneously transmitting synchronous data over discrete lines. Here's an example:

Example 9.13

An output device may be configured to set any one or combinations of all 8 bits of discrete binary data to switch ON/OFF individual, external devices.

Parallel Data Communications mechanisms include computer address, data, and control data buses.

9.7.6.2.3 Serial Data Communications Some systems require the transmission of data to and from external systems at rates that can be accomplished using serial data communications bandwidths. Where applicable, serial data communication approaches minimize parts counts, thereby affecting PC board layouts, weight, or complexity.

Serial data communications mechanisms may be *synchronous* (meaning periodic) or *asynchronous* (meaning sporadic) depending on the application. Serial data communications typically conform to a number of serial data communications protocol standards such as RS-232, RS-422, and the Ethernet.

9.7.7 Understanding Logical and Physical Interfaces

One of the recurring themes of this book is the need to decompose complexity into one or more manageable levels. We do this by first identifying logical/functional interfaces that ask several questions:

Who interacts with whom?
What is exchanged, transferred, and translated?
When does the transference or translation occur?
Under what conditions?
What are the expected results?

Then, we translate the logical/functional connectivity into a physical interface solution that represents how and where the interface will be implemented such as PERSONNEL, EQUIPMENT HARDWARE, and SOFTWARE or combinations of these.

Analytically, system interfaces provide the mechanism for *point-to-point* connectivity. We characterize interface connectivity at two levels: (1) *logical* and (2) *physical*.

Logical interfaces—Represent a direct or indirect association or relationship between two entities. Logical interfaces establish:

1. Who (Point A) communicates with whom (Point B).
2. Under what scenarios and conditions the communications occur.
3. When and where the communications occur.

Logical interfaces are referred to as "generalized" interfaces.

Physical interfaces—Represent physical interactions between two interfacing systems or entities. Physical interfaces express how devices or components (boxes, wires, etc.) will be configured to enable Point A to communicate with Point B. Physical interfaces are referred to as "specialized" interfaces because of their dependency on specific mechanisms (electronics, optics, etc.) required to implement the interface.

Here's an example:

Example 9.14

The Internet provides a mechanism for a User such as a computer equipped with the appropriate hardware and software to communicate with an external Web site. In this context, a *logical* interface or association exists between the User and the Web site without regard to how the computer is *physically* connected via fiber optics, landlines, satellite connections, and so forth.

The preceding discussion highlighted two "levels" of connectivity. This is an important point, especially from a system design perspective involving humans. Enterprise cultures that employ the Plug and Chug … Specify-Design-Build-Test-Fix (SDBTF) Engineering Paradigm have a propensity to jump to defining the physical interfaces (Figure 2.3) before anyone has decided what the interface, in general, is to accomplish. Therefore, you must:

1. Identify which System Elements or entities must associate or interact.
2. Understand why the "need to connect."
3. Determine when the associations or interactions occur.
4. Identify performance-based outcomes—SYSTEM RESPONSES—entities on each side of the interface expect from each other.

9.7.8 Interface Definition Methodology

Principle 9.5

Logical-Physical Interfaces Principle

Logical interfaces establish associative relationships with performance-based outcomes (e.g., *what* is to be achieved). Physical interfaces establish *how* a logical interface will be implemented.

The preceding discussions enable SEs to establish a basic methodology for identifying and characterizing interfaces:

Step 1: Identify *logical* interfaces for each entity. When systems are designed, logical interfaces enable us to acknowledge that an association or relationship concerning *what* is to be accomplished, not *how*. Logical interfaces become a key part of each entity's Context Diagram and logical capability architecture.

Step 2: Identify and define *physical* interfaces for each entity. The physical implementation of a logical or generalized interface requires selection from a range of candidate solutions subject to technical, technology, cost, schedule, and risk constraints. SEs typically conduct one or more trade studies to select the most appropriate implementation.

Step 3: Standardize interfaces. Every unique interface requires specialized expertise and increases the Total Cost of Ownership (TCO) to maintain. Therefore, where practical, select widely accepted interface protocols and standards.

Principle 9.6

Interface Complexity Reduction Principle

Where practical, *minimize* the number of interfaces and to reduce complexity, technical risk, increase reliability, and reduce maintainability cost by standardizing on widely accepted and proven protocols and standards.

9.7.9 Understanding Interface Performance and Integrity

Interfaces, as an entry point, portal, or access point into a system, are *vulnerable* to threats and failures, both internally and externally. Depending on the extent of the physical interface interaction and resulting damage or failure, the interface capability or performance may be compromised or terminated. Our discussion here focuses on understanding interface design performance and integrity. Let's begin by first defining the context of an *interface failure*.

9.7.9.1 Limiting Access to System Interfaces Some interfaces require restricted access to only those devices accessible by authorized Users. In general, these interfaces consist of those applications whereby the User pays a fee for access, data security on a *need-to-know* basis, posting of configuration data for decision-making, and so on.

System access can be implemented and limited via several mechanisms. Examples include: (1) authorized log-on accounts, (2) data encryption/decryption devices and methods, (3) floating access keys, (4) personal ID cards, (5) personal ID scanners, and (6) levels of need-to-know access.

Authorized User Accounts—Employed by Web sites or internal computer systems and require a User ID and a password. If the User forgets the password, some systems allow the User to post a question related to the password that will serve as memory jogger. These accounts also must make provisions to reset the password as a contingency provided the user can authenticate themselves to the computer system via personal ID information and answers to pre-selected security questions.

Communications Security (ComSec)—The physical planning, implementation, and orchestration of communications methods and techniques such as encryption/decryption, shielding, etc. to prevent interception and exploitation by competitors or adversaries.

Data Encryption Methods and Techniques—Employed to encrypt/decrypt data during transmission to prevent unauthorized disclosure. These devices employ data "keys" to limit access. Encryption applications range from desktop computer communications to highly sophisticated banking and military implementations.

Floating Access Key—Software applications on a network that allow simultaneous usage by a subset of the total number of personnel at any given point in time. Since organizations do not want to pay for unused licenses, floating licenses are procured based on projected peak demand. When a user logs onto the application, one of the license keys or tokens is locked until the user logs out. Since some users tend to forget to log out and thereby locking other users from using the key, systems may incorporate time-out features that automatically log out a user and make the key accessible to others in the queue.

Operations Security (OPSEC)—A process that identifies critical information to determine if friendly actions can be observed by adversary intelligence systems, determines if information obtained by adversaries could be interpreted to be useful to them, and then executes selected measures that eliminate or reduce adversary exploitation of friendly critical information. (AcqNotes, 2014)

Personal ID Cards—Magnetically striped or Radio Frequency (RF) ID badges or cards assigned to personnel that allow access to facilities via security guards or access to closed facilities via badge or card readers and passwords.

Personal Authentication Scanners—Systems enabling limited access by authenticating the individual via optical scanners that scan the retina of an eye or thumbprint and match the scanned image against previously stored images of the authorized person.

Physical Security (PhySec)—The physical planning, implementation, and orchestration of ground, sea, air, or space-based assets to prevent physical entry or intrusion into a system.

"Need-to-Know" Access—Restricted access based on the individual's need-to-know. Where this is the case, additional authentication may be required. This may require compartmentalizing data into levels of access.

9.7.9.2 Interface Latency Latency is a critical interface issue for some systems, especially if one interfacing element requires a response within a specified timeframe. As an SE, you will be expected to lead the effort that determines and specifies time constraints that must be placed on interface response time. If time constraints are critical, what is the *allowable* time budget that ensures the overall system can meet its own time constraints?

9.7.9.3 What Constitutes an Interface Failure? There are differing contexts regarding what constitutes an interface failure including degrees of failure. As a general rule, Systems Engineering considers an entity or interface to be in a FAILED condition if it performs outside of its specified performance (Chapter 34). Therefore, an interface might be considered *failed* if it ceases to provide the required capability at a specified level of performance when required as part of an overall system mission. Interface failures may or may not jeopardize a system mission.

9.7.9.4 Interface Failure Types Interfaces fail in a number of ways. In general, physical interfaces can fail in at least

four types of scenarios: (1) disruption of services, (2) intrusion, (3) stress loading, (4) physical destruction, and (5) monitoring.

Disruption of Services can be created by acts of nature, animals, component reliability, poor quality work, lack of proper maintenance, and sabotage. Examples include (1) failed components; (2) cable disconnects; (3) loss of power; (4) poor data transmission; (5) lack of security; (6) mechanical wear, compression, tension, friction, shock, and vibration; (7) optical attenuation and scattering; and (8) signal blocking.

Intrusion examples include (1) unauthorized Electromagnetic Environment Effects (E^3); (2) data capture through monitoring, tapping, or listening; (3) computer viruses; and (4) injection of spurious signals. Intrusion sources include electrical storms and espionage. Intrusion prevention solutions include physical, operations, and communications security and proper shielding, grounding, and encryption.

Stress Loading includes the installation of devices that load, impede, fatigue, or degrade the quality or performance of an interface.

Physical Destruction includes physical threat contact by accident or purposeful action by an external system such as a Human system with the willful intent to inflict physical harm, damage, or destruction to a System, entity, or one of their capabilities or an intrusion by rodents.

Monitoring by external threats via listening devices.

9.7.9.5 Interface Vulnerabilities

Interface integrity can be compromised through inherent design defects, errors, flaws, or vulnerabilities. Interface integrity and vulnerability issues encompass electrical, mechanical, chemical, optical, and environmental aspects of interface design. Today, most interface vulnerability awareness tends to focus on secure voice and data transmissions and network firewalls. Vulnerability solutions include secure voice and data encryption; special, shielded facilities; armor plating; compartmentalization of tanks; cable routing and physical proximity; and operational tactics.

9.7.9.6 Interface Failure Events and Consequences

When you design system interfaces, there are a number of approaches to mitigate the occurrence of interface failures or results. In general, the set of solutions have a broad range of costs. SEs often focus exclusively on the hardware and software aspects of the interface design. Here's an example:

Example 9.15 **Optical Interface Mitigation Example**

Mirrors on vehicles provide a critical interface for the vehicle operator. The mirrors optically "bend" the operator's line of sight, thereby enabling the operator to maneuver the vehicle in close proximity to other vehicles or structures. Vehicles with side mirrors that extend away from the cab are especially vulnerable to being damaged or destroyed.

Hypothetically, one solution is to design outside mirrors with deflectors that protect the mirrors from damage. However, since vehicles move forward and backward, deflectors on the front side of the mirror may limit the operator's line of sight. To minimize damage effects on impact with external systems or objects, mirrors are designed to simply fold away. Thus, we preserve system performance while minimizing EQUIPMENT Element costs.

9.7.9.7 Interface Failure Events, Consequences, Detection, Containment, & Mitigation

Principle 9.7 **Interface Failure Mitigation Principle**

Every interface should be specified and designed to 1) detect failures, (2) contain them to their sources to prevent propagation, and (3) mitigate their effects on mission performance.

The preceding discussion illustrates traditional "Engineering the Box" versus "Engineering the System" concepts introduced in Chapter 1. The reality is: failures are not necessarily *static, self-contained events*; they can have lasting effects and consequences such as the Apollo 13 Accident addressed in Mini-Case Study 1.1.

Interface failures can result in the loss of system C2 and/or critical data disruption or loss; physical injury to SYSTEM operators, maintainers, an others; and physical damage to EQUIPMENT, property, the public, or the environment. Let's examine how interface failures may occur. When an interface fails, there are numerous types of ramifications and consequences. Example effects include electrical shorts or high voltage arcing; depressurization, implosion, or explosion; jet engine compressor blade separation (1) penetrating aircraft cabins causing injury or death or (2) causing loss of hydraulic systems and flight control that lead to a crash (United Airlines Flight 232); oil spills; and so forth. The point here is not only do SEs have to deal with the loss of an interface but also how to (1) detect and (2) contain the failure to prevent it from propagating into other nearby systems (Figure 26.8) with catastrophic results.

How do you prevent and contain interface failure effects?

- The first step begins with the System Performance Specification (SPS). Specification developers tend to

write specification requirements for *ideal* operating conditions and ignore requirements for detecting, containing, and mitigating failures (Principle 9.7).

- The second step consists of conducting a Failure Modes & Effects Analysis (FMEA) (Chapter 34) during the System Design Process based on the selected *physical* design solution, not functional solution. For mission critical applications, the FMEA should be expanded into a Failure Modes & Effects Criticality Analysis (FMECA).

Remember – the results of the two steps above are only as valid as the *competency*, *experience*, and the *diligence* of the Engineers or System Analysts performing the task.

9.8 CHAPTER SUMMARY

During our discussion of system interface practices, we identified the key objectives of interfaces, identified various types, and emphasized the importance of system interface integrity.

Our discussion of *logical* and *physical* ERs should enable you to describe how the System Elements interact at various levels of detail. The logical and physical ERs, in turn, should enable you to assimilate the relationships into an architectural framework. To facilitate the decision-making process, we established the logical ERs or associations between the System Elements via a Logical Architectural representation. Once the logical capability architecture is established, we progressed the decision-making process to the Physical Architecture representation or physical architecture.

We concluded our discussion with a focus on SOI interfaces - types of failures, failure effects, detection, mitigation, and containment.

9.9 CHAPTER EXERCISES

9.9.1 Level 1: Chapter Knowledge Exercises

1. How does the Mission System SEA differ from the Enabling System SEA?
2. Identify the System Elements of the Mission System and its Enabling System.
3. How do the System Elements differ between the Mission System and Enabling System?
4. What is the purpose of an interface?
5. What is a logical interface?
6. What is a physical interface?
7. What problem does developing logical ERs before physical ERs solve?
8. What are the two classes of OE domains?
9. What System Elements comprise a system's Higher-Order Systems domain?
10. What System Elements comprise a system's Physical Environment domain?
11. How do you graphically depict the OE architecture that includes detail interactions of the Higher-Order Systems and Physical Environment domains?
12. What are examples of Human Systems Environment threats and their sources?
13. What are examples of Natural Systems Environment threats and their sources?
14. What are examples of Induced Environment threats and their sources?
15. Boundaries for an OE are relative to what frame of reference?

9.9.2 Level 2: Knowledge Application Exercises

Refer to www.wiley.com/go/systemengineeringanalysis2e

9.10 REFERENCES

AcqNotes (2014), Operations Security, Washington, DC, Department of Defense (D0D), Retrieved on 1/18/13 from http://www.acqnotes.com/Career%20Fields/Operations%20Security.html.

ISO-Architecture (2014), *Survey of Architecture Frameworks* website, Geneva: International Organization of Standards (ISO). Retrieved on 12/13/14 from http://www.iso-architecture.org/42010/afs/frameworks-table.html.

10

MODELING MISSION SYSTEM AND ENABLING SYSTEM OPERATIONS

Every NATURAL and HUMAN SYSTEMS Environment exhibits a fundamental stimulus–response behavior pattern. For example, systems may respond positively to good news. Conversely, a system may respond negatively to threats and employ defensive tactics, pre-emptive, or retaliatory strikes. The response ultimately depends on how your system is designed and trained to respond to various types of inputs—stimuli and information—under specified types of operating conditions and constraints.

This chapter builds on the system architecture concepts in Chapters 8 and 9. Each System of Interest (SOI) coexists, encounters, engages, and interacts with external systems that comprise its OPERATING ENVIRONMENT—namely, HUMAN SYSTEMS, the NATURAL Systems, and the INDUCED Environment.

Development of systems, product, or services that operate and survive successfully in these environments requires that System Developers–System Engineers (SEs), analysts, designers, and specialty engineers - have an understanding of *how* those systems interact with and respond to *stimuli*, *excitations*, and *cues* from external systems in their OPERATING ENVIRONMENT. Responses include patterns of behavior, production of systems, products, or services, as well as by-products.

Chapter 10 provides the foundation for Model-Based Systems Engineering (MBSE).

10.1 DEFINITIONS OF KEY TERMS

- **Construct**—A graphical model or template that can be used to represent entities, architectures, operations, and capabilities.
- **Model**—A graphical and/or mathematical representation of the architectural framework; stimuli, excitations, or cues; sequence of operations and decision logic; and communications of an entity's processing, storage, Command and Control (C2), and performance capabilities.
- **Model-Based Systems Engineering (MBSE)**—"The formalized application of modeling to support system requirements, design, analysis, verification, and validation activities beginning in the conceptual design phase and continuing throughout development and later life cycle phases" (INCOSE, 2007, p. 15).
- **Transfer Function**—A mathematical expression that represents the relationship between a system's outputs, behavioral responses, as a function of its inputs within a bounded set of limitations or constraints.

10.2 APPROACH TO THIS CHAPTER

Our discussion begins with the fundamentals of system behavior. We establish a basic system behavioral model that depicts how an SOI *interacts with* and *responds to* its OPERATING ENVIRONMENT. We expand the discussion with the introduction of six behavioral interaction models that illustrate common type of system interaction models such as open loop and closed loop C2 systems, peer-to-peer data exchange interactions, status and health broadcast interactions, issue arbitration and resolution, and hostile encounter interactions.

Next, we introduce the basic concept of modeling end-to-end mission operations with a simple model of a

System Engineering Analysis, Design, and Development: Concepts, Principles, and Practices, Second Edition. Charles S. Wasson.

Companion Website: www.wiley.com/go/systemengineeringanalysis2e

Car–Driver System. To expand the basic modeling concept to accommodate interactions between entities in the model, we introduce the concepts of *control flow* and *data flow* as a graphical modeling convention. Our discussion includes (1) modeling of integrated PERSONNEL–EQUIPMENT interactions of the MISSION SYSTEM and ENABLING SYSTEM and (2) independent, autonomous systems.

Given this foundational understanding of system modeling, we introduce a series of multi-level modeling constructs or templates that enable us to model the User's Level 0 Enterprise System, SOI, and a PERSONNEL–EQUIPMENT Use Case (UC) Task Sequences Model. Most people view a capability as simply an abstract object and fail to recognize that an operational capability may be manual, semi-automated, or fully automated and consist of three Phases of Operation: Pre-Capability, Capability, and Post-Capability Operations (Table 6.2). This deficiency is reflected in specification capability requirements. To solve this deficiency, we introduce the System Capability Construct that with a phase-based sequence of operations and exception handling to accommodate error conditions.

We close the chapter with a general discussion of MBSE, its application to modeling, its misperceptions, and how to ensure its success.

10.3 THE SYSTEM BEHAVIORAL RESPONSE MODEL

During our discussion of the OPERATING ENVIRONMENT Architecture, Figure 9.3 served as a high-level model to illustrate an SOI's interactions with its OPERATING ENVIRONMENT. To see how this interaction occurs, let's investigate a simple behavioral response model.

Principle 10.1

SYSTEM RESPONSES Principle

Every system responds to stimuli, excitations, and cues in its OPERATING ENVIRONMENT with behavioral actions, products, by-products, services, or combinations thereof.

If we treat the SOI shown in Figure 9.3 as a stimulus–response box, we can create a simple model that represents how the SOI responds to its OPERATING ENVIRONMENT as shown in Figure 10.1. The system model consists of the PHYSICAL Environment Domain (Figure 9.4) and the SOI, both of which are controlled by HIGHER ORDER Systems. The HIGHER ORDER System provides an Organization, allocates Roles and Missions, imposes Operating Constraints, and provides Resources to the SOI. Elements of the PHYSICAL Environment Domain—such as HUMAN SYSTEMS, INDUCED, AND NATURAL SYSTEMS Environments—provide input stimuli into the SOI as well as affect its operating capabilities and performance.

Stimuli, excitations, and cues serve as inputs such as information, data, interrupts, and actions to the SOI's Sensory Receiver. The Sensory Receiver decodes the stimuli and information and supplies them as inputs to the Processor. The Processor adds value to the data by performing User-defined actions.

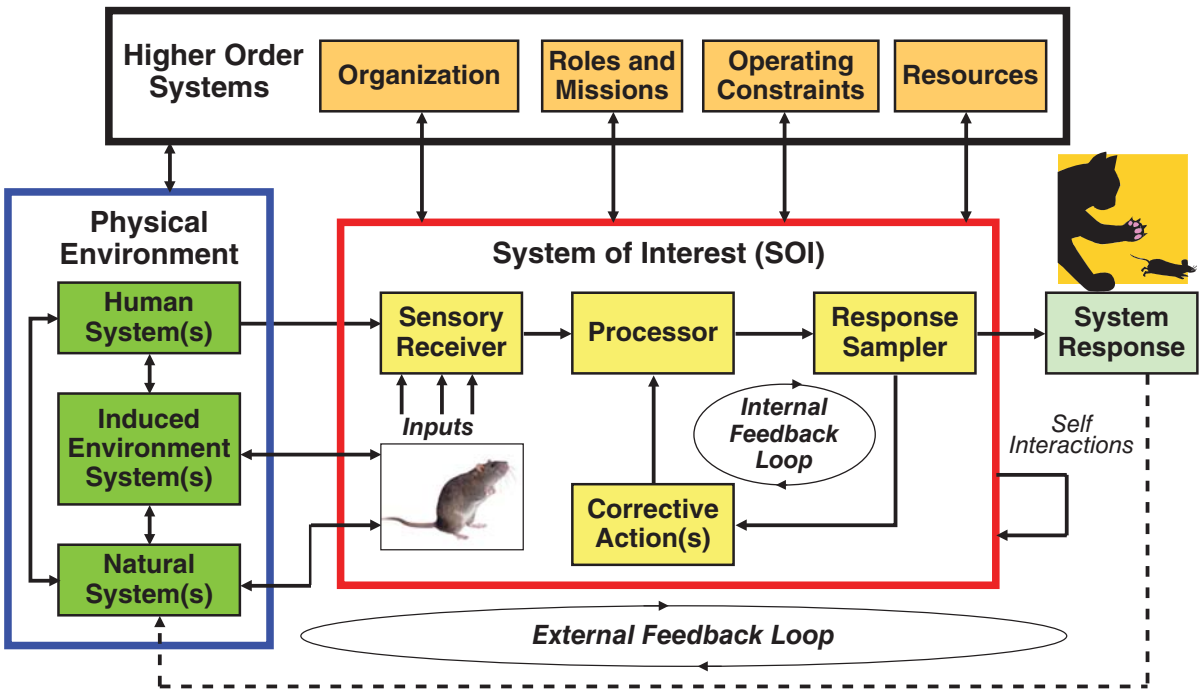


Figure 10.1 System Behavioral Responses Model

The Response Sampler samples the results of the value-added processing and compares those results to OPERATING CONSTRAINTS—namely, mission tasking—established by a HIGHER ORDER System. Based on the results of the Comparison, Corrective Actions are initiated as feedback—Internal Feedback Loop—to the Processor. When the processing is deemed acceptable relative to the OPERATING CONSTRAINTS, SYSTEM RESPONSES are produced. SYSTEM RESPONSES are then fed back to the OPERATING ENVIRONMENT thereby completing the External Feedback Loop.

An SOI such as electromechanical engine, nuclear plant generator, jet engine, and electrical power system are examples that could be represented by this model. As such, we could create a mathematical model to represent the Input–Output Transfer function performed by the SOI.

10.3.1 Several Key Points

Figure 10.1 illustrates several key points regarding system interactions with its OPERATING ENVIRONMENT:

- **Input Data**—External stimuli or data may occur as a *triggering event* such as communications data, an observation, or transfer of information—or as "trend data" over time.
- **Measured or Conditioned System Response**—Internal Control Loop that produces a measured response appropriate for the stimuli, excitations, and cues.
- **System Transfer Function**—The Sensory Receiver, Processor, Response Sampler, and Corrective Action(s) form a system transfer function that shapes the System Response.
- **System Latency**—The time required from the SOI to respond to external stimuli and information until a system response is produced is referred to as system responsiveness, system response time, or system throughput.
- **System Interactions**—Stimuli or data, composed of cues, information, and behavior, as well as the System Response to its OPERATING ENVIRONMENT form a *closed loop* of system interactions.

The preceding discussion focused on a simple throughput model representing an SOIs encounters and interactions with its OPERATING ENVIRONMENT consisting of HUMAN, INDUCED, and NATURAL SYSTEMS Environments. As previously introduced, these encounters and interactions may be characterized as friendly, cooperative, defensive, benign, aggressive, harsh, or hostile.

This leads us to another key concept, Example Constructs of System Behavioral Encounters and Interactions.

10.4 SYSTEM COMMAND & CONTROL (C2) INTERACTION CONSTRUCTS

If we observe and analyze the pattern of interactions between HUMAN SYSTEMS, we can identify some of the primary interaction constructs. In general, examples of common interactions of most friendly systems include the following.

10.4.1 Open Loop Command Interactions Construct

The Open Loop Command Interactions Construct shown in Figure 10.2 represents a simple system in which a HIGHER ORDER System issues C2 Tasking stimuli to the SOI to perform a task and respond to its OPERATING ENVIRONMENT. Observe that no feedback concerning task completion or success occurs.

10.4.2 Closed Loop C2 Interactions

The Closed Loop C2 Interactions Construct shown in Figure 10.3 corrects the feedback deficiency of the Open Loop Command System. In this case, the SOI responds to the HIGHER ORDER SYSTEMS tasking by providing continuous

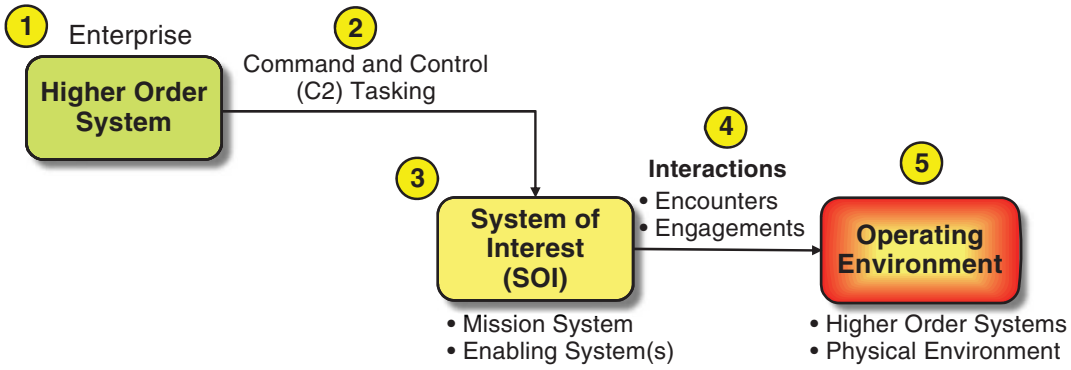


Figure 10.2 Open Loop C2 System Examples

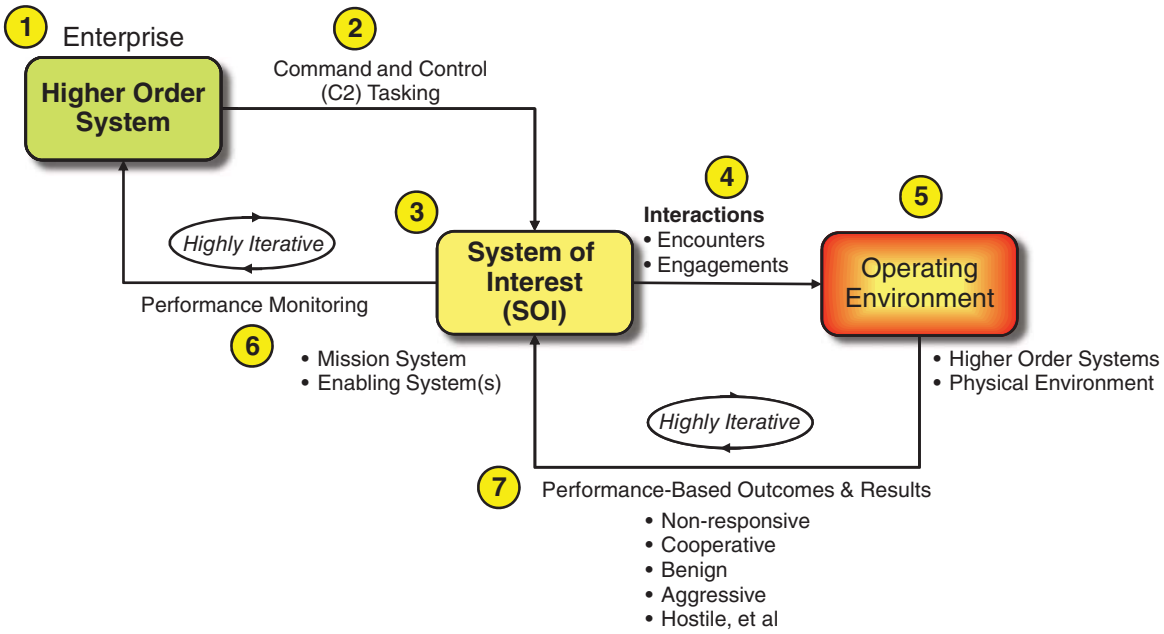


Figure 10.3 Closed Loop C2 System Examples

Performance Monitoring concerning task progress, completion, and success. As part of this process, the SOI monitors the OPERATING ENVIRONMENT's response and feeds that back to the HIGHER ORDER System as part of Performance Monitoring. Example applications of this construct include:

- Enterprise Systems C2– Inter and Intra-organizational tasking
- Enterprise to SOI C2 - mission tasking
- MISSION SYSTEM AND ENABLING SYSTEM C2-PERSONNEL-EQUIPMENT interactions
- EQUIPMENT C2–Subsystem-to-Subsystem interactions

10.4.3 Operational Status and Health (OS&H) Broadcast System Interactions Construct

The Operational Status and Health (OS&H) Broadcast System Interactions Construct shown in Figure 10.4 illustrates a simple system that merely provides *synchronous* (periodic) or *asynchronous* (random) data to a HIGHER ORDER System. The HIGHER ORDER System may or may not acknowledge receipt of the data. Here's an example.

Example 10.1 **Remote Broadcast Weather System**

Remote weather collection systems (SOIs) are installed in the vicinity of an airport to transmit *synchronous* weather data 24 hours per day and 7 days per week at a 1 Hz rate to a HIGHER ORDER SYSTEM at the airport for processing and dissemination to aircraft.

Example 10.2 **PERSONNEL-EQUIPMENT Situational Assessment**

Part of Monitoring, Command, and Control (MC2) requires continuous updates that present Situational Assessment of (1) system performance, and OS&H status and (2) OPERATING ENVIRONMENT conditions. The User – operator or maintainer – has a "need to know" Situational Assessment from the EQUIPMENT to perform their mission tasking. Likewise, the EQUIPMENT's internal MC2 requires Situational Assessment information from SUBSYSTEM ASSEMBLY, SUBASSEMBLY components such as sensors to report to the User to MC2 the EQUIPMENT.

10.4.4 Peer-to-Peer Data Exchange System Interactions

The Peer-to-Peer Data Exchange System Interactions Construct shown in Figure 10.5 represents an interchange between two peer-level, cooperative systems. Here's a hypothetical encounter and interaction scenario:

1. SYSTEM A transmits a Request for Information (RFI) /Status from SYSTEM B.
2. SYSTEM B *may or may* not - dashed line - transmit an Acknowledgement of receipt of the request.
3. SYSTEM B transmits Data to SYSTEM A.
4. SYSTEM A may or may not transmit an Acknowledgement of receipt of the data.

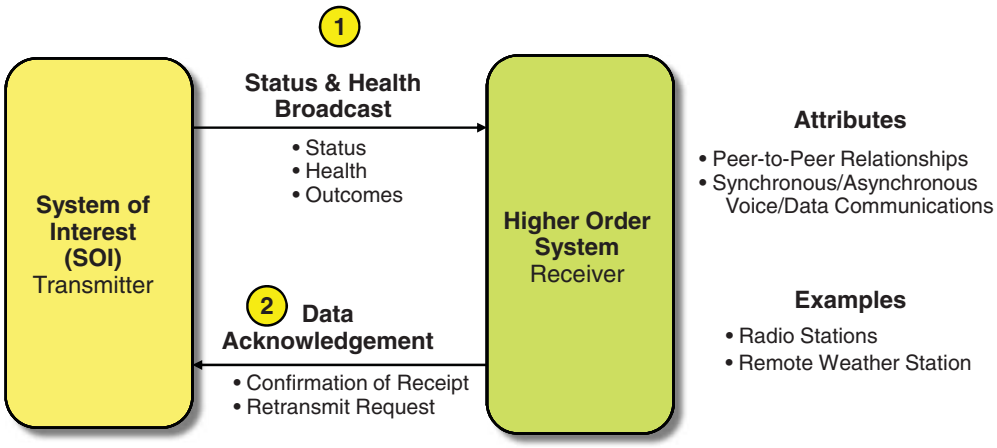


Figure 10.4 Status and Health Broadcast System Interactions Example

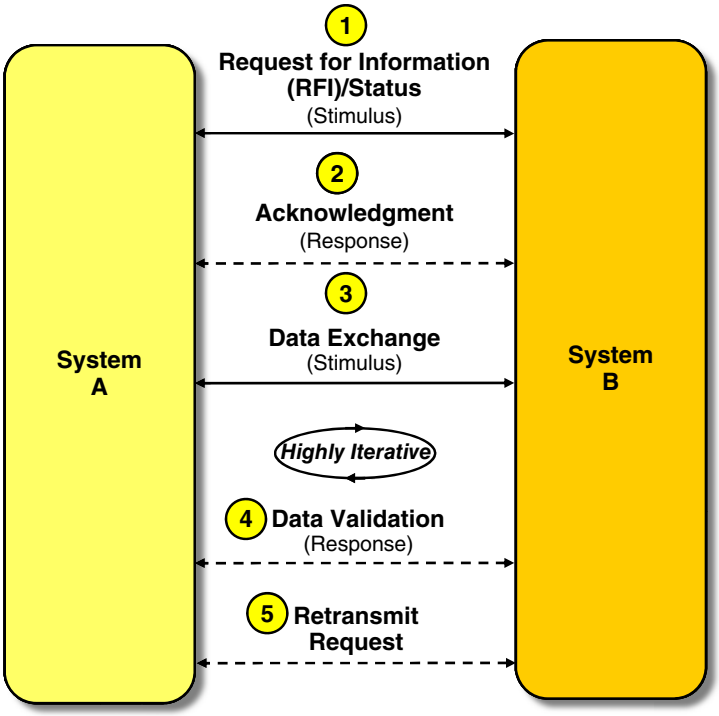


Figure 10.5 Peer-to-Peer Data Exchange Interactions Construct

5. If SYSTEM A does not receive a response or determines the Data is invalid due to errors, it may send a Retransmit Request.

An important point here is to recognize that the phrase *may or may not* is system dependent. If you believe your SYSTEM should Acknowledge requests and receipts, then make it a requirement. Here's an example of peer-to-peer systems interactions.

Example 10.3

Bank Card Transaction Example

A bank card User arrives at an Automated Teller Machine (ATM) location. The ATM displays a welcome and instructs the User to insert their card into the ATM. The ATM reads the card information and returns it to the User. If the account is valid, the ATM issues a Request for Information (RFI) to the User to enter a password to access account information.

On verification of the User's password - Acknowledgement, account access is granted. The ATM issues an RFI to the User concerning what type of transaction the User needs to perform. The User selects from a set of transaction options and enters the amount to be transacted – Data Exchange. The ATM Acknowledges the User's response by parroting the data on the display as it is selected or entered.

The ATM processes the transaction, provides a receipt of the transaction and/or money, and inquires (RFI) if the User needs to perform any additional transactions. If not, the ATM thanks the User for their business – Data Exchange, asks them to return again, and displays a welcome message for the next User.

Observe the series of acknowledgements in this Peer-to-Peer User–ATM information and data exchange.

10.4.5 Issue Arbitration/Resolution System Interactions Construct

Issue Arbitration/Resolution System Interactions Construct shown in Figure 10.6 illustrates a conflict that has been elevated by an SOI to a Higher Order System. Here's a hypothetical encounter and interaction scenario:

1. The SOI issues a Request for Arbitration/Resolution to a HIGHER ORDER System.
2. The HIGHER ORDER System may or may not Acknowledge the request.
3. The HIGHER ORDER System responds by issuing a Resolution Decision as a corrective action to resolve the matter.

Example 10.4

Conflict Resolution Example

1. Within an Enterprise, PERSONNEL may encounter conflicts related to other personnel, working hours, etc. As a result, the individual elevates the matter to their manager—HIGHER ORDER System.
2. A system may require three redundant computers to make a decision. Results from two of the computers are polled and used to validate a third computer's decision. If results from the two polled computers conflict, the third computer has to make a decision that may involve determination of one of the conflicting computers has failed. The question is: *what if results from the other two are non-conflicting, and the supervisory computer is the one that has failed*? The architecture's decision control logic has to be provided to accommodate this situation.

10.4.6 Hostile Encounter Interactions Construct

Issue Arbitration/Resolution System Interactions shown in Figure 10.7 represents a hostile encounter between an SOI and an external system in its OPERATING ENVIRONMENT. Here's a hypothetical encounter and interaction scenario:

1. SYSTEM A performing an Aggressor Role initiates a Hostile action against SYSTEM B.
2. SYSTEM B acting in a Defensive Role responds with Countermeasures as a warning.
3. SYSTEM A responds to SYSTEM B's Countermeasures with a Counter-Countermeasures (CCM) response.

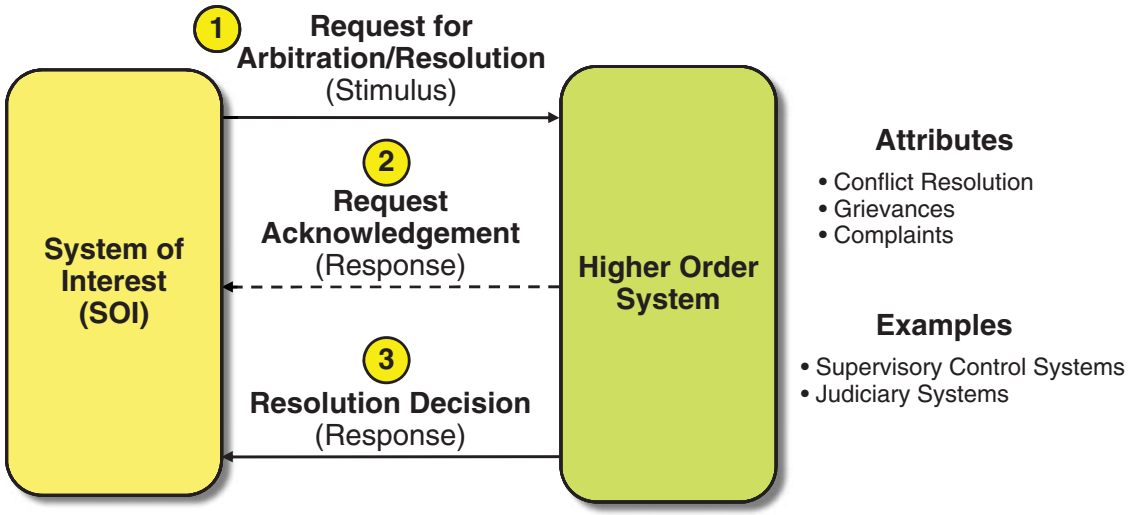


Figure 10.6 Issue Arbitration/Resolution System Interactions Construct

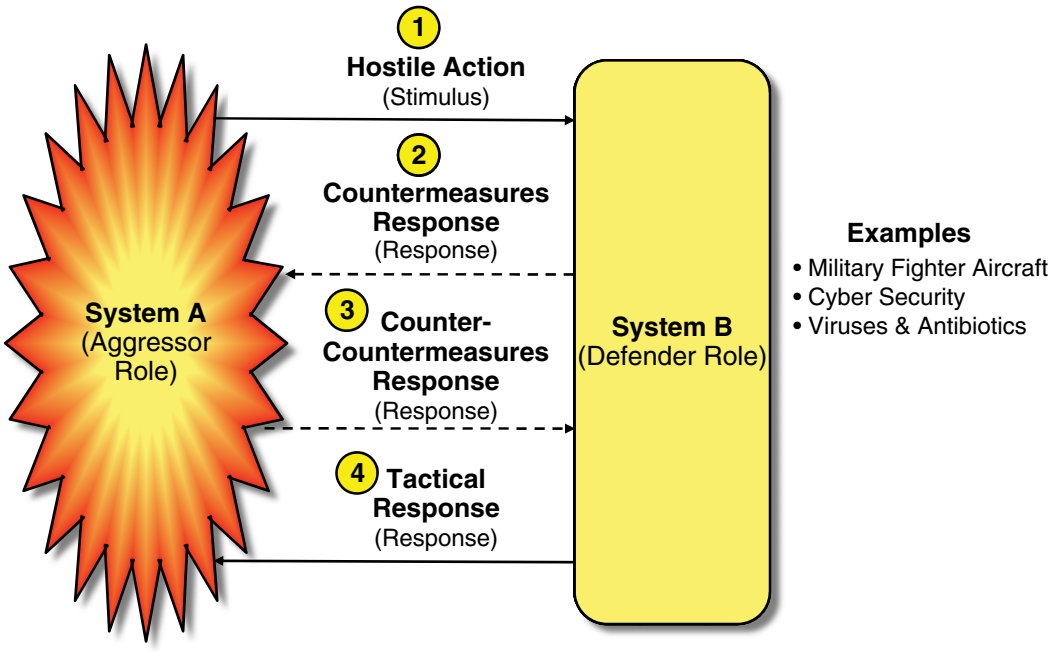


Figure 10.7 Hostile Encounter Interactions Construct

4. Sensing attack, SYSTEM B switches to an Aggressor Role and returns a Tactical Response that forces SYSTEM B to retreat.

10.5 MODELING SYSTEM CONTROL FLOW AND DATA FLOW OPERATIONS

One of the first steps in developing a model is to create a simple, high-level version. As an example, let's create a model for an automobile driver with a mission to go to and from work each day. Figure 10.8 provides an illustrative example.

1. The Driver enters the car at its Initial State, thereby creating the Car–Driver System. The Driver performs Pre-Mission Operations to check out the vehicle to verify everything is working properly.
2. On completion of the Pre-Mission Operations, control flow sequences to Operation Mission Operations representing the drive to the Driver's mission destination.
3. During the drive, the Car–Driver System drives defensively and *interacts* with its OPERATING ENVIRONMENT—HIGHER ORDER SYSTEMS Domain and PHYSICAL ENVIRONMENT Domain.
4. On arrival, the Driver exits the vehicle and locks its doors. When ready to return home, they perform Operations 1.0 through 3.0 again.

This illustrated example represents a simple operations model. However, it does not illustrate the interactions that occur between the Driver and the Car or the tasks each is performing that enable them to interact with each other. This leads to the introduction of our next topic, Operational Control Flow and Data Flow Concepts.

10.5.1 Operational Control Flow and Data Flow Concepts

When you analyze systems, two types of *flows* occur: (1) *control flow* or *workflow* and (2) *data flow*:

- *Control flow or workflow* enables us to understand *how* system operations are sequenced.
- *Data flows* enable us to model information, data, or energy exchanges between system entities such as electrical, optical, or mechanical.

To illustrate these points, let's investigate the control flow and data flow aspects further.

Figure 10.9 depicts intersecting arrows on the left side that represent the graphical convention for control flow and data flow. The top-down vertical arrow represents *control flow*; the horizontal arrow represents *data flow* exchanges between entities.

Applying the *control flow* or *workflow* to Figure 10.8:

- Sequences from the Initial State to Operation 1.0 Pre-Mission Operations

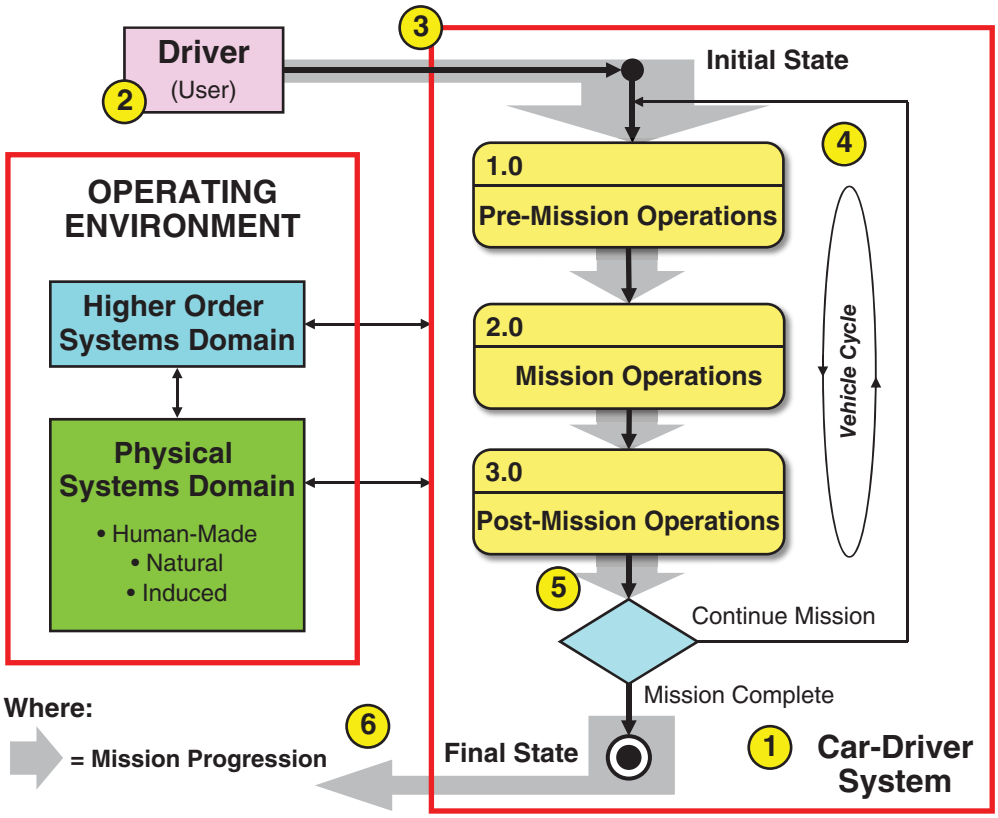


Figure 10.8 High-level Car–Driver System Operations Model

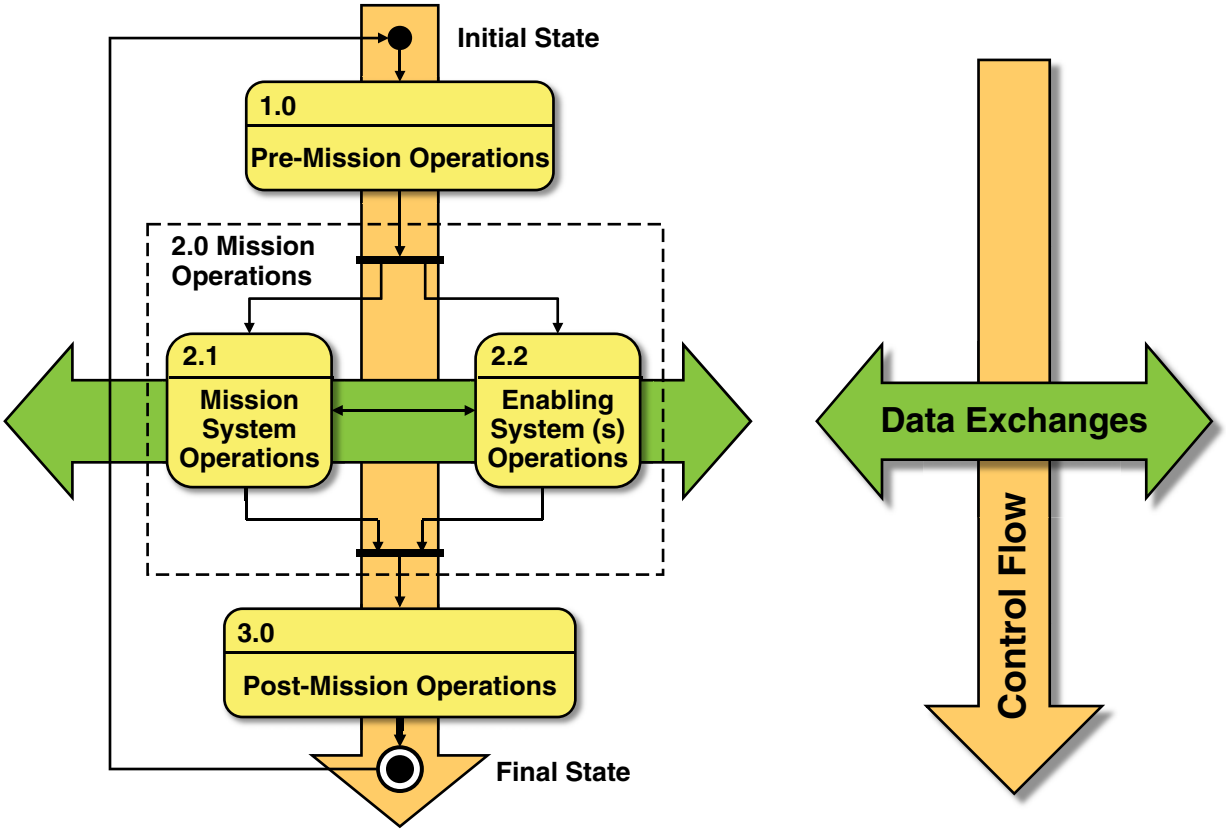


Figure 10.9 Control Flow–Data Flow Graphical Convention

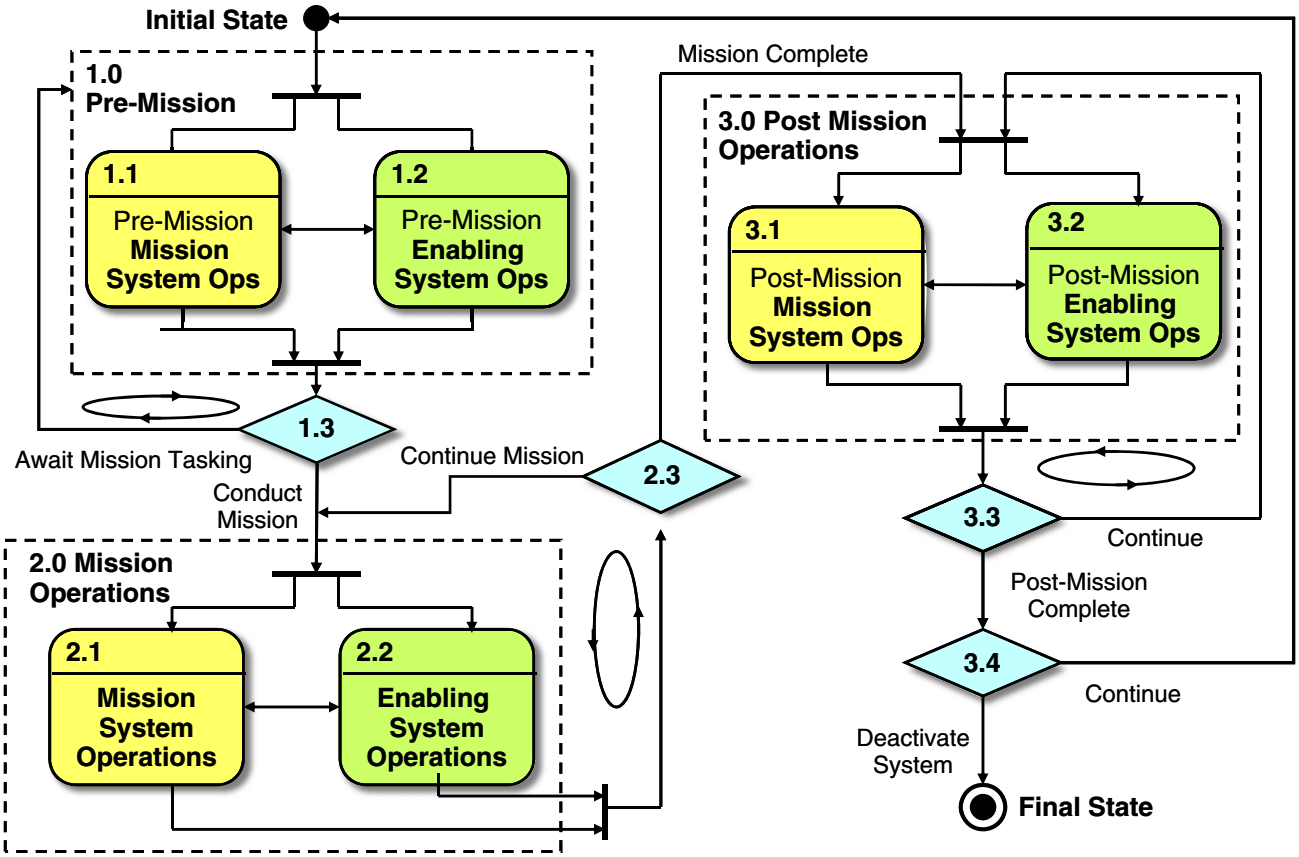


Figure 10.10 Concurrent Multi-phase Operations Model

- Sequences from Operation 2.0 Mission Operations
- Sequences from Operation 3.0 Post-Mission Operations
- Sequences to the Final State

Cyclical systems—Do Until— such as electronic EQUIPMENT when *activated* may employ a feedback loop as indicated by the line from the Final State decision block back to Initial State. From a *data flow* perspective, information is exchanged between Operation 2.1 MISSION SYSTEM Operations and Operation 2.2 ENABLING SYSTEM Operations as a part of Operation 2.0 MISSION Operations.

If we expand the MISSION SYSTEM and ENABLING SYSTEM operational interactions to include all Phases of Operation—Pre-Mission, Mission, and Post-Mission—Figure 10.10 emerges. Each of the Phases of Operation is expanded into concurrent MISSION SYSTEM and ENABLING SYSTEM operations.

Author's Note 10.1

Referring to Figure 10.10, observe 2.0 Mission Operations interactions between the MISSION SYSTEM and its ENABLING SYSTEMS. This interaction represents Figure 6.2 Operation 10 Conduct Mission Operations and Operation 11 Provide Mission Oversight and Support. In the case of an airline, the aircraft – MISSION SYSTEM – flying passengers to another city is under the MC2 (Figure 10.3) of air traffic controllers – ENABLING SYSTEMS.

10.5.2 Modeling Multiple Concurrent Operations

The preceding discussions focused on the SOI executing a sequential end-to-end *control flow* until terminated due to a common time-out, resource depletion, or failure. We expand on that concept to illustrate a more complex system that has multiple SOIs working together to achieve an overall Enterprise mission. Examples include a restaurant, a car dealership, or an airline. Figure 10.11 provides an example.

Example 10.5

Customer Retail Enterprise Example

Let's assume a business such as a retail store – MISSION SYSTEM - in a shopping plaza has normal business hours – Mission Phase of Operations - from 10:00 AM until 9:00 PM.

In preparation for the next business day – Enterprise Pre-Mission Phase of Operations, merchandise may be replenished, moved.

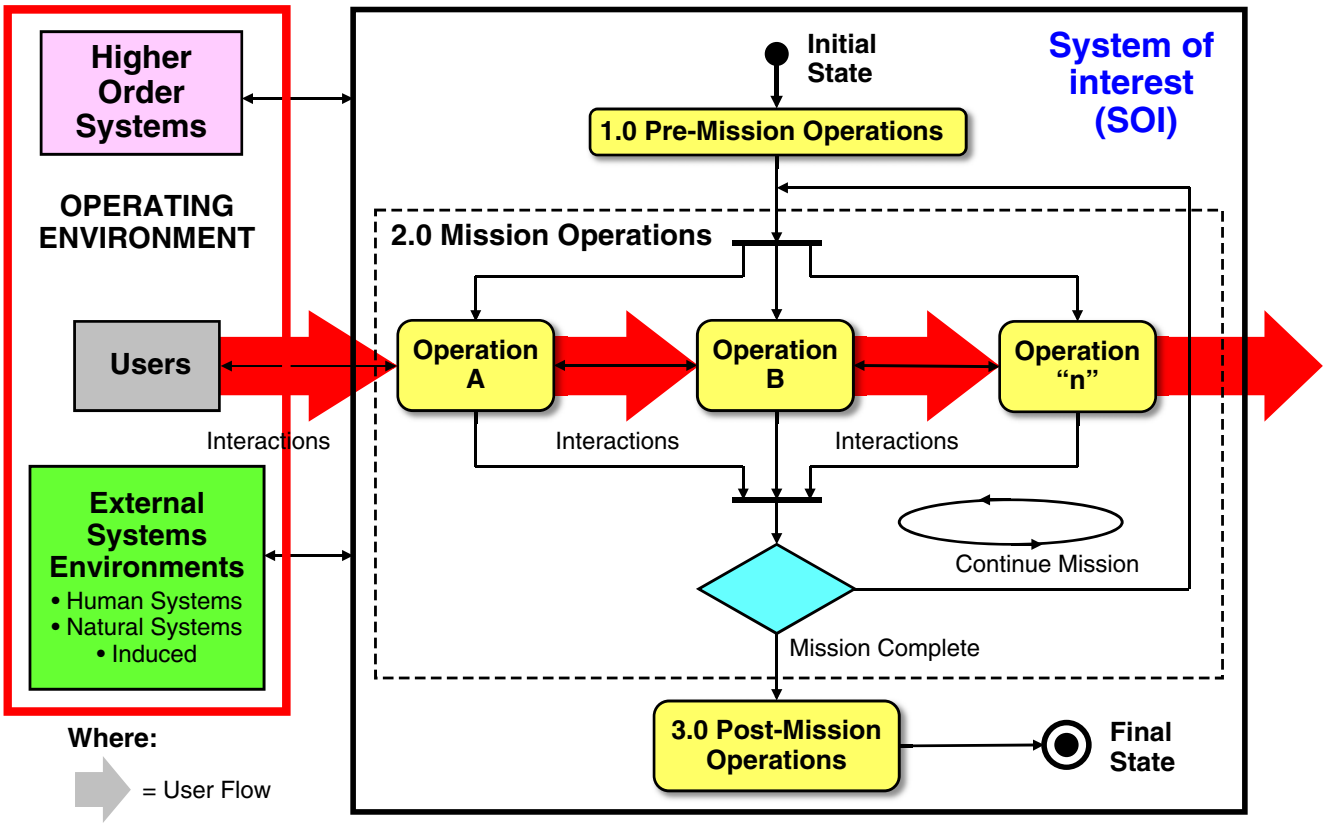


Figure 10.11 MISSION Phase Operations: Multi-user Construct

At 10:00 AM, the store's doors open for business – Enterprise Mission Phase of Operations. At that time each department in the store is staffed with trained sales clerks - ENABLING SYSTEMS PERSONNEL - to assistance customers – MISSION SYSTEMS - in purchasing merchandise.

The customer's workflow consists of traveling to the store – Pre-Mission Phase of Operations, purchasing merchandise – Mission Phase of Operations, and leaving – Post Mission Phase of Operations.

Customers as MISSION SYSTEMS enter the store and have a *horizontal* control flow shopping experience that goes from Operation A to Operation B to Operation "n" before existing the store. Observe the orthogonally of the store's *vertical* control workflow versus the *horizontal* control flow of the customer – intersecting Customer and Enterprise Mission Phases of Operations.

At 9:00 PM, the store completes its Enterprise Mission Phase of Operations, locks the doors, and initiates its Post-Mission Operations such as clean-up and repeats its Pre-Mission phase of Operations for the next business day.

The example above exemplifies automobile dealerships, restaurants, aircraft, hospital, schools, and other types of Enterprise operations. Understand this concept because it provides the foundation for modeling systems, product, or services in the discussions that follow.

The example above MISSION SYSTEM – ENABLING SYSTEM interactions illustrate a key consideration to remember when developing systems—customer interactions and satisfaction. Carlzon (1989) in his text *Moments of Truth* highlights that every interaction a stakeholder such as a User or End User has with your system, product, or service is a "moment of truth" that leaves lasting impressions. This concept exemplifies the Mission System – Enabling System Supply Chain illustrated in Figure 4.1 and its Fitness for Use standards – merchandise – and Customer Acceptance Criteria.

Example 10.6

System, Product, and Service Moments of Truth

- When a User interacts with your system or product such as a computer to turn power on, attempt to load software, find technical answers in operator manuals, create reports, it is a *moment of truth.*
- When that same User contacts the computer manufacturer for service – answers to *how to* questions, it is a moment of truth. Did you talk with a live technical support person or did you get "punch the buttons" recorded messages to submit a question to an obscure community blog – *moments of truth.*

This entire chain of events are *moments of truth* that leave lasting impressions that impacts future purchases. When you sell your products, think about these interactions and their consequences. This is especially true for hardware and software components when they are installed in your system or you connect via e-mail of the Internet. This brings us to our next concept, System Compatibility and Interoperability.

10.5.3 System Interaction Compatibility and Interoperability

System Encounters and Interactions Principle

Principle 10.2 System encounters and interactions with external systems in its OPERATING ENVIRONMENT during an engagement may be cooperative, friendly, benign, competitive, adversarial, hostile, or combination of these.

When two or more systems interact, we refer to the interaction as an *engagement* or *encounter*. Engagements can be characterized with a number of terms. The effects or results of the engagement can be described as positive, benign, negative, damaging, or catastrophic, depending on the system roles, missions, and objectives. Generally, the effects or outcomes can be condensed into a key question. *Was the engagement compatible and interoperable from each system's perspective*? Let's explore the context of both of these terms.

Compatibility and Interoperability Principle

Principle 10.3 Interfaces must be *compatible* in terms of *form* and *fit*. Where applicable, they must be *interoperable* to be able to intelligibly encode/decode, interpret, process, and exchange data.

Compatibility often has different contextual meanings. We use the term in the context of physical capability—form and fit—such as mechanical connections, electrical signals, current capacity, memory storage, and cargo. Observe that we used the operative term *capability*. Possessing a capability does not mean the engagement or interface will be *interoperable*.

When interfaces require information or data exchange, compatibility is a *necessary* but *insufficient* condition in terms of both sides of the interface being able to communicate with each other and understand what is being communicated. That requires *interoperability*.

To illustrate the application of the terms *compatibility* and *interoperability*, consider the following examples of international aircraft voice communications and electronic data communications.

Language Compatibility Versus Interoperability

Example 10.7 Two people from different countries speaking different languages may attempt to communicate—an interaction or engagement between system entities. We could say their voice communications are compatible—transmitting and receiving. However, due to a lack of interoperability, they are unable to decode, process, assimilate, or "connect" what information is being communicated. Due to the criticality of public flight safety, the International Civil Aviation Organization (ICAO) Assembly in October 2013 adopted Resolution 38/8 – "*Proficiency in the English language used for radiotelephony communications*" as the standard language are be used in communications between Air Traffic Controllers (ATCs) and pilots for international travel (ICAO, 2015).

Electronic Communications Compatibility Versus Interoperability

Example 10.8 RS-232 data communications interface between two systems use a standard cable and connectors for transmitting and receiving data. Thus, the interface is physically compatible. However, the data port may not be *enabled* or the receiving system's software capable of decoding and interpreting the information—*interoperability*.

10.5.4 System Interactions Synthesis

As an SE, you must learn to synthesize these interactions in terms of an overall system solution. Figure 10.12 provides an illustration.

Here, we have a diagram that captures the high-level interactions between the SOI, HIGHER ORDER SYSTEM, and the OPERATING ENVIRONMENT. The SOI is illustrated via an Ishikawa or "fishbone" diagram. The diagram includes System Elements that are performance effector factors that must integrate harmoniously to achieve the mission objectives. In combination, the SOI elements produce the SYSTEM RESPONSES Element, which consists of behavior, products, by-products, and services.

In operation, the SOI responds to C2 guidance and direction from the Higher Order Systems Element that consists of ORGANIZATION, ROLES AND MISSIONS, OPERATING CONSTRAINTS, and RESOURCES System Elements. Based on this direction, the SOI System Elements interact with the OPERATING ENVIRONMENT and provide system responses back to the OPERATING ENVIRONMENT and the HIGHER ORDER SYSTEMS Domain Element.

10.5.5 Modeling Operations Summary

During our discussion of system interactions with its operating environment, we described a system's interactions

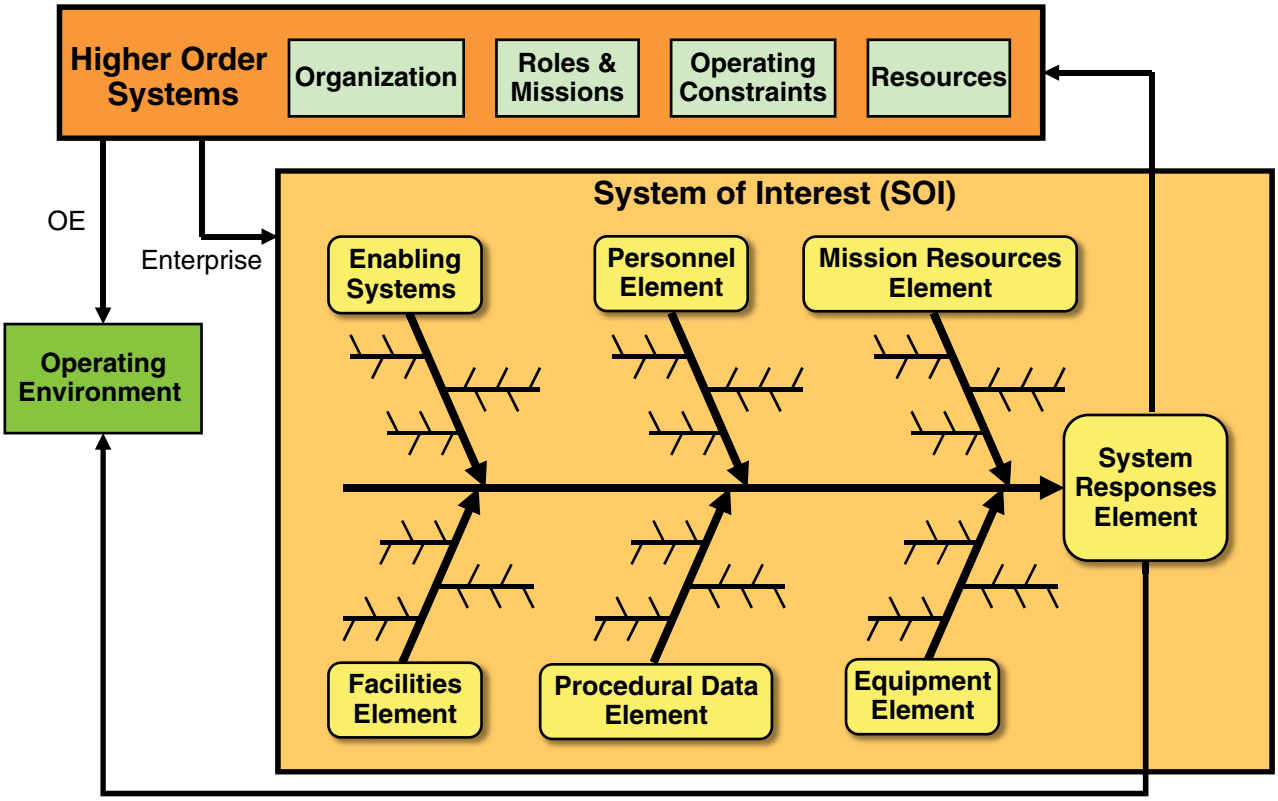


Figure 10.12 Ishikawa Diagram of System Element Contributions to Overall System Performance

via the Behavioral Responses Model. A system's responses are driven by strategic and tactical interactions related to opportunities and threats in the environment. Systems generally interact with cooperative, benign, competitive, or aggressor systems. Based on those responses, we indicated how a system might employ countermeasures and counter-countermeasures to distract, confuse, defend or interact with other systems. We concluded our discussion by highlighting the context of the OPERATING ENVIRONMENT based on the SOI perspective.

10.6 MODELING MISSION SYSTEM AND ENABLING SYSTEM OPERATIONS

People often erroneously ingrained to think of System Engineering and Development as simply the identification and decomposition of functions into arbitrary architectural structures. Beyond that point, magic happens! Both aspects contribute to *ad hoc, endless loop*, Specify-Design-Build-Test-Fix (SDBTF) Engineering Enterprises that are prone to non-compliance with specification requirements, latent defects - design flaws, errors, and deficiencies, and so forth. Sometimes, *uninformed* System Acquirers and System Developers impose *unreasonable* and *impractical* schedules and budgets that force these types of behaviors. Where practicality and informed decisions prevail, there are better ways of producing systems without non-compliances, cost overruns, and missed schedules.

A key tenet of Systems Engineering and Development (SE&D) is making *informed*, *fact-based* decisions related to multi-level specification performance requirements and multi-level system architecture development. The mechanism for supporting informed decisions is modeling the behavioral interactions of the SOI MISSION SYSTEMS, ENABLING SYSTEMS, and their OPERATING ENVIRONMENTS with validated models (Chapter 33).

Some Enterprises model system behavior. Customers are impresses! On inspection, you soon discover that the modelers exist autonomously in a domain separate from software developers. Since software modeling of a system involves algorithmic models as mathematical, logical, and physical representations of systems, common sense says that those same models should be used, where applicable, in the actual software executing in the deliverable system or product. Unfortunately, that is often not the case. The potential dichotomy is as follows: modeling software is used to make SE technical performance decisions; the deliverable software was created independently with little or no collaboration. *Does this sound like sound engineering practices*?

As an introductory overview to Behavioral Model concepts, let's walk through the process and proceed into some of the details. Although there are numerous ways of modeling systems, we will employ a simple approach as the basis for introducing the topic. Remember that we are creating models of the actual, multi-level system as it evolves.

The models we create of the actual system are context based. To illustrate, consider the following system contexts of operation:

- Phase of Operation—Pre-Mission, Mission, and Post-Mission to Modes of Operation
  - Modes of Operation accommodate UCs
    - UCs are implemented by Operational Tasks, each representing an Operational Capability
      - Operational Capabilities are constrained by Allowable and Prohibited
        - Operational Capabilities translate into Specification Requirements

Each of these behavioral modeling contexts applies to and within the SOI, MISSION SYSTEM, and ENABLING SYSTEMS. Given this framework, let's begin with Figure 10.13.

10.6.1 Behavioral Modeling Details

Systems, products, and services can be modeled in any context and level of abstraction. The discussions that follow will illustrate four types of models:

- Generalized User's Multi-Mode Model (Figure 10.13)
- System Element Architecture (SEA) Model (Figure 10.14)
- PERSONNEL–EQUIPMENT UC-Based Task Sequence Model (Figure 10.16)
- System Capability Construct (Figure 10.17)

Let's explore each of these further.

10.6.2 Generalized User's Level 0 Life Cycle Phases Model

One of the first steps in modeling SOI MISSION SYSTEM and ENABLING SYSTEM Operations is to establish its context within the User's Level 0 Enterprise System as shown in Figure 10.13. This model is a construct that can be used to represent SOI Pre-Mission, Mission, and Post-Mission operations and interactions between the MISSION SYSTEMS, ENABLING SYSTEMS, and their OE.

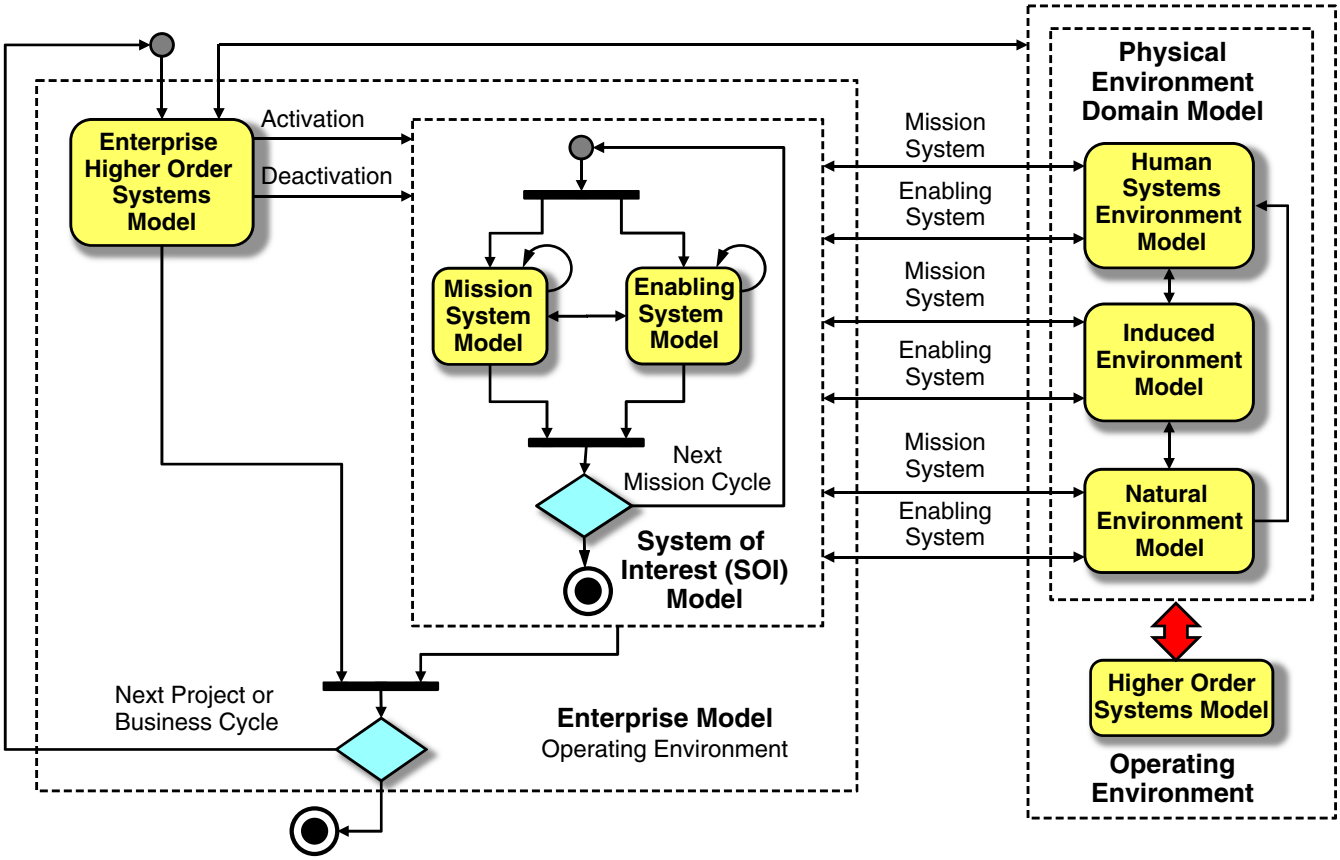
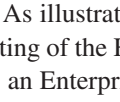


Figure 10.13 Generalized SOI Interactions Construct

As illustrated, an Enterprise-Level model is created consisting of the HIGHER ORDER SYSTEMS Domain and the SOI. As an Enterprise asset, the SOI is *activated* and *deactivated* by the HIGHER ORDER SYSTEMS Domain. *Activation* refers to HIGHER ORDER SYSTEMS authority to make assignments or issue Task Orders (TOs) to conduct missions based on its System Elements—ROLES AND MISSIONS, OPERATING CONSTRAINTS, and RESOURCES.

On activation, the SOI's MISSION SYSTEM and ENABLING SYSTEM models begin with an Initial State—gray circle—and cycles continuously in the current mission until it has been completed. The 270 degree arch with arrowhead at one corner of the model symbolizes cycling of their respective Pre-Mission, Mission, and Post-Mission Phases of Operation until the mission is complete. On completion, *control flow* transitions to the Final State—double circle with gray center.

During MISSION Operations, the SOI Model interacts with external systems in its OPERATING ENVIRONMENT—HUMAN SYSTEMS, INDUCED, and NATURAL ENVIRONMENT System Models. Each of these PHYSICAL ENVIRONMENT Domain models performs under the authority of the OE's HIGHER ORDER SYSTEMS Model.



Author's Note 10.2

As a reminder, please note that due to space restrictions, the bi-directional SOI–OPERATING ENVIRONMENT interaction arrows touching the SOI's dashed boundary represent connectivity to the MISSION SYSTEM and ENABLING SYSTEM models.

A review of the figure leads to a question, *what is the structure of the MISSION SYSTEM and ENABLING SYSTEM Models*? This leads to introduction of our next model, the Generalized SEA Architecture Model.

10.6.3 Generalized SEA Model

Earlier in Figure 8.13, we introduced the SEA that illustrated the relationships among MISSION SYSTEM or ENABLING SYSTEM Elements. The architecture provides the infrastructure for the Generalized SEA Model shown in Figure 10.14. This model serves as template for the MISSION SYSTEM and ENABLING SYSTEM models.

The SEA Model is comprised of the six System Elements that operate concurrently. *Data flow* between each Element's model occurs horizontally.

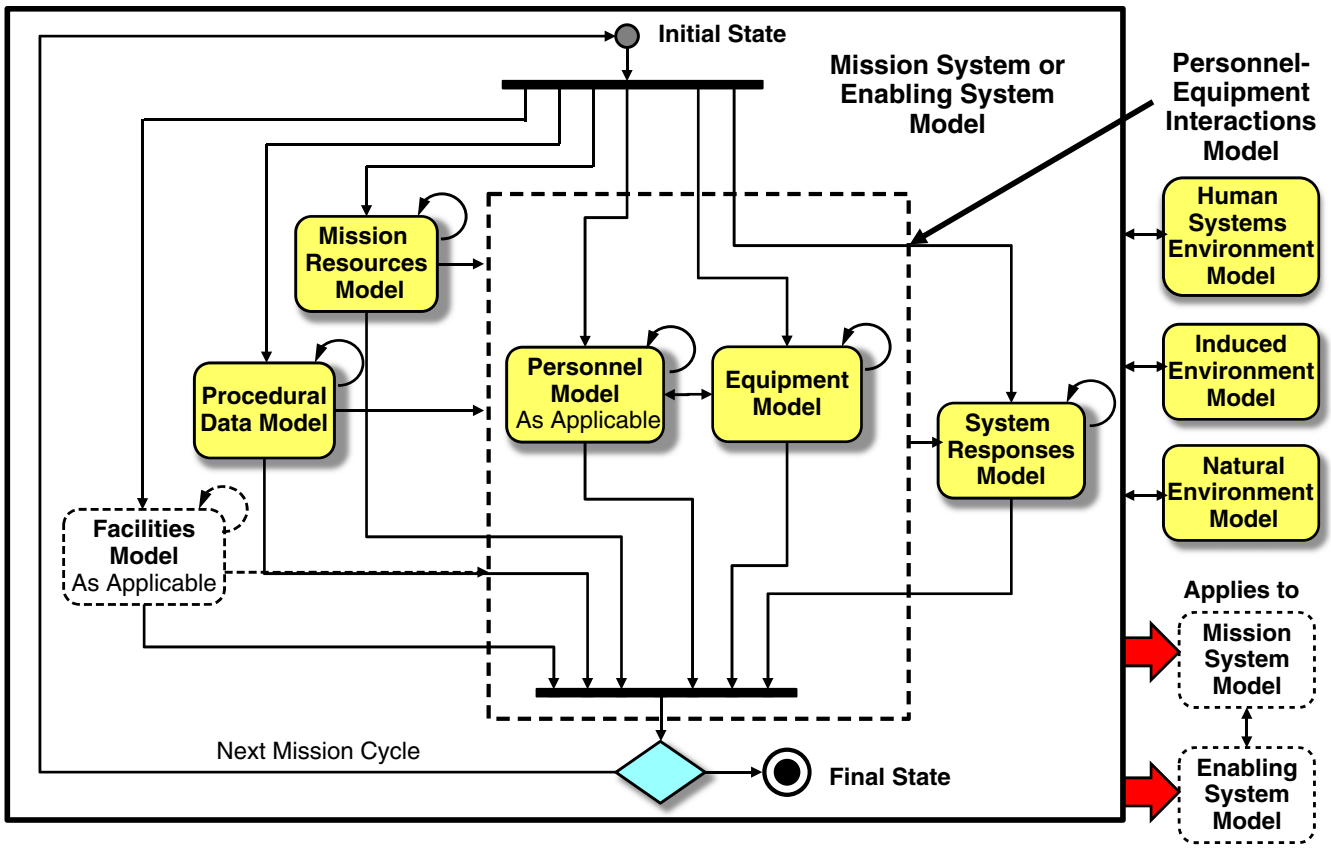


Figure 10.14 Generalized SEA Construct

Observe the dashed boundary encompassing the PERSONNEL and EQUIPMENT Elements. As a *performing entity*, the PERSONNEL–EQUIPMENT interaction is tightly coupled for most HUMAN SYSTEMS. The remaining System Elements—MISSION RESOURCES, PROCEDURAL DATA, SYSTEM RESPONSES, AND FACILITIES Element models—are *supporting entities*. Due to space restrictions, the graphic deviates from standard SysML™ practices and simply uses arrows touching the boxes to represent interactions with external OPERATING ENVIRONMENT—HUMAN SYSTEMS, INDUCED, and NATURAL SYSTEMS Environments.

A review of the figure leads to a question, *what is the structure of the Personnel–Equipment Model*? This leads to introduction of our next model, the PERSONNEL–EQUIPMENT Interactions Model.

10.6.4 Personnel–Equipment UC Task Sequence Model

The Mission Phases of Operation Concept shown in Figure 7.4 highlighted the interactions that need to occur between the MISSION SYSTEM and the ENABLING SYSTEM during all Phases of a mission, especially between their respective PERSONNEL and EQUIPMENT System Elements.

We know that PERSONNEL–EQUIPMENT interactions require some level of synchronization due to the uniqueness of the Equipment design. Specifically, trade-offs must be performed to balance costs, performance, and human factors concerning what the human can do best *versus* what the EQUIPMENT can do best (Figure 24.14). As a result, each UC requires orchestrating a set of interactions between the PERSONNEL and the EQUIPMENT Elements (Figures 10.15 and 10.16).

During Pre-Mission, Mission, and Post-Mission Phases of Operations, the MISSION SYSTEM and ENABLING SYSTEM have their own sets of PERSONNEL–EQUIPMENT interactions. In each interaction, PERSONNEL and EQUIPMENT perform operational tasks that leverage their respective capabilities. Actually, the PERSONNEL–EQUIPMENT interactions represent Mission and System UCs as shown in Figure 10.15.

In performing a UC, the PERSONNEL Element Model, operator, performs C2 tasks that initiate EQUIPMENT Element Model tasks via its interfaces—keyboard, touch screens, mouse, trackball, etc.—physical models. When the EQUIPMENT Element performs its tasks—C2 of its capabilities, it produces performance-based outcomes and behavioral response feedback via physical models, display, lights, audible and visual cautions, warnings, etc., to the PERSONNEL Element Model, operator.

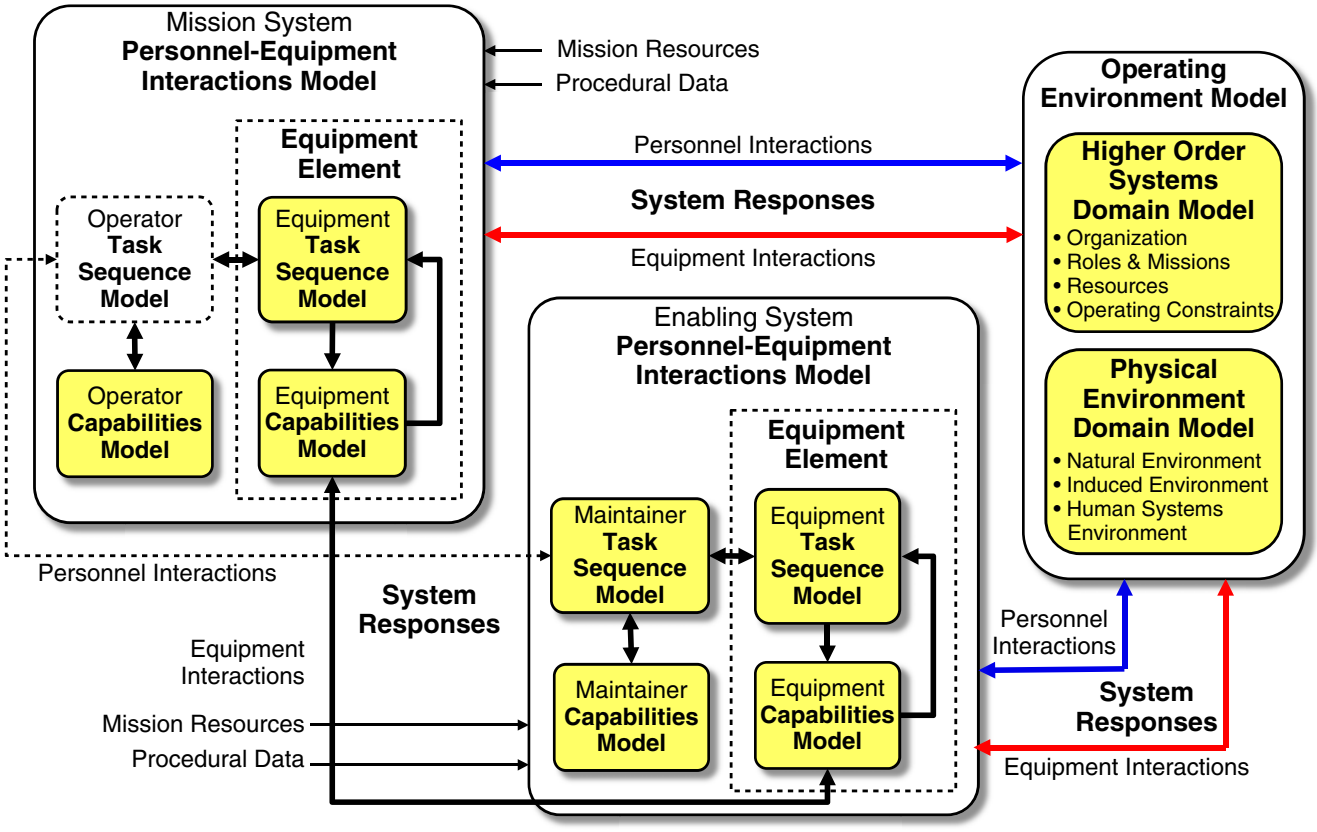


Figure 10.15 PERSONNEL–EQUIPMENT Interactions Illustration

Several key points about the figure are as follows:

1. Both the MISSION SYSTEM and ENABLING SYSTEM have identical, generalized interactions models except for the applicability of the ENABLING SYSTEM Facilities Element (Figure 10.14).
   - In general, each interaction model is based on an operator performing UCs and responding to scenarios unique to the MISSION SYSTEM or ENABLING SYSTEM.
   - To perform a UC, the User - operator or maintainer - must be trained to have specific capabilities to understand how to perform the UCs and operate the EQUIPMENT.
   - In general, the MISSION SYSTEM and ENABLING SYSTEM EQUIPMENT Elements have Task Sequence Models that process User inputs, configure the EQUIPMENT for the UC, and exploit the EQUIPMENT Capabilities Models.
   - The EQUIPMENT Capabilities Models represent time- and performance-based transfer functions that provide outcomes and feedback—results, completion, cautions, warnings, etc.—to the Task Sequence Models.

So, we create the PERSONNEL–EQUIPMENT Task Sequence Model template as shown in Figure 10.16; for a given UC, the PERSONNEL–EQUIPMENT Element interactions within the MISSION SYSTEM, ENABLING SYSTEM, and between the two must be synchronized into an integrated model structure.

Several observations are as follows:

1. Observe that the model has a Mission Phase- and UC-specific context. Recall from our discussions in Chapter 5, that Modes control Configuration States—architectural configurations—and accommodate specific Stakeholder UCs. Each UC consists of a sequence of operational tasks—*automated* or *semi-automated*—to be performed by the System's UC Actors, PERSONNEL and EQUIPMENT System Elements. The model represents the prescribed flow of those tasks. In the case of an aircraft, pilot deviation from a checklist representing required performance of these tasks in a specific sequence could result in catastrophic consequences.
2. Each UC-based task represents a capability the PERSONNEL and EQUIPMENT Elements must provide to accomplish the UC.
3. The EQUIPMENT Element model presented here represents a computer-based system that is capable of performing *manual*, *semi-automated*, or *automated* sequences of pre-defined operations. In the case of an ENABLING SYSTEM performing maintenance, the PERSONNEL Element—User—may have to C2 a test

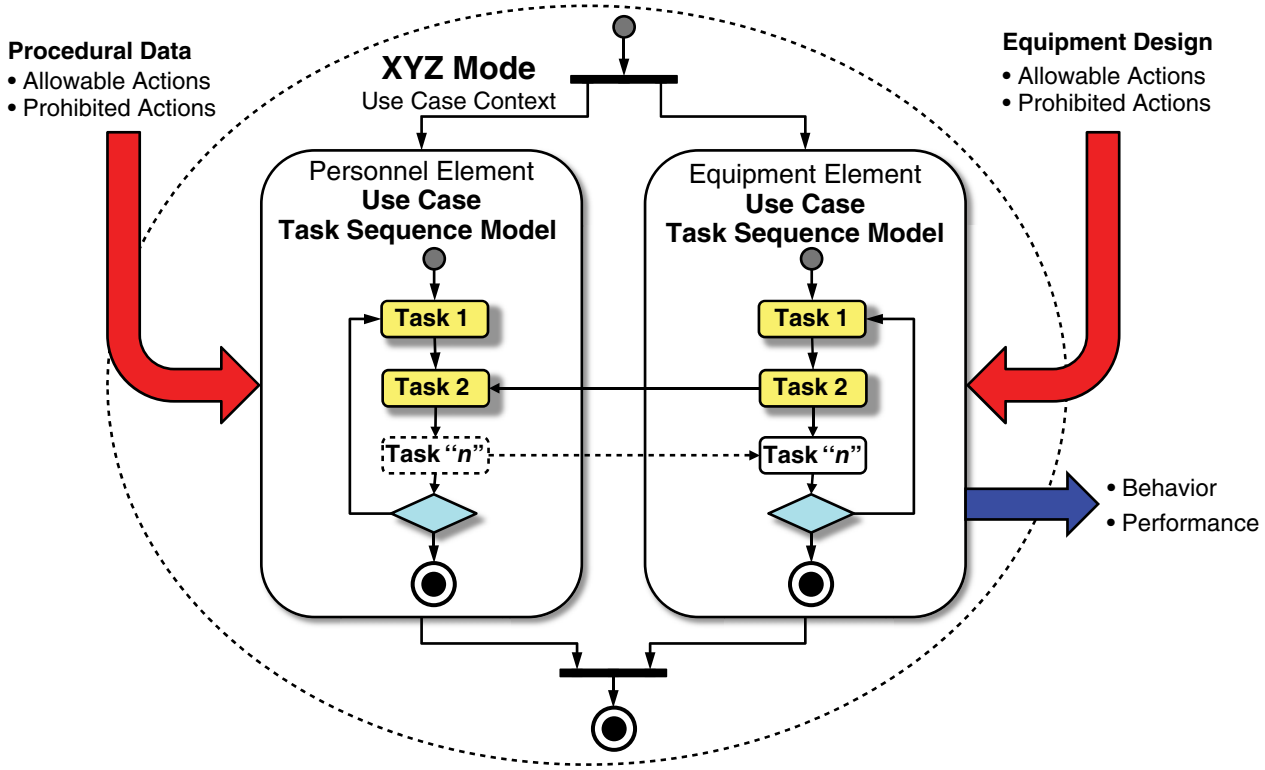


Figure 10.16 PERSONNEL–EQUIPMENT Task Sequence Model

instrument that can only provide readings, not automated sequences of tasks.

10.7 MODELING AN OPERATIONAL CAPABILITY

Engineers, managers, executives, and others talk about capabilities as if they were abstract objects in sales presentations or specifications. However, a capability is more than a *lifeless, abstract* object. Capabilities are the mechanisms that fulfill SPS or EDS requirements and are the *enablers* for mission success. To better understand this point, let's begin with an exploration of the *anatomy* of an operational capability.

10.7.1 Understanding the Anatomy of an Operational Capability

Principle 10.4

System Capability Operations Principle

Every operational capability, as an integrated system, consists of a minimum of three phases of operations:

- Pre-Capability Operations
- Capability Operations
- Post-Capability Operations.

Principle 10.5

Exception Handling and Recovery Operations Principle

Every capability requires a safety or contingency mechanism to detect exceptions, mitigate, and attempt recovery to return to and restore normal operations without doing any harm.

Contextually, a capability is a form of system that performs missions composed of a *minimum* of three phases of operation: Pre-Capability Operations, Capability Operations, and Post-Capability Operations. Figure 10.17 provides an illustration. When unplanned events referred to as exceptions such as Abnormal, Emergency, and Catastrophic scenarios (Figure 19.2) occur, Exception Handling and Recovery Operations are *initiated* to enable recovery, resumption of Normal Operations, and a safe completion of the capability's mission.

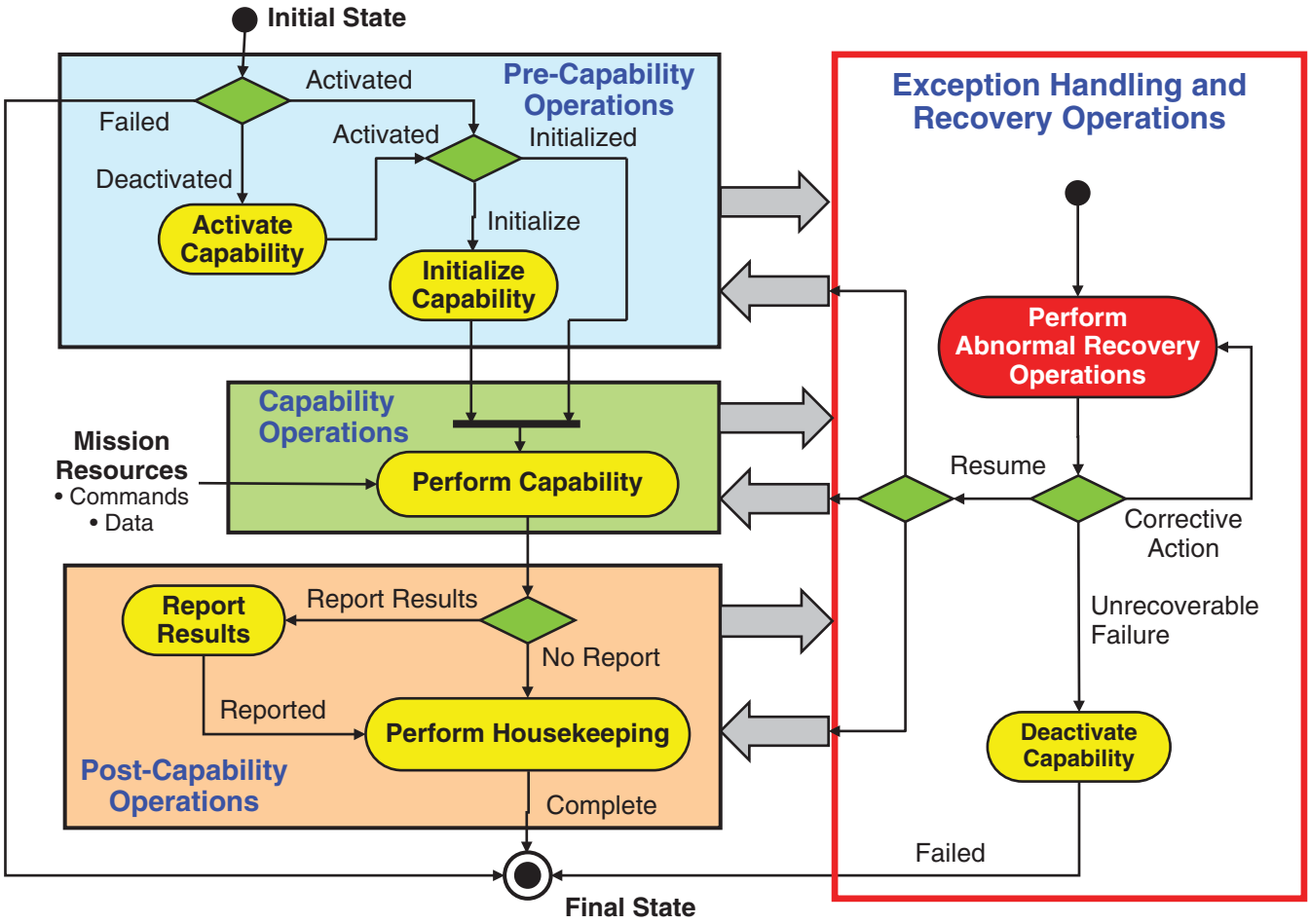


Figure 10.17 System Capability Construct with Exception Handling

Capability Initiation Principle

Principle 10.6 Every operational capability requires an external *trigger* such as a stimulus, excitation, or visual cue to initiate its performance-based outcome processing.

To better understand the point about capabilities being more than abstract objects, Figure 10.17 provides a reference graphic for our discussion. Observe the general *control flow* sequences that begin with a Capability's Initial State. When an external stimulus, excitation, or cue commands that a capability be performed, control flow sequences to Pre-Capability Operations.

10.7.1.1 Pre-Capability Operations Unlike a simple room light capability, which is *activated* by a User flipping a light switch ON/OFF, some systems require more complex capabilities that require a series of decisions to be made before initiation. Decisions include the following:

1. **Energy State Decision**—*What is the capability's current Energy State – Energized or De-energized*? For example, are (1) direct electrical, kinetic, solar, wind, hydro, hydraulic, and other forms of energy or (2) back-up power sources immediately *available* to power the capability on-demand?
2. **Activation State Decision**—*What is the current Activation State of the capability*– ENABLED, DISABLED, STANDBY, *or* HIBERNATION? For example, consider a Mars mission consisting of a mother craft carrying a rover vehicle. The rover vehicle may not be required to be fully operational between Earth and Mars for energy conservation. However, low levels of energy may be required for heating – warmth, periodically reporting OH&S status when interrogated, or receiving software uploads.
3. **Stabilization State Decision**—*When energized, does the capability require a period of stabilization such as warm-up, cool down, electrical or mechanical dampening, and so forth to reach a specified level suitable for performing tasks such as measurements*?
4. **Availability State Decision**—*Is the capability operationally available to perform on-demand*? Is it fully operational, does it exhibit degraded performance, or is it failed—out of specification or destroyed?
5. **Initialization State Decision**—Does the capability require *initialization* such as (1) calibration or alignment of electro-mechanical or optical components or (2) initialization of SOFTWARE parameters such as gravitational constants, pi, or magnetic North Pole movement.

Answers to these questions require establishing Operational Health & Status (OH&S) checks to interrogate the capability to determine the states listed above. For example, an OH&S checkout may be as simple as assessing a set of SOFTWARE "flags," polling HARDWARE configurations of switches or batteries. This may require (1) an OH&S assessment each time the capability is used or (2) an overall Operational Readiness Test (ORT) performed Daily (DORT), Hourly (HORT), or continuous (CORT) in the background.

The Energy, Activation, Stabilization, Availability, and Initialization Decision States are System/Product and application dependent and should be tailored to best fit the needs of the capability. Specific capabilities for your system and its application may require more or less Decision States.

When the Pre-Capability Operations are complete, *control flow* sequences to the Capability Operations. If exceptions occur during Pre-Capability Operations, *control flow* sequences to Exception Handling and Recovery Operations.

10.7.1.2 Capability Operations Capability Operations represent the *core processing* required to produce one or more performance-based outcomes. It may be *asynchronous*—sporadic—or *synchronous*—periodic. Examples include the following:

- Transmit Capability—Broadcast sensor data measurements at a 1 Hz rate.
- Command and Control (C2) Propulsion—Activate, command, and control propulsion based on User C2.

Capability Operations, in general, include tasks such as *processing, storing, navigating, commanding, controlling, transforming, converting, encoding/decoding, opening/closing, assessing*, and so forth. Observe the "ing" suffix for each of the actions. You should recall from our discussion in Chapter 7 that these actions represent Operational States of a capability. This is a very key point that will be applied later in Chapter 22 (Principle 22.8) concerning the development of specification requirement statements.

When Capability Operations are completed via (1) commanded termination, (2) time-out, or (3) resource depletion, *control flow* sequences to the Post-Capability Operations. If exceptions occur during Capability Operations, *control flow* sequences to Exception Handling and Recovery Operations.

10.7.1.3 Post-Capability Operations

Outcome Results Reporting Principle

Principle 10.7 On completion of its required performance, every capability should report successful completion of the task.

Post-Capability Operations represent physical-implementation-driven "Housekeeping" tasks that must be completed prior to the next initiation of the capability. For example:

1. Does completion of the capability require *notification* to the User, external systems, or internal record keeping? If so, in what form, audio, visual, or simply event data logging?
2. In the case of mechanical HARDWARE, are there mechanical arms such as NASA's Spec Shuttle Cargo Bay Arm or a robotic arm (Figure 25.7) that need to be "parked" in a Safe Mode—positioned and locked—to prevent damage to itself or from external systems? For example, an office copier scans a document for copying and then "parks" the scanning sensor at a preset location for reproduction of the copy.
3. In the case of electro-optical systems, does a protective lens cover need to be closed—NASA's Hubble Space Telescope—or an aperture closed to prevent high-intensity light damage to sensitive sensors?
4. Is the capability *fully operational* and ready for its next application?

Answers to these questions drive the need for specific operational tasks to be performed such as those shown in Figure 10.17.

When the Post-Capability Operations are complete, *control flow* sequences to the Final State. If exceptions occur during Postcapability Operations, *control flow* sequences to Exception Handling and Recovery Operations.

10.7.1.4 Exception Handling and Recovery Operations

The control flow sequences of the three preceding operations—Pre-Capability → Capability → Post-Capability Operations represent Cockburn's "main success scenario" (Chapter 5). Inevitably, *disruptions* or *interruptions* occur in the performance of a capability that may be driven by

1. External system interface failures with HUMAN SYSTEMS in its OPERATING ENVIRONMENT.
2. Internal system failures due to interface disconnection, propagation of failures (Figure 26.8), or component failures.

Principle 19.22 addresses these scenarios later.

Some of these failures may be critical, especially for computer operations. For example, the loss of external power may require a back-up power system to enable software applications to save critical data and close out applications before back-up power is lost. So, *how does a capability accommodate these exceptions*? The answer is via a concept referred to as *Exception Handling and Recovery Operations*.

The purpose of Exception Handling and Recovery Operations is to *restore* system operations to a level that enables resumption of the capability to complete a mission. Exception Handling and Recovery Operations are a form of *corrective maintenance* actions addressed later in Chapter 34.

Allocation of Exception Handling and Recovery Operations Requirements

Author's Note 10.3 Human beings instinctively think of EQUIPMENT—HARDWARE and SOFTWARE—as the mechanisms for achieving capability recovery. This point illustrates the fallacy of the SDBTF-DPM Engineering Paradigm Enterprises that allocate SPS or EDS Exception Handling and Recovery Operations requirements directly to HARDWARE and SOFTWARE (Figure 2.3). The reality is All System Elements—PERSONNEL, EQUIPMENT, MISSION RESOURCES, PROCEDURAL DATA, SYSTEM RESPONSES, and FACILITIES—have some contribution and accountability for Exception Handling and Recovery Operations as illustrated in Reason's Swiss Cheese Model introduced later in Figure 24.1. For example, failure to maintain control of an automobile while driving is a Driver–PERSONNEL Element issue, not an EQUIPMENT Element HARDWARE or SOFTWARE problem per se.

When exceptions occur, the challenge question is: *at what point is recovery of a degraded or failed capability futile*? So, key questions that need to be answered include the following:

1. How long or how many attempts are allowable to restore and recover Pre-Capability, Capability, and Postcapability Operations before a decision is made to terminate that effort?
2. How do you safely disable, deactivate, or de-energize the capability?
3. Who and what systems require notification when a capability experiences a disruption, has "failed," and is no longer available?
4. Is the failure a temporary condition such as overheating or freezing that warrants periodic follow-up recovery attempts when conditions have changed?
5. In the interest of energy conservation, should power be removed from a failed capability's devices?

One common scenario that leads to exceptions occurs in computational processing. For example: "divide by zero," applications that have been corrupted, or disconnected interfaces. When processors go into an infinite loop, the only way to recover is to reset the processor. For applications such as space travel, the User cannot easily reset a processor unless

it has a remote *reset* capability. Where communications are disrupted, this becomes a long-distance telecommunications challenge.

To solve the HARDWARE reset capability issues, some systems employ a "Watch Dog Timer." The timer *automatically* performs a HARDWARE *reset* to restart a computer system unless the computer restarts the timer within a specified timeout window such as 1 second. When the computer cycles in an infinite loop, obviously it does not reset the timer with the specified timeout period leading to a SYSTEM *reset* as the only alternative to alleviate the issue.

As part of a SYSTEM's Postmission analysis activities, mission data logs should be analyzed to understand the facts leading to the *exception* events. Consider the following example:

Example 10.9 **Event Data Recovery Example**

A flight data recorder, referred to as the "black box," records vital aircraft systems information such as aircraft performance, sequences of events, and flight environment conditions. Data recorded by the device provide an historical record of events such as PERSONNEL actions and EQUIPMENT performance—that may have led to a specific event for mitigation in future missions via upgrades, training, procedure updates, and so forth.

If the exception is *recoverable*, the recovery should be logged as an event, corrected, and control flow returns to the Pre-Capability, Capability, or Post-Capability Operations that were the source of the exception for continuation of processing. If the exception is *unrecoverable*, the capability is DISABLED and DE-ENERGIZED and directed to the Final State.

10.7.2 System Operational Capability Analysis Rules

To illustrate how the System Capability Construct can be applied to PERSONNEL–EQUIPMENT interactions, let's return to the Car–Driver System illustrated in Figure 10.8. Recognize that the integrated Car–Driver System can be represented by the construct. Similarly, the construct can be used to represent the Car and the Driver individually as shown in Figure 10.18.

In modeling the Car and Driver capabilities, we would synchronize their respective Pre-Mission, Mission, and Post-Mission operations beginning with the Initial State through the Final State.

During these interactions, the Driver can C2 various car capabilities as shown in Figure 10.19. Observe that the Driver can C2 the Engine, Lights, and Radio through "n" capabilities. This illustration represents a model of the Car–Driver System Modes matrix shown earlier in Figure 7.14.

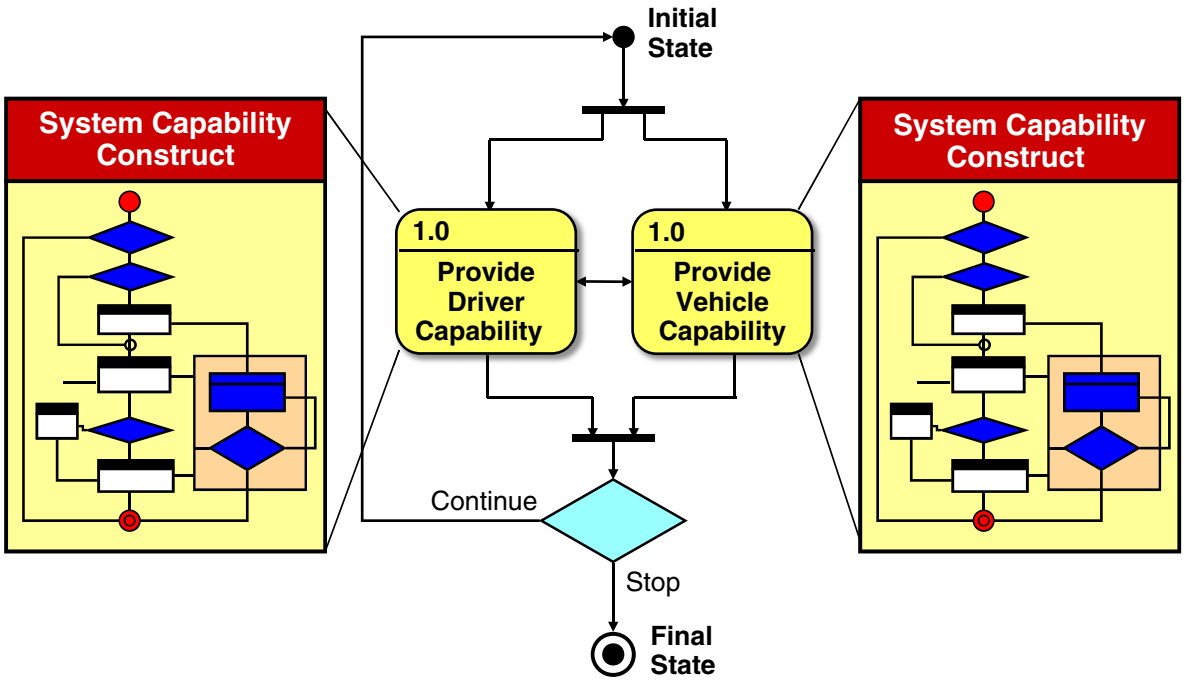


Figure 10.18 Application of the System Capability construct to a Car–Driver System

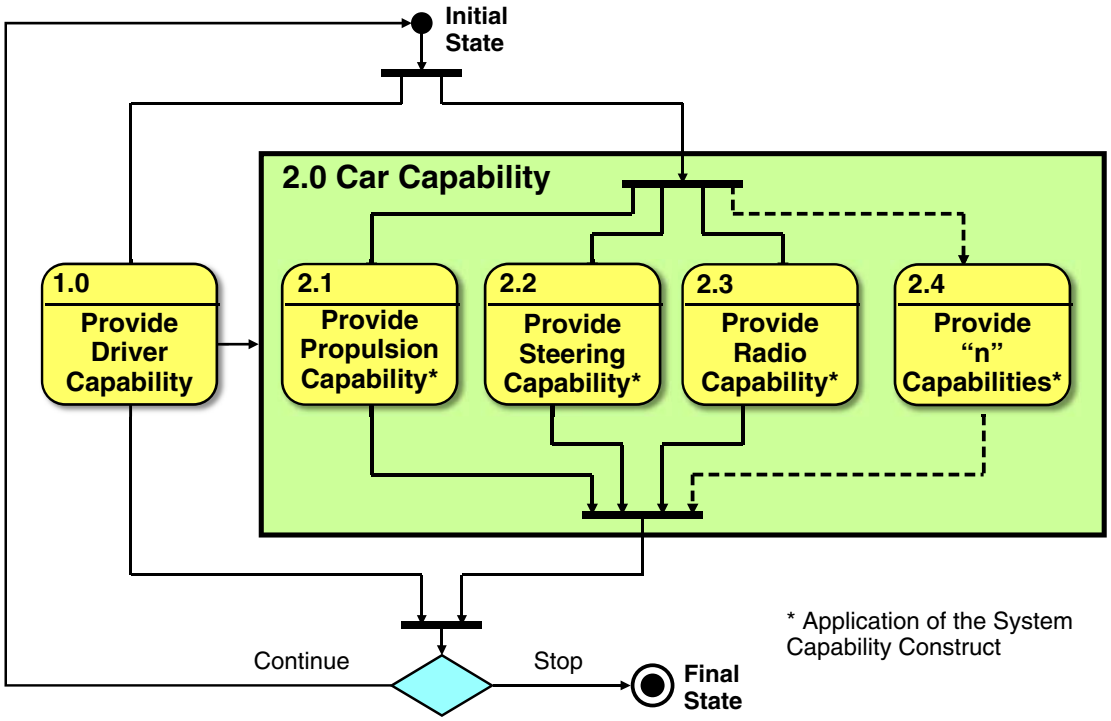


Figure 10.19 Driver Capability C2 of Car Capabilities

If you ask a sampling of engineers how they specify a system's capabilities, most will respond, "We write requirements." Although this response has a degree of correctness, requirements are merely mechanisms for documenting and communicating *what* the User requires in terms of acceptance at delivery. The response you should hear from SEs is: "We specify system capabilities and levels of performance to achieve specific outcomes required by the User."

When Engineers focus on writing requirement statements (*requirements-centric approach*) rather than specifying capabilities (*capability-centric approach*), requirements statements are what you get. When you attempt to analyze specifications written with a requirements-centric approach, you will encounter a "wish list" domain that is characterized by random, semi-organized thoughts; overlapping, conflicting, replicated, and missing requirements (Figure 20.3); ambiguous statements subject to interpretation; compound requirements; Statement of Work (SOW) tasks; mixtures of goals and requirements.

In contrast, specifications written by SEs who employ model-based requirements derived from coherent structures of system capabilities generally produce documents that *eliminate* or *reduce* the number and types of deficiencies 0. These specifications also tend to require less maintenance and facilitate verification during the system integration and test phase of the program.

If you analyze and characterize work products of the requirements-centric approach, several points emerge:

- Lack of understanding of the end product—bounding system capabilities—and its interactions with its OPERATING ENVIRONMENT.
- Lack of training and experience in how to identify and derive capability-based requirements.
- Poor understanding of the key elements of a requirements statement.

We will address the last two points in Chapter 11. Our discussion in this chapter focuses on the first item: understanding system capabilities.

Our discussion of the automated or semi-automated system capability construct and its application as a generic template enables us to establish the System Operational Capability analysis rules identified in Table 10.1.

10.7.3 The Importance of the System Capability Construct

At the start of this chapter, we contrasted traditional, ad hoc *endless loop*, Plug and Chug … SDBTF-DPM Engineering

TABLE 10.1 System Operational Capability Analysis Rules

Rule	Title	System Analysis and Design Rule
CAP_1	Multi-phase Operations	Every automated or semi-automated System capability consists of at least three types of operations: Pre-Mission, Mission, and Post-Mission
CAP_2	Capability Activation	Each capability must be *enabled* or *activated* to perform its intended mission. If not, the capability remains *deactivated* or *disabled* until the decision is revisited in the next cycle
CAP_3	Capability Initialization	Each automated or semi-automated System capability, when *activated* or *enabled*, may require initialization to establish a set of initial conditions that enable the capability to perform its mission. If initialization is not required, workflow progresses to the next cycle, Perform Capability
CAP_4	Perform Capability	Each automated or semi-automated System capability must perform a primary mission that focuses on accomplishing a performance-based outcome that is documented as a requirement in the SPS or Entity Development Specification (EDS)
CAP_5	Exception Handling	Each automated or semi-automated System capability should provide a mechanism for recognizing, recording, and processing exceptions and errors
CAP_6	Exception Recovery Operations	Every automated or *semi-automated* System capability should provide an exception handling mechanism to enable recovery from exceptions and errors without having to restart the System or result in destructive consequences to the system, life, property, or the environment
CAP_7	Exception Recovery Attempts	As each recovery operation is attempted, the System must decide whether additional recovery attempts are justified
CAP_8	Completion Notification	Every automated or semi-automated System capability may require automatic or manual notification of completion of the capability's mission and outcome to internal and external entities
CAP_9	Capability Safety and Security	Every automated or semi-automated System capability may require a set of operations or actions to safely and securely store or stow the capability to protect it from external threats or itself
CAP_10	Capability Cycling	Once a capability has cycled through its planned operations, it may: Continue to cycle—Do Until—until terminated by an external command, timed completion, or resource depletion Sequence to a Final State when either of the preceding conditions occur

Paradigms that focus on "deriving and writing" specification requirements from random thoughts from personal experiences. Our discussion of the System Capability Construct serves as compelling objective evidence as to *why it is important to structure and specify a capability and then translate it into a requirements statement*. Operations within the structure serve as a graphical checklist for specification requirements, namely, *what* is to be accomplished and *how well*—performance—without stating *how* it is to be implemented.

As our discussion illustrated, a capability consists of a series of operational tasks, decisions, inputs, and outcomes. Each of these operations is translated into a specification capability requirement statement and is integrated into a set of requirements that bound the total set of capabilities. Without the *capability-centric focus*, a specification is nothing more than a set of random, loosely coupled text statements with missing requirements representing overlooked operations within the capability construct.

Does this mean every operation within the construct must have a requirements statement? No, you have to apply good judgment and identify which operations require special consideration by designers during the capability's implementation.

Author's Note 10.4

Bounding and Specifying a Capability

Remember the old adage: *If you do not tell someone what you want, you cannot complain about what gets delivered*. If you forget to specify a specific operational aspect of a capability, the System Developer will be pleased to accommodate that requirement for a price—and, in some cases, a very large price. Therefore, do your homework and make sure all capability requirements are complete. The capability construct provides one approach for doing this, but the approach is only as good as your efforts to define it.

Our discussion introduced the concept of *automated* or *semi-automated* system capabilities. We described how most HUMAN SYSTEM capabilities can be modeled using the System Capability Construct as a template. The construct provides an initial framework for describing requirements that specify and bound the capability.

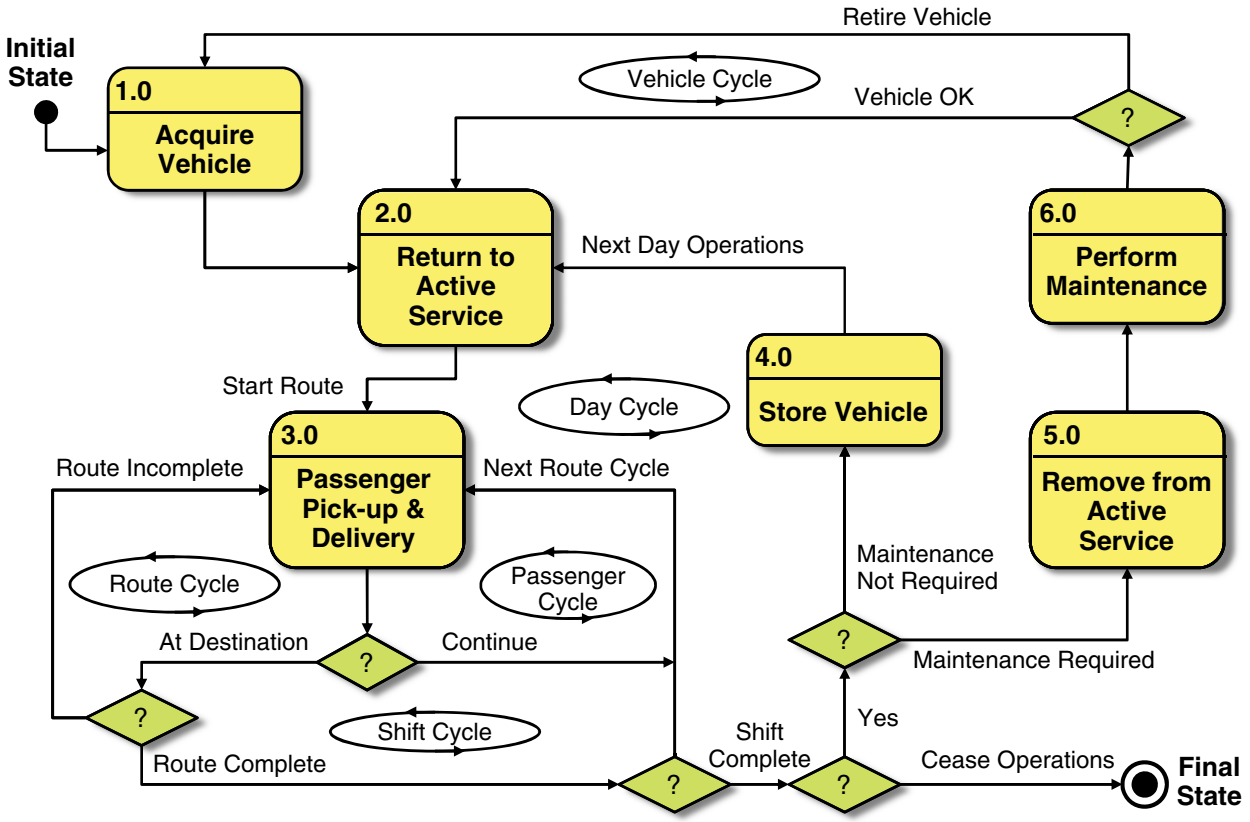


Figure 10.20 Business Operational Cycles within Cycles

10.8 NESTED OPERATIONAL CYCLES

While businesses have daily, *cyclical* operations at the enterprise level, entities within the business may have iterative cycles. If you investigate the context of the MISSION SYSTEM's application, analysis reveals several *embedded* or *nested* operational cycles. Let's explore an example of this type.

Let's assume we have a city bus transportation system that makes a loop around the city several times a day, every day of the week, seven days a week. As part of each route, the bus makes scheduled stops at designated passenger pickup and drop-off points, passengers board, pay a token, ride to their destination, and exit the vehicle.

At the end of each route, the vehicle is returned to a maintenance facility for routine *preventive maintenance.* If additional *corrective maintenance* is required, the vehicle is removed from active service until the maintenance action is performed. During maintenance, an assessment determines if the vehicle is scheduled for replacement.

- If repairable, the vehicle is returned to *active service.*
- If the vehicle is to be replaced, a new vehicle is acquired.

The current vehicle remains *inactive* service until it is decommissioned, which may or may not be linked to the new vehicle entering service, depending on business needs.

Figure 10.20 provides an operational model for this example to illustrate nested operational cycle within cycles. The six operational cycles, which are assigned reference identifiers, include a Vehicle Life Cycle, a Daily Schedule Cycle, a Driver Shift Cycle, a Route Cycle, a Passenger Cycle, and a Maintenance Cycle. Depending on the structure of the business, a seventh Vehicle Fleet Cycle may be applicable. Such is the case with aircraft, delivery vehicles, rental cars, police cars. While there are numerous ways of creating this graphic, the primary message here is *learn to recognize* embedded operational cycles that include the integration of the MISSION SYSTEM with the ENABLING SYSTEM during various cycles.

10.9 MODEL-BASED SYSTEMS ENGINEERING (MBSE)

Concepts for Modeling and Simulation (M&S) of system capabilities and performance as a means to identify, bound,

and specify system capabilities; assess and resolve Critical Technical and Operational Issues (CTIs/COIs); etc. have been in existence since the 1950s and 1960s. Part of the challenge was the maturity and performance of computer hardware technologies. Computer hardware technologies performance, however, was only one aspect of the system Problem Space. Other Problem Space issues related to the evolution and maturity of Systems Engineering and software development education and training, processes and methods, the need for a universal descriptive language for characterizing systems, and cross-platform and operating system compatibility, interoperability, and portability of system M&S designs and data.

Over several decades—1970–1990s—computer hardware and software technologies and methods maturity increased significantly. Organizations began creating M&S standards in the 1980s such as the DoD. What was needed was a universal descriptive language for characterizing systems and data standards for M&S software. As a result, work on the Universal Modeling Language (UML™) began in the 1990s at Rational Software for software intensive systems. However, due to the critical, tightly coupled link between Systems, Hardware, and Software Engineering, industry and government recognized that UML™ required additional features to serve as a descriptive language for SE applications to systems.

In 2001, the International Council for Systems Engineering (INCOSE) began a SysML™ Initiative to customize UML™ for SE applications. Subsequently, this leads to the development of Systems Modeling Language (SysML™) as a subset of UML™. In September 2006, the Object Management Group (OMG) released the OMG SysML™ Version 1.0 as an available specification in September 2007. Once the OMG standard for SysML™ was established, tool vendors began development and certification of MBSE tools to the OMG UML™ and SysML™ standards.

As a result of these advancements, one aspect of the original *problem space* remains. During the introduction to this text, we highlighted and contrasted the traditional, ad hoc, never-ending Plug and Chug … SBTF Engineering Paradigm many organizations continue to use. This is driven by the *erroneous* perception that since the System Engineering Process is *iterative* and *recursive*, (Figure 12.3 and 14.2) so is the SBTF Engineering Paradigm. Therefore, it must be SE. Not true!!

Wasson (2011) observes that since MBSE concepts and publicity have gained the attention of customers as the next panacea for improving organizational and contract performance, some SBTF organizations that refused to learn SE methods view MBSE tools as promotional "brochureware" publicity to secure business. They rush out and purchase MBSE tools with the perception that if they are using a SysML™ MBSE tool, they are, by definition, performing Systems Engineering. The inference here is that "if you own a paint brush, by definition, you must be an artist."

The reality is UML™ and SysML™ provide the descriptive language for modeling systems via certified MSBE tools. In general, they do not supply the requisite SE knowledge and methodology in analyzing and defining System / Entity, Missions, Operations, Architectures, Capability, and Specifications concepts that are required by the User to competently create, organize, and structure system information in the tool using UML™ and SysML™. That knowledge is the focus of this text—*System Engineering, Analysis, and Development: Concepts, Principles, and Practices*.

Lacking this requisite knowledge, MBSE tools become amateurish drag and drop graphics of incoherent systems. Unfortunately, when the ad hoc Plug and Chug … SBTF Enterprises fail because of this lack of SE knowledge and willingness to learn it, Systems Engineering, MBSE, MBSE tools, UML™, and SysML™ become casualties of bad publicity though no fault of their own. As Wasson (2011) observes, MBSE requires due diligence research; strategic and tactical planning, training, rollout, and implementation for its success; and long-term management commitment and support.

In summary, MBSE is a very powerful method, useful, and beneficial method for modeling and simulating system architectures and interfaces, operations, capabilities, etc. Consider its application to your Enterprise but do it right. Plan for its success; MBSE is not a "flavor of the month" activity.

10.9.1 Simple MBSE Example

To illustrate the essence of MBSE, consider the graphic shown in Figure 10.15, Here, we model the interactions within a Mission System or Enabling System between the repsective Personnel - Equipment Elements for a given Mode of Operation.

On entry from the previous Mode of Operation, the Mode has an Initial State that sequences via a SysML™ "fork" to Personnel Element and Equipment Element operations. Personnel Element and Equipment Element Operations begin with an Initial State and sequence through a series of tasks based on *Allowable* and *Prohibited* Actions. Each processing cycle sequences through a control or staging point decision block until a stimulus, excitation, or cue terminates the cycle. Concurrent task processing with each Element culminates in the SysML™ "join" as noted by its Final State, and subsequent control flow shifts back to the Mode via its SysML™ "Join" and termination as noted by its Final State.

10.10 CHAPTER SUMMARY

In summary, we have introduced the concept of Modeling Mission System and Enabling System operations. Our discussions introduced:

- A simple behavioral system response mode that illustrated the overall concept of *how* a system responds to *stimuli*, *excitation*, and *cues* in its Operating Environment and produces behavioral responses, systems, products, by-products, and services.
- Six behavioral interactions constructs common to most systems.
- Graphical *control flow* and *data flow* concepts for sequencing a system operations model and exchanging data between entities within the model.
- A series of top-down SysML™ templates that represent how we can model a Mission System and Enabling System interactions with its Operating Environment.
- The System Capability Construct that represents what is required to prepare, perform, and complete a capability for *single-use* or *cyclical* application.
- The concept of MBSE and its application to modeling Mission System and Enabling System operations.

In closing, our discussions here provide the foundation for Chapters 11–15 System Specification Concepts that address the development of multi-level specifications via the translation of UC-based tasks and capabilities into specification requirements statements.

10.11 CHAPTER EXERCISES

10.11.1 Level 1: Chapter Knowledge Application Exercises

1. What is a *model*?
2. How does a model represent a System or Entity?
3. What is a *transfer function*?
4. What is Model-Based Systems Engineering (MBSE), its purpose, descriptive language, and application to SE?
5. If MBSE is a tool that satisfies an SE solution space, define the problem space it is intended fill.
6. How do you develop a simple behavioral response model for a system?
7. What are the six basic types of behavioral interactions constructs?
8. How do you model multi-phase system operations?
9. How do you define, describe, and represent multi-level modeling constructs or template for modeling Mission System and Enabling System operations.
   - What is the purpose of the system behavioral response model (Figure 10.1)?
   - What are the key elements and interfaces of the model?
10. What is system *compatibility*?
11. What is system *interoperability*?
12. Identify and describe six types of system interactions with its Operating Environment?
13. What is meant by a system's *control flow*?
14. What is meant by a system's *data flow*?
15. How are *control flow* and *data flow* related?
16. What is the User's Level 0 Enterprise Model?
17. What is the System Element Architecture (SEA) Model?
18. What is the Personnel–-Equipment Interactions Model?
19. What is the System Capability Construct?
20. Graphically depict, label, and describe the operations within the System Capability Construct (Figure 10.17).
21. Using the System Capability Construct (Figure 10.17) as a template, develop textual and graphical descriptions of its application to a tablet computer, smartphone, or other devices.
22. What are system operational cycles?

10.11.2 Level 2: Chapter Knowledge Application Exercises

Refer to www.wiley.com/go/systemengineeringanalysis2e

10.12 REFERENCES

Carlzon, Jan (1989), *Moments of Truth: New Strategies for Today's Customer-Driven economy*, New York, NY: Harper Business.

ICAO (2013), Resolution A38/8 – Proficiency in the English language used for radiotelephony communications, 38th Session of the ICAO Assembly (September 24 – October 4,

2013), Montreal: CA: International Civil Aviation Organization (ICAO). Retrieved on 5/19/15 from http://www.icao.int/safety/lpr/Pages/Language-Proficiency-Requirements.aspx

INCOSE (2007), *Systems Engineering Vision 2020*, version 2.03, TP-2004-004-02, San Diego, CA: International Council on System Engineering (INCOSE). Retrieved on 1/17/14 from http://www.incose.org/ProductsPubs/pdf/SEVision2020_20071003_v2_03.pdf.

Wasson, Charles S. (2011), *Model-Based Systems Engineering (MBSE): Mirage or Panacea*, 5th Annual INCOSE Great Lakes Conference, Dearborn, MI, November 2011, http://www.incose.org/michigan/2011INCOSEGLRegionalPresentations.zip.

11

ANALYTICAL PROBLEM-SOLVING AND SOLUTION DEVELOPMENT SYNTHESIS

Throughout Part I, "System Engineering and Analysis Concepts," we sequenced through a series of chapters that provide an analytical perspective into *how to* conceptualize, analyze, organize, and characterize systems. These discussions provide the foundation for Part II, "System Design and Development Practices," which enables us to translate a Stakeholder's vision that could be abstract into a deliverable system, product, or service that can be *verified* and *validated* as meeting their needs.

11.1 DEFINITIONS OF KEY TERMS

- **Synthesis**—The analysis and assimilation of specification operational, capability, and interface requirements into candidate architectural solutions for evaluation, selection, and development of an optimal solution within environmental, design and construction, technology, budgetary, schedule, and risk constraints.
- **Synthesis**—The creative process that translates requirements (performance, function, and interface) into alternative solutions resulting in a physical architecture for the "best-value" design solution, made up of people, products, and process solutions for the logical, functional grouping of the requirements (FAA, 2006, Vol. 3, p. B-12)

A Word of Caution 11.1

Please note that the Synthesis definitions above summarize Chapters 1–11. As a level of abstraction, they do not correct the SDBTF-DPM Engineering Paradigm quantum leap represented by in Figure 2.3. Chapter 11 introduces a new methodology referred to as the Four Domain Solutions illustrated in Figure 2.3 to avoid the pitfalls of the SDBTF-DPM Paradigm.

11.2 PART I: SYSTEM ENGINEERING AND ANALYSIS CONCEPTS SYNTHESIS

Part I, "System Engineering and Analysis Concepts," embodied several key themes that System Engineers (SEs), Systems Analysts, managers, and executives need to understand when developing an Enterprise, system, product, or service. In Chapter 2, we highlighted the fallacies of the *ad hoc*, *inefficient*, and *ineffective* Plug and Chug … Specify-Design-Build-Test-Fix (SDBTF)-Design Process Model (DPM) Engineering Paradigm with deeply rooted origins in the Scientific Method. Although the SDBTF-DPM Paradigm may be fine as a Research & Development (R&D) and as an educational instructional model, a different type of problem-solving and solution-development model is required for Systems Engineering and Development (SE&D).

SE&D is intended for knowledge-based *application* of maturing technologies, not Research and Development (R&D). When individuals and Enterprises employ the SBTF-DPM paradigm, latent defects such as design flaws, errors, and deficiencies increase.

Unfortunately, these individuals and Enterprises often *erroneously believe* they are applying Systems Engineering methods and their customers, in good faith, believe them.

There is, however, a more *efficient* and *effective* approach to SE&D. We can employ a methodology that overcomes the quantum leap from requirements to the physical solution addressed earlier in Figure 2.3.

System Engineering Analysis, Design, and Development: Concepts, Principles, and Practices, Second Edition. Charles S. Wasson.

Companion Website: www.wiley.com/go/systemengineeringanalysis2e

Author's Note 11.1

Shifting Engineering Paradigms

Let's be clear about the SE&D Paradigm proposed here. When Enterprises and individuals have deeply ingrained paradigms that are part of the organizational culture, it can be a challenge shifting the paradigms due to the Not Invented Here (NIH) syndrome. Where these conditions exist, a mental model – the Earth is flat - *groupthink* rejects the notion that there are better approaches. They will acknowledge that the SDBTF-DPM Engineering Paradigm is ad hoc, chaotic, inefficient, ineffective, and unscalable for different sized systems … "but that's the way we have always done it." You can introduce new technologies, design methods, and so forth; however, if the Enterprise paradigm rejects acceptance, it takes effective, visionary leadership to shift the paradigm and then over a period of time.

Given this introduction, let's recap key aspects of the analytical thought process addressed in Part I:

- What is the problem space Users need to resolve via new system development or upgrades to fulfill their Enterprise mission performance objectives?
- What are the *boundary conditions* and *constraints*—operational, Enterprise, and regulatory—imposed by the System Acquirer on a system, product, or service in terms of missions within a prescribed Operating Environment?
- Given the set of boundary conditions and constraints, how does the User envision:
  a. D*eploying*, *operating*, *maintaining*, *sustaining, retiring, and disposing* of the system, product, or service.
  b. Performing missions within specific time and resource limitations.
- Given: (1) deployment; (2) operation, maintenance, and sustainment (OM&S); and (3) retirement/disposal constraints; what *behavioral responses* and performance-based *outcomes* does the User require to achieve mission success?
- Given those behavioral responses and performance-based outcomes, how can the deliverable system be physically and cost-effectively produced to perform those missions with acceptable risk?

11.3 SHIFTING TO A NEW SYSTEMS ENGINEERING PARADIGM

The SDBTF-DPM Engineering Paradigm is characterized as *ad hoc, chaotic*, *inconsistent*, *inefficient*, and *ineffective.* As an endless loop with no apparent completion, it typically results in cost and schedule overruns, increases technical risk, and is unpredictable. One of the problems is that Engineers pre-maturely take a quantum leap or shortcut from requirements to physical solution without understanding *interim* steps (Figure 2.3).

Project Managers deplore the endless loop performance with one step forward, two steps backward progress and continual *redesign* and *rework*. What is needed is a problem-solving and solution development methodology that approaches technical decision-making in a logical and insightful manner in which decisions are made with *minimal redesign* and *rework*. *How do we accomplish this*?

If we distill and analyze the points made in Chapter 11's Introduction key steps for a new methodology emerge:

- Step 1—Understand the User's Operational Needs, Problem, or Issue Space.
- Step 2—Bound and Specify the User's Problem and Solution Spaces.
- Step 3—Understand How the User Intends to Employ the System.
- Step 4—Model System Engagements and Behavioral Interactions with Its Operating Environment.
- Step 5—Determine a Cost-Effective, Acceptable Risk, Physical Implementation.

11.3.1 Step 1—Understand the User's Operational Needs

Boundary conditions and constraints are imposed by the Enterprise that owns or acquires the system, product, or service to accomplish missions with one or more performance-based outcome objectives. The following chapters provided foundational concepts in understanding User Enterprise roles and missions, User Stories, Use Cases (UCs), and UC scenarios. These concepts were addressed earlier in the following chapters:

- Chapter 4: "User Enterprise Roles, Missions, and System Applications"
- Chapter 5: "User Needs, Mission Analysis, Use Cases, and Scenarios"

11.3.2 Step 2—Bound and Specify the User's Problem and Solution Spaces

Once we understand the User's Operational Needs, we need to *define* and *specify* the problem space boundaries. Then, partition the problem space into one of more practical and affordable solution spaces that have an acceptable level of risk. These concepts were addressed earlier in Chapters 4 and 5

- Chapter 4: "User Enterprise Roles, Missions, and System Applications"

- Chapter 5: "User needs, Mission Analysis, Stories, Use Cases, and Scenarios"

Later in Part 2, Chapters 19–23, "System Specification Practices," will address how specifications are used to bound and specify the system, product, or service's solution space(s).

11.3.3 Step 3—Understand How the User Intends to Employ the System

Once we understand the User's operational needs, problem or issue space, we need to understand how the User envisions *deploying*, *operating*, *supporting*, *sustaining*, and *retiring/disposing* of the System. This enables us to build an analytical framework of interactions between the User, system, product, or service and external systems in its Operating Environment. The following chapters provided foundational concepts:

- Chapter 5: "User needs, Mission Analysis, Stories, Use Cases, and Scenarios"
- Chapter 6: "System Concepts Formulation and Development"
- Chapter 7: "System Command & Control (C2)—Phases, Modes, and States of Operation"

11.3.4 Step 4—Model System Behavioral Interactions with Its Operating Environment

The analytical framework representing *how* the User intends to use a system, product, or service enables us construct and model capability-based behaviors and response outcomes required to engage and behaviorally interact with the external systems in its Operating Environment. The following chapters provided the framework and modeling of a system, product, or service, its engagements, and behavioral interactions.

- Chapter 8: "System Levels of Abstraction, Semantics, and Elements"
- Chapter 9: "Architectural Frameworks of the System of Interest (SOI) and Its Operating Environment"
- Chapter 10: "Modeling Mission and Enabling Systems Operations and Behavior"

11.3.5 Step 5—Determine a Cost-Effective, Acceptable Risk, Physical Implementation

Once the System's behavioral responses and interactions with external systems are understood and defined, the question is: *how do we physically implement cost-effective, acceptable risk solution to perform those missions*? The following chapters provide the physical architectural framework foundation for understanding how systems, products, or services are physically implemented:

- Chapter 8: "System Levels of Abstraction, Semantics, and Elements"
- Chapter 9: "Architectural Frameworks of the System of Interest (SOI) and Its Operating Environment"

We used the phrase "acceptable risk" in Chapter 1 to define SE. As an abstract phrase, what does *acceptable* risk really mean? Consider Mini-Case Study 11.1 below:

Mini-Case Study 11.1

Acceptable Risk – Case of the Apollo 12 Lightning Strike

"The Apollo 12 space vehicle was launched on November 14, 1969, at 11:22 a.m.e.s.t, from launch complex 39A at Kennedy Space Center, Florida. At 36.5 seconds and again at 52 seconds, a major electrical disturbance was caused by lightning. As a result, many temporary effects were noted in both the launch vehicle and spacecraft. Some permanent effects were noted in the spacecraft and involved the loss of nine non-essential instrumentation sensors. All noted effects were associated with solid-state circuits, which are the most susceptible to the effects of a discharge.

Analysis shows that lightning can be triggered by the presence of the long electrical length created by the space vehicle and its exhaust plume in an electric field which would not otherwise have produced natural lightning. Electric fields with sufficient charge for triggered lightning can be expected to contain weather conditions such as the clouds associated with the cold front through which the Apollo 12 vehicle was launched. *The possibility that the Apollo vehicle might trigger lightning had not been considered previously.*

The Apollo space vehicle design is such that a small risk of triggered lightning is *acceptable*. In accepting this minimal risk for future flights, launch rule restrictions have been imposed with respect to operations in weather conditions associated with potentially hazardous electric fields." NASA (1970, p. 1).

By inspection, these themes range from the abstract, visionary concepts to the physical implementation; this is not coincidence. This progression is intended to illustrate a methodology that enables SEs and System Analysts to:

- Evolve a System Design Solution from abstract vision to physical realization.
- Avoid the pitfalls of the quantum leaps or shortcuts from requirements to physical solution shown in Figure 2.3.

These steps form the basis for our next topic, Introduction to the Four Domains of Problem-Solving and Solution Development.

11.4 THE FOUR DOMAIN SOLUTIONS METHODOLOGY

Four Domain Solutions Principle

Principle 11.1 A SYSTEM or ENTITY's design regardless of Level of Abstraction is composed of Four Domain Solutions sequenced in a logical workflow—Requirements, Operations, Behavioral, and Physical—based on decision dependencies to *minimize redesign* and *rework*.

If we simplify and reduce these thematic groupings, we find that they represent four types of solution domain dependencies for logical Problem-Solving and Solution Development actions. Table 11.1 illustrates the mapping between Part I's SE AND ANALYSIS CONCEPTS themes and the four domain solutions.

The Four Domain Solutions represent a logical problem-solving and solution-development methodology for "bridging the gap" between a User's abstract vision and the physical realization of the system, product, or service. Each domain solution elaborates decisions established by its predecessor and expands the level of detail of the *evolving* System Design Solution as illustrated in Figure 11.1.

In summary, the SE Problem-Solving and Solution Development Methodology enables us to elaborate an abstract Mission into successive levels detail to select the optimal Physical Domain Solution. This approach enables us to avoid ad hoc SDBTF-DPM Paradigm "quantum leaps" (Figure 2.3) from Requirements to Physical Solution that are inefficient,

TABLE 11.1 Linking Part I SE and Analysis Concept Themes into Part II System Design and Development Practices

Step	Thematic Objective	Outcome
1.	Understand the User's operational needs, problem, or issue space(s)	Problem definition
2.	Bound and specify the user's problem and solution space(s)	Requirements Domain Solution
3.	Understand how the user intends to deploy; operate, maintain, and sustain (OM&S), and retire/dispose of the system	Operations Domain Solution
4.	Model system logical/behavioral interactions with its OPERATING ENVIRONMENT	Behavioral Domain Solution
5.	Determine a cost-effective, acceptable risk, physical implementation	Physical Domain Solution

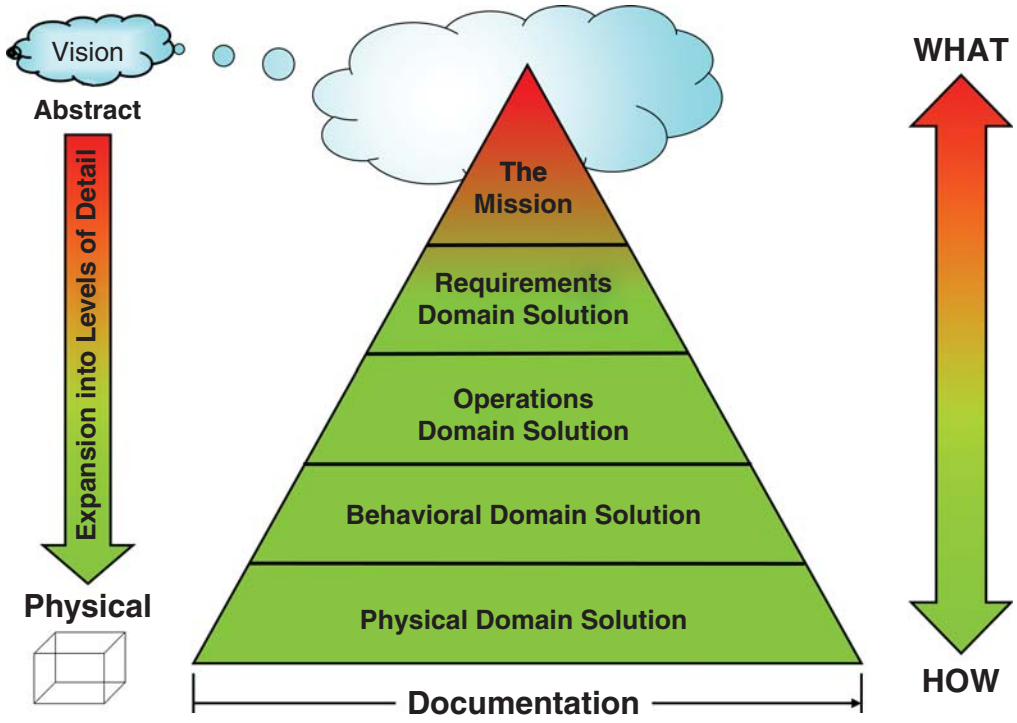


Figure 11.1 Development and Evolution of a SYSTEM/ENTITY's Solution Domains

ineffective, and often lead to project cost and schedule overruns.

This allows us to make several observations about SE&D workflow over time:

1. The Mission as defined by the Opportunity/Problem Space forms the foundation for the User to *formulate*, *bound*, and *specify* a Requirements Domain Solution in which requirements, technology, development cost and schedule, and risk are in balance.
2. As the Requirements Domain Solution evolves and matures, it establishes the basis for *conceptualizing*, *developing*, and *maturing* the Operations Domain Solution. This solution is developed collaboratively with the User to collect, define, and review User Stories, UCs, and Scenarios that will be used to derive SYSTEM capabilities and subsequently specification requirements.
3. As the Operations Domain Solution evolves and matures, it establishes the basis for *conceptualizing*, *developing*, and *maturing* the Behavioral Domain Solution. The solution defines *how* the System is envisioned to engage and interact with systems in its OPERATING ENVIRONMENT.
4. As the Behavioral Domain Solution *evolves* and *matures*, it establishes the foundation for developing and maturing the Physical Domain Solution based on physical components, and their technologies and application knowledge are readily available.

From a workflow perspective, the design and development of the System Design Solution - composed of the Requirements, Operations, Behavioral, and Physical Domain Solutions-*evolves* and *matures* from the *abstract* to the *physical* over time. However, the workflow progression consists of numerous *iterations* of feedback loops (Figure 14.2) to preceding solutions to reconcile Critical Operational and Technical Issues (COIs/CTIs). As a result, we symbolize the iterative loops of System Domain Solution shown at the left side of Figure 11.2.

11.4.1 Workflow Sequencing of the System Domain Solution

Figure 11.2 illustrates how the System Domain Solution is developed, evolves, and matures over time. The Requirements Domain Solution is initiated first, either in the form of a contract SPS or an ENTITY's Development Specification (EDS). The sequencing occurs as follows:

1. When the Requirements Domain Solution is understood and reaches a sufficient *level of maturity*, initiate development of the Operations Domain Solution. This includes:
   a. Development of the System's Concepts of Operation (ConOps)—deployment, operations, maintenance, sustainment, retirement, and disposal.

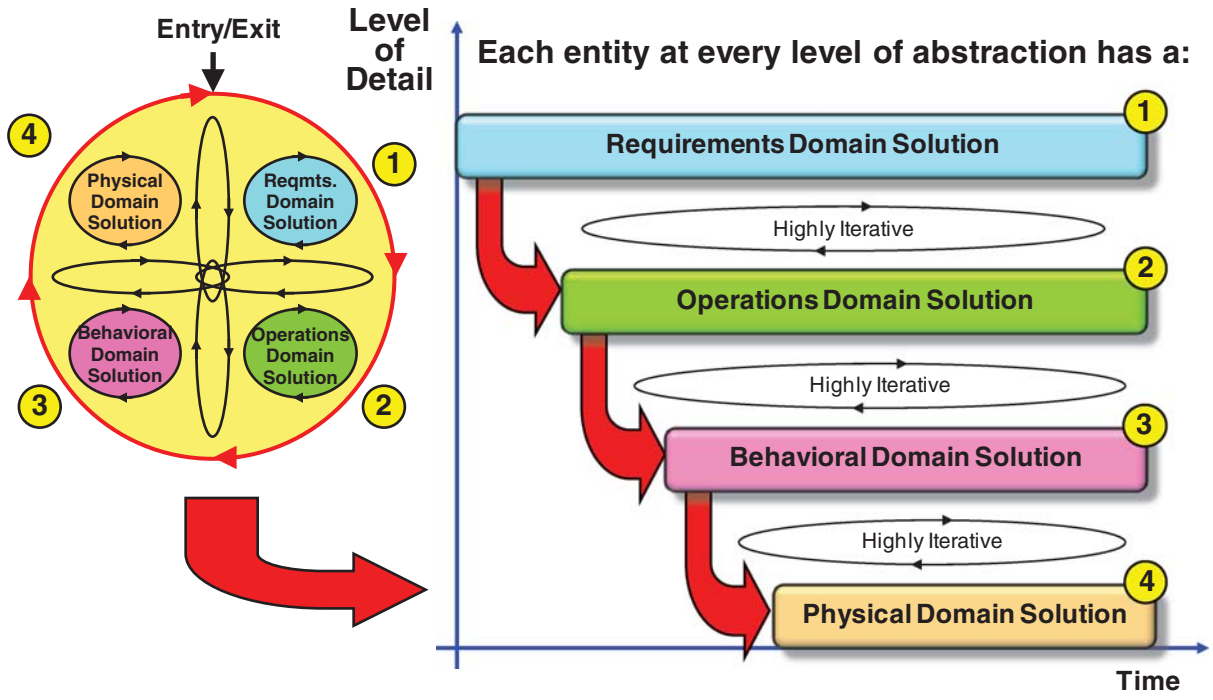


Figure 11.2 System Design Solution Domain Time-Based Implementation

b. Formulation, evaluation, and selection of an optimal Operational Architecture from a set of viable candidates.

2. When the Operations Domain Solution reaches a sufficient *level of maturity*, formulate, evaluate, and select an optimal Behavioral/Logical Solution from a set of viable candidates.
3. When the Behavioral/Logical Domain Solution reaches a *level of maturity*, formulate, evaluate, and select an optimal Physical Domain Solution from a set of viable candidates.
4. Once initiated, the Requirements, Operations, Behavioral, and Physical Domain Solutions *evolve concurrently*, *mature*, and *stabilize* as a fully integrated System Design Solution. Therefore, the Word of Caution 11.1 noted earlier.

11.4.2 Implementation of the Four Domain Solutions

To illustrate how the Four Domain Solutions methodology is implemented and its workflow from the *abstract* to the *physical* and *iterative* feedback loop interdependencies between each Domain Solution, refer to Figure 14.2 that forms the framework for the Systems Engineering Process.

Heading 11.1

The methodological flow of the Four Domain Solutions precludes the quantum leap problem from Requirements to Physical illustrated in Figure 2.3. This brings us to a pivotal point in SE—understanding how Archer's DPM introduced in Chapter 2 can be *valid* yet *deficient* in fostering a concept perceived *incorrectly* by Plug and Chug … SDBTF-DPM Engineering Paradigm Engineers, managers, and executives to be System Engineering. We now shift our focus to address these deficiencies.

Principle 11.2

Task-Based Outcome and Rewards Principle

If you reward Engineers for *activities*, you get … activities. If you equip Engineers with the right processes, tools, and methods to accomplish performance-based outcomes, you get … *performance-based outcomes*.

You may ask: *how is Archer's DPM deficient as a problem-solving and solution development methodology? Engineering is highly iterative; that's how it is performed*. Conceptually, this is true. However, Archer's DPM (Figure 2.9) is an abstract, observational model that iterates *within* the development of each of the Four Domain Solutions. Contrary to misguided interpretations, the DPM is an endless loop, analytical model that has *validity* and *relevance* to SE but not *closure* to the Problem Space or Solution Space it is perceived to solve.

Reexamine Archer's DPM in Figure 2.9; analyze its contents. *What do you observe*? Data Collection, Analysis, Synthesis, Development, and Communication. *What is common across each of these items*? These are Engineering *activities*, not *outcomes*! Now, factor in the feedback loops of the DPM into the Plug and Chug … SDBTF Engineering Paradigm and what do you get? The Plug and Chug … SDBTF-DPM Paradigm (Figure 2.8).

Is there any wonder why systems and projects fail due to technical compliance, cost, and schedule issues? Because Enterprises, organizations, projects, and Engineers are out wandering around doing what they are tasked to do - "perform activities!"

Objective evidence of this paradigm is reflected in project schedule line items such as: Write Specification, Design PC Board, Order Components, Test HW or SW, and so forth. Abstract activities such as these illustrate why System Acquirers impose Earned Value Management System (EVMS) requirements to measure progress as a basis for assessing the risk of on-time project completion within budget. Key EVMS tools include an Integrated Master Plan (IMP) and its supporting Integrated Master Schedule (IMS), which are outcome result-driven progress measurement tools. The IMS, as an integrated network of outcome-based tasks and dependencies, provides a measurement snapshot of the status, progress, completion, and risks and their impact on the development of the overall schedule.

Engineers often deplore IMPs and IMS for a variety of reasons, some of which are warranted! Simply stated, the IMS and IMP are based on *accountability*. Engineers are required to demonstrate *objective evidence* that timely and substantive technical decisions are being made that actually produce outcome-based results as planned! It is hard to produce results and deliver a SYSTEM or PRODUCT on time and within budget if all you do is "perform activities." Recognize and appreciate the difference!

How do the Four Domain Solutions correct Archer's DPM "activities" paradigm? Figure 11.3 provides an illustration.

As a high-level abstraction, Archer's DPM is *valid* from the perspective of orthogonal activities performed within each of the Four Domain Solutions. For example, the Requirements, Operations, Behavioral, and Physical Domain Solutions have their own embedded Data Collection, Analysis, and Synthesis activities that enable the sequential flow of the Four Domain Solutions. However, Archer's DPM is inappropriate as an overall SE&D outcome-based problem-solving and solution development methodology. Based on this explanation, the fallacy of the Plug and Chug … SDBTF-DPM Engineering Paradigm should be obvious based on a focus on *activities*, not *outcomes*. It also illustrates a *major* factor that contributes to Engineering's reputation with Project Managers and Executives—deserved or not—

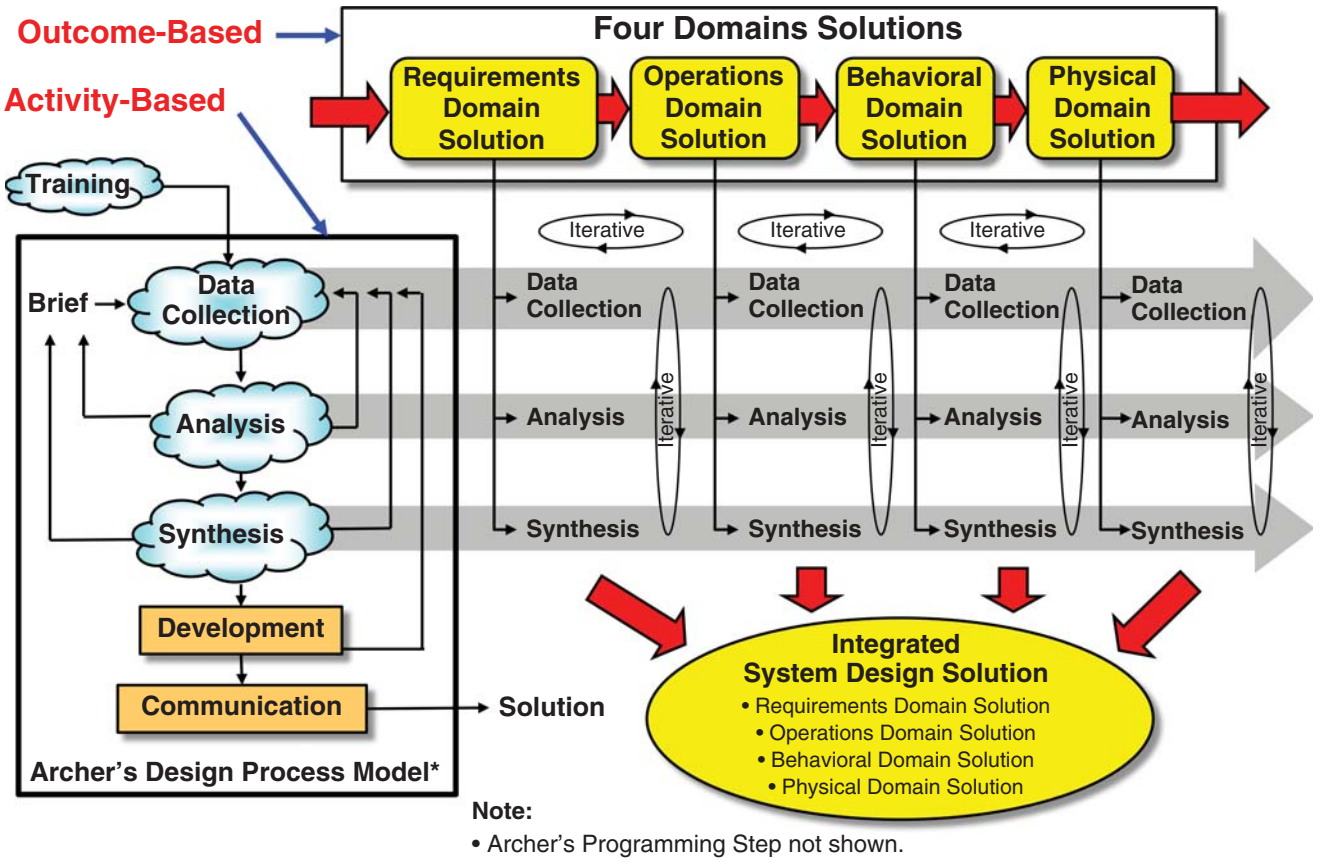


Figure 11.3 Illustration of the Relationship of Archer's DPM to Each of the Four Domain Solutions

for being unable to finish a system or product's design within project schedule or budget constraints, assuming those were realistically achievable at project start.

11.5 CHAPTER SUMMARY

Chapter 11 synthesizes our discussions in PART I, "SYSTEM ENGINEERING AND ANALYSIS CONCEPTS," and establishes the foundation for PART II, "SYSTEM DESIGN AND DEVELOPMENT PRACTICES." The introduction of the Requirements, Operations, Behavioral, and Physical Solution Domains, coupled with chapter references in each domain, encapsulates the key analytical concepts that enable SEs and System Analysts to *think about*, *communicate*, *analyze*, and *organize* systems, products, and services for SE&D.

Remember:

- The Four Domain Solutions provide a logical decision dependency method that *shifts* the *ad hoc*, *chaotic*, *inconsistent*, *inefficient*, and *ineffective* Engineering SDBTF Paradigm inferred by Archer's DPM to a new level of System Thinking (Chapter 1).
- The Four Domain Solutions Methodology consists of a series of logically sequenced decision dependencies—Requirements → Operations → Behaviors → Physical—that form each SYSTEM or ENTITY's design solution and *minimize* redesign and rework.

Figure 11.4 illustrates the essence of SE. Observe that the Four Domain Solutions introduced in Figures 11.1 and 11.2 are expanded top-down into four levels. The analysis for each Phase of Operation consists of the following solutions.

- **Requirements Domain Solution**—Specifies and bounds performance-based objectives, outcomes, and results to be achieved for each Mission Phase or subphase of Operation. Requirements at all levels of abstraction must be traceable to User Mission Requirements.
- **Operations Domain Solution**—The User performs Use Case (UC)-based Operational Tasks (OTs) required to accomplish the mission. Each OT requires that the User Monitor, Command, and Control (MC2) a system, product, or service's Modes of Operation.

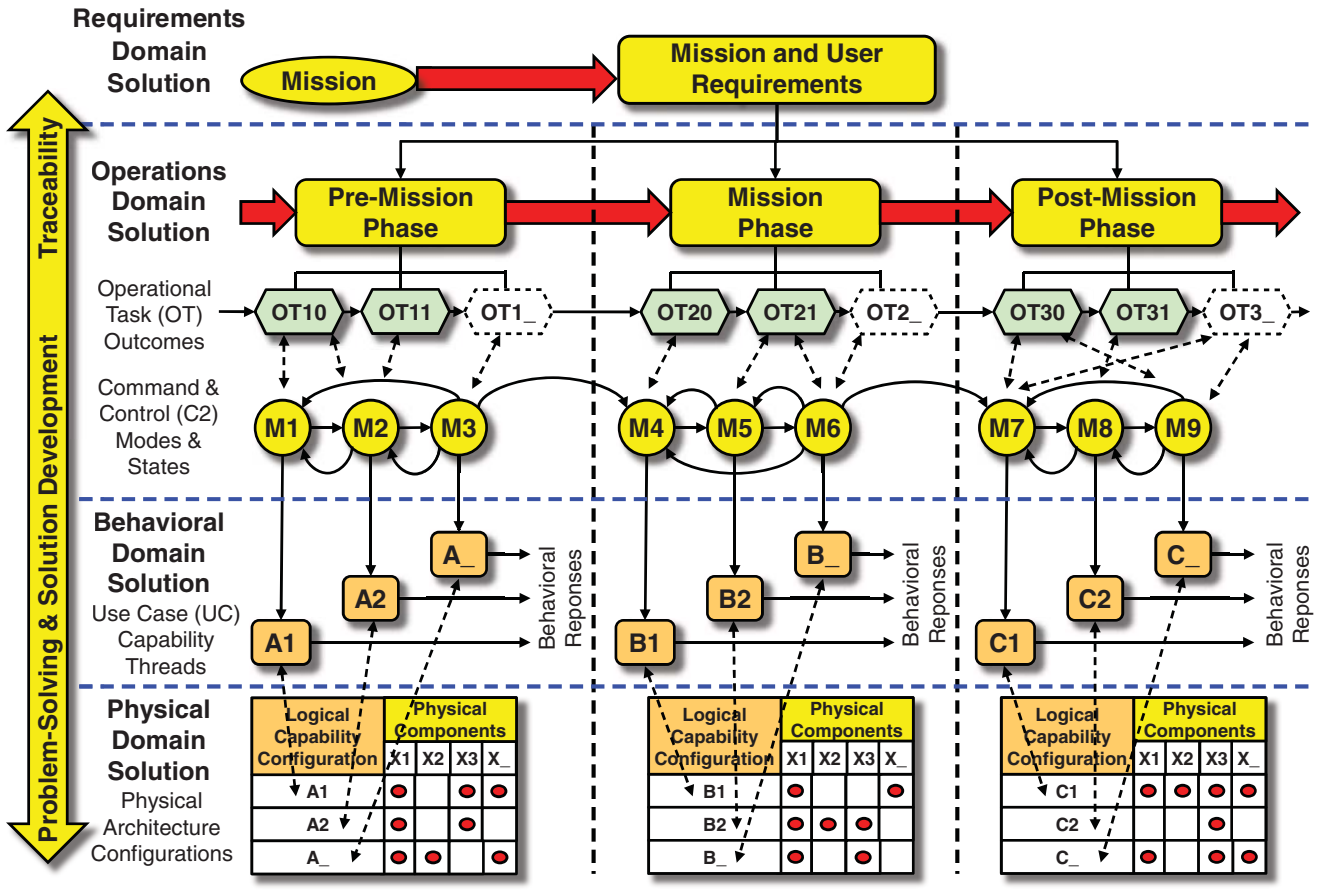


Figure 11.4 Conceptual Overview of System Engineering & Analysis Synthesis

Operations and OTs must be traceable to and compliant with the Requirements Domain Solution.

- **Behavioral Domain Solution**—For a given Mode of Operation, characterize the stimulus–response behavior required to achieve specific UC and Scenario performance outcomes. This requires configuration of sets of a system, product, or service's logical capabilities via User and System C2 to enable UCs and Scenarios to be achieved. We represent the Behavioral Domain Solution via methods and tools such as Model-Based Systems Engineering (MBSE). Logical capabilities and performance-based outcomes must be traceable to the Operations Domain Solution and compliant with the Requirements Domain Solution.
- **Physical Domain Solution**—The Physical Domain Solution is composed of physical components that have been selected to implement the Behavioral Domain Solution. Components must be traceable to the Behavioral Domain Solution and compliant with the Requirements Domain Solution.

As a bonafide problem-solving and solution development methodology, the Four Domain solutions serve as the "engine" for the Systems Engineering Process Model (Figure 14.1) introduced later in Chapter 14.

Conceptually, as the System Design Solution *evolves*, we can create an overview of the SYSTEM or PRODUCT's System Design Solution using the matrix shown in Figure 11.5. Each bubble identifier links technical requirements, descriptions, decisions, etc. that can be tailored as Applicable or Not Applicable to the specific SYSTEM or PRODUCT.

Author's Note 11.2

Figure 11.5 and Its Application

Please note that on inspection, you may say that there is no way anyone could have the time to address all of the bubble identifiers in Figure 11.5. The reality is that every SDBTF-DPM Paradigm project typically spends time in an *ad hoc*, *inefficient*, and *ineffective* manner in each of these area making technical decisions. These time increments add up. So, there should be nothing new here. Remember - this chart should be tailored for each specific SYSTEM or PRODUCT. Many of the bubble IDs may be Not Applicable (N/A) to a specific SOI. The graphic simply serves as a high-level, visual audit checklist to

Where:
(X) = Reference to Required Operational Capabilities
▭ = Scenario-Based Operational Capabilities

Operational Mode	Phase			System Elements						Operating Environ.					
										Man-Made Systems					
	Pre-Mission	Mission	Post-Mission	Mission Resources	Procedural Data	Equipment	Personnel	Facilities	System Responses	Friendly I/Fs	System Threats	Induced	Natural	Design & Construct. Constraints	Architectural Configuration
Power-Off	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Power-Up / Initialize	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
Configure	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45
Calibrate / Align	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60
Training	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75
Normal Operations	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90
Abnormal Operations	91	92	93	94	95	96	97	98	99	100	101	102	103	104	105
Safing	106	107	108	109	110	111	112	113	114	115	116	117	118	119	120
Analysis	121	122	123	124	125	126	127	128	129	130	131	132	133	134	135
Maintenance	136	137	138	139	140	141	142	143	144	145	146	147	148	149	150
Power-Down	151	152	153	154	155	156	157	158	159	160	161	162	163	164	165

Figure 11.5 Matrix Summarizing MISSION SYSTEM and ENABLING SYSTEM Design Synthesis

ensure various aspects of System Development have been addressed.

The intent of the matrix in Figure 11.5 is two-fold:

1. To provide a analytical framework of technical decisions and solutions that need to be addressed during SE&D - knowledge creation.
2. To use the analytical framework as a checklist for auditing a technical project performance during various reviews to identify deficiencies.

The second point is crucial. You do not want to discover these deficiencies during System Integration, Test, and Evaluation (SITE) or for the User to discover these during the Operations, Maintenance, and Sustainment Phase. As we shall address later in Chapter 13, the cost-to-correct latents defects increases almost exponentially downstream in the System Development Phase.

The matrix illustrates the overall C2 of the SYSTEM or PRODUCT as a function of its relationships between Modes and Phases of Operation, System Element capabilities, OPERATING ENVIRONMENT conditions, Design and Construction Constraints, and Physical Architecture Configuration. For example:

- *Is the Calibrate/Align Mode allowed in the Pre-Mission, Mission, or Post-Mission Phases*?
- If so:
  - What capabilities are required for each System Element for a given set of OPERATING ENVIRONMENT conditions?
  - Are there any unique Design and Construction Constraints implications or ramifications?
  - What behavioral and physical architectural configurations and Allowable/Prohibited Actions are required?

To illustrate answers to these questions, let's assume that each of the Bubble IDs in Figure 11.5 represents a knowledgebase of technical decision information. For example:

- Bubble ID 110 would define the Procedural Data requirements and solutions to support the Safing (small caps) Mode of Operation.
- Bubble ID 86 would identify System Threats requirements and solutions for the Normal Operations Mode.

Based on fundamental analytical knowledge of what a System is, who its Users are, and how they intend to deploy, operate, maintain, sustain, retire, and dispose of it, we are now ready to proceed to PART II, "SYSTEM DESIGN AND DEVELOPMENT PRACTICES."

11.6 REFERENCES

FAA SEM (2006), *System Engineering Manual*, Version 3.1, Vol. 3, National Airspace System (NAS), Washington, DC: Federal Aviation Administration (FAA).

NASA (1970) – R. Godfrey, et al., *Analysis of Apollo 12 Lightning Incident*, N72-73978, February 1970, Washington, DC: NASA. http://klabs.org/history/ntrs_docs/manned/apollo/19720066106_1972066106.pdf. Retrieved on 3/19/13.

PART II

SYSTEM ENGINEERING AND DEVELOPMENT PRACTICES

12

INTRODUCTION TO SYSTEM DEVELOPMENT STRATEGIES

The award of a system development contract to a System Developer or Services Provider or the initiation of a project charter for commercial product development signifies the beginning of the System Development Phase. This phase covers all activities required to meet the provisions of the contract; produce the end item deliverable(s); and deploy or distribute the deliverables to the designated contract delivery site.

On Contract Award or project charter start, a project transforms itself from a proposal organization to a System Developer or Services Provider organization. The challenge is for the selected System Developer to "live up to their proposal claims" that enabled them to capture the System Development effort. This requires the Enterprise to demonstrate that they can *competently* deliver the proposed system on time, within budget in accordance with the provisions of the contract and risk understood by the Acquirer.

Chapter 12 focuses on *how* a proposed SYSTEM is developed under a contract, project charter, or task and delivered to the User. We explore *how* the System Developer or Services Provider evolves the visionary and abstract set of Stakeholder requirements through the various phases of system development to ultimately produce a deliverable system, product, or service. The "system" might be a a spacecraft, a smartphone, a mass mailing service, a trucking company, a hospital, a symposium or may others.

Author's Note 12.1

The System Development Phase described here, in conjunction with the System Procurement Phase (Chapter 3), may be repeated several times before a final system is fielded. For example, in some business domains, the selection of a System Developer may require a sequence of System Development Phase contracts to *evolve* and *mature* the SYSTEM requirements in stages and subsequently "down select" a field of qualified contractors to one or two contractors.

For a System Services Provider contract, project charter, or task, the System Development Phase might require developing or adapting reusable system operations, processes, and procedures to support the User's mission and support services for the System Operations, Maintenance, and Sustainment (OM&S) Phase. For example, a computer Services Provider may win a contract or task to deliver "outsourced" support services for an Enterprise's computer maintenance program. The delivered services may be a "tailored" version similar to projects the contractor provides to other organizations.

The key to successful development begins with formulation and development of technical strategies that enable the System Developer or Services Provider to transform

System Engineering Analysis, Design, and Development: Concepts, Principles, and Practices, Second Edition. Charles S. Wasson.

Companion Website: www.wiley.com/go/systemengineeringanalysis2e

the User's operational needs into a physical System Design Solution. As an introductory overview, this chapter provides the high-level infrastructure for Chapters 12–18 that comprise the SYSTEM DEVELOPMENT STRATEGIES series.

12.1 DEFINITIONS OF KEY TERMS

- **Corrective Action** The set of tasks required to correct latent defects in specification contents, errors, or omissions; designs flaws, errors, or deficiencies; component workmanship and defective materials or parts; or correct flaws, errors, or omissions in test procedures.
- **Discrepancy Report (DR)** A report that identifies a condition in which a document or test results indicate a noncompliance with a capability and performance requirement specified in a performance or item development specification.
- **Developmental Test & Evaluation (DT&E)** "Test and evaluation performed to:
  a. Identify potential operational and technological limitations of the alternative concepts and design options being pursued.
  b. Support the identification of cost-performance trade-offs.
  c. Support the identification and description of design risks.
  d. Substantiate that contract technical performance and manufacturing process requirements have been achieved.
  e. Support the decision to certify the system ready for operational test and evaluation." (MIL-HDBK-1908B, p. 12)
- **Design Requirements** Requirements specified via drawings, schematics, wire lists, notes, or Engineering Bill of Materials (EBOM) concerning the configuration of components and connections, and instructions.
- **Developmental Configuration** "The contractor's design and associated technical documentation that defines the evolving configuration of a configuration item during development. It is under the developing contractor's configuration control and describes the design definition and implementation. The developmental configuration for a configuration item consists of the contractor's released hardware and software designs and associated technical documentation until establishment of the formal product baseline" (MIL-STD-973, 1992 Cancelled para. 3.30).
- **First Article** "First article includes pre-production models, initial production samples, test samples, first lots, pilot models, and pilot lots; and approval involves testing and evaluating the first article for conformance with specified contract requirements before or in the initial stage of production under a contract" (MIL-HDBK-1908B, p. 12).
  "Includes pre-production models, initial production samples, test samples, first lots, pilot models, and pilot lots; and approval involves testing and evaluating the first article for conformance with specified contract requirements before or in the initial stage of production under a contract" (DAU, 2012, p. B-85).
- **Functional Configuration Audit (FCA)** "An audit conducted to verify that the development of a configuration item has been completed satisfactorily, that the item has achieved the performance and functional characteristics specified in the functional or allocated configuration identification, and that its operational and support documents are complete and satisfactory" (SEVOCAB, 2014, p. 132 - Copyright 2012 by IEEE. Used by permission). (Source: ISO/IEC/IEEE 24765 :2010).
- **Independent Test Agency (ITA)** An independent organization employed by the Acquirer to represent the User's interests and evaluate how well the *verified* system satisfies the User's *validated* operational needs under field operating conditions in areas such as *operational utility*, *suitability*, and *effectiveness*.
- **Operational Test & Evaluation (OT&E)** Field test and evaluation activities performed by the User or an ITA under actual OPERATING ENVIRONMENT conditions to assess the operational utility, suitability, availability, usability, efficiency, and effectiveness (Principle 3.11) of a system based on validated User operational needs. The activities may include considerations such as training effectiveness, logistics supportability, reliability and maintainability demonstrations, and efficiency.
- **Physical Configuration Audit (PCA)** "An audit conducted to verify that a configuration item, as built, conforms to the technical documentation that defines it" (SEVOCAB, 2014, p. 221 - Copyright, 2012, IEEE. Used with permission) (IEEE 828-2012 IEEE Standard for Configuration Management in Systems and Software Engineering, 2.1).
- **Problem Report (PR)** A document that (1) identifies a *non-compliance*, discrepancy, discrepancy, or problem and (2) characterizes the event including the configuration and sequence of steps performed that may have led to the problem. A PR does not identify or speculate about the source or root cause of the problem or its corrective action. Determination of root causes and corrective actions are accomplished as separate tasking using investigative, forensic, and analytical methods.

- **Product Development Team (PDT)**—A multi-discipline team accountable for the development of a specific system or entity such as a PRODUCT, SUBSYSTEM, ASSEMBLY, or SUBASSEMBLY. PDT leadership and membership varies as a function of the key specialties required for integrated decision-making as a function of the System Development Process (Figure 12.2).
- **Proof of Concept**—Verification and Validation (V&V) that a *strategy* will accomplish the required outcomes. For example, create a communications network that can detect problems and reroute messages or signals without disruption.
- **Proof of Principle**—V&V that an *idea* is realistically achievable and feasible. For example, can millions of cars be designed to safely fly in the air simultaneously and arrive at their destinations without incident?
- **Proof of Technology**—V&V that a new or existing *technology* is sufficiently mature and rugged for use in a specific application or implementation of a system in terms of size, weight, strength, and performance—accuracy, precision, reliability, environmental, and mass production.
- **Quality Record (QR)** A document such as a memo, email, report, analysis, meeting minutes, and action items that serves as *objective evidence* that a planned task action, event, or decision has been accomplished and completed.
- **Source or Originating Requirements** – Requirements released by a System Acquirer specifying Stakeholder operational needs that serve as the frame of reference for acquisition of a system, product, or service.

12.2 APPROACH TO THIS CHAPTER

Successful development of systems, products, or services requires insightful implementation of a System Development Strategy based on proven processes, methods, and tools. The strategy needs to answer a key question: *How does a project get from Contract Award to delivery and acceptance by the System Acquirer or User with the available resources, expertise, and acceptable risk*? During the Request for Proposal (RFP), this question has two key points:

1. The System Acquirer and/or User wants a level of confidence that the set of qualified System Developer or Services Provider offerors *understand* the User's Problem Space and Solution Spaces and can *competently* develop a deliverable system, product, or service on time and within budget to meet the RFP and subsequent contract requirements.
2. Each System Developer or Services Provider is challenged with how do we apply our Enterprise capabilities to *efficiently* and *effectively* develop the system, product, or service with current technologies on time, within budget and acceptable risk while minimizing the Total Cost of Ownership (TCO) of the SYSTEM.

To address these two points, competent System Developers establish multi-faceted technical strategies that enable them to transform a set of abstract User requirements into the physical realization of a system, product, or service.

You will discover that implementing SE methods in antiquated, Enterprise "stovepipes" that purposely limit SE interactions with the customers—Stakeholder Users and End Users—may not necessarily lead to *customer satisfaction*. As a result, SE is often relegated to a *reactionary* environment that responds to System Acquirer RFP requirements or contract System Performance Specification (SPS) requirements. *What if these requirements do not accurately and completely document the Customer's prioritized set of needs, demands, or requirements*?

Traditional Engineering (SE) begins with the assumption that a System Acquirer or User will provide a base set of requirements such as a DoD Statement of Objectives (SOOs) or an RFP System Requirements Document (SRD). You need to understand what the User is really thinking. Technically, the Enterprise that wins many System Development efforts is one that understands the *underlying lessons learned* that motivate those needs and are able to clearly articulate and communicate that understanding in their proposal.

Chapter 12 serves as an Introduction to the System Development Strategies consisting of Chapters 12–18 as illustrated in Figure 12.1. As the central focal point for this series of chapters, Chapter 12 establishes the foundation and infrastructure via its System Development Workflow Strategies. This raises a question: *How do we establish a robust System Development Workflow Strategy that will improve our probability of success and achieve the outcomes established by the User's constraints—technical, cost, schedule, and risk*? The answer resides in several supporting strategies discussed in the following chapters:

- Chapter 13 establishes the strategy for *verifying* component compliance to specifications, designs, and test procedures and *validating* that those components and the final SYSTEM will satisfy the User's operational needs.
- Chapter 14 establishes an SE Process Model that serves as the multi-level *problem-solving* and *solution development* methodology for developing entities at every level of abstraction. This model serves as the core problem-solving and solution-development methodology to:

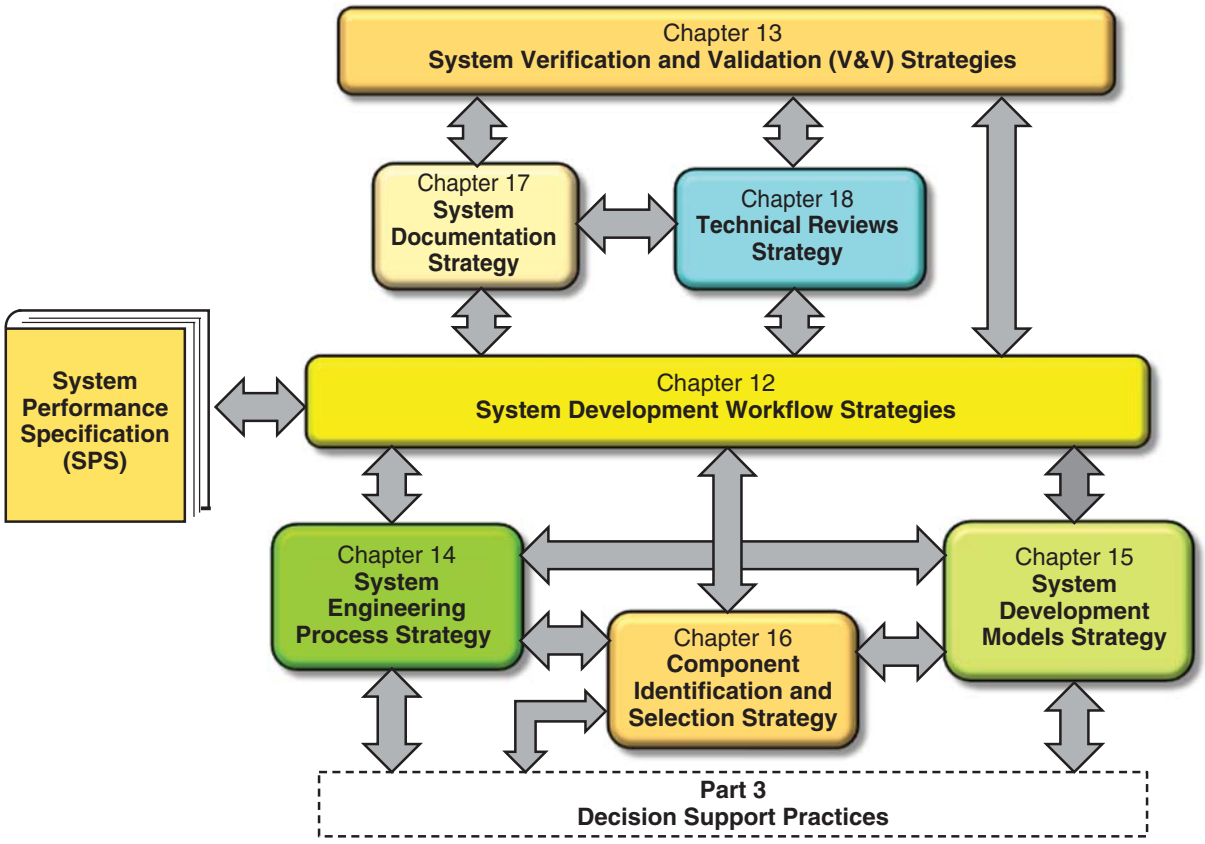


Figure 12.1 Overview Graphic of the System Development Strategies

- Correct Engineering *efficiency* and *effectiveness* problems created by the SDBTF-DPM Engineering Paradigm.
- Avoid the quantum leaps (Figure 2.3) from requirements to a single, point design solution.

- Chapter 15 addresses various types of System Development Models that represent strategies for how a project approaches design and development of SYSTEMS, SUBSYSTEMS, ASSEMBLIES, SUBASSEMBLIES, and so forth.
- Chapter 16 introduces the System Configuration Identification and Component Selection Strategy for physically implementing the SYSTEM each entity's System Design Solution.
- Chapter 17 addresses documentation strategies for capturing key technical decision artifacts—specifications, designs, drawings, and so forth—that facilitate development of the System Design Solution.
- Chapter 18 establishes the strategy for reviewing and assessing the progress, status, maturity, and risk of the evolving System Design Solution and each of its multi-level components at various decision-making *staging* or *control points* during the System Development Phase.

12.3 SYSTEM DEVELOPMENT WORKFLOW STRATEGY

The System Development Workflow Strategy shown in Figure 12.2 establishes the primary roadmap of project technical activities required to get from Contract Award or project charter start to System Acceptance and delivery of a system, product, or service. In general, our discussion implements the general System Development Phase description provided in Chapter 3.

The System Development Phase consists of a series of workflow processes required to translate the contract SPS into the deliverable System Design Solution. The primary infrastructure for the SE&D Workflow Strategy originates from general Engineering practice:

- From a System Developer's perspective, their mission is to (1) design the SYSTEM; (2) procure and/or develop SYSTEM components; (3) integrate, test, and evaluate the SYSTEM; and (4) demonstrate the SYSTEM's compliance to its SPS and documentation—for example, System Verification—for the Acquirer and User prior to final acceptance and delivery.

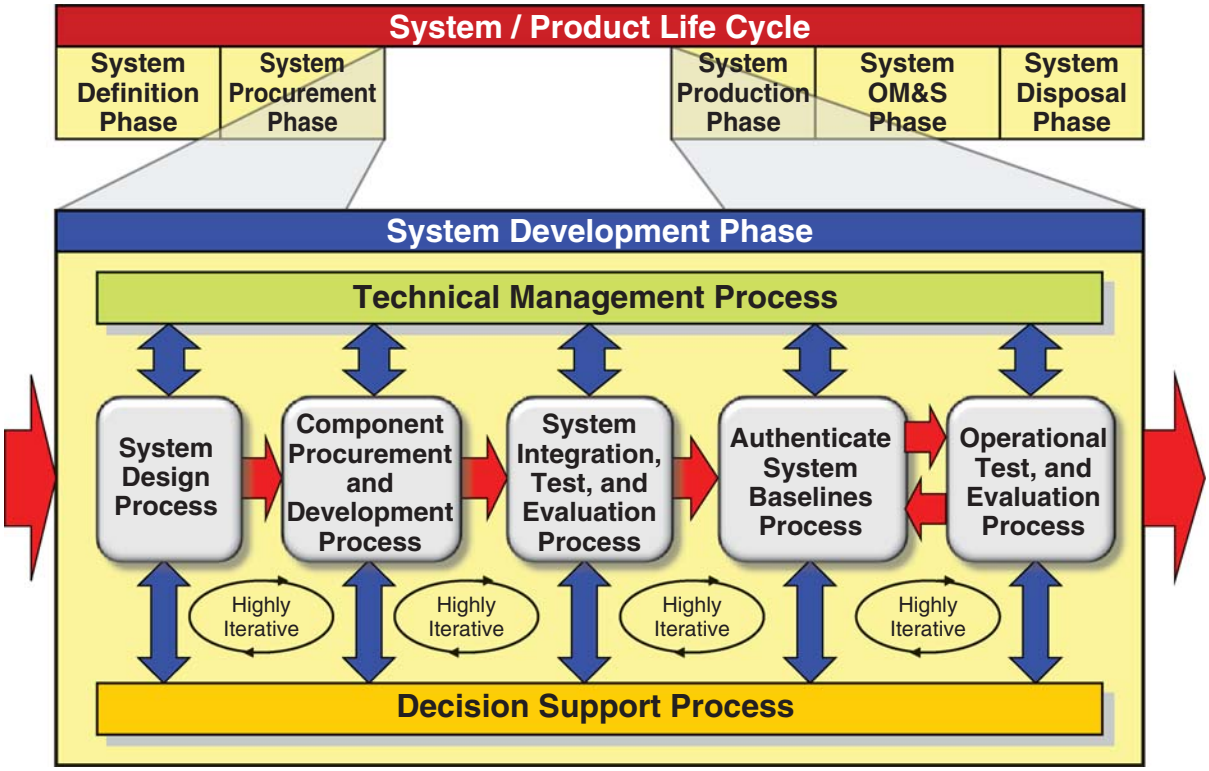


Figure 12.2 The System Development Process Workflow

- From the User's perspective, their mission is to *validate*—System Validation—that the System fulfills the User's operational needs.
- From a System Acquirer's contract perspective, User *validation* is *optional* unless it is explicitly specified in the contract.

Given the preceding points including the need to accommodate validation options, we establish the System Development Workflow consisting of the five sequential processes illustrated in Figure 12.2. These include the:

- System Design Process
- Component Procurement and Development Process
- System Integration, Test, and Evaluation (SITE) Process
- Authenticate System Baselines Process
- (Optional) Operational Test & Evaluation (OT&E) Process

Author's Note 12.2

System Development Process Workflow Direction

Although the general left-to-right workflow appears to be sequential in Figure 12.2, there are *highly iterative* feedback loops (Figure 13.6) that connect back to earlier processes to initiate refinements and corrective actions.

The System Development Workflow processes are supported by two enabling processes, Technical Management and Decision Support:

- The primary objective of the Technical Management Process is to plan, organize, staff, resource, orchestrate, and control Product Development Teams (PDTs) tasked with accountability for delivering their assigned entities—Products, Subsystems, Assemblies, and so forth—within technical, technology, cost, schedule, and risk constraints.
- The primary objective of the Decision Support Process is to provide meaningful data to support *informed* technical decision-making within each of the workflow processes through the development and use of analyses and trade studies such as Analysis of Alternative (AoA), prototypes, models, simulations, tests, proof of concept or technology demonstrations or methods. Part 3, "Decision Support Practices," Chapters 30–34 address Decision Support Process activities.

When the System Development Phase is completed, the workflow progresses to the System Production Phase or System OM&S Phase, whichever is applicable.

In summary, the SE&D Workflow Strategy, as a high-level technical project management process, provides the infrastructure for developing systems, products, or services. The challenge is: *How do we leverage this infrastructure to transform SPS requirements into a System Design Solution—drawings, parts lists, etc.—sufficient for internal development and/or external procurement of components*? To answer this challenge, we need to establish a logical, technical strategy that enables us to "Engineer" the System including all of its constituent components. This brings us to the Multi-Level Systems Design & Development Strategy.

12.4 MULTI-LEVEL SYSTEMS DESIGN AND DEVELOPMENT STRATEGY

If we expand the SE Design Process; Component Procurement and Development Process; and SITE Process shown in Figure 12.2 into lower-level strategies, Figure 12.3 emerges.

Observe that Figure 12.3 depicts a multi-level SYSTEM architectural decomposition, component development, and SITE Strategy. This strategy represents *what has to be accomplished* without regard to the temporal—time-based—aspects. The reason is that there are different System Development Models that can be employed for the implementation at the System Development Workflow Strategy level and for specific entities—PRODUCTS AND SUBSYSTEMS—within the SYSTEM. We will defer a discussion on this topic until Chapter 15.

Referring to Figure 12.3:

- The left side represents how the SE Design Process is accomplished. SE Analysis and Design, concepts, principles, and practices are employed to decompose and partition SPS requirements into architectural levels of abstraction entities such as PRODUCTS, SUBSYSTEMS, ASSEMBLIES, and so forth. SPS requirements are *allocated* and *flowed down* to lower levels via their respective Entity Development Specifications (EDSs) such as a SUBSYSTEM EDS or and ASSEMBLY EDS and traced vertically back to the Acquirer's *source* or *originating requirements*.
- The bottom center represents how the Component Procurement and Development Process acquires or develops components using SE Design Process work products such as designs, drawings, wiring lists, parts lists, and so forth.

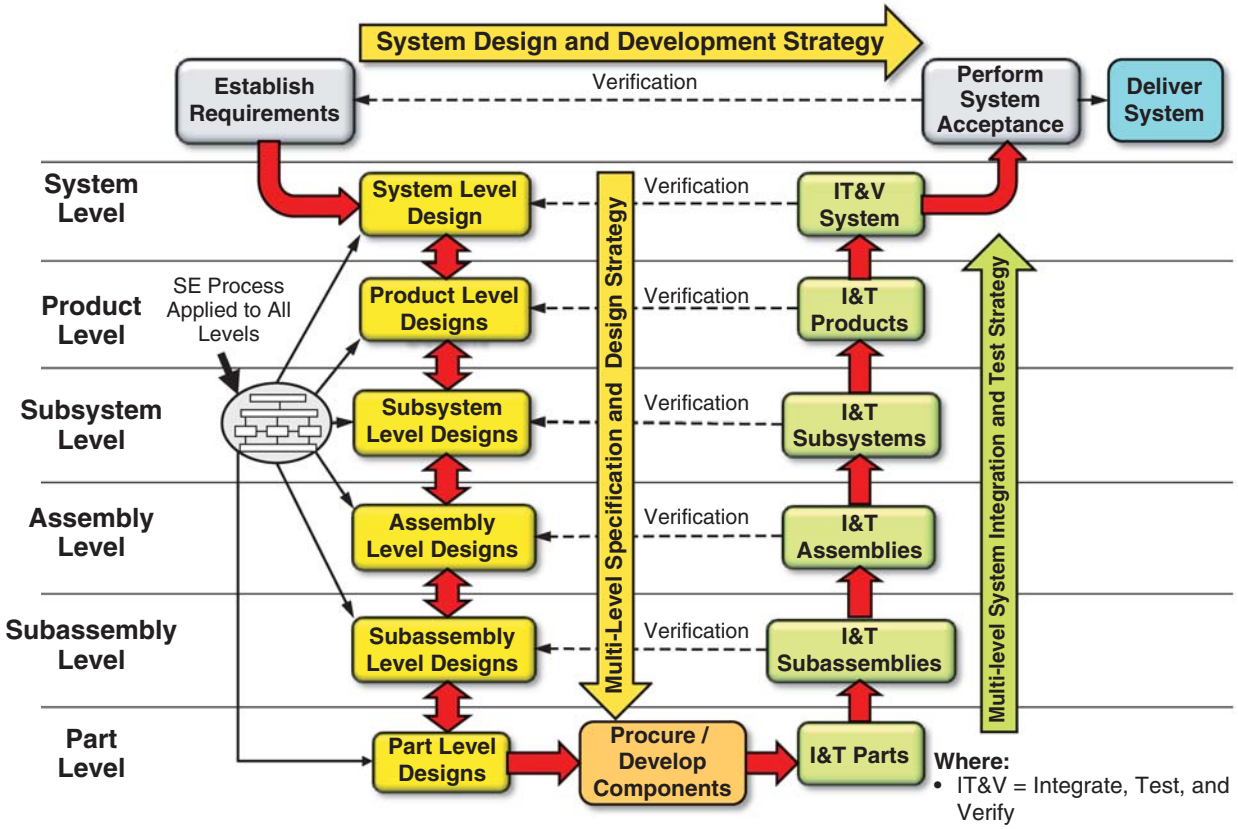


Figure 12.3 Multi-Level System Design & Development Strategy

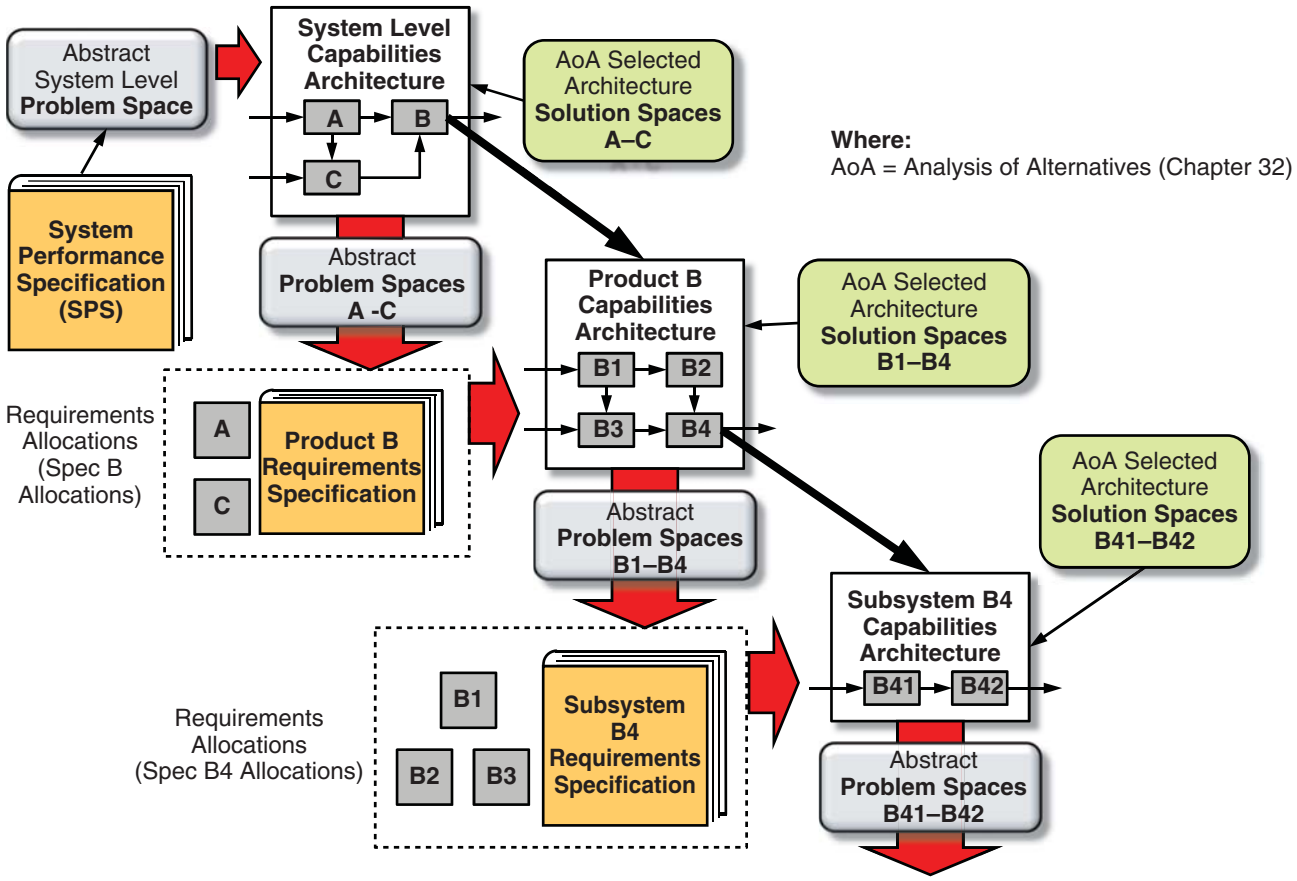


Figure 12.4 Multi-Level System Engineering Problem Solving and Solution Development Strategy

- The right side represents how the SITE Process is implemented to verify technical compliance of entities at various levels of abstraction to their respective EDSs and ultimately to the SPS.

On inspection, the top-down decomposition of the left side of Figure 12.3 may appear simple. For *ad hoc* Plug and Chug … SDBTF Engineering Paradigm Enterprises, this is accomplished by "hacking" their way to a System Design Solution. Despite the simplistic *top-down* appearance, a lot of collaboration and interaction occurs between and within each of the levels of abstraction. So, *how do we establish a strategy that avoids the ad hoc Plug and Chug … SDBTF and enables us to effectively deal with the multi-level collaboration and iterations*? After all, any entity within any Level of Abstraction represents a Problem Space to be solved via decomposition or partitioning into lower-level Solutions Spaces. Surely, there is a common problem-solving/solution development method that we can apply. The answer is *yes* as indicated by the oval icon—The SE Process Strategy—in the left side center of Figure 12.3 detailed in Chapter 14.

12.4.1 Architectural Decomposition Strategy

System Design Principle

Principle 12.1 System design is a *highly iterative*, *collaborative*, and *multi-level* process with each Level of Abstraction dependent on the maturation, stability, and integrity of higher-level specification requirements and design decisions.

To better understand what we would expect an SE problem-solving and solution development method to accomplish, Figure 12.4 provides an example. Several key points emerge from the figure:

- Specifications for entities at each level of abstraction represent an abstract Problem Space to be solved.
- Multi-discipline SEs analyze each Problem Space, formulate several viable candidate Solution Spaces, evaluate each solution via an AoA (Chapter 32), and select an optimal solution.

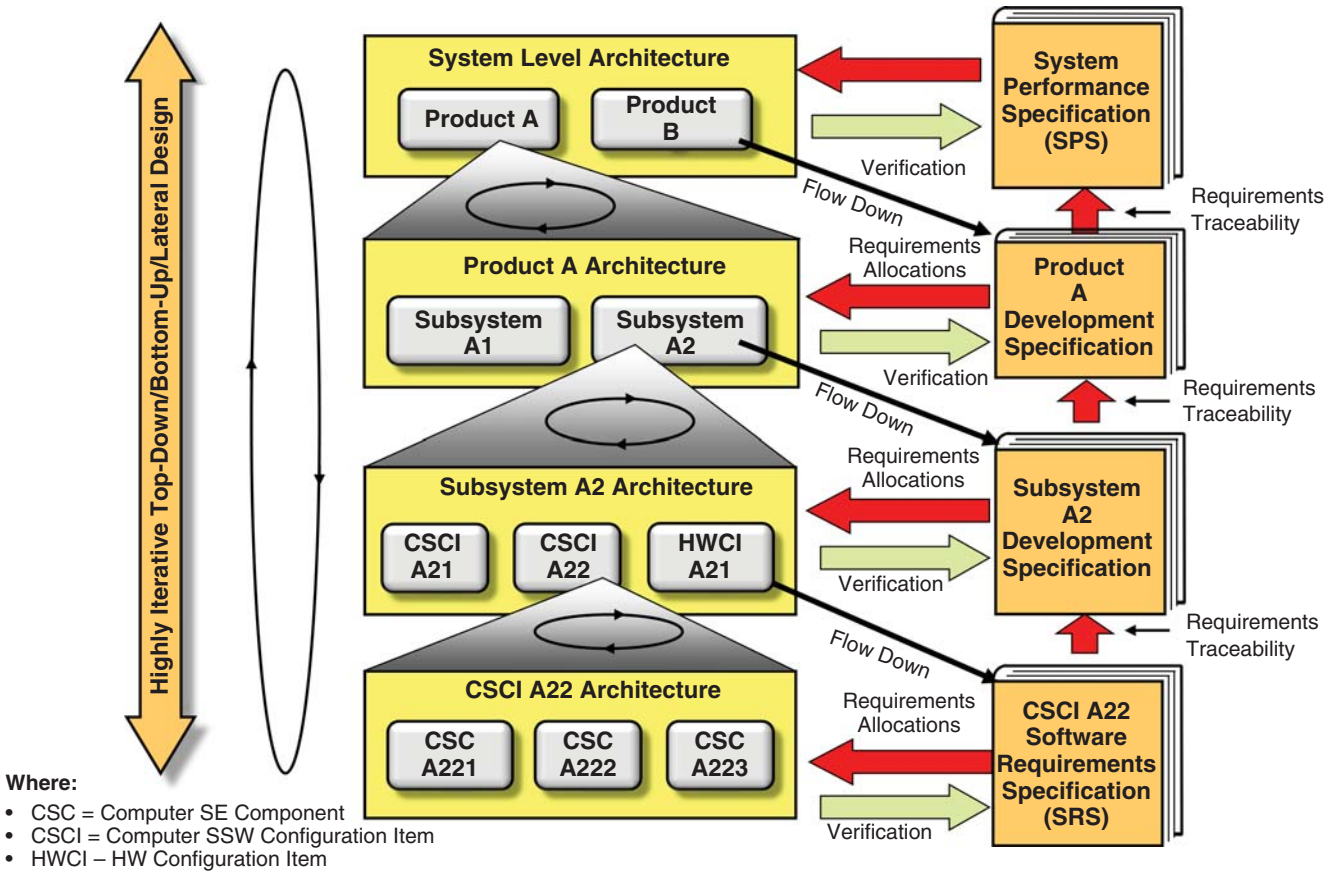


Figure 12.5 Multi-Level System Design Strategy

- The multi-level decomposition is based on *capabilities*, not physical entities. Ultimately, the architectural capability requirements will be allocated to physical system entities.

As the SE Design Process Strategy *evolves* and *matures* the System Design Solution, Figure 12.5 illustrates the typical outcomes of requirements allocations, flow down, and traceability. Observe that Engineering Designs at all Levels of Abstraction—SYSTEM, PRODUCT, and SUBSYSTEM—are:

- Derived from (1) analysis of their respective SPS or EDS requirements and (2) formulation and selection of the architecture that structurally serves as the basis for allocating the requirements to lower levels.
- Traceable to higher-level EDS and SPS requirements and subsequently to Acquirer *source* or *originating* requirements.
- *Highly iterative* within and between Levels of Abstraction.
- Are verified for compliance to their respective SPS and EDS requirements.

12.4.2 Multi-level SITE Technical Strategy

As components become available from internal development, subcontractors, and vendors, we need a strategy that enables

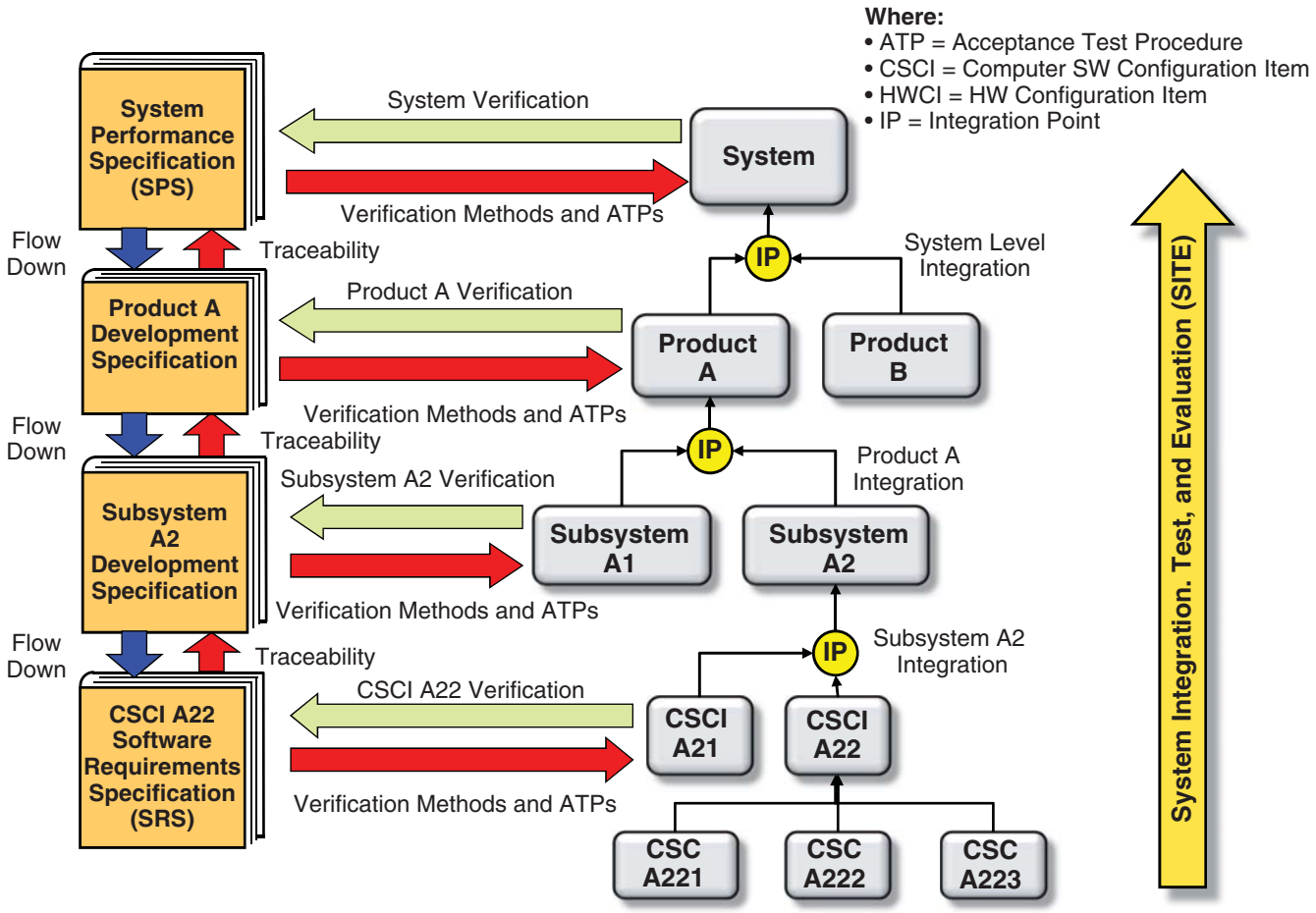


Figure 12.6 System Integration, Test, & Evaluation (SITE) Strategy

us to integrate PARTS bottom-up into SUBASSEMBLIES, SUBASSEMBLIES into ASSEMBLIES, ASSEMBLIES into SUBSYSTEMS, SUBSYSTEMS into PRODUCTS, and PRODUCTS into the deliverable SYSTEM. The challenge is: *How do we ensure that we are integrating entities into higher levels of complexity that are free from latent defects—design flaws, errors, and deficiencies - and comply with their EDS or SPS requirements*? This brings us to the need for a SITE Strategy shown in Figure 12.6.

Observe how SITE is accomplished in a generalized, bottom-up, workflow that integrates components at various levels of abstraction. Each integration step encompasses *verification* of each Entity for compliance to its EDS requirements culminating SYSTEM Level verification for compliance to its SPS.

12.4.3 DT&E Strategy

DT&E Principle

DT&E is performed by the System Developer to:

Principle 12.2

- Mitigate Developmental Configuration technology, design, and other risks
- Provide insights and assurance that the evolving System Design Solution will comply with its SPS requirements.

DT&E serves as a technical risk mitigation strategy to ensure that the *evolving* and *maturing* System Design

Solution, including its components, will be compliant with its *SPS* requirements. *So, what is DT&E and how does it apply to the System Development Workflow Strategy*?

The DAU *T&E Management Guide* (2005, p. B-6), for example, states that the objectives of DT&E are to:

1. "Identify potential operational and technological capabilities and limitations of the alternative concepts and design options being pursued.
2. Support the identification of cost-performance tradeoffs by providing analyses of the capabilities and limitations of alternatives.
3. Support the identification and description of design technical risks.
4. Assess progress toward meeting Critical Operational Issues (COIs), mitigation of acquisition technical risk, achievement of manufacturing process requirements and system maturity.
5. Assess validity of assumptions and conclusions from the AoA.
6. Provide data and analysis in support of the decision to certify the system ready for OT&E.
7. (In the case of Automated Information Systems (AISs)), support an information systems security certification prior to processing classified or sensitive data and ensure a standards conformance certification."

DT&E is performed throughout the System Design, Component and Procurement, and SITE Processes shown in Figure 13.6. Using the Supply Chain Concept (Figure 4.1), each process *verifies* that the Developmental Configuration of the *evolving* and *maturing* System Design Solution *complies* with the SPS and lower-level EDS requirements. This is accomplished via in-process and project reviews, Proof of Principle, Proof of Concept, and Proof of Technology demonstrations; engineering models: simulations, brass boards; and prototypes.

12.4.4 When Is the System Design Solution Complete?

Principle 12.3

System Design Solution Completion Principle

Contractually, a System Design Solution is not considered complete until it has been formally *verified* as *compliant* with its System Performance Specification (SPS) by the System Acquirer or User acceptance.

Technically, a System Design Solution is not complete until all *latent defects* such as design flaws, errors, and deficiencies are removed; most systems exist between these two extremes even after fielding.

Enterprises and engineers tend to believe that the System Design Solution is complete when approved at the Critical Design Review (CDR). Technically, the System Design Solution has a *level of maturity* at the CDR that is yet to be proven until formal System Acceptance is complete. Prior to that event, the evolving System Design Solution, which was placed under formal Configuration Management following the CDR, is subject to formal baseline change management procedures.

Our discussion to this point has addressed a generalized System Development Workflow Strategy. In fact, Enterprises that employ the Plug & Chug … SDBTF-DPM Engineering Paradigm will contend that this is what they do. However, remember that one of the fallacies of the SDBTF-DPM Paradigm is the *ad hoc, endless loop* that never seems to come to completion. Since one of the *major* challenges of System Development is the *elimination of latent defects* prior to User acceptance and field usage, SDBTF-DPM projects have a tendency to casually defer their "discovery" until SITE when insufficient time allows this to occur. This will be a key point of discussion in Chapter 13.

How does this discussion relate to the System Development Workflow Strategies? The reality is: the SITE Process is the *last line of defense* before User acceptance. The more time you spend testing the System/Product, the more latent defects are identified and corrected. However, this leads to a challenging question: *How much testing is both necessary and sufficient*? You can never eliminate all defects, especially on large, complex systems; however, you need to ensure that all *mission-critical defects* are eliminated to the greatest extent practical.

Author's Note 12.3

For an interesting perspective on this topic, refer to the Epilog for observations by Brian Muirhead, Jet Propulsion Laboratory (JPL) Chief Engineer.

What is the optimal amount of testing? There are no magic answers; it depends on your system and its

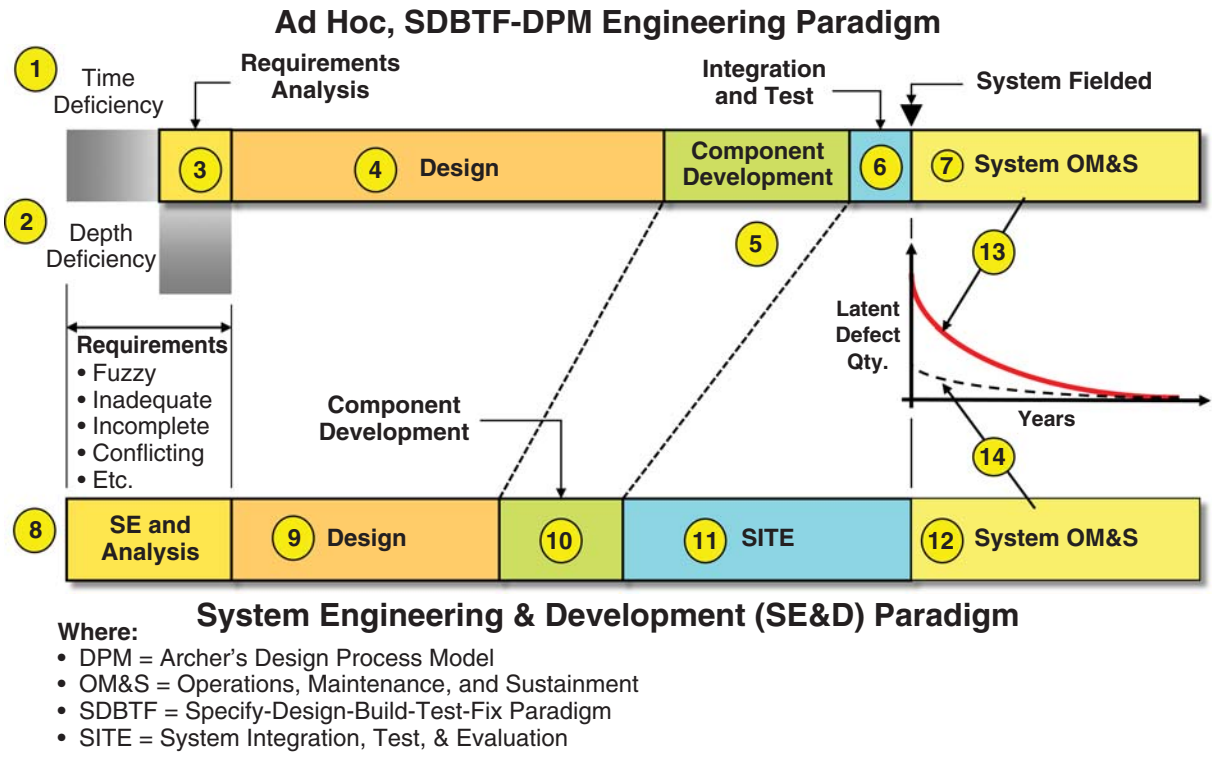


Figure 12.7 Latent Defects - *Ad Hoc*, SDBTF-DPM Engineering Paradigm Bonafide Versus Bonafide SE and Development (SE&D) Paradigm

complexity; competency of the designers, manufacturing personnel, testers; material and component integrity; and many other factors. As a general rule of thumb, some suggest at least 40% of the total project schedule is a *nominal* starting point; it could be as low as 20% or require 60%. This brings us to a key differentiating point that contrasts the SDBTF-DPM Paradigm versus SE&D. Let's explore this point further.

Figure 12.7 consists of two parallel tracks: the upper track for ad hoc, SDBTF-DPM Paradigm Enterprises and the lower track for SE&D Enterprises. Both tracks originate with the same fuzzy, inadequate, incomplete User needs, and requirements.

In general, due to a lack of an outcome-based problem-solving and solution development methodology leading to an integrated System Design Solution, SDBTF-DPM Enterprises apply their ad hoc, endless loop methods. Designs are not well integrated, interfaces are ill defined, and so forth. As a result, schedules begin of overrun completion dates. Finally, out of frustration the project rushes the design into Component Procurement and Development and subsequently into SITE. Remember our earlier discussion of Figure 2.2 (Mini-Case Study 2.1) and redesign efforts to deal with the latent defects?

Since the System Design Process overran its schedule, the length of the SDBTF-DPM SITE is significantly shortened to ensure "on-time delivery." When delivered, the SYSTEM/PRODUCT has a large quantity of latent defects—*unknown–unknowns*—as indicated by the curve on the right side. The User may be required to issue a maintenance contract to *discover* and *eliminate* latent defects, which could take months or years. This adds another dimension of complexity—who is *accountable* for paying for corrective actions to eliminate the defects.

Now, consider an SE&D Enterprise, which employs a *problem-solving* and *solution development* methodology based on the SE Process Model addressed in Chapter 14. Although the SE&D Design Phase may be same or slightly longer, it completes the design on schedule. When the

SYSTEM/PRODUCT enters SITE, the project activities are properly focused where they should be—*verifying* compliance to specification requirements, *testing* the design's robustness of the design to drive out weak components, and *eliminating* any remaining latent defects. On completion of SYSTEM/PRODUCT acceptance, it may still contain some latent defects; however, the quantity should be small compared to the SDBTF-DPM Paradigm – right side of Figure 12.7. The User is satisfied and the Enterprise's performance and reputation are highly regarded.

12.5 CHAPTER SUMMARY

Chapter 12 provided: (1) an overview of the System Development Strategies addressed in Chapters 12–18 and (2) introduced the central framework of systems, the System Development Workflow Strategies.

In general, SYSTEM/PRODUCT development processes are common across industry and government. Both types of Enterprises specify, design, build, integrate, and test the SYSTEM/PRODUCT. However, there are pronounced differences between Commercial Product Development and Contract System Development as shown in Figure 5.1.

Next, we defined the System Development Process Workflow and identified a sequence of five key processes (Figure 12.2) used to develop a SYSTEM/PRODUCT. These include the:

- System Engineering Design Process
- Component Procurement and Development Process
- SITE Process
- Authenticate System Baselines Process
- (Optional) OT&E Process

To perform the SE Design Process, we established an overall Multi-Level SE Technical Development Strategy (Figure 12.3), which encompasses both System Design and System Integration, Test, & Evaluation (SITE). We created a Multi-Level Problem Solving and Solution Development Strategy (Figures 12.4 and 12.5) for performing System Design and a Multi-Level SITE Technical Strategy (Figure 12.6). Since a key aspect of SE&D is risk mitigation, we established an overall DT&E Strategy addressed later in Chapter 13.

As our final topic, we addressed the question: *When is the System Design Solution Complete*? Enterprises and Engineers often believe that the System Design is complete when it is approved at a CDR. A System Design Solution is not officially complete until it has been *verified* and *validated*, if required, and accepted by the System Acquirer. Technically, a System Design is not considered complete until all latent defects have been removed.

12.6 CHAPTER EXERCISES

12.6.1 Level 1: Chapter Knowledge Exercises

1. What are the workflow sequence steps in System Development Phase?
2. What is the Developmental Configuration?
3. When is the Developmental Configuration initiated? When is it complete?
4. What is a *first article* system? Is there only one instance of a first article system?
5. What is DT&E?
6. What is the objective of DT&E?
7. When is DT&E performed during the System Development Phase?
8. Who is responsible for performing DT&E?
9. What is OT&E?
10. When is OT&E performed during the System Development Phase?
11. What is the objective of OT&E?
12. Who is responsible for performing OT&E?
13. What is the System Developer's role in OT&E?

12.6.2 Level 2: Chapter Knowledge Application Exercises

Refer to www.wiley.com/go/systemengineeringanalysis2e

12.7 REFERENCES

DAU (2005), *DAU Test and Evaluation Management Guide*, 5th ed., Ft. Belvoir, VA: Defense Acquisition University (DAU) Press.

DAU (2012), *Glossary: Defense Acquisition Acronyms and Terms*, 15th ed. Ft. Belvoir, VA: Defense Acquisition University (DAU) Press. Retrieved on 6/1/15 http://www.dau.mil/publications/publicationsDocs/Glossary_15th_ed.pdf.

MIL-HDBK-1908B (1999), *DoD Definitions of Human Factors Terms*, Washington, DC: Department of Defense (DOD).

MIL-STD-973 (1992, Canceled 2000), *Military Standard: Configuration Management*, Washington, DC: Department of Defense (DoD).

SEVOCAB (2014), *Software and Systems Engineering Vocabulary*, New York, NY: IEEE Computer Society. Accessed on 5/19/14 from www.computer.org/sevocab.

13

SYSTEM VERIFICATION AND VALIDATION (V&V) STRATEGY

Principle 13.1

Engineering Truth Principle

If Engineering fails to perform its job well, System Users – PERSONNEL, the EQUIPMENT, the general public, and the environment may be placed at risk.

Engineering has a simple truth as noted by Principle 13.1. You can create the most "elegant" Engineering design; however, if it fails and/or hurts people or the customer does not like it, *elegance* is *inconsequential*.

A colleague tells a story that during the National Aeronautics and Space Administration (NASA) Apollo Space Program, astronauts travelled to field centers to meet with Engineers, designers, to discuss the progress, status, and issues concerning development of the launch vehicle, space capsule, and components. Standing before the gathering, the individual introducing the astronaut(s) made a simple statement: *if you do not do your job well, the astronaut standing before you could die*.

As the astronauts met and shook hands with the individuals, it was a very sobering experience physically shaking hands with someone totally dependent on *how well* you did your job no matter how large or small. Every project is not an Apollo Project; however, the Engineering *integrity* and *accountability* as a professional and as a human is no less.

With the preceding paragraph as a theme, this chapter addresses multi-level System Verification and Validation (V&V). In general:

- Verification seeks to answer the question: Is the system or product being developed in compliance with its contract, specification, and requirements?
- Validation seeks to answer the question: Will the system, product, or service being developed to fulfill its Users' operational needs?

For many years, Enterprises believed that V&V were activities you performed *after* a system, product, or service was developed prior to delivery. That approach results in many problems including increased costs, overrun budgets, schedule slips, risk, and factors. Even worse as noted in the NASA example earlier, people could die, be injured, or be placed at risk if the Engineering was not performed properly. As discussions will reveal, to ensure the technical *validity* and *integrity* of the system, product, or service, V&V must be performed by everyone beginning on Day #1 and continuing throughout the System Development Phase. Then, when the system, product, or service is fielded, V&V are transferred to the User to continue.

13.1 DEFINITIONS OF KEY TERMS

- **Analysis** (Verification Method)—"Use of analytical data or simulations under defined conditions to show theoretical compliance. Used where testing to realistic conditions cannot be achieved or is not cost-effective" (INCOSE, 2011, p. 129).
  "Use of mathematical modeling and analytical techniques to predict the compliance of a design to its requirements based on calculated data or data derived from lower system structure end product validations." (NASA, 2007, p. 266)

System Engineering Analysis, Design, and Development: Concepts, Principles, and Practices, Second Edition. Charles S. Wasson.

Companion Website: www.wiley.com/go/systemengineeringanalysis2e

Certification—"Written assurance that the product or article has been developed and can perform its assigned functions in accordance with legal or industrial standards. The development reviews and verification results form the basis for certification; however, certification is typically performed by outside authorities, without direction as to how the requirements are to be verified. For example, this method is used for electronics devices via CE certification in Europe and UL certification in the United States and Canada" (INCOSE, 2011, p. 130).

- **Classification of Defects**—"The enumeration of possible defects of the unit or product, classified according to their seriousness. Defects will normally be grouped into the classes of critical, major or minor: however, they may be grouped into other classes, or into subclasses within these classes" (MIL-STD-105E, p. 2).
- **Critical Staging or Control Point**—A key milestone such as a technical review or audit programmed into a project schedule to assess the status, progress, maturity, or risk of the evolving System Design Solution as a basis for grant an authority to proceed including resource commitments. Comparable in concept to commercial industry stage–gate reviews.
- **Deficiency**—"Operational need minus existing and planned capability. The degree of inability to successfully accomplish one or more mission tasks or functions required to achieve a mission or mission area. Deficiencies might arise from changing mission objectives, opposing threat systems, changes in the environment, obsolescence, or depreciation in current military assets. 2. In contract management—any part of a proposal that fails to satisfy the government's requirements" (DAU, 2011, p. B-64).

  "Deficiencies consist of two types:

  - Conditions or characteristics in any item which are not in accordance with the item's current approved configuration documentation.
  - Inadequate (or erroneous) item configuration documentation, which has resulted, or may result, in units of the item that do not meet the requirements for the item" (MIL-STD-973, 1992, p. 10).
- **Demonstration** (Verification Method)—"A qualitative exhibition of functional performance, usually accomplished with no or minimal instrumentation" (INCOSE, 2011, p. 130).

  "A qualitative exhibition of functional performance, usually accomplished with no or minimal instrumentation." (NASA, 2007, p. 275)
- **Developmental Design Verification**—A rigorous process that evaluates a System or Entity using one or more Verification Methods by collecting and presenting performance outcomes and data as objective evidence that a design complies with its specification or design requirements. You only verify a design once. Refer to Product Verification for verification of production instances of a previously verified design.
- **Deviation**—Refer to Chapter 19's Definitions of Key Terms.
- **Discrepancy**—A statement documenting the performance variance between what has been observed or measured versus specified requirements.
- **IV&V**—"Systematic evaluation of software products and activities by an organization that is not responsible for developing the product or performing the activity being evaluated." (SEVOCAB, 2014. P. 149 - Copyright, 2012, IEEE. Used by permission.) (Source: ISO/IEC/IEEE 24765:2010, 2010).
- **Inspection** (Verification Method)—"Visual examination of the item (hardware and software) and associated descriptive documentation which compares appropriate characteristics with predetermined standards to determine conformance to requirements without the use of special laboratory equipment or procedures" (Adapted from DAU, 2012, pp. B-108).

  "An examination of the item against applicable documentation to confirm compliance with requirements. Inspection is used to verify properties best determined by examination and observation (e.g., paint color, weight)" (INCOSE 2011, p. 129).
- **Product Verification**—A type of verification in which each physical instance of a System or Entity—for example, model and serial number—is (1) implemented in accordance with a *previously verified* Developmental or Production Design, (2) verified to be functioning as required, and (3) deemed to be free from latent, workmanship, and material defects.
- **Requirements Verification Traceability Matrix (RVTM)**—"Matrix correlating requirements and the associated verification method(s). The VRTM (RVTM) defines how each requirement (functional, performance, and design) is to be verified, the stage in which verification is to occur, and the applicable verification levels" (FAA SEM, 2006, Vol. 3, p. B-14).
- **Similarity** (Verification Method)—The process of demonstrating, by traceability to source documentation, that a previously developed and verified SE design or item applied to a new program complies with the same requirements thereby eliminating the need for design level reverification.
- **Test** (Verification Method)—The act of executing a formal or informal scripted procedure, measuring and recording the data and observations, and comparing to expected results for purposes of evaluating a system's

response to a specified stimuli in a prescribed environment with a set of constraints and initial conditions.

"An action by which the operability, supportability, or performance capability of an item is verified when subjected to controlled conditions that are real or simulated. These verifications often use special test equipment or instrumentation to obtain very accurate quantitative data for analysis" (INCOSE, 2011, p. 130).

- **Testbed**—"A system representation consisting of actual hardware and/or software and computer models or prototype hardware and/or software" (DAU, 2012, p. B 229).
- **Validation of Records** (Verification Method)—The process of demonstrating, by presentation of authenticated verification data, that a previously verified design complies with its specification and design requirements, that is, drawings, thereby eliminating the need for System- or Entity-level design reverification, assuming that no changes to the Product Baseline have been made.
- **Validation Table**—"A listing of all requirements that describes if a requirement has been validated, where the requirement may be found, source of validation, corrective action to be taken if necessary, and the corrective action owner" (FAA SEM, 2006, Vol. 3, p. B-14).
- **Verification**—Refer to Chapter 2's Definitions of Key Terms.
- **V&V**—"The process of determining whether the requirements for a system or component are complete and correct, the products of each development phase fulfill the requirements or conditions imposed by the previous phase, and the final system or component complies with specified requirements." (SEVOCAB, 2014, p. 347 - Copyright, 2012, IEEE. Used by permission.) (Source: ISO/IEC/IEEE 24765:2010, 2010).
- **Waiver**—Refer to Chapter 19's Definition of Key Terms.

13.2 APPROACH TO THIS CHAPTER

Chapter 12 established a technical *strategy* or *roadmap* from logically transforming a User's abstract operational needs and vision into a deliverable system, product, or service. Implementation of this strategy, however, has a probability of success. *Why*?

Overall, the probability of success of any strategy is dependent on the education, training, discipline, processes, methods, and tools employed by project personnel. Even in the best of situations, humans are naturally prone to make mistakes, based on interpretations, miscommunications, and so forth. Without a robust strategy of *checks* and *cross-checks* to eliminate or reduce *latent defects*—design errors, flaws, and deficiencies—the probability of success is reduced significantly and diminishes with increasing system complexity and project size. If a system, product, or service is delivered to the User with *latent defects*, the User's mission can be jeopardized. The net result can be injury or loss of life to its Users - operators, maintainers, and, and the general public, or damage or destruction to the EQUIPMENT, property, or the environment.

From the System Acquirer's and User's perspectives, there are three key technical outcomes that determine system success: (1) compliance to specification requirements; (2) satisfaction of the User's operational need in terms of its operational utility, suitability, usability, availability, effectiveness, and efficiency (Principle 3.11); and (3) a system that is free of *latent defects*, workmanship, and component integrity issues.

Chapter 13 introduces the V&V strategy for System Engineering and Development (SE&D) with a focus on ensuring that the three key technical outcomes above are achieved. Key questions to be addressed by the System Acquirer, User, and Developer are:

1. Is the system, product, or service being developed in *compliance* with the contract, project charter, or task order specification or design requirements?
2. What *objective evidence* is being collected to prove compliance?
3. What *methods* do we employ to prove compliance for the least cost and schedule impact?
4. What assurance do we have that the deliverable system or product will be compliant with its requirements and free of latent defects?

On System Acceptance and delivery:

1. How do we – Acquirer and User - know that the system, product, or service will satisfy our documented operational need(s) in terms of resolving one or more Problem Space(s)? - Critical Operational or Technical Issues (COIs/CTIs)
2. How do we know that the system or product delivered will identically match its documentation to facilitate maintenance?
3. How do we ensure that the SYSTEM or PRODUCT can be easily maintained?

For many years, System Development was a simple, two-step approach:

Step 1—Design, develop, and test the System or Product.

Step 2—Verify compliance to requirements. If not, go back to Step 1 and repeat until complete.

This exemplifies the *ad hoc*, *endless loop*, Plug and Chug … Specify–Design–Build–Test–Fix (SDBTF). Design Process Model (DPM) Engineering Paradigm (Chapter 2) in action. The two steps above worked until Enterprises driven by global competition and profit came to a stark realization. *Inefficient* and *ineffective* System Development, especially Engineering design that resulted in large amounts of rework or scrap, became prohibitively expensive.

In commercial production environments, warehouses of *unusable* components stood as a monument to these methods. Root cause analysis such as the *5 Whys* Analysis (Chapter 5) revealed that *latent defects* resulting from poor Engineering design methods and quality control were a major technical issue. During the 1980's to minimize waste and improve profitability, Enterprises believed that "documenting" their processes would solve the problem. They did correct some of the waste and profitability issues. However, as addressed earlier in Chapter 2, processes did not correct latent defects in poor SDBTF-DPM Engineering Paradigms.

How do latent defects migrate into a multi-level System Design Solution? The DAU (2005) illustrates the point very succinctly using Figure 13.1. Observe how errors *proliferate* through the System Development Phases. Two key points:

1. Errors originate in the form of *incorrect* and *poorly written* specification requirements; latent defects—misinterpretation of specification requirements, design errors, design flaws, and deficiencies; inadequate, marginal, or weak components or materials; poor manufacturing processes and workmanship practices; poor test procedures; and inadequate testing.
2. Failure to correct errors when they occur results in a *proliferation* of errors *downstream*. Error proliferation downstream, in turn, translates into unnecessary costs and schedule slips that impact delivery, profitability, and customer satisfaction.

Author's Note 13.1

Proliferation of Latent Defects

Observe that Figure 13.1 illustrates how latent defects multiply as they progress in time "down the rapids" of SE&D. In that context, Figure 13.1 merely addresses the technical aspects of latent defects. The cost-to-correct latent defects increases similarly. Principle 13.2 and Table 13.1 will address this point later. To further illustrate the challenges of eliminating *latent defects*, consider the graphic illustrated in Figure 13.2. Latent defects enter the User Needs Analysis; the System Design Process; Component Procurement & Development Process; and System Integration, Test, and Evaluation (SITE) Strategy (Figure 12.3). Latent defects that are left *undiscovered* migrate with the deliverable system, product, or service to the User. Unless the User discovers them during the Deployment and Operations, Maintenance, and Sustainment (OM&S) Phases, they may lie dormant as hazards that culminate in system incidents or accidents when specific conditions occur (Figure 24.1).

Observe the Analytical Time Deficiency and Depth annotations on the left side of Figure 13.2. Metaphorically, these are the "headwaters" where latent defects enter SE&D. Wasson's Task Significance Principle 23.1 introduced later very succinctly addresses how SDBTF-DPM Paradigm managers and executives naively impose unrealistic time restrictions on these critical analyses. Then, act surprised when it is discovered that the User's Problem Space was so poorly understood.

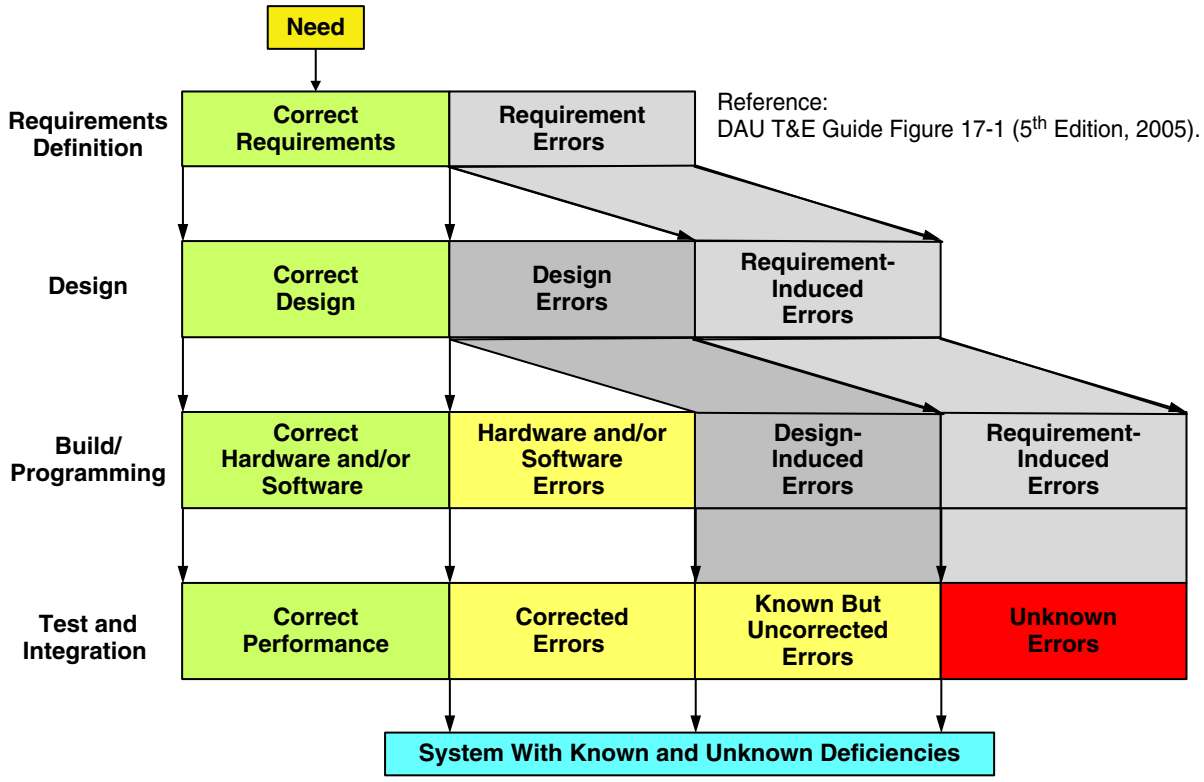


Figure 13.1 The Error Avalanche. Source: DAU (2005, Figure 17-1, p. 17-3)

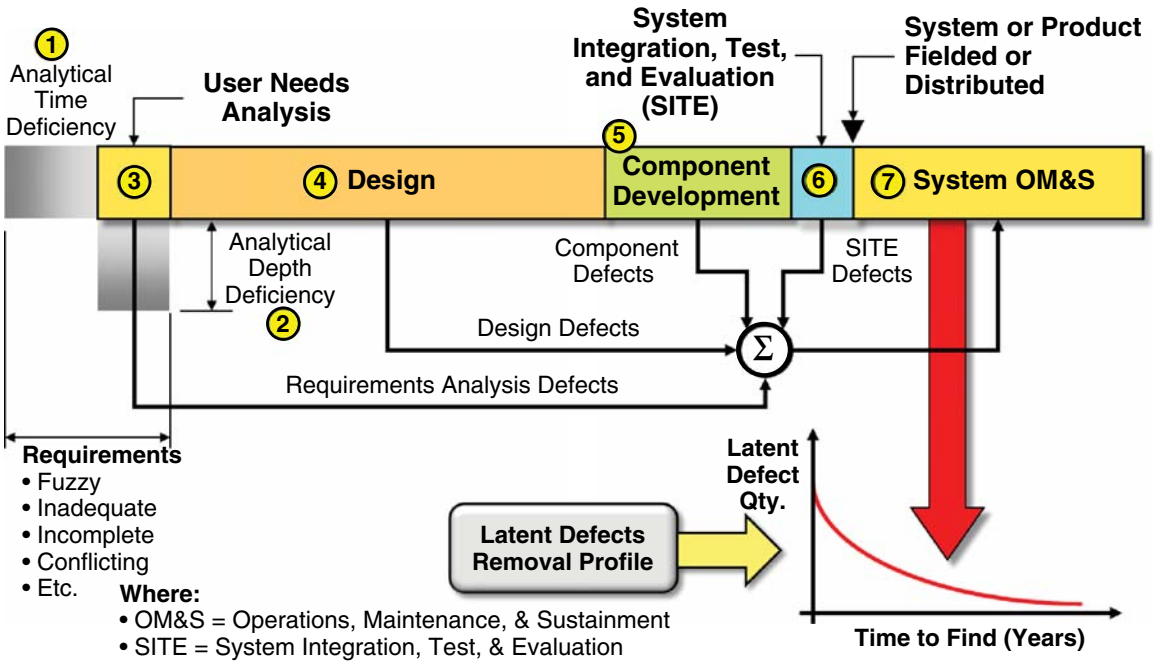


Figure 13.2 The Cumulative Effects of Undiscovered Latent Defects

Principle 13.2

Latent Defects Cost Principle

Identify and correct latent defects immediately. Depending on when a latent defect enters the System Development Phase, the cost-to-correct may increase as much as a 100 times by the time System Integration, Test, & Evaluation (SITE) is performed.

With the increasing *complexity* and growing expenses of software development, Dr. Barry Boehm (1981, pp. 39–40) published results of studies in his 1981 text, *System Engineering Economics*, to quantify the cost-to-correct latent software defects as a function of the System Development Phase processes. The cost-to-correct concept was labeled by some as the 100X Cost Rule for software. Table 13.1 provides results from a 2004 NASA study (Stecklein, et al, 2004) comparing the cost to correct latent defects in software and systems.

Dr. Boehm (1981, p. 41) notes that the data, reflecting a sequential approach – Requirements Design → Code → Development Test → Acceptance Test → Operation, has a trade-off curve indicating less cost using a "first-cut" prototyping approach in lieu of more time-defining requirements. The point is that you have a choice: (1) correct latent defects such as design errors, design flaws, or deficiencies early or (2) pay an *exponentially* increasing cost as you progress through each of the System Development Phase processes (Figure 12.2).

TABLE 13.1 Comparison of Software and Systems Cost Factors to Correct latent Defects

Life Cycle Phase	Software Cost Factors	Systems Cost Factors
Requirements	1X	1X
Design	5X–7X	3X–8X
Build	10X–26X	7X–16X
Test	50X–177X	21X–78X
Operation	100X–1000X	29X–1615X

Source: Steckleinet al. (2004), Table 13, p. 10.

The 100X Cost Rule proliferated throughout industry, government, and textbooks for decades until 2006. Dr. Boehm (Shull, et al., 2006) and his colleagues with the Center for Empirically-Based Software Engineering (CeBASE) offered the following guidance in a paper titled "What We Have Learned about Fighting Defects":

1. "Item 1' Finding and fixing a severe software problem after delivery is often 100 times more expensive than finding and fixing it during the requirements and design phase.
2. Item 1.1. Finding and fixing non-severe software defects after delivery is about twice as expensive as finding these defects predelivery." (CeBASE, 2006, p. 3)

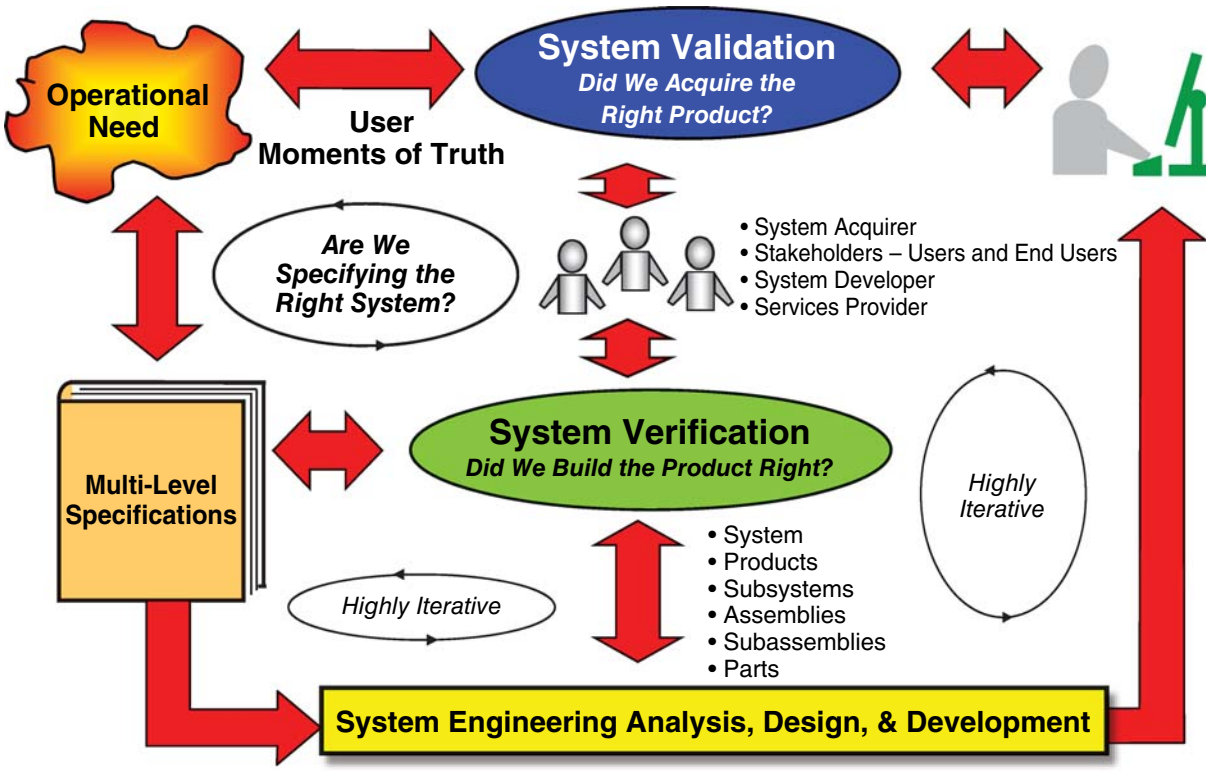


Figure 13.3 System V&V Concept Overview

Most Engineers receive little or no SE education and minimal training about avoiding, minimizing, and eliminating latent defects. After all, Engineers spend four years obtaining a degree learning to innovate and create elegant designs, not eliminating latent defects unless the outputs fail to accomplish the required performance values! As a result, the Defect Quantity (Figure 13.2) in fielded systems or products may take years to *discover, isolate,* and *eliminate* as illustrated by the graph in the lower right corner. Hopefully, these are found with *minimal* impact to mission and System performance without catastrophic injury or loss of life!

Given this backdrop concerning the importance of delivering systems, products, and service that are free from latent defects, let's begin our discussion with an introductory overview of System V&V Concepts.

13.3 SYSTEM V&V CONCEPTS OVERVIEW

Principle 13.3

Verification Principle

Verification answers the Stakeholder and Developer's question: Are we developing a SYSTEM, ENTITY, or task in compliance with its specified requirements?

Principle 13.4

Validation Principle

Validation answers the User's question: Did we acquire the right SYSTEM, ENTITY, or work product to satisfy our operational needs?

Figure 13.3 provides a SYSTEM-Level illustration of the V&V concept. In general, *System Verification* seeks to answer the question:

- Was the system, product, or service developed in compliance with its Multi-level Specifications?

Author's Note 13.2

Colloquial Characterizations of Verification

For the Verification question above written as "Was the system built right?" the problem with this statement is:

1. "Built" refers to the Fabrication, Assembly, Integration, and Test—Verification—(FAIT) of each individual component and higher levels of integration.
2. What qualifies as "right"? Is anything considered "right" in the eyes of the beholder acceptable? Using sound Engineering practices? In accordance with standards? User specification requirements? All of the above?

Verification encompasses more than "built right." This includes *everything* required to perform SE and Development (SE&D) of a system, product, or service and prove compliance. In the truest sense of the term, *verification* is a method requiring *independent* evaluation, comparison, and assessment. "You said you were going to do this. Where's the objective evidence—Quality Records (QRs) - to prove it? Does it comply—meet or exceed—the specified requirements?"

Returning to Figure 13.3, observe the *focus on compliance* to Multi-Level Specifications. The *implicit assumption* here is that the System Acquirer translated the User's Operational Need—Problem Space—into a set of *well-defined* System Requirements Document (SRD) requirements that *specify and bound* a Request for Proposal (RFP) Solution Space in terms of their accuracy, preciseness, and completeness.

Author's Note 13.3

System Acquirer Specifications

Many times a System Acquirer's RFP will include an SRD. The offerors then submit a proposal that includes a DRAFT System Performance Specification (SPS). In turn, the offeror's proposal including the SPS will be evaluated. If selected, the offeror's SPS requirements will be negotiated and finalized as part of an element of a contract.

Once the system, product, or service has completed System-Level Verification, the System Acquirer or User may choose to perform System Validation, which answers the question:

- *Does the System, product, or service satisfy our Operational Need*?

If the answer at system or product delivery turns out to be No, then the *implicit assumption* about how well the original SRD and subsequently the SPS *specified* and *bounded* the Solution Space(s) in terms of resolving the Problem Space was *flawed* or *incorrect*.

Author's Note 13.4

Colloquial Characterizations of Validation

Again, as in Author's Note 13.2, the colloquial question is often: *Did we acquire the "right" system to satisfy our Operational Need*? What constitutes "right"? Either the system, product, or serve meets the need or it does not. "Right" infers there is some magical threshold for comparison. The central question then becomes: *where is the "threshold" articulated in a document that defines "rightness"* (Principle 13.3)?

Unless the need is well documented, humans tend to shift the threshold for satisfaction, especially when they discover they acquired the *wrong system* or *capabilities* required to fulfill their operational need. "We will know it when we see it" is a common expression. User performance-based outcome Operational Requirements Documents (ORDs) or Capability Development Document (CDD), Statements of Objectives (SOOs), Test and Evaluation Master Plans (TEMPs) that include Key Performance Parameters (KPPs), Measures of Effectiveness (MOEs), and Measures of Suitability (MOSs) are key to establishing operational need thresholds for determining what constitutes system, product, or service success in the User's mind.

As an introductory overview, this provides a basic understanding of the System V&V concept.

Enterprises and Engineers often proliferate myths about System Verification that are untrue. That brings us to *Debunking the SE Document Centric Myth*.

13.3.1 Debunking the V&V Myth

Principle 13.5

Life Cycle V&V Principle

V&V are performed *continuously* and *relentlessly* throughout every System/Product Life Cycle Phase, contract/subcontract, project charter, or task.

Some SDBTF-DPM Enterprises and engineers have a *misperception* that V&V are only performed on the completed System prior to System Acceptance. This is *factually incorrect* and *misguided*! V&V are performed throughout every phase of the System/Product Life Cycle by the System Acquirer, User, System Developer, Services Provider, and subcontractor Enterprises.

Unfortunately, this myth proliferates when uninformed people with limited understanding of V&V create graphics similar to the one shown in Panel A of Figure 13.4. Observe the symbolic V-Model (Figure 15.2) in Panel A with boxes labeled "Verification" and "Validation." This is a *partial* truth; however, the problem is one of context. The boxes should be labeled "System-Level Verification" and "System Validation" as shown in Panel B. The incorrect labeling in Panel A is often shown in presentations as well as documents posted on the World Wide Web. Then, others, who are equally *uninformed*, *naively* proliferate these incorrect forms (misinformation). Learn to recognize and appreciate the difference!

In summary, it is important to remember that System Developer V&V are performed *simultaneously*, *continuously*, and *relentlessly* beginning with release of a System Acquirer's Request for Proposal (RFP), contract award, project charter, or task order approval, whichever is applicable. Then, continues through System delivery and acceptance by the System Acquirer. On System acceptance, the System Acquirer / User accepts accountability for V&V for the remainder of the System/Product Life Cycle.

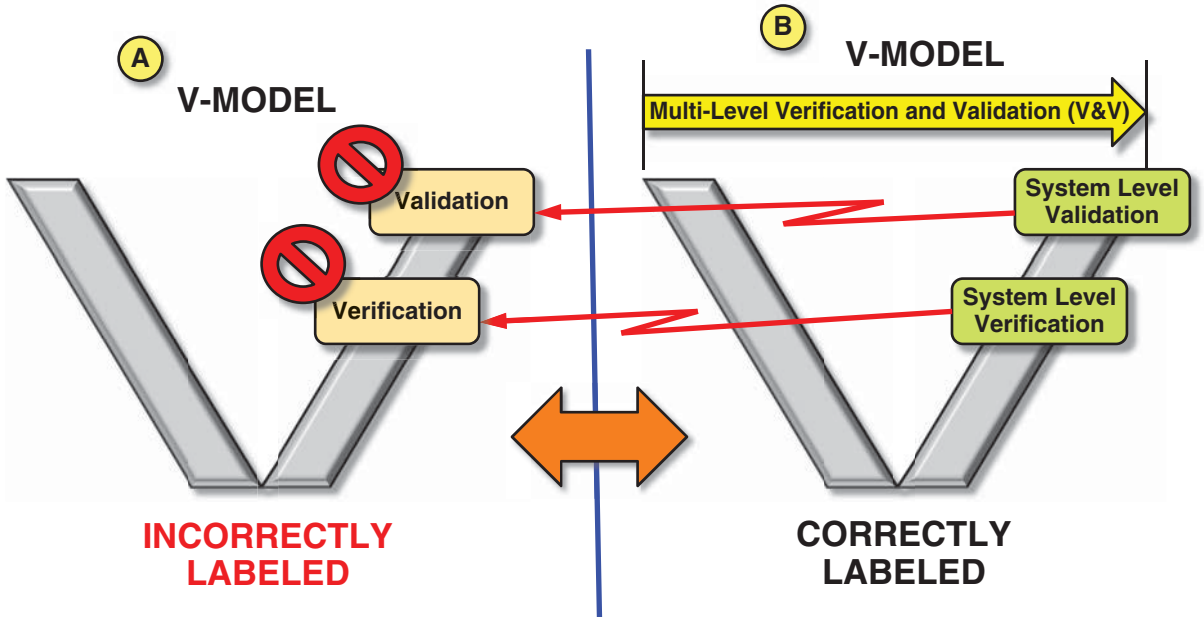


Figure 13.4 How Incorrectly Labeled V&V Graphics unwittingly Promote Misinformation

13.3.2 V&V Activities throughout the System/Product Life Cycle

V&V Applicability Principle

Principle 13.6 V&V applies to every stage of system development beginning with the proposal phase and continuing after Contract Award until system delivery and acceptance completion.

Defects Elimination Principle

Principle 13.7 The number of *latent defects* in the fielded system, product, or service is dependent on (1) a willingness and commitment or provide resources to avoid and eliminate them; (2) personnel competency—experience, knowledge, and training; (3) employment of the *right* tools, processes, and methods; and (4) time available to perform the job correctly.

V&V activities are performed throughout the System/Product Life Cycle. They are:

1. Performed throughout the duration of a contract, project charter, or task order.
2. Conducted for any type of work activities and their work products such as proposals, contracts, demonstrations, prototypes, models, and simulations.
3. Conducted as *formal* and *informal* conferences, meetings, technical reviews, audits demonstrations, and tests.
4. Performed to review decision artifacts QRs at *critical staging, control, or gating points* to assess *compliance* of the evolving system design solution to technical plans, specifications, and tasks.
5. Performed periodically to assess the *quality and integrity of* the *evolving and maturing multi-level* System Design Solution in terms of:
   a. Technical compliance with requirements.
   b. Traceability to User source or originating requirements.
   c. Validity of the solution to solve the User's Problem Space.
   d. Consistency and completeness of decision artifacts that guide SE&D.
6. Performed to identify *latent defects* such as design errors, flaws, and deficiencies.

Observe that Item 4 complies with the ISO 9001 requirements but *does not* assess the *validity* of the System Design Solution. Item 5 assesses the technical validity of the System Design Solution. When technical reviews and audits are conducted, Items 4 and 5 should be addressed simultaneously.

13.3.3 V&V Standards

From a Systems Engineering perspective, three key standards establish guidance for organizational and project V&V activities: ISO/IEC 15288:2008, the Capability Maturity Model Integration (CMMI), and ISO 9001:2008. Let's briefly explore each of these.

13.3.3.1 ISO 15288:2008 V&V Requirements

ISO/IEC 15288:2008 establishes requirements for V&V processes in the following sections:

- Para. 6.4.6—Verification Process.
- Para. 6.4.8—Validation Process.

13.3.3.2 CMMI V&V Requirements

Chapter 2 introduced the role of the Carnegie Mellon CMMI Institute administration of several CMMI models. These models include Services (CMMI-SVC), Development (CMMI-DEV), and Acquisition (CMMI-ACQ). The CMMI-DEV establishes guidance for establishing and improving their organizational V&V capabilities in two key areas (CMMI-DEV, 2010):

- Process Area 21—Validation (Maturity Level 3).
- Process Area 22—Verification (Maturity Level 3).

13.3.3.3 ISO 9001:2008 V&V Requirements

V&V should be an integral part of any Enterprise's Organizational Standard Process (OSP). For those Enterprises that have attained or plan to attain ISO 9001 certification, V&V are key elements of an organizational Quality Management System (QMS). ISO 9001:2008, for example, establishes the following requirements for QMS V&V activities:

- Section 7.2.2—Review of Requirements Related to the Product.
- Section 7.3.5—Design and Development Verification.
- Section 7.3.6—Design and Development Validation.

13.3.3.4 What the Standards Require

System Development V&V Strategy Principle

Principle 13.8 A System Development V&V Strategy requires three elements:

1. A roadmap consisting of incremental V&V activities to progress from Contract Award to system, product, or service delivery and acceptance
2. A task-based plan of action for implementing the strategy synchronized to milestone-driven events, accomplishments, and criteria.
3. Documented *objective evidence* of task completions—that is, QRs—that demonstrates that you accomplished the planned results.

In general, these standards require Enterprises and projects to perform three types of actions:

1. Plan activities to be performed with measurable work product outcomes.
2. Perform the planned activities.
3. Produce QRs—for example, work products—as objective evidence that you accomplished what you planned to do. QRs include work products such as conference meeting minutes, presentations, documents, drawings, and email.

Given this introductory overview of System V&V, let's explore each topic in-depth beginning with *System Verification Practices*.

13.4 SYSTEM VERIFICATION PRACTICES

System Verification Principle

Principle 13.9 *System verification* assesses the compliance of a work product's attributes, characteristics, and performance-based outcomes to one or more contract, project charter, specification, or task requirements.

13.4.1 System Verification Overview

Verification includes activities that assess compliance of:

- Project work products, such as plans, schedules, budgets, technical decisions, specifications, designs, prototypes, and test results, for consistency, adequacy, completeness, and traceability to contract or task requirements
- The deliverable system, product, or service to its specification requirements

When *latent defects* or *discrepancies* in system outcomes and performance are identified, corrective actions are taken leading to *reverification* to ensure that latent defects or discrepancies have been eliminated.

System Acquirers and System Developers have two perspectives of V&V. For example:

- **System Acquirer V&V Perspective** - Continuously *verify* System Developer work product compliance to contract or specification requirements.
- **System Developer V&V Perspective** - Continuously *verify* compliance to contract, project charter, and specification work products as well as those developed externally by subcontractors, vendors, and Service Providers.

Referring to Figure 13.3, the User or their System Acquirer technical representative translates the User's Operational Need into a set of SRD requirements that form the basis for developing a system, product, or service.

Every industry employs various types of documents to capture and document Customer or User requirements. From a System Verification perspective, it is critically important throughout the System Development Phase to ensure that the *technical integrity* of the *evolving* and *maturing* System Design Solution is *preserved* and not *compromised* by the introduction of *latent defects*. As one of several conditions of acceptance, the System Acquirer representing the User will attest that the System Developer delivered what was required (by contract) or demanded (by marketplace surveys or testimonials) to meet the User's or consumer marketplace's operational need(s).

Observe that compliance *verification* does not ensure that the deliverable system, product, or service is what the Stakeholder Users and End Users actually need to satisfy their operational needs; that is System Validation. Verification only proves that a system, product, or service produces performance-based outcomes that comply with specification requirements, which *may or may not* reflect the User actual operational need(s). Those specification requirements are subject to *how well* the specification developer *specified* and *bounded* the system, product, or service as a Solution Space or one of several Solution Spaces intended to satisfy a User's Problem Space.

If we expand the System V&V Overview shown in Figure 13.3 to illustrate how V&V are applied to SE&D, Figure 13.5 emerges. Observe that the expansion:

- Highlights the importance of User/End User collaboration and feedback, Decision Support, SE&D, and a series of Technical Reviews.
- Introduces the concepts of Developmental Test and Evaluation (DT&E) and Operational Test and Evaluation (OT&E) discussed later in this section.

During the System Development Phase, the *evolving* and *maturing* System Design Solution is prototyped, modeled, simulated, and tested to mitigate risk and collect performance data to validate design decisions. These activities are referred to as DT&E of the Developmental Configuration (Chapters 12 and 16). The intent is to *incrementally verify* that the System Design Solution:

- is technically *compliant* with specification and design requirements.
- is *traceable* to the User's source or originating requirements.
- has acceptable risk.

On completion of System Verification, the system, product, or service may undergo System Validation assessments by the User. System Validation may be required by contract or be performed via test marketing in the case of consumer products or User field exercises of contract-based systems (Figure 5.1).

13.4.2 System Verification Objectives

Since SE&D requires transformation of *abstract* User operational needs into specifications that lead to a System Design Solution, component selection, and SITE, humans require a mechanism to express and communicate how the SYSTEM or PRODUCT will be developed to ensure consistency of thought.

Work products document technical decision artifacts—specifications, drawings, and test procedures. Until a working prototype and actual deliverable system is physically available to instrument and measure, documentation is the only stable form of knowledge expression that *factually* represents a consensus of agreement within the System Developer's Enterprise or with Users. Therefore, verification is *dependent* on these work products as a frame of reference to (1)

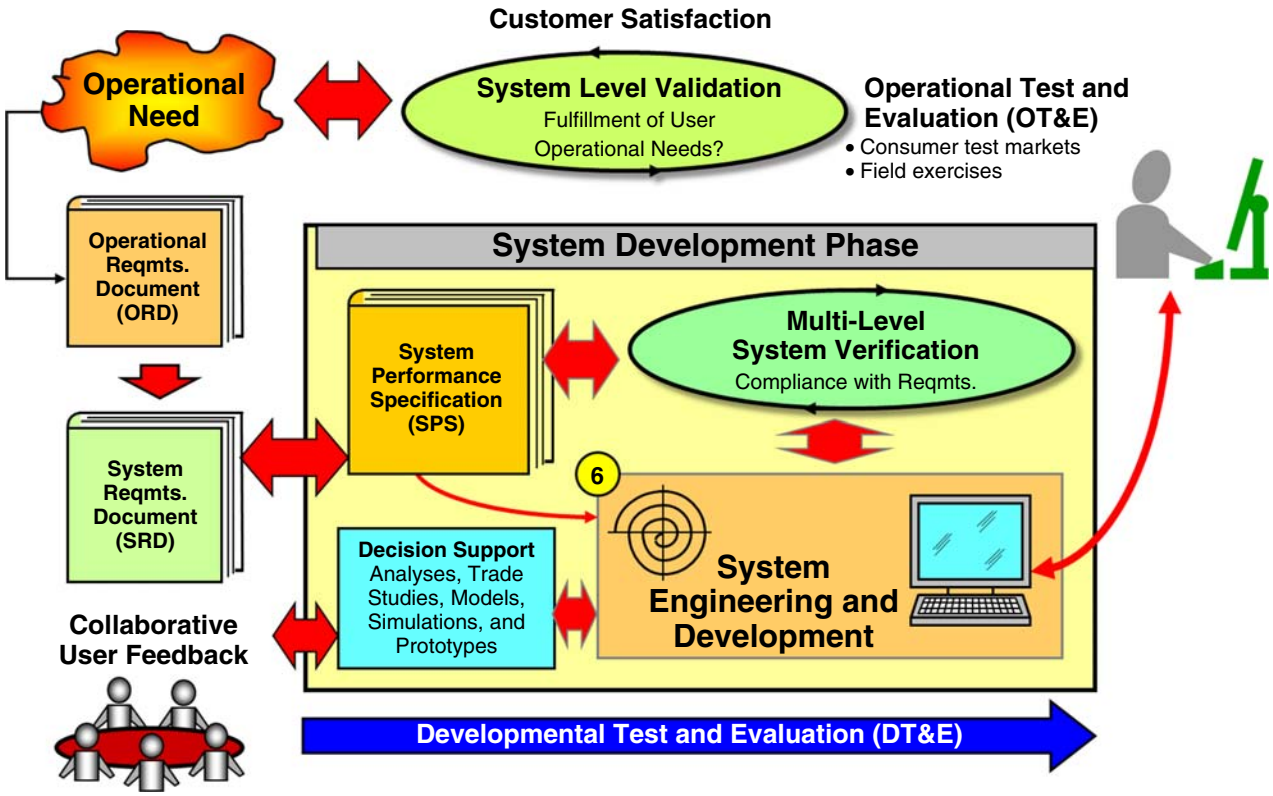


Figure 13.5 System V&V: Programmatic Perspective

ensure specification compliance and (2) eliminate what is or is not a latent defect.

One of the erroneous myths promulgated by the *ad hoc* SDBTF-DPM Paradigm (Chapter 2) Enterprises and engineers is the notion that SE focuses on producing documents. *Absolutely not true*! SE focuses on developing a compliant System Design Solution; documentation is simply a means to document decision-making process artifacts. Until the System Design Solution manifests itself in the form of physical components, the only objective evidence available for evaluating and verifying decisions and designs are the document artifacts. Recognize and appreciate the difference between creating documents versus documenting decision artifacts!

Given an overview of System Verification, let's briefly explore some types of *verification*.

13.4.3 Types of System Verification

The development of a System Verification strategy requires more than simply performing the activity. Two types of verification are used to answer specific questions:

- **Design Verification** answers the question: If the components of a system, product, or service are ideal, does the architectural arrangement and interconnectivity of those components representing its design configurations produce the required behavioral responses and performance based outcomes?
- **Product Verification** answers the question: If a system or product's design configuration has been verified to produce the required behavioral responses and outcomes, does the workmanship, material composition, and integrity of a specific instance of the system or product reliably and predictably produce the same results?

13.4.3.1 Developmental Configuration Design Verification Design Verification includes prototyping, such as breadboards and brassboards, and subjects a new System or Product to a set of functional and environmental operating conditions to demonstrate its compliance and robustness to accommodate failure and continue to operate.

We know that verifying every system, product, or service for compliance to its specification requirements is very expensive and time-consuming. In fact, it is impractical and doesn't make sense to *reverify* a design that has already been proven to be compliant. So, *how do we solve the problem*?

13.4.3.2 Developmental Configuration or Production Article Verification *Systems Thinking* (Chapter 1) helps us recognize that once we have *verified* a new system or product's design, the design never changes from one deliverable instance to another until a decision is made to change the design and produce a new model. So, if the design has been verified—Design Verification—the only remaining variable to be verified is *detection* and *correction* of workmanship and material defects. That brings us to the second type of System Verification, *Product Verification*:

- *Product Verification* verifies that each deliverable system or product is fully operational based on performance of a limited number of tests in an ambient—room temperature—environment.

As an illustration of Product Verification, consider the following example.

Example 13.1 **Product Verification: Portable Electronic Device**

A portable electronic device is designed to *automatically* perform self-test diagnostics when it is powered up. If the self-test indicates that all capabilities are fully operational, it ends with a desktop display. If not, the display indicates a problem via a light or error code. During production, each unit is typically powered up to ensure that it is fully operational and then packaged for delivery to stores.

13.4.3.3 Production Design Verification Once a system or product has been fielded for a period of time, the User may decide to contract for production in small to mass quantities. Remember that just because the Developmental Configuration complies with its SPS requirements does not necessarily mean that it can be produced *cost-effectively*. That requires another form of System Development project to make it producible and cost-effective. Typically, this requires another contract to modify the Developmental Configuration baselined at System acceptance delivery.

When the Production System Design is completed, Production Design Verification is conducted against the Production SPS requirements. Once the Production Design has been *verified* and a decision is made to mass produce the system or product, production units are verified via a subset of Production Design Verification referred to as Product Verification in the same manner as the original Developmental Configuration systems.

Given these insights into Developmental Design Verification, Product Verification, and Production Design Verification, let's investigate how V&V are performed.

13.4.4 Verification Methods

Principle 13.10 **Verification Methods Principle**

Verification methods consist of Inspection, Examination, Analysis, Demonstration, and Test or combinations of these; Similarity is permitted by some Enterprises.

The process of verifying multi-level SE design in compliance to the SPS or Entity Development Specification (EDS) requires standard verification methods that are well defined and understood. Verification includes five (5)

commonly recognized methods: (1) Inspection, (2) Examination, (3) Analysis, (4) Demonstration, and (5) Test. A sixth method, Verification by Validation of Records, is permitted as a verification method in some business domains.

Before we describe each of the verification methods, be aware that each verification method has a cost in terms of labor and time. Therefore, you should use the *least* number of verification methods to prove compliance.

Let's begin our discussion with *Verification by Inspection*.

13.4.4.1 Verification by Inspection NASA (2007, p. 86) defines *Verification by Inspection* as:

> "The visual examination of a realized end product. Inspection is generally used to verify physical design features or specific manufacturer identification. For example, if there is a requirement that the safety arming pin has a red flag with the words "Remove Before Flight" stenciled on the flag in black letters, a visual inspection of the arming pin flag can be used to determine if this requirement was met."

To illustrate Verification by Inspection, consider the following example.

Example 13.2 **Verification by Inspection: Metal Plate Hole Pattern Layout**

A drawing specifies that a metal plate consists of 6 holes 1.0'' ± 0.1'' in diameter each laid out in a dimensional pattern shown on the drawing. Verification by Inspection would consist of verifying that:

- The hole pattern and centers comply with the drawing. ✓
- A quantity of 6 holes are drilled in the plate as specified by a drawing. ✓
- Hole #1 is 1.0'' ± 0.1'' in diameter. ✓
- Hole #2 is 1.0'' ± 0.1'' in diameter. ✓
- And so forth.

13.4.4.2 Verification by Examination Verification by Examination consists of a detailed visual examination of a component or specimen to determine the integrity of its workmanship, material composition, or defects. Verification by Examination has a number of contexts that are determined by the operating environment. Consider the following examples:

Example 13.3 **Verification by Examination: Human Experience and Judgment Factors**

Humans perform microscopic and visual analysis due to their superior ability to detect and make experience-based judgment decisions about specific types of abnormalities. When compared to machines, for example, a human's ability to detect fractures or cracks in jet engine compressor blades or analysis of mammography for breast cancer is superior to machines. Humans simply exploit what machines do best to enable us to exercise capabilities we have. This is exemplified in Figure 24.14 and Principle 24.13.

Example 13.4 **Verification by Examination: Harsh Environment Conditions**

Robots equipped with an array of cameras and sensors are sent to Mars, underground sewers, and nuclear reactor buildings to report conditions in a harsh environment that may not be conducive to human health or survival.

Example 13.5 **Verification by Examination: Human Health Conditions**

Non-invasive camera devices with RF transmitters or surgical tools inserted into the human body to enable an operator to view and assess health conditions along a pathway such as an intestinal or cardiovascular tract.

13.4.4.3 Verification by Demonstration Verification by Demonstration is typically performed without instrumentation. The SYSTEM or PRODUCT is exercised in various facets of operation for witnesses to observe and document the results. Demonstration is often used in operational scenarios involving reliability, maintainability, human engineering, and final on-site acceptance following formal verification.

The NASA *SE Handbook* (2007, p. 86), for example, describes Verification by Demonstration as follows:

> "Showing that the use of an end product achieves the individual specified requirement. It is generally a basic confirmation of performance capability, differentiated from testing by the lack of detailed data gathering. Demonstrations can involve the use of physical models or mockups; for example, a requirement that all controls shall be reachable by the pilot could be verified by having a pilot perform flight-related tasks in a cockpit mockup or simulator. A demonstration could also be the actual operation of the end product by highly qualified personnel, such as test pilots, who perform a one-time event that demonstrates a capability to operate at extreme limits of system performance, an operation not normally expected from a representative operational pilot."

To illustrate Verification by Demonstration, consider the following example:

Example 13.6 **Verification by Demonstration: Accessibility of Organizational Forms**

A specification requirement states that Engineering forms on a Web site are required to be accessible by personnel within a maximum of 3 mouse clicks. The System Developer develops a rapid prototype of the Web site. They collaborate with the User to elicit feedback about the design. Prior via a demonstration to verify accomplishment of the requirement.

13.4.4.4 Verification by Test *Verification by Test* is used as a verification method to collect instrumented data measurements and results that the performance of a SYSTEM or PRODUCT complies with its specification or design requirements. Testing requires creating a prescribed set of operating environment conditions and conducting the test in accordance with approved and baselined test procedures.

The NASA *SE Handbook* (2007, p. 86), for example, describes Verification by Test as follows:

"The use of an end product to obtain detailed data needed to verify performance, or provide sufficient information to verify performance through further analysis. Testing can be conducted on final end products, breadboards, brass boards, or prototypes. Testing produces data at discrete points for each specified requirement under controlled conditions and is the most resource-intensive verification technique."

Testing can be very expensive. When Inspection, Analysis, and Demonstration-*individually or collectively* - are *insufficient* in producing objective evidence required to prove compliance (Figure 22.1), testing may be required. Remember that the objective is to demonstrate compliance with the least number of verification methods including tests. To illustrate this point, observe how the following Verification by Test example provides the basis for our next topic Verification by Analysis.

Example 13.7 **Verification by Test: Electric Motor Speed and Torque**

The specification for a small electric motor includes requirements to achieve a specified speed and torque for a given set of loading and OPERATING ENVIRONMENT conditions. The System Developer places the Unit Under Test (UUT) in a test fixture, instruments the UUT, and conducts a *single* test to collect both speed and torque measurements. During the test, the motor's speed and torque data are documented. Results of the test are compared to the specified motor speed and torque performance requirements as verification of compliance.

13.4.4.5 Verification by Analysis Verification of some specification requirements related to performance parameters such as efficiency, effectiveness, time dependence, and characteristics may not be directly measurable by test. They can be, however, derived from test measurement data. If not, *why are they requirements*?

The NASA SE Handbook (2007, p. 86), for example, defines Verification by Analysis as:

- "The use of mathematical modeling and analytical techniques to predict the suitability of a design to stakeholder expectations based on calculated data or data derived from lower system structure end product verifications. Analysis is generally used when a physical working prototype; engineering model; or Fabrication, Assembled, Integrated, and Tested (FAIT) SYSTEM or PRODUCT is not available. Analysis includes the use of modeling and simulation as analytical tools. A model (in this context) is a mathematical representation of a SYSTEM or PRODUCT. A simulation is the manipulation of a model."

Verification by Analysis, which is typically documented in a formal technical report, is placed under formal configuration control and may or may not be a deliverable. To illustrate an example of Verification by Analysis, let's continue with the electric motor from Example 13.7.

Example 13.8 **Verification by Analysis: Electric Motor Efficiency**

The small electric motor in Example 13.7 is required to have an *efficiency* equal to or greater than 80% for a specific set of loading and environmental conditions. Based on advanced verification planning, the System Developer extracts the recorded motor speed and torque test data and computes the engine efficiency. Results are documented in an analysis presented as objective evidence of compliance to the efficiency requirement.

13.4.4.6 Verification by Validation of Records As stipulated earlier in Principle 16.7, new SYSTEM Design should be a *last resort* when all other options—existing in-house or legacy designs and vendor catalog items—have been exhausted in terms of meeting the specification requirements. For example, consider legacy designs. If (1) the original design was *verified* as being compliant with specification capability requirements and environmental performance requirements that were *equal to or exceeded* the current specification requirements and (2) no modifications have been made or are planned to be made to the original design, then *Verification by Validation of Records* may be an option.

Verification by Validation of Records simply requires the presentation of objective evidence of the original design's specification requirements, verification data, and compliance as a *condition* for verification of the new System or Product.

Example 13.9 **Verification by Validation of Records: Processor Board**

A System Developer has a contract to develop a Sensor System for an aircraft. Due to the complexity of the Sensor System, the conceptual architecture calls for a Processor Board that will serve as a controller for the sensor. An Engineering analysis is conducted and has determined that the Processor Board used on a previous system: (1) fully complies with the new SPS requirements and (2) has already been verified. A decision is made to *reuse* the Processor Board without modifications

in the new design, retrieve the verification records for the original design, and present the records as objective evidence for proof of compliance to the new specification.

Author's Note 13.5

Be aware that the Verification by Validation of Records may be easier said than done, especially if the Enterprise has very poor discipline in maintaining records from the past. When a decision such as this is made, you should always initiate a search immediately for copies of the records. Do yourself and your Enterprise a favor. Avoid the professional embarrassment of getting to formal verification and then being unable to find the records! Then, incur the expense of reverifying a legacy design.

13.4.4.7 Verification Method Selection

Principle 13.11

Verification Method Selection Principle

Select the *least* number of verification methods required to prove compliance to a single requirement.

Every verification method has an impact in terms of (1) the *quantity* of requirements verified, (2) labor cost and schedule duration, and (3) risk. In general, the cost to perform these verification methods varies significantly. Relative costs by verification method are:

1. Inspection (least cost)
2. Validation of Records (low cost)
3. Examination (low cost)
4. Analysis (low to moderate cost)
5. Demonstration (moderate cost)
6. Test (moderate to high cost)

When proposing a new system, strive to identify the method with the lowest cost, schedule, and risk that will provide compelling, objective evidence that a requirement has been satisfactorily accomplished. If you have a requirement that you can verify by Analysis (low to moderate cost), *why would you want to commit to Test* (moderate to high cost) as the *verification method and drive up your proposal price and potentially risk losing the contract?*

13.4.5 Verification Compliance Strategies

People often believe you have to verify each specification or design requirement … *and you do*! However, there are two different concepts here: (1) the collection of verification data to prove compliance and (2) the process of proving compliance. Let's examine this point further by starting at the end and working backwards.

13.4.5.1 Specification Compliance Verification Strategy

A formal review such as a DoD System Verification Review (SVR) (Chapter 18) requires that you present Quality Records (QRs)—measurements, data logs, etc. These items serve as objective evidence of verification results that have been authenticated by witnesses and compliance to specification and design requirements. The SVR reviews *authenticate* data collected in accordance with the Terms and Conditions (Ts&Cs) of the contract, specification requirements, prescribed test procedures approved by the User or project, and legal, ethical, and moral principles.

So, *how did you collect the data*? This brings us to the first part, Verification Data Collection Strategy.

13.4.5.2 Verification Data Collection Strategy

Principle 13.12

Compliance Testing Principle

Leverage each test to collect as much data that will enable you to verify compliance to the largest number of requirements.

We could perform separate verification tasks for each specification requirement. However, that can become very expensive and time-consuming. Let's step back and apply *systems thinking*.

We know that a specification specifies and bounds operational requirements—integrated sets of capability requirements—and discrete capability requirements. When the SYSTEM or ENTITY is *operational*, combinations of capabilities subject to *Allowable* and *Prohibited Actions* (Figure 7.13) are physically configured by the System Architecture to produce performance-based outcomes. In other words, the System Architecture configures "chains" of Use Case (UC) capabilities.

Each capability was derived from those UC outcomes via Model-Based Systems Engineering (MBSE), for example, and translated into specification requirements. Therefore, we should be strive to *instrument* SYSTEM or ENTITY capabilities at specific test points to collect data that can be used to provide compliance to several specification requirements. Leverage a single test to verify XX requirements.

Given an in-depth understanding of System Verification Practices, let's shift our focus to System Validation Practices.

13.5 SYSTEM VALIDATION PRACTICES

Whereas Verification asks if we built the work product *right*—in compliance with its specified requirements—Validation answers the Acquirer's question: *Did we acquire the right system to meet the User's validated operational needs*? Validation employs a number of methods for System

Developers to collaborate with the Users. Let's begin with an overview of *Validation*.

Principle 13.13

System Validation Principle

System validation assesses a Stakeholder's satisfaction with a work product's performance-based outcome(s) to their documented operational needs and expectations.

13.5.1 System Validation Overview

Validation, like verification, occurs throughout the System Development Phase beginning with contract award or project charter start and continuing until system acceptance and delivery. Validation, however, is not limited to external Stakeholders; it also applies to internal Stakeholders within the System Developer's Enterprise.

As levels of internal Stakeholders decompose or partition the SYSTEM into lower levels of abstraction—SYSTEM into PRODUCTS, PRODUCTS into SUBSYSTEMS, and so forth—each entity within each level of abstraction represents a Stakeholder Problem Space that requires one or more lower-level Solution Spaces (Figure 4.7).

Solution Space teams such as internal developer teams, subcontractors, or vendors develop SYSTEMS or PRODUCTS to satisfy or contribute to satisfying the higher-level Problem Space need. As in the case of external Stakeholders, internal Stakeholders must also ask the question: *will the Solution Space entities we have acquired fulfill our operational needs*?

Referring to Figure 13.3, the User's question at the SYSTEM Level following the completion of System Verification becomes: *Did we acquire the right system to fulfill our operational needs*?

Based on the Validation Overview, let's define objectives Validation is intended to accomplish.

13.5.2 System Validation Objectives

Validation, like Verification, applies to the System or any of its lower-level entities and work products. So, validation objectives are written for general application for all contexts. Validation objectives include:

- Evaluating and assessing the multi-level System Design Solution or one of its Entities during System Development to determine if it will satisfy its User's operational needs. Remember that a User could be the external User or System Developer SE, Engineer, designer, or tester.
- Conducting field exercises with the *actual* Users that have been trained to operate the deliverable system, product, or service to conduct missions.

13.5.3 Validation Methods

Validation employs a number of methods such as User interviews, prototyping, demonstration, qualification tests, test markets, and field trials. In general, validation consists of any collaborative method that enables the Acquirer (contextual role) of a system, product, or service to provide *candid*, *constructive* feedback that an evolving work product will satisfy their documented operational need(s). Two key points are as follows:

1. Observe usage of the term System Acquirer (role) in the earlier text. There are two contexts here:
   a. System Acquisition (role) Context—A System Acquirer external to a System Developer representing the technical interests of the User that acquires a system, product, or service via a contract.
   b. Entity Acquisition (role) Context—An individual or team *within* the System Developer's project that allocates specification requirements to a lower-level team or issues a subcontract to develop an ENTITY—PRODUCT, SUBSYSTEM, and ASSEMBLY—in accordance with the specification.
2. Note the use of the term *documented* operational needs. Humans tend to change their minds about *what they said* or *intended to say*. To avoid any misinterpretation by either party:
   a. Document the mutual understanding of User operational needs prior to contract award.
   b. Document the criteria that constitute system acceptance in the contract.

13.5.4 System Validation: System Developer Context

Validation employs User feedback mechanisms to keep the activities that produce work products *focused* on the key factors *critical* to the Voice of the Customer (VOC) and their satisfaction. Agile Development User Stories and Quality Function Deployment (QFD) (Chapter 5) analyses serve as examples for driving out User operational needs, preferences, and outcome-based performance requirements.

The scope of validation encompasses more than a User function. Remember that the SPS provides a basis to decompose a high-level Problem Space into lower levels of Solution Spaces (Figure 4.7), even within the System Developer's program Enterprise. In this context, the User (role) is the higher-level team assigned to the PRODUCT, SUBSYSTEM, ASSEMBLY, and SUBASSEMBLY Problem Space. As the lower-level Solution Space "designs" evolve, higher-level Users must *validate* that the *evolving* and *maturing* item will satisfy their needs. Consider the following example.

Mini-Case Study 13.1

Internal Validation with a Project

A Product Development Team (PDT), which is assigned accountability for developing a Control Station Performance Specification, has a Problem Space to resolve. The IPT analyzes the Problem Space and decomposes it into several potential Solution Spaces (Figure 4.7). Requirements

for Computer Software Configuration Item (CSCI) #1 are allocated and documented in the CSCI #1's Software Requirements Specification (SRS) and assigned to the CSCI #1 PDT.

As CSCI #1's design is formulated, its PDT employs *iterative* rapid prototyping methods such as Spiral Development (Figure 15.4) to generate sample operator displays and transitions for evaluation. Users are invited via Acquirer contract protocol to participate in a demonstration to review and provide feedback. The outcome of the evaluation is to validate that the evolving display designs and transitions satisfy the User's operational needs. Results of the demonstration are documented as a QR and used to support System Developer decision-making.

Example 13.10

An SE is tasked to perform a Stakeholder Needs Analysis. The SE performs *validation* by talking with the User or task source to:

- Fully understand the Critical Operational or Technical Issue (COI/CTI) to be resolved.
- Scope areas for investigation, objectives, and constraints.
- Understand how the results are to be documented to ensure the deliverable work product will meet the task source's needs.

As a final reminder –*System validation* continues after the *work product* such as a PRODUCT and SUBSYSTEM is delivered until the User's intended operational need is satisfied. Beyond that point, "gap" analysis becomes the focal point for assessing gaps in the system, product, or service's capabilities.

13.5.5 System Validation: User Context

The ultimate proof of a System Validation resides with the User community. How this is accomplished is typically unique to the business domain and Users. For example:

- Commercial product developers *test market* ideas and new products during development and incorporate their feedback. After product release to the public, surveys, sales, interviews, and other methods are used to collect customer satisfaction data.
- Military and government System Acquirers and Users employ the services of an Independent Test Agency (ITA). The ITA validates the system based on User PERSONNEL trained by the System Developer. During validation, the system or product is subjected to actual field OPERATING ENVIRONMENT conditions and scenarios. This is referred to as OT&E and will be discussed later in this chapter.

13.6 APPLYING V&V TO THE SYSTEM DEVELOPMENT WORKFLOW PROCESSES

The preceding discussions provide a general overview of the V&V concepts and technical methods. Given this understanding, let's shift our attention to applying V&V strategies to Chapter 12 Introduction to SYSTEM DEVELOPMENT STRATEGIES processes.

To fully understand how V&V apply to these processes, let's employ a rigorous example that you can use as the basis for tailoring a simpler model that fits your project's needs. Figure 13.6 provides a reference for our discussions. Bubble identifiers are provided to serve as navigational aids for our discussion.

13.6.1 System Specification Process: V&V Strategy

The primary objectives of the System Specification V&V Strategy are to:

- Understand the Problem Space—Critical Operational Issue (COI) or Critical Technical Issue (CTI)—an Acquirer or User needs to solve.
- Understand how the Acquirer's SPS specifies and bounds the User's Solution Space that leads to resolution of a mission or existing system COI or CTI.
- Ensure that an Acquirer's SPS *accurately* and *sufficiently* bounds and specifies the *essential* requirements for the Solution Space to be filled by a system, product, service, or capability upgrade.

During the User's System/Product Life Cycle - System Procurement Phase, Operational Needs identified by the User and System Acquirer are documented in the SPS. This is a critical step. The reason is that by this point the System Acquirer, in collaboration with the User, partitions an Enterprise Problem Space (Figure 4.3) such as a COI or CTI related to current capabilities into one or more Solution Spaces to be filled by systems, products, services, and upgrades. Each Solution Space is then bounded and specified in terms of *what* the system, product, or service is to accomplish and *how well* but not *how* to design it.

If *human errors* in Engineering judgment are made concerning bounding and specifying a Solution Space, they manifest themselves as latent defects in the requirements documented in the SPS. For example, has the Problem Space been accurately identified, or is it a *symptom* of a larger Problem Space (Principle 2.1)? Therefore, the challenge question for the System Acquirer, User, and ultimately the System Developer is: *Have we specified the right Solution Space (system or entity) to satisfy one or more User operational needs (Problem Space)? How do we answer this question*?

SPS requirements are subjected to Requirements Validation against the Operational Need to validate that the *right* solution space description has been *concisely, accurately,* and *completely* bounded by the SPS.

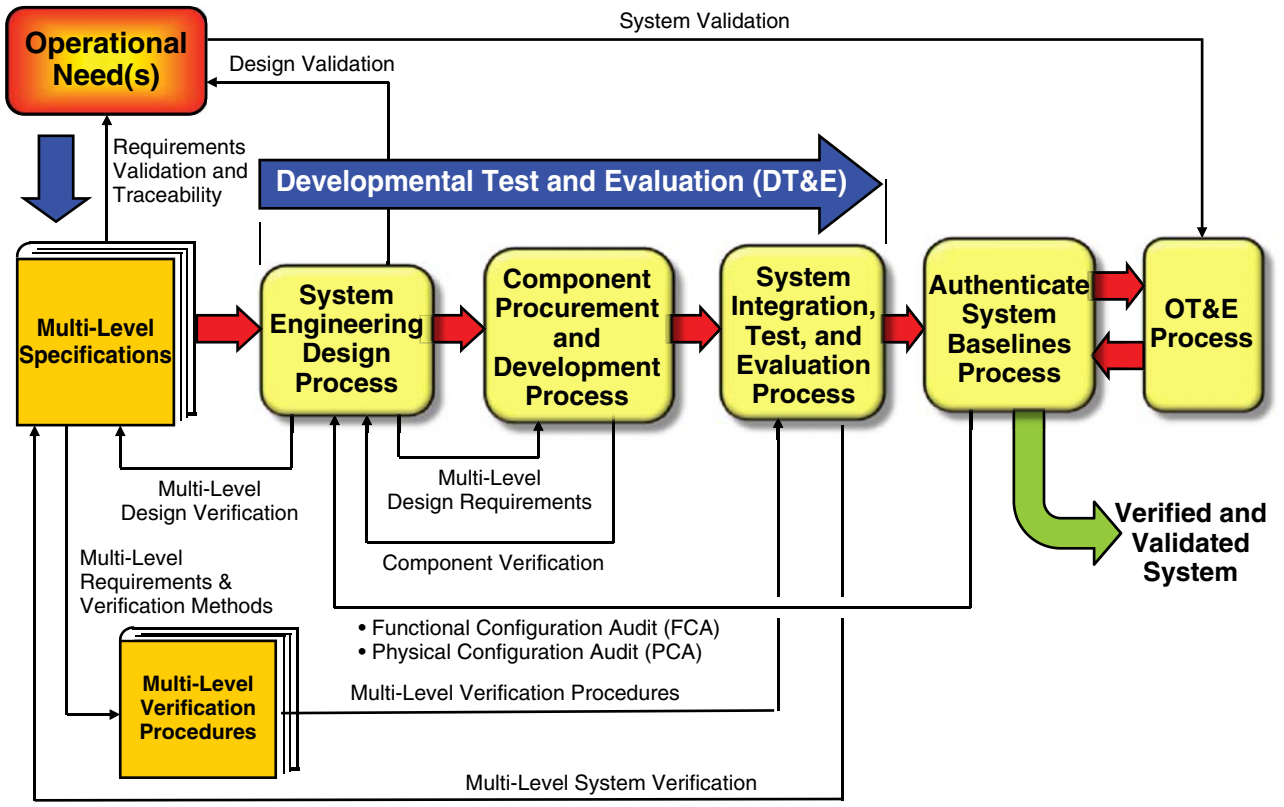


Figure 13.6 SE Design Process V & V Strategy Applied to the System Development Processes Workflow (Figure 12.2)

A Word of Caution 13.1

Please note that any discussions with the Acquirer and User regarding the SPS requirements validation require tactful professionalism and diplomacy. In effect, you are validating that the Acquirer performed their job correctly.

On the one hand, they may be grateful you may have identified any potential deficiencies in their assessment and brought it to their attention. Conversely, you may offend them!

Approach any discussions in a tactful, well-conceived, professional manner. Develop strong rapport with the System Acquirer/User in advance of a procurement action to build *confidence* and *acceptance* of your recommendations.

13.6.2 SE Design Process: V&V Strategy

The primary objectives of the SE Design Process V&V Strategy are to:

1. Verify that the multi-level System Design Solution is evolving, maturing, traceable to, and compliant with its specification or task requirements.
2. Validate that the System Design Solution will satisfy the Stakeholder's operational needs.
3. Verify that design requirements—drawings and parts lists—are accurate, compliant, and mature for procurement of physical components.

Let's investigate how the System Design Process V&V Strategy is applied to the System Development Workflow Processes (Figure 12.2).

When the SPS requirements have been validated via analysis and interviews with the User, the SPS serves as *source* or *originating* requirements inputs to the System Design Process. Throughout the System Design Process, Design Verification is performed on the evolving System Design Solution by tracing allocated requirements back to the SPS and prototyping design areas for risk mitigation and COI/CTI resolution. Design Validation activities are performed to confirm that the User and System Acquirer, as the User's technical representative, agree that the evolving System Design Solution will satisfy their operational needs.

Author's Note 13.6

Please note that despite the location of the System Design Process within the workflow sequence, Design Verification, and Design Validation activities—that is, Design V&V—are not considered complete until the System has been verified,

validated, and legally accepted by the User via the System Acquirer in accordance with the contract, project charter, or task requirements.

Design V&V activities, which are performed throughout the SE Design Process, are:

- Used to elicit System Acquirer and User validation feedback, acceptance, and approval, as appropriate based on prototypes and mock-ups and technical demonstrations.
- Accomplished via major technical reviews (e.g., SDR, SSR, PDR, and CDR) addressed later in Chapter 18.

The SE Design V&V Strategy culminates with a SYSTEM Level Critical Design Review (CDR) to assess the status, maturity, and risk of committing project resources to proceed to the Component Procurement and Development Process.

13.6.3 Component Procurement and Development Process: V&V Strategy

The primary objective of the Component Procurement and Development V&V Strategy is to verify that the *internally* developed or *externally* procured components are fully compliant with their design requirements—drawings, parts lists, and wiring lists. Let's investigate how the Component Procurement and Development V&V Strategy (Figure 13.6) is implemented.

During Component Procurement and Development, design requirements from the SE Design Process serve as the basis for procuring, fabricating, coding, and assembling system components. Each internally developed or externally procured PART-Level hardware or software component undergoes Component Verification for compliance to its Design Requirements such as drawings, parts lists, schematics, wiring lists, and software design.

Component procurement and development *verification* activities occur in several ways:

1. Vendor or internal development verification following component FAIT. Vendor verification of components may be accomplished by System Developer on-site witnesses or witnessed by vendor QA and authenticated via a Certificate of Compliance (C of C) delivered with the component to the System Developer.
2. System Developer verification via receiving inspection of external vendor products such as components and raw materials based on procurement "Fitness-for-Use" criteria (Figure 4.1).
3. System Developer verification of internally produced or modified components.

Externally procured components and materials undergo receiving inspection verification that the components comply with their procurement or product specifications. The verification may be accomplished by:

- Random selections of component samples for analysis and test.
- Inspection of Certificates of Compliance (CofCs) issued by the vendor's Quality Assurance (QA) organization.
- Sampled testing or 100% testing of each component.

In all cases, component material and performance discrepancies such as deficiencies, incorrect mass properties, and substandard work quality – workmanship - are recorded as Discrepancy Reports (DRs) and dispositioned for subsequent corrective action, as appropriate.

As internally developed components are verified and externally procured components pass receiving inspection, they are stored until required for integration into higher-level entities such as SUBASSEMBLIES, ASSEMBLIES, SUBSYSTEMS, PRODUCTS, or the SYSTEM. When ready, PARTS are integrated into higher-level entities such as SUBASSEMBLIES for entry into the SITE verification.

13.6.4 SITE Process: V&V Strategy

The primary objectives of the SITE V&V Strategy are to verify that components and levels of integrated entities—PRODUCTS, SUBASSEMBLIES, ASSEMBLIES, and SUBSYSTEMS—are:

1. Tested using a test configuration that is documented and under CM Control.
2. Compatible and interoperable.
3. Performance compliant with their specification requirements.
4. Ready for integration at higher levels, as appropriate.

Let's investigate how the SITE V&V Strategy (Figure 13.6) is implemented.

The SITE Strategy (Figure 12.6) provides the basis to verify that each entity at various levels of integration—for example, PART, SUBASSEMBLY, ASSEMBLY, SUBSYSTEM, PRODUCT, and SYSTEM Levels:

- Performs and produces results that comply with its specified performance-based capability requirements prior to, during, and after being subjected to the specified operational and OPERATING ENVIRONMENT conditions.
- Is compatible and interoperable with internal entities within the System's Architecture and with external system entities, if required.

To implement the SITE Process, Verification Methods and Requirements specified in the SPS are used to develop System Verification Procedures. Verification methods such as Inspection, Analysis, Demonstration, Test, and Similarity (conditional) are specified as verification requirements for each SPS requirement. Each System Test Procedure prescribes the test environment configuration, OPERATING ENVIRONMENT (initial and dynamic), data inputs, and expected test results required to produce compliance data to support verification of each SPS or lower-level specification requirement.

During SITE, the System Developer formally verifies the SYSTEM using preapproved test procedures with QA, Software Quality Assurance (SQA), representatives of the System Acquirer, User, and System Developer present as witnesses. Multi-level verification activities compare *actual* test data and results against the System Verification Procedures. SITE culminates with a formal System Verification Test (SVT) to assess compliance of the overall SYSTEM to SPS requirements. If discrepancies are identified between SPS required performance and *actual* performance results, a DR is recorded and submitted for corrective action.

13.6.5 Authenticate System Baselines Process (First Pass): V&V Strategy

The primary objective of the Authenticate System Baselines Process V&V Strategy is to ensure that the System or its components:

- Comply with its SPS or lower-level specification requirements.
- Identically match its assembly drawings, design drawings, cable wiring lists, parts lists, and documentation.

Let's investigate how the Authenticate System Baselines Process V&V Strategy (Figure 13.6) is implemented.

When the SVT is completed, workflow progresses to Authenticate System Baselines Process as the first of two potential passes:

- First Pass—Assesses the System Design Solution Developmental Configuration results immediately following the SVT at the conclusion of the System Development contract, project charter, or task.
- Second Pass—Assesses any "delta" changes to the Developmental Configuration that may have occurred during User System Validation following the First Pass and prior to System Acceptance and delivery, depending on contract requirements.

The process consists of Configuration Management (CM) (Chapter 16) Audits followed by an SVR to assess the results of the CM audits. The SVR reviews the QRs of a Functional Configuration Audit (FCA) and a Physical Configuration Audit (PCA). The FAC and PCA authenticate the System Design Solution documentation baselines—for example, Developmental Configuration—for *accuracy*, *consistency*, and *completeness* with the actual *verified* System:

- The FCA reviews and validates authenticated QRs—for example, performance-based capability (functional) test results—from SPS and EDS requirements compliance verification. During the FCA, *discrepancies* between SPS or lower-level specification requirements and test results—for example, non-compliances—are noted in DRs and referenced in FCA Conference Minutes for corrective action and closure of the DR.
- The PCA reviews and validates QRs that entities, for example, PARTS, SUBASSEMBLIES, ASSEMBLIES, SUBSYSTEMS, PRODUCT, and the SYSTEM, physically comply with *design requirements*—for example, dimensional drawings, parts lists, schematics, and wiring lists. Since the PCA may require "after the fact" disassembly of the verified System to perform the validation, it may be cost-effective to perform incremental PCAs at each Integration Point (IP). During the PCA, discrepancies between *design requirements* such as noncompliances are noted in DRs and referenced in PCA Conference Minutes for corrective action and closure of the DR.

Entry criteria for the SVR require successful completion of the FCA and PCA for scheduling the event. System Acquirer contracts may or may not require the conduct of an FCA, PCA, or both. However, it is advisable that Enterprise command media require at least an FCA and possibly a PCA for legal record purposes whether required by contract or not. In any case, the SVR marks the completion of Developmental Test and Evaluation (DT&E).

At this juncture, several options may be available to the System Acquirer and User and established by the contract:

- **Option #1**—The System Acquirer may perform final acceptance of the SYSTEM or PRODUCT at a designated facility such as the System Developer or designated field site.
- **Option #2**—The System Developer's contract may require delivery of a SYSTEM or PRODUCT to a User's designated site for installation, integration, and checkout prior to final System Acquirer and User acceptance.
- **Option #3**—The System Acquirer may require delivery of the SYSTEM or PRODUCT to a User's designated field site for OT&E by an ITA.

Option 3 requires that the System Acquirer and User establish a System OT&E Validation Strategy.

13.6.5.1 OT&E: Validation Strategy

OT&E Principle

Principle 13.14 OT&E enables Users to assess *how well* a system, product, or service satisfies their operational needs based on independent field trials with trained User PERSONNEL operating the system in a realistic OPERATING ENVIRONMENT.

The primary objectives of the OT&E Validation Strategy are to enable the User to:

1. Conduct field trials that employ the system, product, or service to perform representative missions in a realistic OPERATING ENVIRONMENT; component integrity issues; and manufacturing process and workmanship issues.
2. Evaluate how well the system, product, or service:
   a. Satisfies their operational needs
   b. Resolves COIs/CTIs from the User's perspective
   c. Reveals any latent defects such as design errors, flaws, or deficiencies related to OM&S

OT&E activities are typically conducted on large, complex systems such as aircraft and military acquirer activity systems. The theme of OT&E is: *Did we acquire the right system or product to satisfy our operational need(s)*? OT&E consists of subjecting the *test articles* to actual field environmental conditions with operators from the User's Enterprise. An ITA designated by the Acquirer or User typically conducts this testing. To ensure independence and *avoid* conflicts of interest, the contract *precludes* the System Developer from *direct participation* in OT&E; the System Developer may, however, provide maintenance support, if required.

Since OT&E is dependent on how well the Users perform with the Developmental Configuration SYSTEM or PRODUCT, the System Developer typically trains the User's PERSONNEL to safely operate and maintain the system, product, or service. During the OT&E, the ITA oversees User's personnel mission operations and SYSTEM usage based on operational Use Cases (UCs) and scenarios under actual field OPERATING ENVIRONMENT conditions. UCs and scenarios are structured to evaluate system operational *utility, suitability*, *availability*, and *effectiveness*.

ITA PERSONNEL monitor and instrument the SYSTEM to:

- Assess how well the system, product, or service resolves the Stakeholder User and End User COI(s) or CTI(s) that motivated the need for the system, product, service, or upgrade.
- Observe the Human–System interactions and responses, efficiency, effectiveness, and so forth.

Let's investigate how the System Validation Strategy (Figure 13.6) is implemented.

Validation Independence Principle

Principle 13.15 To ensure independence and avoid a conflict of interest, OT&E is typically performed by an ITA with User PERSONNEL that have been trained to operate and maintain the SYSTEM under actual field OPERATING ENVIRONMENT conditions based on scripted scenarios.

System Validation activities (Figure 13.6) referred to as OT&E assess *how well* the fielded system performs missions in its prescribed OPERATING ENVIRONMENT as originally envisioned by the User.

During the OT&E, the System Developer is normally precluded from being present and participating in the activity. Generally, the System Developer is kept informed by the User or System Acquirer's Contracting Officer (ACO) about the evolving results of OT&E. Qualification tests should be conducted by the USER personnel under field conditions to assess not only the SYSTEM (EQUIPMENT Element performance) but also the overall Human Systems Integration (HSI) (PERSONNEL Element) effectiveness. This includes evaluation of MOEs, MOSs, COIs, and CTIs (Chapter 5).

If OT&E determines that a documented *deficiency* is unspecified in the original contract's SPS, this is a critical issue for the System Acquirer and User. For example, *did the User or Acquirer overlook a specific capability as an operational need and failed to document it via requirements in their System Requirements Document (SRD)*?

This point reinforces the need to perform a credible Requirements Validation activity (Figure 13.6) prior to or immediately after Contract Award to *avoid* surprises during System Acceptance. If the deficiency is not within the scope of the contract, the System Acquirer may be confronted with modifying the contract and funding additional design implementation and efforts to incorporate changes to correct the deficiency. Any latent *defects* discovered during System Validation are recorded as Problem Reports (PRs) and submitted to the appropriate decision authority for disposition and corrective action, if required.

Verified System Modifications Principle

Principle 13.16 Any uncoordinated, unapproved, and unverified modifications to a *verified* Developmental Configuration of a Physical System without prior authorization during OT&E effectively *invalidate* the Developmental Configuration SVR results.

If deficiencies are discovered during OT&E, corrective actions that require rework of the physical SYSTEM may

be required to be performed by the System Developer, either under their existing contract or a modified contract. If this occurs, the SYSTEM may be returned to the System Developer's facility to undergo SITE to achieve a new version verified system.

The reality is minor modifications of the fielded system, product, or service will probably have to be made during OT&E. When this occurs, the ITA, System Acquirer, and System Developer must have "open" lines of communication and decision-making channels. Since any modification typically has legal ramifications concerning ownership, accountability, roles, and authorities, contact your Contracts organization for guidance.

Legally, the challenge question is has the *validated* System been *altered* or *modified* from its SVR authenticated baseline (First Pass) by the ITA, User, or System Developer? If this occurs, the physical SYSTEM may no longer be in *compliance* with its verified system documentation. This leads us back to the Authenticate System Baselines Process—Second Pass.

13.6.6 Authenticate System Baselines V&V Strategy (Second Pass): V&V Strategy

The primary objective of the Authenticate System Baselines V&V Strategy—Second Pass is to verify that any changes to the system, product, or service baselines have been updated and authenticated. Let's investigate how the Authenticate System Baselines V&V Strategy—Second Pass (Figure 13.6) is implemented.

During the Authenticate System Baselines Process—Second Pass, V&V activities assess the *validated* SYSTEM against its System Design Solution documentation. If modifications have been made during OT&E, this may require conducting another PCA. The assumption here is that any changes were verified prior to installation and checkout on the fielded SYSTEM during OT&E. On successful completion of the PCA, a follow-up SVR should be considered to:

- Resolve any outstanding Post-OT&E, FCA/PCA issues.
- Recertify the results of the Post-OT&E, FCA and PCA, if required.
- Assess SYSTEM readiness for final acceptance.

On successful completion of the Authenticate System Baselines Process—Second Pass, the Verified and Validated System should be ready for formal System Acceptance by the System Acquirer, as the User's representative, and subsequent delivery to the User.

13.6.7 System Acceptance

Final acceptance of the system, product, or service by the System Acquirer representing the User is accomplished in accordance with contract, project charter, or task requirements.

A Word of Caution 13.2

Refurbishment and Deliverables If the validated system is to be delivered following an OT&E activity, chances are it has blemishes such as scratches from operating in a field environment. The question is *what level of refurbishment, such as spot painting, is permitted by the contract*? Some Users may accept First Article systems assuming that they have been refurbished to meet specific standards. Always consult your contract, project charter, or task for guidance.

13.7 INDEPENDENT VERIFICATION & VALIDATION (IV&V)

Due to technical issues such as risk, interoperability, safety, and health related to large, complex, expensive systems, government organizations such as the US DoD, DoE, and NASA may issue IV&V contracts to assess the work of System Developers during the System Development Phase. The IV&V contractor is tasked to provide an independent assessment to the System Acquirer that the provisions of the System Development contract are being implemented properly.

13.7.1 Need for Independence

IV&V can be an important supporter to the System Acquirer in preventing these problems. Potential hardware and software latent *defects* such as design flaws, deficiencies, or human errors are sometimes missed due to limited personnel availability and skills, incorrect requirements, or changes in hardware or software platforms. Critical flaws can result in cost overruns or even catastrophic mission failure that poses safety risks to the general public and the NATURAL ENVIRONMENT.

13.7.2 Degree of Independence

A common question is: *How "independent" must an IV&V Enterprise be*? ISO/IEC/IEEE 24765 (2010) describes an IV&V organization as "technically, managerially, and financially independent of the development organization" (SEVOCAB, 2014, p. 149 - Copyright 2012, IEEE. Used by permission).

13.7.3 Benefits of IV&V

Users and System Developers often ask *why should they go to the expense of performing IV&V, either by contract or by internal assessments? What's the Return on Investment (ROI)*? There are several reasons; some are objective and others subjective. In general, IV&V:

1. Improves system or product safety.
2. Provides increased visibility into the System Development Process.

3. Identifies non-essential requirements and design features.
4. Assesses compliance between specification and performance.
5. Identifies potential risk areas.
6. Reduces the quantity of latent defects such as design flaws, errors, defective materials or components, and workmanship problems.
7. Helps reduces development, operations, and support costs.

When performed competently and essential results reported constructively, IV&V can be of benefit to both the Acquirer and the System Developer. Depending on the role assigned by the Acquirer to the IV&V contractor, adding another contractor into the System Development Process may require the System Developer to plan for and obtain supplemental resources. The challenge for the System Developer may be dealing in an environment whereby the Acquirer believes the IV&V contractor is not "earning their keep" unless they find a lot of *microscopic deficiencies*, even if the work products are more than adequate technically, professionally, and contractually.

Example 13.11 **NASA IV&V**

As an example, software is playing an increasing role in day to day. For each NASA mission or project to execute successfully, it is imperative that the software operates safely and within its designed parameters. Failure of a single unit of mission critical software within a NASA mission can potentially result in loss of life, dollars, and/or data. IV&V serves as a mechanism to underscore the importance of software safety and helps ensure safe and successful NASA missions.

13.7.4 Is IV&V a Help or Hindrance?

People typically view IV&V activities as *unnecessary* tasks that consume critical skills, cost, and schedule resources that could be better spend on additional system or product capabilities. Contrary to this shortsighted mind-set, IV&V activities should result in a higher-quality product and typically reduce costly rework – cost of IV&V effort versus cost of rework. Project Managers (PMs), Technical Directors (TDs), Project Engineers, SEs, and others who are accountable for technical project performance and customer acceptance must live with the consequences of their decisions.

Some people apply IV&V and view it as a pathway leading to success; others view it as an unnecessary hindrance. Project performance tends to correlate with these two perspectives. The bottom line is as follows: invest in correcting defects such as design flaws, errors, discrepancies, and deficiencies up front or pay significantly more to correct problems at higher levels of integration.

Author's Note 13.7 Ensure that *checks* and *balances* are in place to verify that the system development effort will produce systems, products, and services that comply with contract requirements. IV&V activities serve as one option to accomplish this. *Does IV&V guarantee success?* Absolutely not! Like most human activities, the quality of the IV&V effort is *only as good as* the competency of the personnel who perform the work, methods and tools used, and the resources allocated to the activity. Perform due diligence and select qualified and competent IV&V vendors that provide good value.

13.8 CHAPTER SUMMARY

During our discussion of system V&V practices, we defined V&V; its objectives; how, when, and where V&V is accomplished; who is accountable; and methods for conducting V&V activities. In support of this overview, Chapter 18 addresses specific verification activities that support key decision control points. Key points of discussion include:

1. Verification (Principle 13.3) asks the question: *Are we developing the system in compliance with the specified requirements*?
2. Validation (Principle 13.4) asks the question: *Did we acquire the right system to satisfy our operational needs*?

Standards that specify V&V requirements expect a project to 1) define performance-based outcome plans and tasks for what they want to accomplish, 2) perform to the plan, and 3) produce work products and QRs as objective evidence of performance to the plan and accomplishment of each task.

QRs include items such as meeting and technical review minutes, technical audit results, analyses, trade study reports, and modeling and simulation (M&S) results.

V&V are performed throughout every contract, project charter, or task from initiation to completion. The notion that V&V are only performed at the end of a project (Figure 13.4 left side) is a *factually incorrect myth*. As evidence, refer to Figure 13.6 concerning the application of V&V to the System Development Processes Workflow, if implemented properly.

There are three types of System Verification: (1) Developmental Design Verification, (2) Product Verification, and (3) Production Design Verification.

- Developmental Design and Production Design Verification demonstrate technical compliance to specification or design requirements.
- Once a Developmental Design and Production Design have been verified, the only remaining technical compliance issue is the detection of (1) manufacturing process and workmanship issues and (2) material defects in each physical implementation of the design.

The primary verification methods consist of (1) Inspection, (2) Examination, (3) Analysis, (4) Demonstration, (5) Test, and (6) Validation of Records.

Compliance verification of each and every requirement requires presentation of QRs as objective evidence obtained from one or more verification methods. Since every verification method has a cost and time required to perform, select only the least number of methods that prove compliance for the least cost.

The System Acquirer approves which verification methods are deemed acceptable.

V&V are ongoing activities performed relentlessly every day by project personnel—professionals—accountable for *eliminating* or *reducing defects*, such as design flaws, errors, and deficiencies. V&V are not reserved exclusively for scheduled events on someone's schedule.

IV&V is an approach used on large complex projects by the System Acquirer to employ the services of an external Enterprise specializing in V&V to assess the work of a System Developer.

13.9 CHAPTER EXERCISES

13.9.1 Level 1: Chapter Knowledge Exercises

1. What is verification, what is its primary objective, and what constitutes Verification success?
2. When is verification started and when does it end? At what levels of abstraction are V&V performed?
3. Who is accountable for performing verification?
4. Name six verification methods, define each method, and scope the activities required to accomplish the verification method.
5. What is validation, what is its primary objective, and what constitutes Validation success?
6. When is validation started and when does it end?
7. Who is accountable for performing validation?
8. What is the difference between V&V?
9. What is the 100X Software Rule and what are its implications to System Development?
10. Why is the 100X Software Rule important concerning latent defects?
11. What is IV&V and how is it applied?
12. How do you apply IV&V internally?

13.9.2 Level 2: Chapter Knowledge Application Exercises

Refer to www.wiley.com/go/systemengineeringanalysis2e

13.10 REFERENCES

Boehm, Barry W. (1981), *Software Engineering Economics*. Englewood Cliffs, NJ: Prentice-Hall.

CMMI-DEV (2010), *Capability Maturity Model Integration (CMMI) for Development*, Version 1.3, Pittsburgh, PA: Carnegie-Mellon University – CMMI Institute.

DAU (2012), *Glossary: Defense Acquisition Acronyms and Terms*, 15th ed. Ft. Belvoir, VA: Defense Acquisition University (DAU) Press. Retrieved on 6/1/15 from http://www.dau.mil/publications/publicationsDocs/Glossary_15th_ed.pdf.

FAA SEM (2006), *System Engineering Manual*, Version 3.1, Vol. 3, National Airspace System (NAS), Washington, DC: Federal Aviation Administration (FAA).

INCOSE (2011), *System Engineering Handbook*, TP-2003-002-03.2.2, Seattle, WA: International Council on System Engineering (INCOSE), Version 3.2.2.

ISO 9001:2008 (2008) *Quality management systems – Requirements*, International Organization for Standardization (ISO). Geneva, Sweden.

ISO/IEC 15288:2008 (2008), *System Engineering - System Life Cycle Processes*, Geneva: International Organization for Standardization (ISO).

ISO/IEC/IEEE 24765:2010 (2010), *Systems and software engineering—Vocabulary*, International Organization for Standardization, Geneva: ISO Central Secretariat.

MIL-STD-105E (1989), Military Standard: *Sampling Procedures and Tables for Inspection by Attributes*, Washington, DC: Department of Defense (DoD).

MIL-STD-973 (1992, Canceled 2000), Military Standard: *Configuration Management*, Washington, DC: Department of Defense (DoD).

Stecklein, J. (NASA); Dabney, J. (NASA); Dick, B. (Boeing); Haskins, B. (Boeing); Lovell, R. (Northrop Grumman); and Moroney, G. (Wylie Labs) (2004), *Error Cost Escalation Throughout the Project Life Cycle*, Table 13, NASA Technical Reports Server (NTRS), Washington, DC: NASA. http://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/20100036670.pdf. Retrieved on 1/17/14.

NASA SP 2007-6105 (2007), *System Engineering Handbook*, Rev. 1., Washington, DC: National Aeronautics and Space Administration (NASA). Retrieved on 5/1/13 from https://acc.dau.mil/adl/en-US/196055/file/33180/NASA%20SP-2007-6105%20Rev%201%20Final%2031Dec2007.pdf.

SEVOCAB (2014), *Software and Systems Engineering Vocabulary*, New York, NY: IEEE Computer Society. Accessed on 5/19/14 from www.computer.org/sevocab.

Shull, Forrest; Basili, Vic; Boehm, Barry; Brown, A. Winsor; Costa, Patricia; Mikael; Lindvall, Dan Port; Rus, Ioana; Tesoriero , Roseanne; and Zelkowitz , Marvin (2003), *What We Have Learned About Fighting Defects*: NSF Center for Empirically Based Software Engineering CeBASE). Retrieved on 4/9/14 from http://www.cs.umd.edu/%7Emvz/pub/eworkshop02.pdf.

14

THE WASSON SYSTEMS ENGINEERING PROCESS

If we investigate how organizations develop systems, the responses range from the traditional, *ad hoc* Plug and Chug … Specify–Build–Test–Fix (SBTF) Design Process Model (DPM) Paradigm methods to authentic SE-based models. Wasson (2012a) observes that due to the dilution of the SE discipline concerning *what it is* or *is not* SE, most Enterprises that employ the SBTF paradigm mistakenly believe they are performing SE and boldly proclaim their capabilities to Acquirers who mistakenly believe and accept the approach as SE. Then, both parties are unable to understand why programs fail and unfortunately blame SE as the culprit.

Since World War II, considered to be the beginning of modern SE, organizations such as the US Army, USAF, DoD, and IEEE developed a variety of SE Processes that have helped advance the state of SE practice. Research of the evolution of these models reveals how SE advanced, and newer models attempted to correct deficiencies in previous models and Enterprise practices at the time. Since Aerospace & Defense (A&D), for example, required a significant amount of rigor and traceability due to national security and lives at risk, commercial domains *erroneously* attached a bureaucratic paperwork stigma to SE.

Since the 1980s, the highly competitive global marketplace and economy have forced many commercial Enterprises to rethink their Engineering, System Development, and Quality System (QS) paradigms. Driven by the need to improve SYSTEM/PRODUCT performance, customer satisfaction, and profitability to survive, they are discovering that SE provides a key solution for achieving their business objectives.

Humans, by nature, generally deplore structured methods and will go to great lengths to avoid them without understanding (1) why they exist and (2) how they benefit from them. While traditional, *ad hoc* SBTF Paradigm engineering methods may be successful on simple, small systems and products, their lack of *consistency* and *scalability* of these methods to large, complex programs employing dozens or hundreds of people can lead to dysfunctional projects characterized by *disorder* and *chaos*. The same is true for development of upgrades to an existing system, product, or service. So, the question is: *Does a simple methodology exist that is explicit, easily understood, scalable, and can be applied for all size projects in any type of business domain*? The answer is yes!

14.1 DEFINITIONS OF KEY TERMS

- **Behavioral Domain Solution**—A multi-faceted, technical design that describes a SYSTEM or ENTITY's behavioral responses to encounters, interactions, and reactions to *stimuli*, *excitations*, or *cues* from external SYSTEMS within its OPERATING ENVIRONMENT. Decision artifacts include logical capabilities and architecture, SySML™ Sequence and Activity Diagrams, design descriptions and Requirements Traceability Matrices (RTMs).
- **Hypothesis**—"A proposition tentatively assumed in order to draw out its logical or empirical consequences and test its consistency with facts that are known or may be determined" (Merriam-Webster, 2013).
- **Iterative Characteristic**—An attribute that describes the interactions between each of the SE Process Model's elements.

System Engineering Analysis, Design, and Development: Concepts, Principles, and Practices, Second Edition. Charles S. Wasson.

Companion Website: www.wiley.com/go/systemengineeringanalysis2e

- **Operations Domain Solution**—A multi-faceted, technical solution of a SYSTEM or ENTITY that describes how the System Developer, in collaboration with the User(s) and System Acquirer, envision deploying, operating, maintaining, sustaining, retiring, and disposing of the system. Decision artifacts include ConOps document containing an Operational Architecture and Operational Concept Descriptions (OCDs), M&S, and RTM.
- **Physical Domain Solution**—A multi-faceted, technical design of a SYSTEM or Entity that describes its selected physical implementation. Decision artifacts include design descriptions, RTMs, conference and review minutes, assembly drawings, schematics, wiring lists, parts lists, test cases, and procedures.
- **Point Design Solution**—A solution selected by reading specification requirements and immediately jumping to a single physical design without due consideration of (1) how the User plans to use the SYSTEM and (2) expects it to interact with or respond behaviorally to its User(s) and external SYSTEMS in its Operating Environment or (3) an Analysis of Alternatives (AoA) in making those decisions.
- **Recursive Characteristic**—An attribute of the SE Process Model that enables it to be applied to any entity within a SYSTEM regardless of level of abstraction.
- **Requirements Domain Solution**—A hierarchical framework of requirements *traceable* to User *originating* or *source* requirements that bound and specify the capabilities, performance, interfaces, environmental conditions, design and construction constraints, quality factors, and verification methods that characterize a SYSTEM or ENTITY to be developed. Decision artifacts include stakeholder identification, user stories, use cases and scenarios, specifications, Design Criteria Lists (DCLs), RTMs, analyses, trade studies, Models & Simulations (M&S), key decision artifacts, conference and review minutes used to derive the requirements and their levels of performance.
- **SE Process Model**—A construct derived from a *highly iterative*, problem-solving–solution development methodology that can be applied *recursively* to multiple levels of system design.
- **Solution Domain**—A unique requirements, operations, behavioral, or physical solution traceable to User source or originating requirements that expresses how a system, product, or service is bounded and specified, envisioned to operate, engage, and respond behaviorally to external systems, and be physically implemented.

14.2 APPROACH TO THIS CHAPTER

Chapter 14 introduces the Wasson SE Process Model, its underlying problem-solving and solution development methodology, and application to development of a SYSTEM or ENTITY's design. To better understand what an SE Process is, we:

- Provide a brief background discussion of the evolution of SE Processes that have enabled SE to advance as a discipline.
- Highlight shortcomings in current SE process paradigms that drive the need to shift to a new level of SE performance.
- Illustrate how the Scientific Method contributes to and influences the *ad hoc* Plug and Chug … SBTF Paradigm.

To address the need for a new level of SE Process performance, we introduce the Wasson SE Process Model as a solution. We provide a graphical and text description of the model and address its two characteristics: *highly iterative* and *recursive.* We illustrate the model's *highly iterative* internal activities *iterate* and how the model applies *recursively* to multiple levels of abstraction within the System Design Process shown in Figures 12.4 and 14.9.

Given an understanding of the Wasson SE Process Model, we provide a high-level example of how the model is applied to the development of a Tablet Computer System.

We conclude the chapter with a summary discussion concerning the strength of the Wasson SE Process Model.

14.3 EVOLUTION OF SE PROCESSES

Since World War II, several types of SE processes have evolved. Organizations such as the US Department of Defense (DoD), the Institute of Electrical and Electronic Engineers (IEEE), the International Council on Systems Engineering (INCOSE), and others have documented a series of SE process methodologies. Specific examples include the AFSCM 375-5 (1966), US Army FM 770-78 (1979), DoD MIL-STD-499B DRAFT (1994), and IEEE 1220-1994. Each of these SE process methodologies highlights key aspects its developers considered fundamental to Systems Engineering practice at a specific time.

14.3.1 Commonly Used SE Process Model

One of the commonly used SE Processes today originates from AFSC 375-5 (1966) that evolved over many years into DoD MIL-STD-499B DRAFT (1994), which was canceled

in 1994 before approval. The key activities of the process are:

1. Requirements Analysis
2. Functional Analysis and Allocation
3. Synthesis
4. System Analysis and Control

At the time it was created, the process advanced SE thinking. However, newcomers to SE view its primary features—Requirements Analysis, Functional Analysis and Allocation, and Synthesis—as being too abstract to understand. Additionally, two of the features—Requirements Analysis and Functional Analysis—are relevant but are insufficient to meet today's needs for Engineering system, product, or services. Specifically:

1. **Requirements Analysis**—Infers that requirements exist and can be analyzed. Perhaps this might be the case for an Electrical, Mechanical, and Software Engineering after multi-discipline SE teams have flowed specification requirements down for a printed circuit board, designing a software module. The reality is requirements begin in the form of *abstract*, *visionary* User mission objectives, UCs, and scenarios that will be ultimately translated into capability-based requirements. Although seasoned SEs who have applied the MIL-STD-499B DRAFT process may understand the context and scope of Requirements Analysis, this does not mean that its title explicitly communicates what SEs are required to understand or what Requirements Analysis entails.
2. **Functional Analysis and Allocation**—Although this topic is relevant to defining system behaviors, it is deficient in terms of meeting today's SE needs. In Chapter 3, stated that a *function* identifies an *action* to be performed to achieve an *outcome*. An action, however, does not embody *performance*, a separate attribute. In contrast, a *capability* encompasses both *function* and *performance*.
   When SEs and others say they are going to perform Functional Analysis and Allocation and identify functions, that activity is the easy part. The challenge is often *quantifying* the level of performance within the context of the System and User priorities coupled with cost and schedule impacts.
   Additionally, flowing down specification requirements allocations to lower levels is more than simply "decomposition of functions." Performance allocations related to the "functional requirements" are often a major issue. Assigning a performance value to a functional requirement can be difficult; allocating the performance value to lower level components can be even more challenging. The reality of needing to simultaneously allocate and flow down both function and performance illustrates the *inadequacy* of performing Functional Analysis as a primary SE activity:

Although the SE standards noted above advanced the state of the practice in SE, in the author's opinion, no single SE process captures the actual steps performed in Engineering a System, Product, or Service.

Over the years, SEs and Enterprises have often formulated their own variations of the SE Process. Here's the challenge: what many Enterprises and SEs believe or perceive to be an SE Process implementation is actually an nothing more than the *ad hoc*, *endless loop* paradigm referred to by Wasson (2012a) as the Plug & Chug … SDBTF Paradigm. The paradigm, which is intended as a *problem-solving and solution development* methodology, reflects a convolution of the Scientific Method of Inquiry and Archer's Design Process Model (1965).

To better understand this last point as a backdrop to the Wasson SE Process Model, let's briefly explore the relationship between the Scientific Method and the Engineering Design Process employed by Engineers.

14.3.2 Comparison of Archer's Design Process and the Scientific Method to Engineering Design

Archer's DPM (1965) shown earlier in Figure 2.9 and the Scientific Method provide a key foundation as a *general* problem-solving and solution development method. Although the operative term is *scientific* in Scientific Method, it is not unique to science and has general application to any type of problem or issue requiring scientific inquiry or investigation. As introduced earlier in Chapters 2 and 11, Engineers enter the workforce with an in-grained culture that is exemplified by a traditional SDBTF-DPM Engineering Paradigm that is perceived to be SE.

To better understand how this occurs, let's explore a comparison of similarities between the Archer's DPM (1965), the Scientific Method, and Engineering Design Process. Table 14.1 provides an illustrative comparison.

Observe several key points about the Engineering Design Process:

Step 1: Identify and Analyze Requirements

Step 2: Research Legacy Designs and Vendor Catalogs

Step 3: Hypothesize System or Entity Design Solution

Step 4: Develop Prototype, Model, or Simulate Areas of Interest or Risk

Step 5: Develop Test Procedure

Step 6: Test the Prototype, Model, or Simulation and Document Results

TABLE 14.1 A Comparison of Similarities Between Traditional Engineering Design Process, Archer's DPM (1965), and the Scientific Method

Sequence	Scientific Method	Archer's DPM (1965)	Traditional Engineering Design Process
Step 1	State the question, issue, or problem you need to answer	Data Collection	Identify and analyze requirements
Step 2	Conduct background research related to the question, issue, or problem	Data Collection & Analysis	Research legacy designs and vendor component catalogs for potential solutions
Step 3	Formulate an hypothesis	Synthesis	Hypothesize System or Entity Design Solution
Step 4	Design experiment to validate the hypothesis	Synthesis & Development	Develop prototype, model, or simulate areas of interest or risk
Step 5	Develop experiment test procedure	Development	Develop test procedure
Step 6	Conduct experiment and document results	Development	Test the prototype, model, or simulate and document the results
Step 7	Analyze experiment results	Analysis	Analyze lab test results
Step 8	Develop reasoned conclusions concerning the validity of the hypothesis	Synthesis	Redesign, rework, or tweak System or Entity design
Step 9	Repeat Steps 2–8 as necessary	Repeat Steps 2–8 as necessary	Repeat Steps 2–8 as necessary.
Step 10	Communicate the results	Communication	Review, approve, and release design

Step 7: Analyze Lab Test Results

Step 8: Redesign, Rework, or Tweak System or Component Design

Step 9: Repeat Steps 2 through 8 (endless loop)

Step 10: Review, Approve, and Release Design

The preceding points illustrate the *ad hoc, endless loop* SDBTF-DPM Paradigm employed by some Enterprises and Engineers. *Is there any wonder why*:

- Teams are often *dysfunctional* and exhibit *chaos* due to of a lack of consensus into how to perform SE design as a result of a lack of Engineering education?
- Engineers always have to redesign some aspect of the SYSTEM or ENTITY during the System Integration, Test, and Evaluation (SITE) Phase due to:
  a. "Quantum leaps" from requirements to a point design solution (Figure 2.3) resulting in rework and redesign at significant cost (Chapter 12 - Boehm).
  b. *Incompatibility* and *interoperability* issues resulting in cost overruns and schedule problems?
- Project Managers complain that Engineers "gold-plate" their designs as evidenced by overrun project budgets and schedules and have to be told when to stop working on a design?

Principle 14.1

Problem-Solving–Solution Development Principle

A quantum leap from requirements to a point design solution does not reflect Problem Solving or Solution Development.

System development success can only be as good as the process employed and its human knowledge, implementation, and adherence to the process. Since human decision-making is compounded by politics, no SE Process is perfect. However, SEs can establish a process for a project and exercise "intellectual control" (McCumber and Sloan, 2002, p. 4) over its performance. This brings us to this chapter's topic, the Wasson SE Process Model, as a solution to shift current SE Paradigms to a new level of performance.

14.4 THE WASSON SE PROCESS MODEL

Whereas the Scientific Method serves as a universal problem-solving and solution development methodology for *scientific inquiry* and *investigation*, the Wasson SE

Process Model enables SEs, Engineers, System Analysts, et al to:

- Understand the Problem or Opportunity Space a User needs to resolve at any level of abstraction.
- Analyze, bound and specify, and decompose a contextual Problem Space at any level of abstraction – SYSTEM, PRODUCT, SUBSYSTEM, - into one or more Solution Spaces.
- Formulate, develop, evaluate, and select each of the SYSTEM/ENTITY's Four Domain Solutions.
- Balance the SYSTEM/ENTITY's technical, technology, development and life cycle cost, schedule, and support solutions and risks.

These points illustrate what the Wasson SE Process Model accomplishes for SE&D. Additionally, the Model provides additional benefits related to our discussions earlier in Chapter 2. It:

- Provides a common problem-solving and solution development method that is Engineering discipline independent and can be employed by multi-discipline teams.
- Overcomes the quantum leap from requirements to single point design solution (Figure 2.3) problems related to the *ad hoc, endless loop* SBTF-DPM Engineering Paradigm.

Let's begin with a discussion of the model's methodological-based structure.

14.4.1 Wasson SE Process: Methodological-Based Structure

The conceptual foundation for the Wasson SE Process originates from the Four Solution Domains - Requirements, Operations, Behavioral, and Physical Domain Solutions – introduced in Chapter 11. Although the Four Domain Solutions provide a strategic roadmap to develop and characterize a SYSTEM/ENTITY's requirements, operations, behavior, and physical implementation; the challenge is they require additional steps related to:

- Understanding the SYSTEM/ENTITY's contribution to the User's Problem Space.
- Ensuring that the overall System Design Solution performance is *optimal* to meet the User's mission needs.

The underlying methodology for the model consists of the following steps:

- Step 1: Understand the Problem/Opportunity and Solution Spaces.
- Step 2: Develop the Requirements Domain Solution.
- Step 3: Develop the Operations Domain Solution.
- Step 4: Develop the Behavioral Domain Solution.
- Step 5: Develop the Physical Domain Solution.
- Step 6: Evaluate and Optimize the System Design Solution.

Figure 14.1 provides a graphical representation of the Wasson SE Process Model. Within the model, workflow activities include a number of *iterative* dependencies as illustrated in Figure 14.2 coupled with staged Verification and Validation (V&V) as well as corrective action, if necessary.

As a problem-solving and solution-development method, the Wasson SE Process Model applies to the System and Entities at any level of abstraction or entity with a level. Therefore, role-based terms such as System Acquirer, User, and System Developer are *contextual*. Consider the following example:

Example 14.1 **Contextual System Acquirer, User, and System Developer Roles**

A System Acquirer (role) contracts with a System Developer (role) Enterprise to develop a SYSTEM.

Within the System Developer's project organization, a PRODUCT-Level Team (Acquirer role) allocates requirements for developing a SUBSYSTEM to an internal SUBSYSTEM Product Development Team (PDT) or subcontractor performing a System Developer (contextual role) to develop and deliver a SUBSYSTEM.

Step 1: Understand the Problem/Opportunity and Solution Spaces

The first step of the Wasson SE Process Model is to simply understand the User's Problem/Opportunity and Solution Spaces. SEs and System Analysts need to understand and validate *how* the User envisions employing the SYSTEM to accomplish performance-based outcomes to achieve mission objectives. This requires understanding:

1. The SYSTEM/ENTITY's role in the User's Level 0 SYSTEM.
2. How the User plans to deploy; operate, maintain, sustain (OM&S); retire, and dispose of the SYSTEM/ENTITY - its Use Cases (UCs) and scenarios.
3. The SYSTEM/ENTITY's interfaces and interactions with external SYSTEMS such as HUMAN SYSTEMS, NATURAL SYSTEMS, and INDUCED Environments (Chapter 9).
4. The expected performance-based behavioral responses and outcomes to interactions with external systems in its OPERATING ENVIRONMENT.

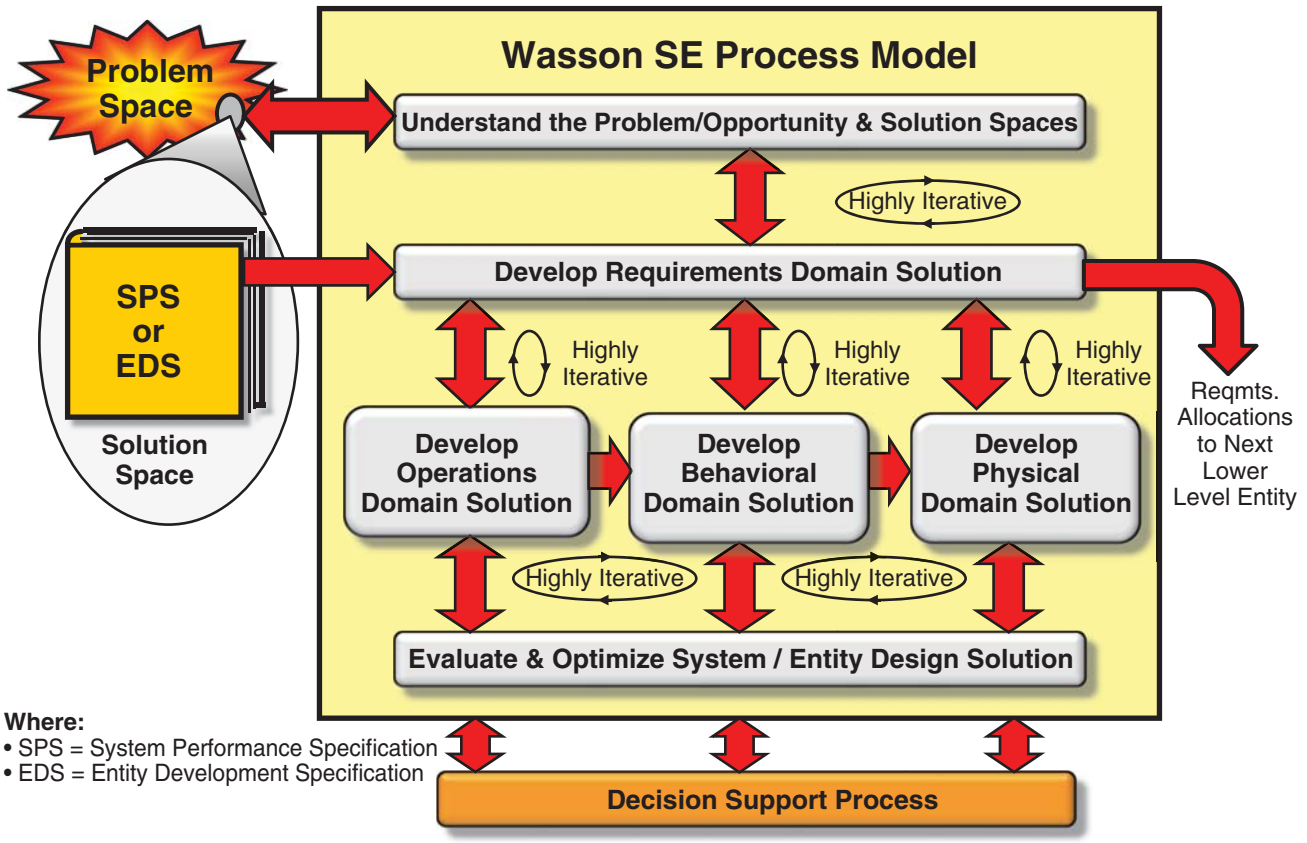


Figure 14.1 The Wasson System Engineering Process Model

5. The SYSTEM/ENTITY's Mission Event Timeline (MET) related to Pre-Mission, Mission, and Post-Mission operations.

Work products of the Understand the Problem/Opportunity and Solution Spaces include:

- Identification of the User's Problem Statement (Chapter 4).
- Definition of User missions and performance-based objectives and outcomes (Chapter 5).
- A Context Diagram (Figure 8.1) illustrating SYSTEM's context within its OPERATING ENVIRONMENT and interfaces.
- Partitioning of the Problem/Opportunity Space into one or more Solution Spaces (Figure 4.7).
- Identification of Key Performance Parameters (KPPs), Measures of Effectiveness (MOEs), and Measures of Suitability (MOSs) (Chapter 5).
- Technology constraints.
- Cost constraints.

Step 2: Formulate, Select, and Develop the Requirements Domain Solution

As the understanding of the SYSTEM's or ENTITY's Problem/Opportunity and Solution Spaces *evolves* and *matures*, the next step is to formulate, evaluate, select, and mature the Requirements Domain Solution.

The Requirements Domain Solution consists of a hierarchical set of requirements derived from a User's *source* or *originating* requirements as shown in Figure 14.3. These requirements are typically documented in marketing analysis documents and contract documents such as Statement of Objectives (SOO) and System Requirements Document (SRD). Our objective is to derive requirements to a lower level that is explicitly *sufficient* for allocating each requirement directly to *one and only one* architectural entities.

The Requirements Domain Solution is represented by a Requirements Architecture consisting of a multi-level framework of requirements traceable to the User's *source* or *originating* requirements. When the requirements are partitioned into specifications such as an SPS and lower level EDSs, the framework of specifications is referred to as a Specification Tree (Figure 19.2). Since some systems may contain 10's of thousands of requirements, the Spec Tree provides a simpler representation of the Requirements Domain Solution.

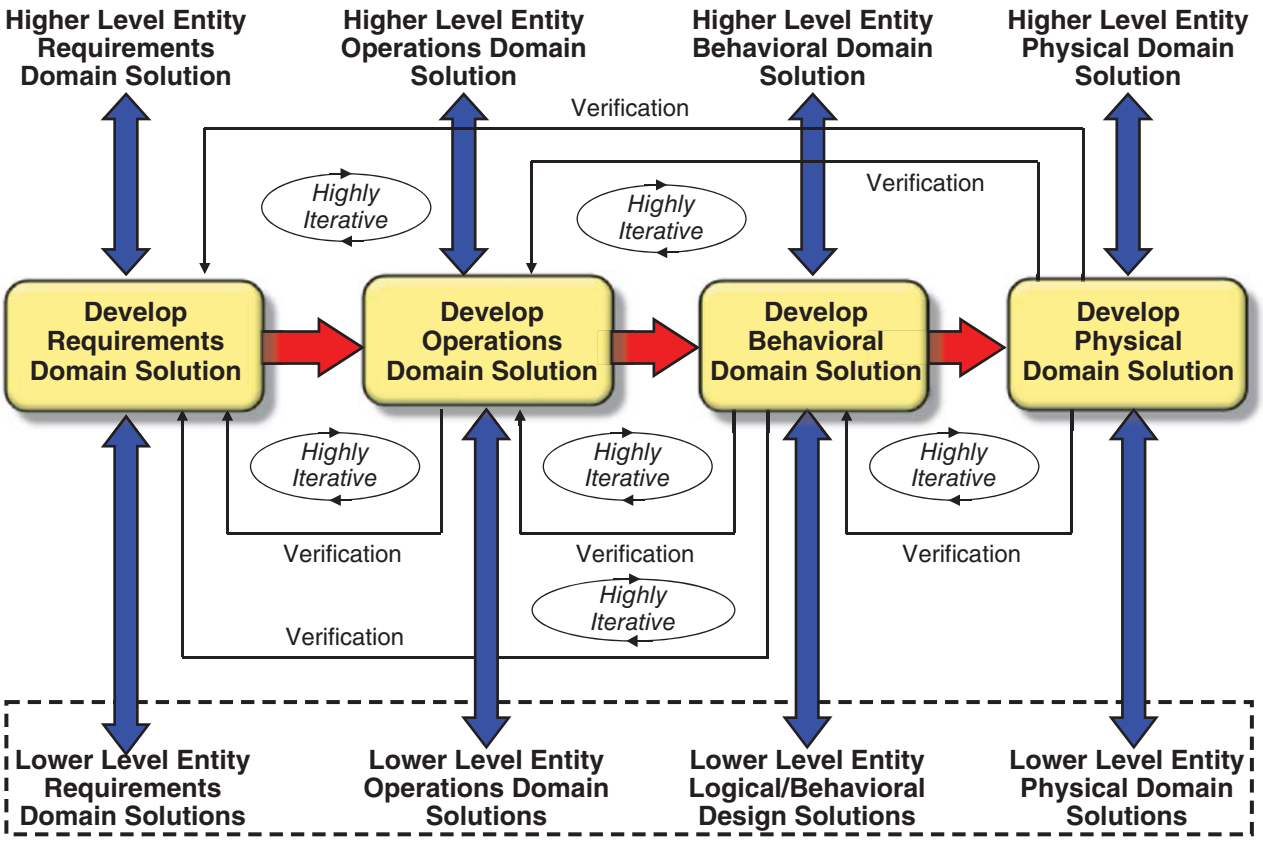


Figure 14.2 Iterative Dependencies within the Wasson SE Process Model

The Requirements Domain Solution encompasses more than a Spec Tree and its specifications. It also includes the analyses, trade-studies, Models & Simulations, and so forth that were used to create the specifications, especially requirement performance values.

Referring to Figures 14.1 and 14.2, SYSTEM/ENTITY Requirements Domain Solution activities include:

- Understanding, analyzing, bounding, and specifying the SYSTEM/ENTITY's Solution Space, operational capabilities, interfaces, environmental, and other constraints.
- Iterating with the Understand the ENTITY's Problem/Opportunity and Solution Spaces to reconcile COIs/CTIs.
- Deriving SPS or ENTITY EDS requirements to lower levels within the document for direct allocation and flow down to lower level architectural entities. For example, a PRODUCT EDS to A SUBSYSTEM EDS and a SUBSYSTEM EDS to an ASSEMBLY EDS.
- Ensuring traceability to higher level specifications such as the SPS or lower level specifications such as PRODUCT, SUBSYSTEM, or ASSEMBLY Level EDSs.

Requirements Domain Solution *work products* include:

- *Quality Records* (QRs) that document technical decision artifacts, User Design Criteria Lists (DCLs – Chapter 17), conference and review minutes, analyses, and technical reports.
- Specification Tree consisting of the SPS and lower level ENTITY development specifications.
- Supporting analyses, models, and simulations.

Step 3: Formulate, Select, and Mature the Operations Domain Solution

As the SYSTEM/ENTITY's Requirements Domain Solution *evolves* and *matures*, System Developers initiate the activities to formulate, evaluate, select, and mature the SYSTEM/ENTITY's Operations Domain Solution from a set of viable candidate solutions (Chapter 32). In general, the SYSTEM Level Operations Domain Solution captures *how* the User envisions deploying, operating, maintaining, sustaining, retiring, and disposing of the SYSTEM. Operational concepts for each of these activities are documented in the SYSTEM/ENTITY's Concept of Operations (ConOps) document or Theory of Operations that serves as the focal point to guide early development of the System Design Solution.

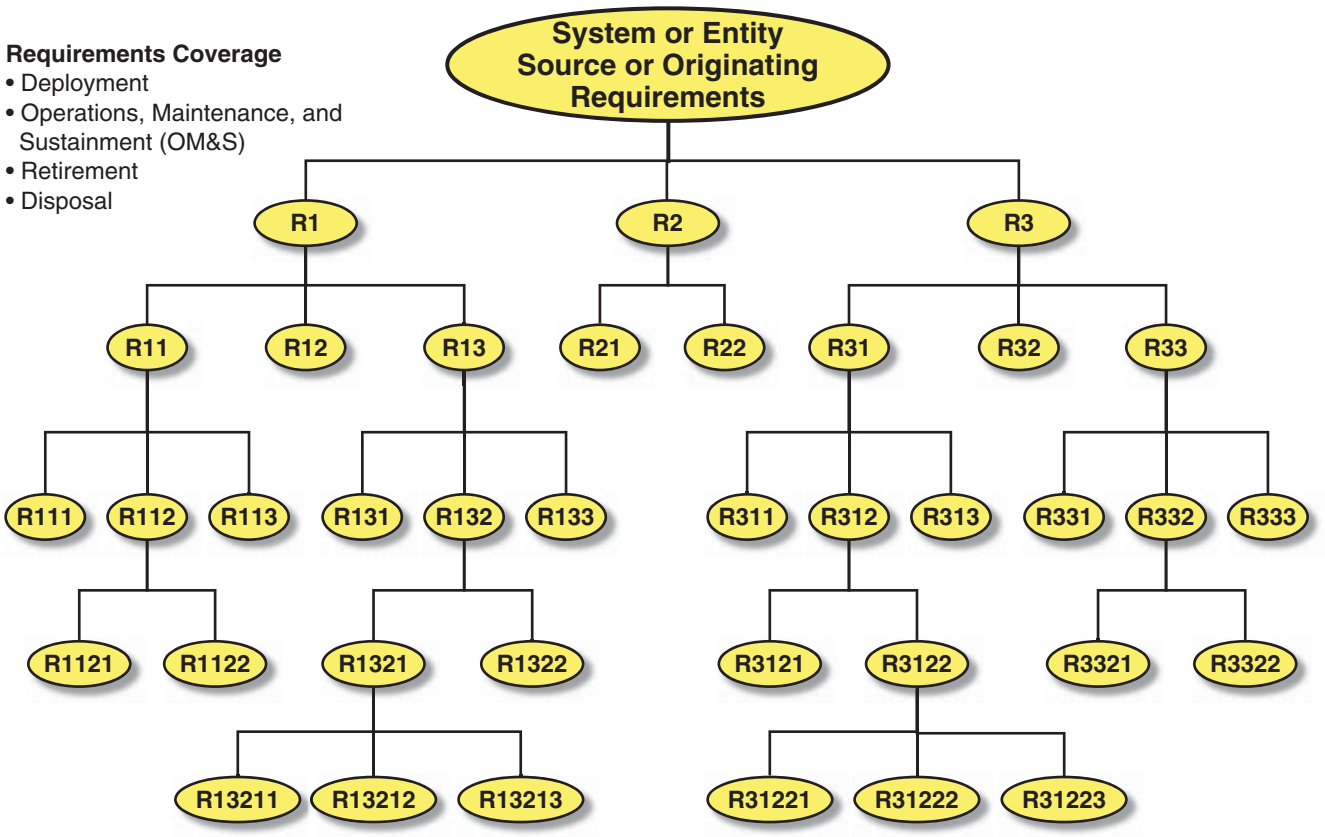


Figure 14.3 Example Requirements Domain Solution Structure

The Operations Domain Solution serves as a critical step in expressing the Conceptual System Design. Typically, the ConOps Document serves that purpose. One of the key topics in the ConOps is the Operational Architecture shown in Figure 14.4.

Operational Architectures can be expressed in a number of forms and media. Two key points need to be expressed and communicated by an Operational Architecture:

1. What is required to deploy, operate, maintain, and sustain (OM&S), retire, and dispose of a system, product, or service similar to Figures 6.1 and 6.2. This graphic depicts "A Day in the Life Cycle of a System" as well as "A Day in the Life Cycle of a System's Mission."
2. Large systems with mega budgets often create artist or architectural renderings or cartoon illustrations depicting real-life images of the system's interactions with external systems in its OPERATING ENVIRONMENT such as the one shown in Figure 14.4. These illustrations provide an enhanced understanding of a system and its interactions with external systems in its OPERATING ENVIRONMENT. For example, compare the System Block Diagram (SBD) at the bottom of the page with the cartoon graphic at the top of the page containing the same information.

Referring to Figures 14.1 and 14.2, SYSTEM/ENTITY Operations Domain Solution activities include:

1. Continuously monitoring the SYSTEM/ENTITY's Requirements Domain Solution for updates.
2. Establishing the Level 0 User's Operational Architecture that characterizes interactions between the SYSTEM/ENTITY and friendly, benign, or hostile Systems and threats in its OPERATING ENVIRONMENT.
3. Developing and modeling the end-to-end – Pre-Mission → Mission → Post-Mission - sequences of operations and tasks that comprise the SYSTEM/ENTITY's mission cycle.
4. Identifying the SYSTEM/ENTITY's performance-based outcomes to be accomplished.
5. Synchronizing operational tasks with the MET.
6. Verifying *coverage, consistency, completeness,* and *compliance* to the respective Requirements Domain Solution specification requirements.
7. Reconciling COIs/CTIs with the Requirements and Behavioral Domain Solutions.

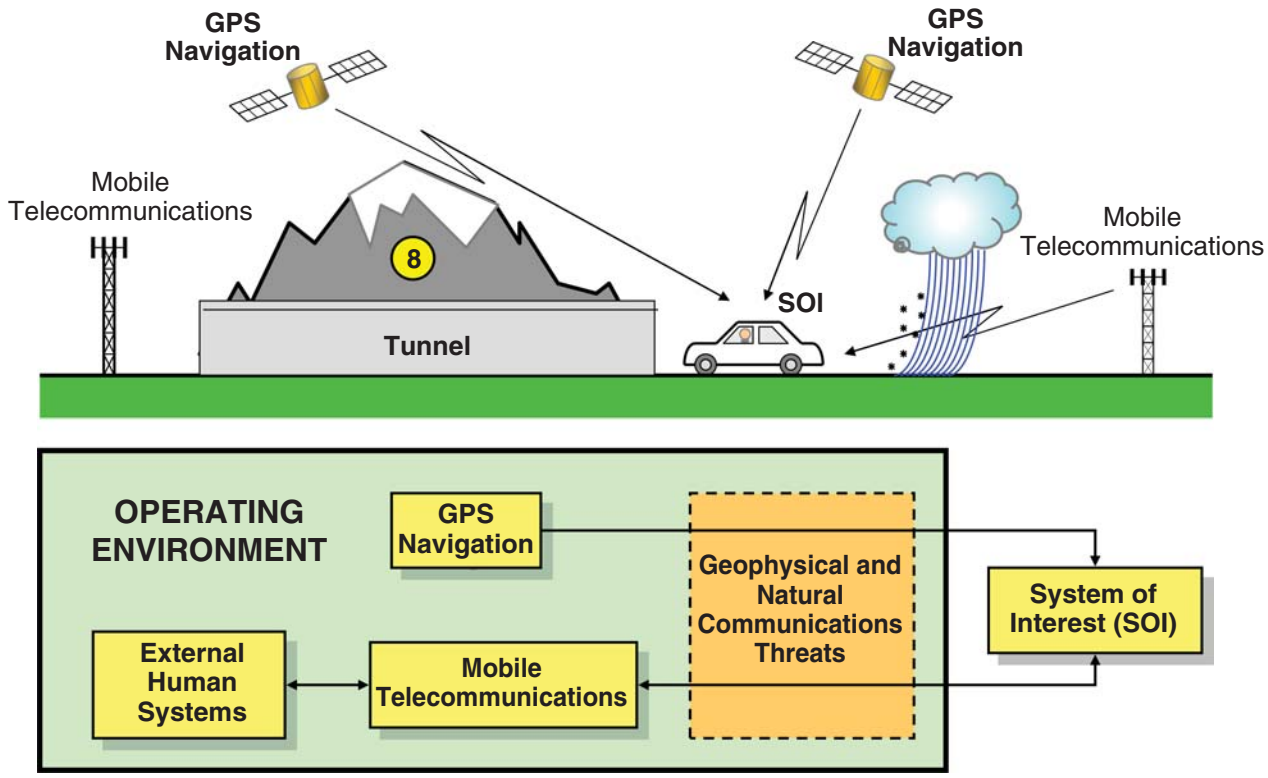


Figure 14.4 Operational Architecture Example

8. Integrating the solution into the next higher level (Figure 14.12) such as the User's Level 0 SYSTEM or the project's SYSTEM, PRODUCT, ASSEMBLY, and SUBASSEMBLY Operations Domain Solutions.
9. Responding to change initiated by the Requirements, Behavioral, and Physical Domain Solutions.

The Operations Domain Solution work products consist of documented decision artifacts such as:

- *Quality Records* (QRs) that document technical decision artifacts, such as conference and review minutes, analyses, and technical reports.
- The ConOps that provides Operational Concept Descriptions (OCDs) for deploying, operating, supporting, sustaining, retiring, and disposing of the SYSTEM/ENTITY (Chapter 6).
- The Operational Architecture that identifies the SYSTEM/ENTITY's mission cycle including interactions with external SYSTEMS— SysML™ Actors— within its OPERATING ENVIRONMENT.
- Establishment of the SYSTEM or PRODUCT's Developmental Configuration Baseline (Chapter 16) including Requirements Domain Solution work products that have been reviewed, approved, and released.
- Mission Event Timeline (MET).
- System Phases, Modes, and States of Operation (Chapter 7).

Step 4: Formulate, Select, and Mature the Behavioral Domain Solution

As each ENTITY's Operations Domain Solution *evolves* and matures, System Developers initiate activities to formulate, evaluate, select, and mature the Behavioral Domain Solution from a set of viable candidate solutions (Chapter 32). In general, the Behavioral Domain Solution describes *what* has to be accomplished in terms of logical interactions and sequences of capability-based tasks required to produce the desired performance-based outcomes.

Referring to Figures 14.1 and 14.2, SYSTEM/ENTITY Behavioral Domain Solution activities include:

1. Modeling SYSTEM/ENTITY internal capability *transfer functions* and interactions with external SYSTEMS in its OPERATING ENVIRONMENT throughout its mission cycles such as the example shown in Figure 14.5.
2. Synchronizing SYSTEM/ENTITY operations with the MET such as the example shown in Figure 14.6.
3. Reconciling COIs/CTIs with the Requirements, Operations, and Physical Domain Solutions (Figure 14.2).
4. Ensuring its:

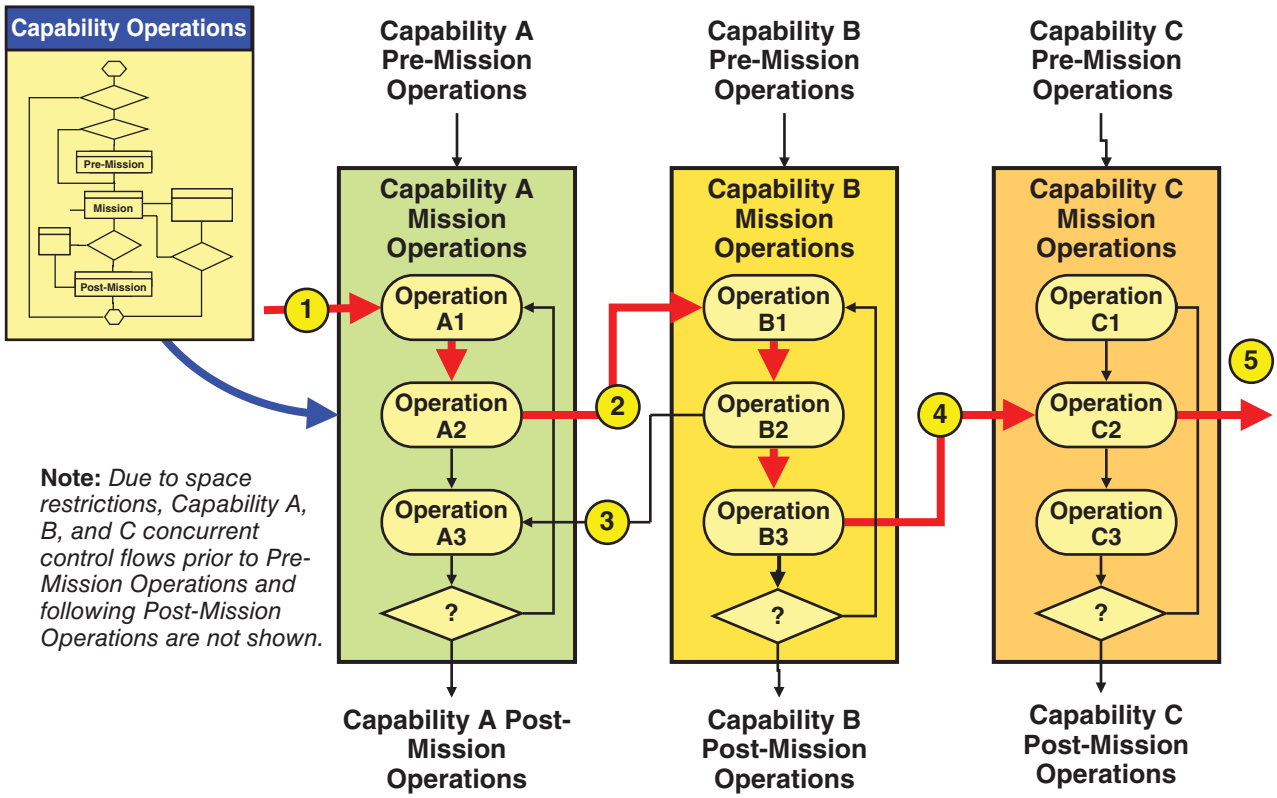


Figure 14.5 Behavioral Capability Architecture Illustrating Mission Phase of Operation Sequential Control Flow and Data Flow Interactions

a. Horizontal *consistency* with and *traceability* to the Requirements and Operations Domain Solutions (Figure 14.2).
b. Vertical *consistency* with and *traceability* to the next higher-level Behavioral Domain Solution (Figure 14.12).

5. Responding to changes from the Evaluate and Optimize the System Design Solution Process.

Behavioral Domain Solution work products consist of decision *artifacts* such as:

1. Quality Records (QRs) that document technical decision artifacts such as conference and review minutes, analyzes, trade studies, models, and simulations.
2. The Logical Capability Architecture that depicts the configuration and interactions between SYSTEM/ENTITY capabilities.
3. Models of SYSTEM/ENTITY behavior consisting of N2 Diagrams (Figure 8.11) and SysML™ Sequence and Activity Diagrams (Figures 5.10, 5.11, and 14.5).
4. Update of the SYSTEM or PRODUCT's Developmental Configuration Baseline (Chapter 16) to include Behavior Domain Solution work products that have been reviewed, approved, and released.

Step 5: Formulate, Select, and Develop the Physical Domain Solution

As the Entity's Behavioral Domain Solution *evolves* and matures, System Developers initiate activities to formulate, evaluate, select, and mature the Physical Domain Solution from a set of viable candidate solutions (Chapter 32). In general, this solution implements the Behavioral Domain Solution via physical components that have been architecturally configured and selected to provide the capabilities required to accomplish User missions.

Referring to Figures 14.1 and 14.2, SYSTEM/ENTITY Physical Domain Solution activities include:

- Continuously monitoring the SYSTEM/ENTITY's Requirements, Operations, and Behavioral Domain Solutions for updates.
- Formulating, evaluating, and selecting an optimal Physical Architecture via an AoA based on a set of viable candidate architectures.
- Linking Behavioral Domain Solution capabilities to the Physical Architecture components as shown in Figure 14.7.

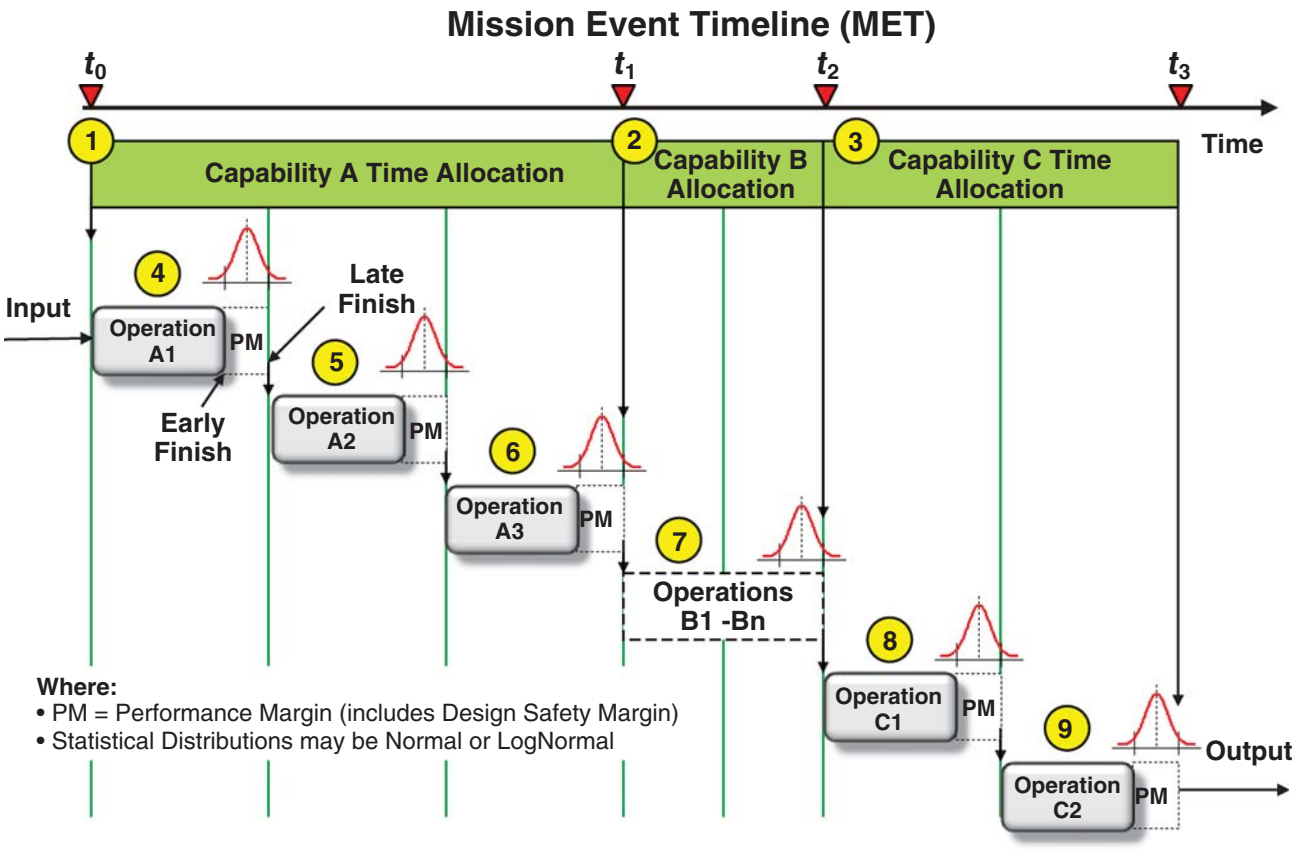


Figure 14.6 Synchronization of MET Time Performance Allocations to Use Case Based Behavioral Capabilities

- Establishing performance budgets and design safety margins for physical attributes such as electrical power, size, and weight.
- Finalizing selection of physical components via Make versus Buy versus Buy–Modify decisions (Chapter 16) to attain the best mix of components within the architecture to achieve technical and project performance, costs, technology, schedule, and risk objectives.
- Translating the Physical Architecture into a detailed System Design Solution consisting of hardware assembly drawings, schematics, wiring diagrams, and software designs that meet *necessary* and *sufficiency* criteria for procurement and/or development.
- Assessing the solution's *compatibility* and *interoperability* with external SYSTEMS in its OPERATING ENVIRONMENT.
- Reconciling COIs/CTIs with the Requirements and Behavioral Domain Solutions.
- Integrating the solution into the next higher level (Figure 14.12) such as the project's SYSTEM, PRODUCT, ASSEMBLY, and SUBASSEMBLY Physical Domain Solution.
- Ensuring consistency and completeness with the Operations Domain Solution.
- Ensuring traceability to the Requirements Domain Solution.
- Responding to changes requested by the Evaluate and Optimize System Design Solution Process.
- Update of the SYSTEM or PRODUCT's Configuration Baseline (Chapter 16) to include Physical Domain Solution work products that have been reviewed, approved, and released.

Referring to Figures 14.1 and 14.2, each Physical Domain Solution consists of decision artifacts documented in work products such as the SYSTEM/ENTITY's:

- Quality Records (QRs) that document technical decision artifacts such as conference and review minutes, analyzes, trade studies, models, and simulations.
- Physical System Architecture (Chapter 8).
- Matrix linking Behavioral capabilities to Physical Architecture components (Figure 14.7).
- Product Work Breakdown Structure (PWBS).
- System/Segment Design Description (SSDD).

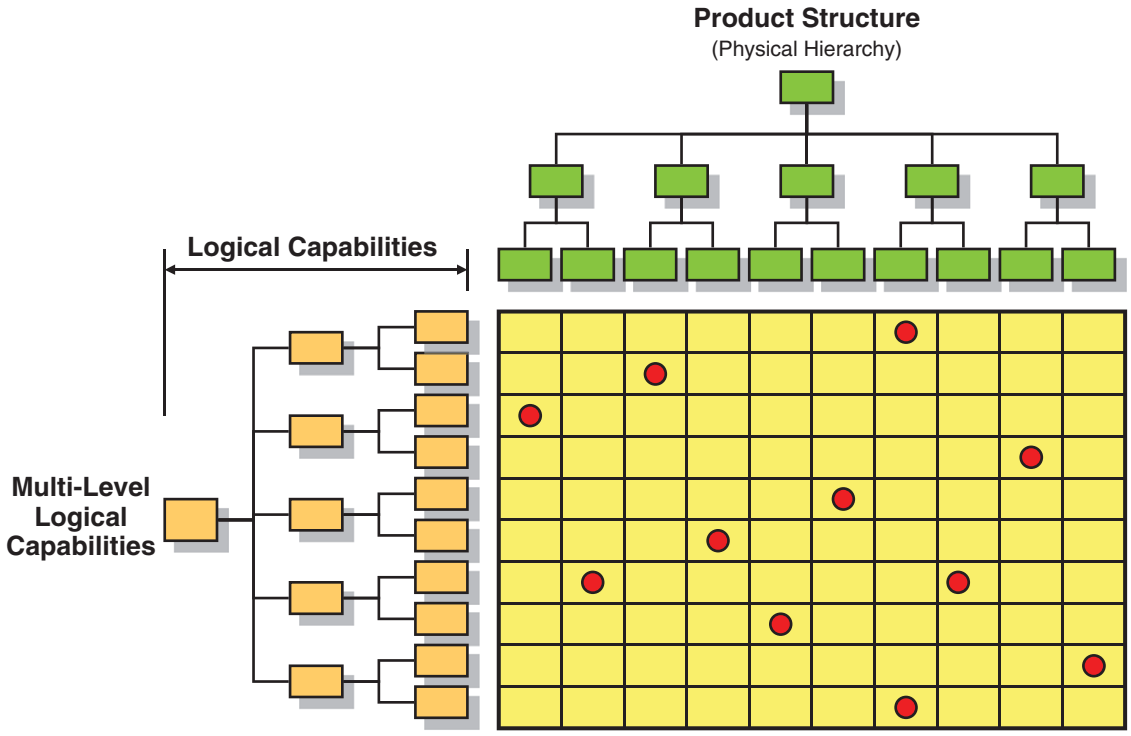


Figure 14.7 Linking Behavioral Domain Solution Capabilities to the Physical Domain Solution's Product Structure

- Hardware Design Description (HDD).
- HW Interface Control Documents (ICDs).
- System Design Descriptions (SDD).
- SW Interface Design Descriptions (IDDs).
- SW Database Design Descriptions (DBDDs).
- Component procurement specifications.
- Design requirements such as assembly drawings, parts lists, schematics, cabling diagrams, and wiring lists.
- HW and SW Test Cases (TCs) and procedures for verifying the Physical Domain Solution.

Step 6: Evaluate and Optimize the Entity's Total Design Solution

As the Physical Domain Solution *evolves* and *matures*, the Evaluate and Optimize the System Design Solution process *continuously* assesses the Operations, Behavioral, and Physical Domain Solutions. Observe that we said System Design Solution, which represents the Developmental Configuration, will ultimately include all levels of abstraction over time. Since we start at the top level, the first instance of the Developmental Configuration begins with the SYSTEM Level followed by PRODUCTS and SUBSYSTEM levels as illustrated in Figure 8.4.

The objective of this step is to review and respond to requests from PDTs at various levels of abstraction to provide meaningful data that will support *informed* decision making in areas such as (1) achievement of specification requirements performance, (2) analysis of COIs/CTIs, and (3) evaluation of viable candidate solutions.

Author's Note 14.1

Organizational Decision Support

It is important to note here the organizational relationship between a PDT and Decision Support. Decision Support:

- May be performed by (1) a System Analysis team within the project, (2) an independent Enterprise organization external to the project, or (3) a subcontractor.
- Provides specialized analytical methods, tools, expertise, models & simulations, and prototyping capabilities that the PDT may not possess.

In their multi-discipline role, a PDT uses the data provided by Decision Support to make *informed* decisions that reflect the definition of SE:

- Achieve the right balance in terms of System Design Solution technical performance and compliance.
- Apply the right technology.

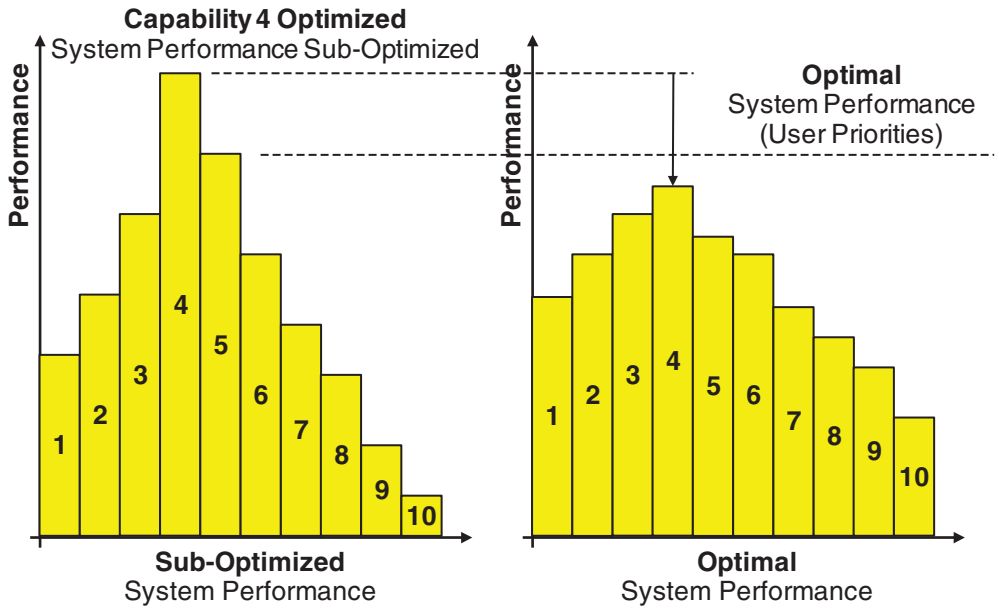


Figure 14.8 Optimized versus Optimal Performance Example

- Minimize development and life cycle costs.
- Determine acceptable risk.

Chapters 30–34 focus on Decision Support Practices.

You will encounter people who contend that it is impractical to *optimize* a system, product, or service for a diverse set of operating scenarios and conditions—it can only be *optimal*. For a prescribed set of User priorities and OPERATING ENVIRONMENT conditions, you can mathematically optimize a SYSTEM. The challenge is that these conditions are often *independent* and *statistically random* occurrences in the OPERATING ENVIRONMENT. As a result, a system, product, or service performance may not be *optimized* across all sets of random variable conditions as shown in Figure 14.8. So, people characterize the SYSTEM's performance as *optimal* in dealing with these random variables.

Author's Note 14.2

System Design Solution Technical Accountability

Developing the Requirements, Operations, Behavioral, and Physical Domain Solutions simply implements a technical strategy. The success of any process is determined by the knowledge, experience, and discipline of the humans performing the work. Ultimately, someone has to be accountable. Therefore, assign accountability for developing and maintaining the overall System Design Solution for the project and its multi-level Requirements, Operations, Behavioral, and Physical Domain Solutions. Within each PRODUCT, SUBSYSTEM, and ASSEMBLY PDT, such as an Integrated Product Team (IPT), assign accountability for the ENTITY-Level Design Solution and its Requirements, Operations, Behavioral, and Physical Domain Solutions. Ultimately, the Project Engineer, Lead SE (LSE), and IPT Leads are accountable.

14.4.2 Exit Criteria

Since the SE Process Model is highly *iterative* and subject to development time constraints of the ENTITY, *exit criteria* in general are determined by the *level of maturity* and *risk* assessed at a Critical Design Review (CDR) (Figure 14.9) to commit resources to proceed to the Component Procurement & Development Phase (Figure 12.2). Risk considerations include *level of urgency* and *confidence* in terms of correcting *latent defects*. Refer to Principle 12.3 for a more explicit answer.

14.4.3 SE Process Model: Work Products and QRs

The Wasson SE Process Model supports the development of numerous System/Product Life Cycle phase *work products* – deliverable system or product - and QRs. When applied to the development of the SYSTEM or ENTITY, the SE Process Model produces four categories of work products for each entity: (1) the Requirements Domain Solution, (2) the Operations Domain Solution, (3) the Behavioral Domain Solution, and (4) the Physical Domain Solution.

General examples of work products and QRs include specifications, Specification Tree, verified and validated Models & Simulations (M&S), architectures, analyses, trade studies, technical reports, drawings, verification records, and meeting minutes.

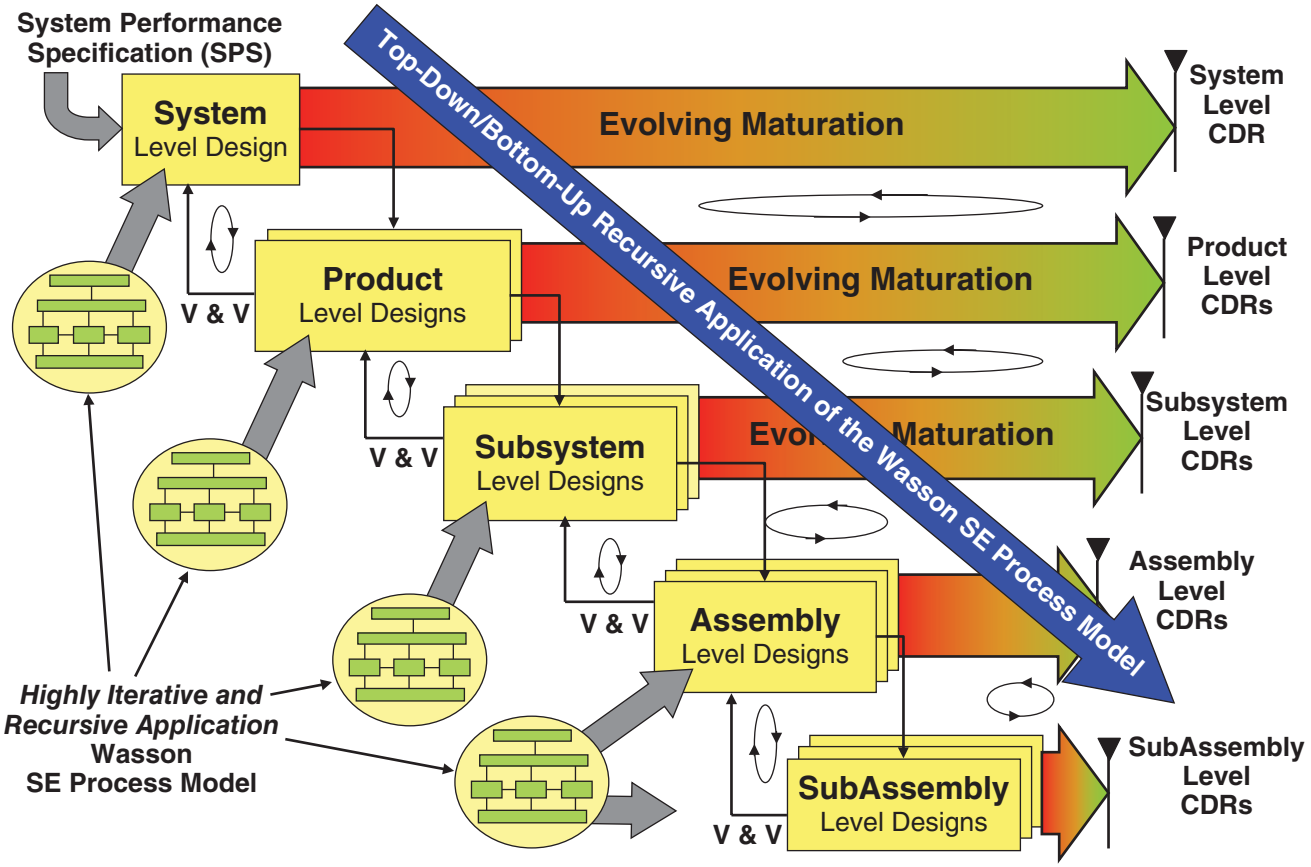


Figure 14.9 Recursive Application of the Wasson SE Process to the Multi-level System Design Process

Author's Note 14.3

SE Process Purpose

People often confuse the purpose of any SE Process. They believe that the SE Process is established to create documentation; this is factually *incorrect* and *misguided*!

The purpose of the SE Process Model is to establish a Problem-Solving–Solution Development methodology to solve problems and produce an *optimal* solution that satisfies contract requirements within technical, technology, cost, schedule, and risk constraints. *Work products* and *QRs* are simply artifacts of the technical decision-making process. They are simply a means to an end, not a destination.

Applying Principle 11.2:

- If you reward people for producing documentation, you get documentation and a System Design Solution that may or may not meet specification requirements.
- If you reward people producing the SE Process Model's Four Domain Solutions, the result should be a SYSTEM/ENTITY that complies with and is traceable to its specification or design requirements supported by documentation artifacts that provide objective evidence of its *integrity* and *validity* of the solution to meet the User's operational needs.

14.5 WASSON SE PROCESS MODEL CHARACTERISTICS

As a problem-solving and solution development methodology applicable to a SYSTEM or ENTITY at any level of abstraction, the Wasson SE Process Model is characterized as being *highly iterative* and *recursive*. To better understand the context of these descriptors, let's explore each one.

14.5.1 Highly Iterative Characteristic

The Wasson SE Process Model, when applied to a specific entity within a level of abstraction, is characterized as *highly iterative* as illustrated earlier in Figure 14.2. Although sequence of steps in the methodology has a workflow progression, each step also has feedback loops that allow a return to preceding steps to reassess decisions when Critical Operational or Technical Issues (COIs/CTIs) issues

are encountered. As a result, the horizontal, left-to-right workflow progression over time through the Four Domain Solutions and their feedback loops (Figure 14.2) illustrate the *highly iterative* characteristic within the System/Entity.

14.5.2 Recursive Characteristic

Principle 14.2

Decision-Making Stability Principle

Mature and stabilize decision-making at higher-levels as soon as practical to enable decision-making at each successive lower levels to mature and stabilize.

Referring to Figure 14.9, observe that the Wasson SE Process Model applies to every level of abstraction. We refer to this as its *recursive* characteristic, meaning the problem-solving–solution development methodology has universal application to any ENTITY within the SYSTEM regardless of level of abstraction.

To better understand how the Wasson SE Process Model applies to System Development, Figure 14.9 shows the V-Model implementation of the SE Design Process. Observe the *recursive characteristic* represented by the SE Process oval icon application to each level of abstraction. As a reminder, the multi-level SE Design Process shown in Figure 14.9 illustrates two key points that require emphasis:

1. In general, each level of the System Development Process matures in time ahead of lower levels (Figure 11.2). It is the LSE's responsibility to mature and bring stability to the decision-making at higher levels to enable successively lower levels of the System Design Solution to *mature* and *stabilize* (Principle 14.2).
2. The multi-level System Design Solution is not considered complete until ENTITIES at all levels of abstraction are mature and have been reviewed for compliance and Engineering best practices and corrective actions completed (Figure 14.9). If a COI/CTI is discovered at any level of abstraction that impacts specification requirements compliance or the design and its interfaces, corrective actions could have a rippling effect *upward* as high as the SYSTEM Level of abstraction.

To illustrate the *highly iterative and recursive* characteristics of the Wasson SE Process Model, let's build on the Four Domain Solutions concept introduced in Figures 11.1 and 11.2. Figure 14.10 provides a starting point. Observe the four alternating gray quadrants annotated with symbols representing interactions among the Four Domain solutions - **R** (Requirements), **0** (Operations), **B** (Behavioral), and **P** (Physical) (ROBP).

Now, let's assume we have the SYSTEM shown in Figure 14.11 consisting of Products 1 and 2. Product 1, a large, complex design, consists of Subsystems 11 and 12; Product 2 consists of Subsystems 21 and 22. We apply the SE Process Model to each ENTITY as illustrated by the ROBP cyclical iterations. Application of the Wasson SE Process Model to each ENTITY continues to lower levels of abstraction until the System Design Solution is mature and

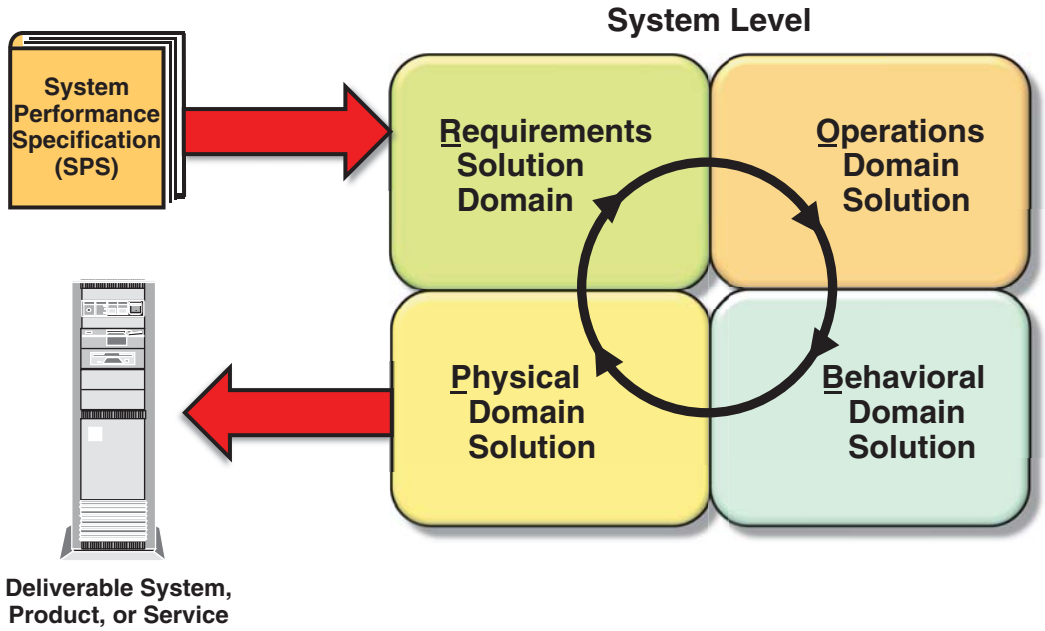


Figure 14.10 Quadrant Representation Symbolizing the Sequential Dependencies of the Four Solution Domains within the Wasson SE Process Model

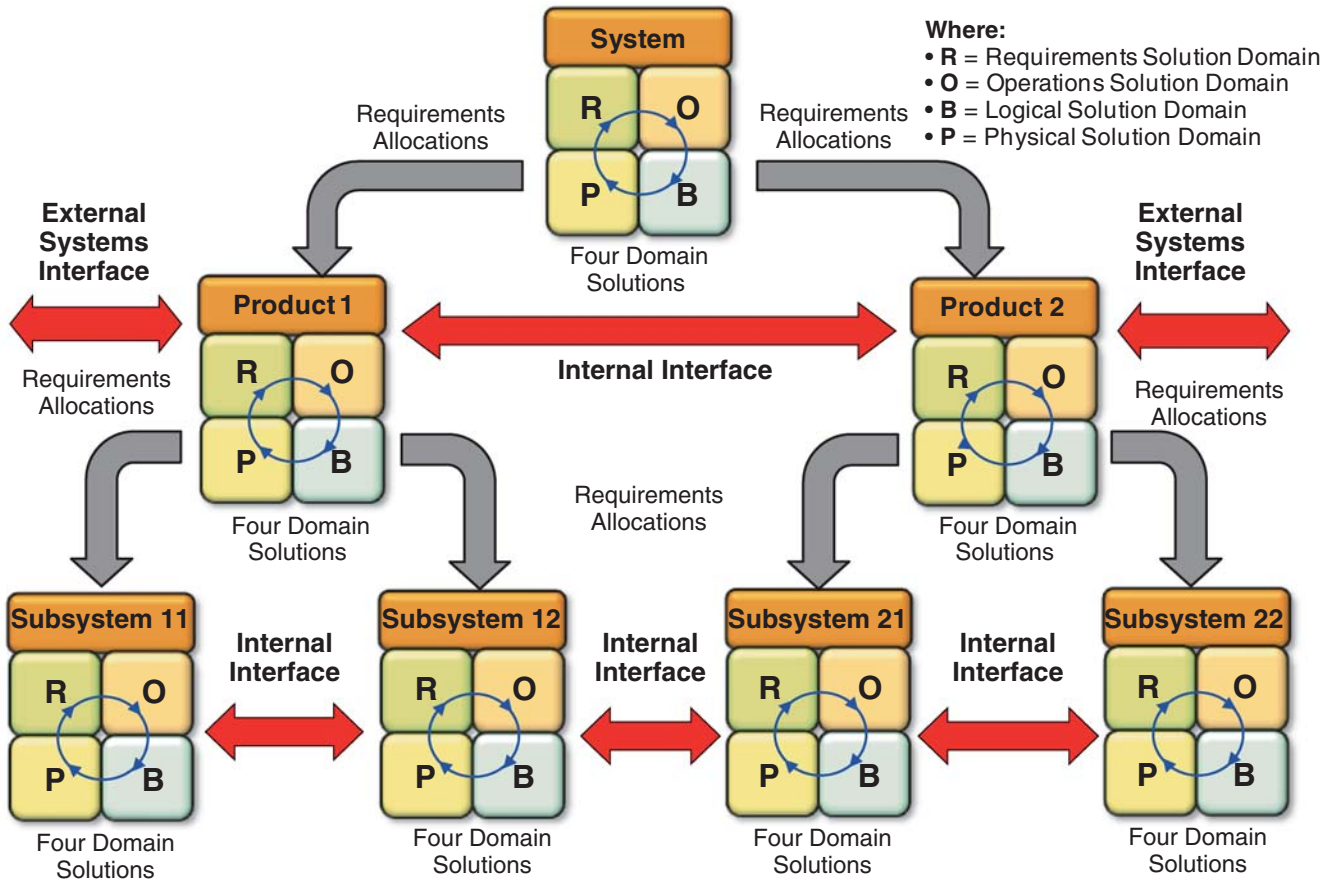


Figure 14.11 Recursive Application of the Wasson SE Process to System Levels of Abstraction and Entities Within Each Level

ready to commit to implementation. Please note that due to space restrictions, External Interfaces to Products 1 and 2 are not show to Subsystems 11 and 22 where the physical interactions actually occur.

Figure 14.12 represents the state of the System Design Solution at completion. Here, we see a multi-level framework that depicts the *horizontal workflow* progression over time. Vertically, the ROBP Domain Solutions are decomposed into various levels of abstraction. Collectively, the framework graphically represents the SYSTEM, which, by definition, is the integration of multiple levels of capabilities to achieve a higher level purpose—*emergence*—that is greater than their individual capabilities.

Leveraging the *highly iterative* and *recursive* characteristics of the Wasson SE Process, System Developers evolve the System Design Solution over time from the SPS into a series of workflow progressions through each level of abstraction until the Developmental Configuration is mature. Figure 14.13 illustrates how the total System Design Solution *iterates* and *evolves* through the Four Domain Solutions culminating with the Critical Design Review (CDR) at the center of the graphic.

Symbolically, the inner loops of the spiral represent *increasing* levels of Engineering design detail at lower levels until the CDR is conducted. Each loop of the spiral culminates in a technical review that serves as a critical *staging* or *control* point for assessing the SYSTEM/ENTITY's status, progress, maturity, and risk for committing resources to initiate the next level of abstraction. Each loop also includes a breakout point to permit cyclical iterations and reconciliation of COIs/CTIs from lower levels.

Author's Note 14.4

Finalization of the System Design Solution

Figure 14.13 reflects the iterations of the SE Design Process shown in Figures 12.4 and 14.9. This characterizes the System Design Process (Figure 12.3) from Contract Award through CDR when the total System Design Solution is approved and released to the Component Procurement and Development Process. However, please note that the System Design Solution *is not contractually finalized* until the First Article SYSTEM or PRODUCT has been integrated, tested, verified, validated (optional), and formally accepted by the Acquirer or User.

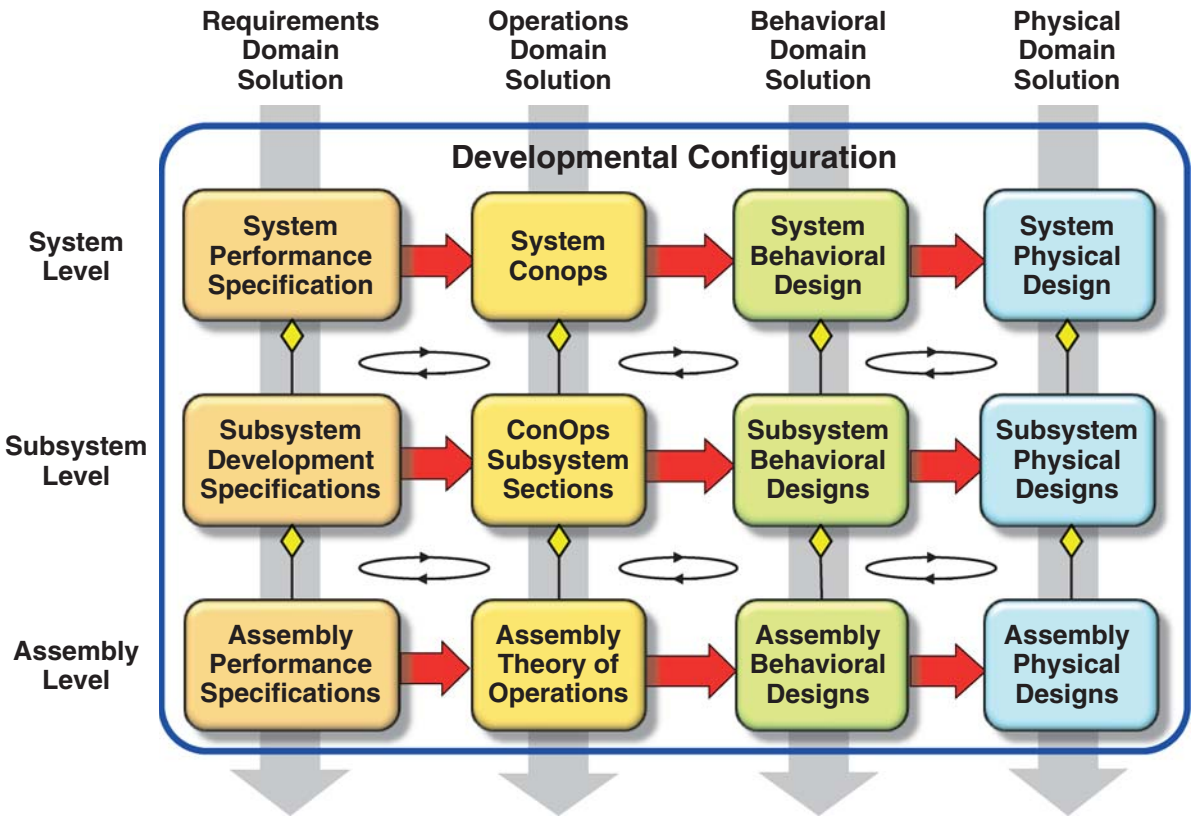


Figure 14.12 System Design Solution Framework Illustrating Integration of the Four Domain Solutions

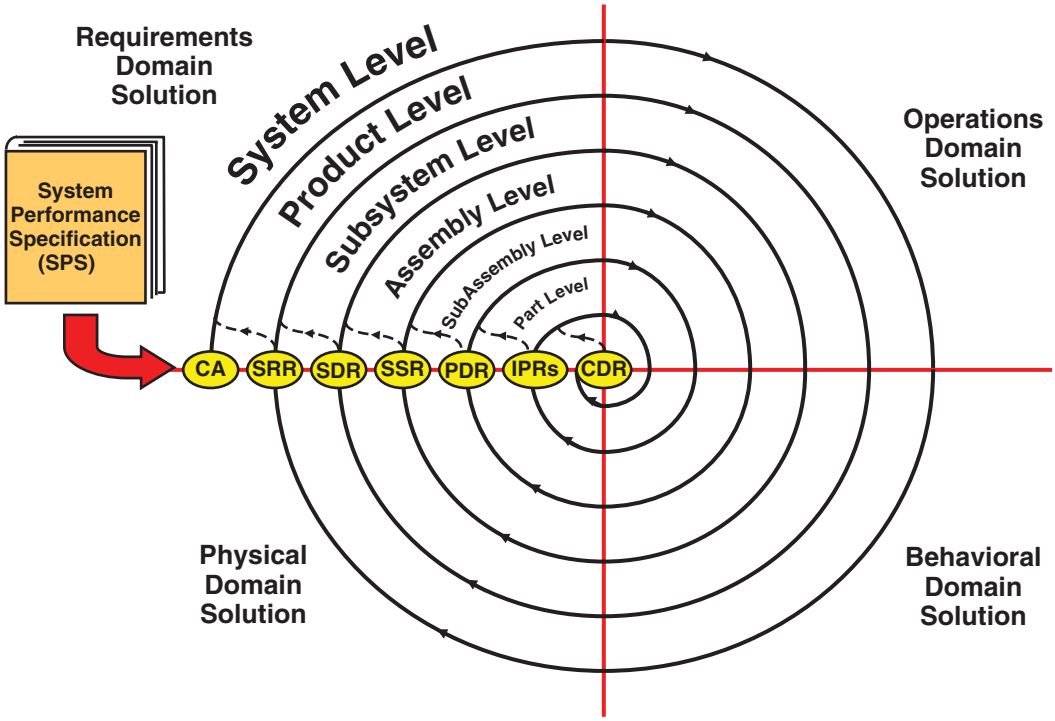


Figure 14.13 The Multi-Level System Design Process Spiral with Breakout Points at Each level

14.6 APPLICATION OF THE WASSON SE PROCESS MODEL

To better understand the application of the Wasson SE Process Model, let's use a simple Tablet Computer System example.

14.6.1 Understand the User's Problem/Opportunity and Solution Spaces

Problem Statement: *If I had a tablet computer that is small and portable, I could research the Internet, send and receive e-mail, and perform work from remote locations.*

Once the Stakeholders are understood, SEs and System Analysts interview and collaborate with the Users. These discussions seek to to understand *how* the User envisions deploying, operating, maintaining, sustaining, retiring, and disposing of a new SYSTEM prior to Request for Proposals (RFPs) or other procurements are released. For each of the System/Product Life Cycle Phases, we identify and document their UCs, expected performance-based outcomes, priorities, estimated frequency, and potential scenarios.

Next, we collaborate the User to formulate a Problem Statement, bound and partition the Problem Space into one or more Solution Spaces (Figure 4.7), develop a System Context Diagram (Figure 8.1) with external interfaces, and delineate what is/is not part of the Tablet Computer System.

14.6.2 Develop the Tablet Computer's Requirements Domain Solution

Based on analysis of the Problem Statement, we use the Context Diagram to clearly delineate (1) the Tablet Computer System context relative to its OPERATING ENVIRONMENT and (2) boundaries concerning what is/is not part of the SYSTEM. We analyze the User UCs and scenarios and derive a basic set of SYSTEM capabilities that we translate into SPS requirements. Examples of the requirements include:

- Physical Characteristics
  - Size—6″ × 9″ × ½″
  - Power—Internal, rechargeable, battery with external 110 vac and 60 Hz AC adapter
  - Display—8″ LED backlit, touch screen, and 1024 × 768 resolution
  - Wireless Communications—802.11a/b/g/n Wi-Fi
  - Cellular Communications—Bluetooth X.X
  - Memory Capacity—16 GB expandable to 64 GB
  - Weight—0.5 lb. max.
- General Use Cases (UCs)
  - Perform web searches
  - Send/receive e-mail
  - Take personal notes
  - Maintain an appointment schedule
  - Determine GPS location
  - Download software updates
  - Download e-books, newspapers, music, and games
  - Conference with external parties in real-time video
  - Photograph and store images; make movies
  - Link real-time video to other Users
  - Create photo albums
  - Create, retrieve, save, and delete text document, presentation, and spreadsheet types of applications
  - Print out documents
- Environmental Constraints

14.6.3 Develop the Tablet Computer's Operations Domain Solution

Based on an analysis of the SPS requirements, we formulate, evaluate, and select an Operational Architecture via an Analysis of Alternatives (AoA) (Chapter 32) from a set of viable candidates. Consider the following example.

Operations Domain Solution Scenario Example

Example 14.2 Assume that the User's mission is to go to a wi-fi hotspot with free Internet accessibility to enjoy coffee and perform research. An initial UC narrative description might read as follows:

Power down the computer, store it in a backpack, drive to a coffee shop, remove computer from backpack, power up computer, access local Wi-Fi, perform research, create and save files, power down and pack up computer, drive home, unpack computer, power up computer, print out files, and read e-mail.

Using content such as Example 14.2 above, we begin to populate a ConOps document (Chapter 6).

14.6.4 Develop the Tablet Computer's Behavioral Domain Solution

Based on the Operations Domain Solution, we need to define how the Tablet Computer System should interact with and respond to the User. For example, the User opens the laptop, presses the Power button, and the computer boots up, initializes, and displays the desktop. If the User wants to search the web, they select an Internet icon, which initiates a web browser application, and so forth. Likewise for other types of applications such as email.

From an SE perspective, we employ tools such as SysML™ to model the interactions between the Actors—computer, User, facility power, facility Wi-Fi, and external SYSTEMS. Observe that the focus here is on *what* has to be accomplished and *how well*, not *how* to design the physical SYSTEM.

We also update the System Design Description (SDD) that describes the Logical/Behavioral Architecture, components, interface details, and performance allocations.

14.6.5 Develop the Tablet Computer's Physical Domain Solution

Given a Behavioral Domain Solution in the form of behavioral interaction models, we formulate, evaluate, and select via an AoA an *optimal* Physical Domain Solution from a set of viable candidate architectures. This includes the Tablet Computer's SYSTEM, SUBSYSTEM, and Physical Architecture; interfaces—mechanical, electrical, data, protocol, and data; and performance allocations such as weight and power.

Next, we map the capabilities identified in the Behavioral Domain Solution's Logical Architecture to the Physical Architecture component(s) shown in Figure 14.7. The mapping can be implemented via a simple spreadsheet or tool.

Next, we allocate and flow down SPS or lower level specification requirements to the physical components and trace requirements back to the User's *source* or *originating* requirements.

Finally, we update the System Design Description (SDD) that describes the Physical Architecture, components, interface details, and performance allocations.

14.6.6 Initiate the Tablet Computer's Decision Support Process

Selection of the Operational, Behavioral, and Physical Domain Solutions requires evaluation and selection via an AoA of a viable set of candidates. SEs and System Analysts, as generalists and generalist–specialists, require System Analysis and Specialty Engineering support from Electrical, Mechanical, and Software Engineering disciplines; Human Factors; Reliability, Maintainability, and Availability (RMA); and Safety to support informed PDT decision making.

Where specialized expertise is required, we initiate the Decision Support Process to analyze, prototype, model, and simulate conceptual designs of the computer. The Decision Support Process provides analytical and performance-based feedback such as system effectiveness, COIs/CTIs, performance values for the computer's specification requirements, and timing. These inputs facilitate SE Process operational and technical decisions that may impact specifications and performance allocations.

14.6.7 Evaluate and Optimize the Tablet Computer's System Design Solution

Principle 14.3

Suboptimization Principle

Avoid degrading or suboptimizing overall SYSTEM or ENTITY technical, cost, or risk performance just to *optimize* a lower level specific aspect.

Since decisions at all levels of abstraction can impact overall SYSTEM performance, SEs need to ensure that the total System Design Solution is "optimized" for the User(s) and their operational needs. Remember that the intent is to *avoid* a condition referred to as *suboptimization* in which a SUBSYSTEM and ASSEMBLY, for example, is optimized at the expense of the overall SYSTEM-Level performance.

For our Tablet Computer System example, we could increase the battery capacity for longer operation but at the expense of User *undesirables* such as increased weight and physical product size without User-*desired* performance improvements.

14.7 THE STRENGTH OF THE WASSON SE PROCESS MODEL

Observe that the Wasson SE Process Model provides a logical, step-wise, problem-solving and solution development model that precludes the requirements "quantum leap" to a point design solution illustrated in Figure 2.3. Each step in the model focuses on *convergent* decision making to achieve specific *outcomes*. This ensures that technical dependency decisions are sequenced properly without taking pre-mature design shortcuts. Overall System Design Solution integration is improved thereby resulting in fewer System Integration, Test, & Evaluation (SITE) incompatibility and interoperability issues. The quantity of latent defects should be reduced leading to delivery on-time and within budget.

In contrast, the traditional, *ad hoc, endless loop* Plug & Chug … SDBTF-DPM Engineering Paradigm wanders around *inefficiently* and *ineffectively* "performing activities." Then, when the system goes into System Integration, Test, & Evaluation (SITE), incompatibility and interoperability issues emerge due to poor design integration. Additionally, the quantity of latent defects is greater, which results in significant cost and schedule overruns to correct.

14.8 CHAPTER SUMMARY

Chapter 14 introduced the Wasson SE Process Model as a problem-solving–solution development methodology that:

- Is characterized by explicitly labeled activities that are easy to understand.
- Applies to any type of business domain—medical, energy, A&D, transportation, and telecommunications.
- Overcomes the fallacies of the *ad hoc* SDBTF–DPM Paradigm

Our discussion described how the Wasson SE Process Model integrates the ROBP Domain Solutions at any level of abstraction into a *highly iterative*, multi-level framework (Figure 14.12). The power of the SE Process Model is its recursive characteristic application to any level of abstraction.

In summary, the Wasson SE Process Model:

1. Is an analytical problem-solving–solution development methodology that can be applied to any type

of User problem, issue, or concern, not just Systems Engineering and Analysis. This encompasses government, healthcare, transportation, energy, and so forth that may not require Engineering *per se*.

2. Is *scalable* to any type or size of SYSTEM or project. In contrast, the *ad hoc* Plug and Chug … SBTF-DPM Paradigm is acknowledged as being *unscalable* to any size project, labor-intensive, subject to chaos, and prone to latent defects such as design flaws, errors, and deficiencies.
3. Applies *iteratively* and *recursively* to any level of abstraction or ENTITY within each level.
4. Includes multiple *control* or *staging points* to *verify* and *validate* decisions prior to commitment to the next stage of SYSTEM/ENTITY design activities—Requirements to Operations, Operations to Behavior, and Behavior to Physical. Breakout points for each activity facilitate corrective actions to previous activities.
5. Is a convergent technical decision-making-centric process that captures decision *artifacts* in the form of *work products* such as specifications and designs. Contrary to myths and misguided perceptions promulgated by *ad hoc* SBTF-DPM Engineering Paradigm Enterprises, the focus is on making timely and informed technical decisions instead of producing documents. Learn to recognize the difference!
6. Establishes the analytical linkages for problem-solving–solution development. Inevitably, schedules may not permit full implementation. Therefore, learn to apply SE knowledge, experience, and insightful wisdom to *tailor* the process by *scaling* the formalities while *preserving* the underlying methodology—Requirements to Operations to Behavioral to Physical. Observe that we said "*tailor* the process by *scaling* the formalities." Tailoring and scaling formalities do not mean taking shortcuts.

14.9 CHAPTER EXERCISES

14.9.1 Level 1: Chapter Knowledge Exercises

1. What is the Wasson SE Process Model?

2. What problems with existing Engineering paradigms is it intended to correct?

3. What are the key elements of the SE Process Model?

4. What are the sequences, interdependencies, and relationships within SE Process Model?

5. What are the steps of the underlying SE Process Model methodology?

6. What is meant by the SE Process Model's *highly iterative* characteristic?

7. What is meant by the SE Process Model's *recursive* characteristic?

14.9.2 Level 2: Chapter Knowledge Application Exercises

Refer to www.wiley.com/go/systemengineeringanalysis2e

14.10 REFERENCES

AFSCM 375-5 (1966), *Systems Engineering Management Procedures*, Wright-Patterson Air Force Base (WPAFB), OH: Air Force Systems Command.

Archer, L. Bruce (1965), *"Systematic Methods for Designers," Design*, London: Council on Industrial Design.

FM-770-78 (1979), *US Army Field Manual: System Engineering Fundamentals*, Washington, DC: Department of Defense (DoD).

IEEE Std 1220™-2005 (2005), *IEEE Standard for the Application and Management of the Systems Engineering Process*, New York: Institute of Electrical and Electronic Engineers (IEEE).

Merriam-Webster (2013), *Merriam-Webster On-Line Dictionary*, www.Merriam-Webster.com, Retrieved on 6/8/13, 2013 from http://www.merriam-webster.com/dictionary/hypothesis.

MIL-STD-499B Draft (1994), *Military Standard: Systems Engineering*, Washington, DC: Department of Defense (DoD).

McCumber, William H. and Crystal Sloan (2002), *Educating Systems Engineers: Encouraging Divergent Thinking*, Rockwood, TN: Eagle Ridge Technologies, Inc.

Wasson, Charles S. (2012a), *System Engineering Competency: The Missing Course in Engineering Education*, American Society for Engineering Education (ASEE) National Conference, San Antonio, TX. Accessed on 6/12/13 from http://www.asee.org/public/conferences/8/papers/3389/view

Wasson, Charles S. (2012b), *Formulation and Development of the Wasson Systems Engineering Process Model*, American Society for Engineering Education (ASEE), Southeast Section Regional Conference, Mississippi State University. Accessed on 5/12/13 from http://se.asee.org/proceedings/ASEE2012/Papers/FP2012was181_589.PDF

15

SYSTEM DEVELOPMENT PROCESS MODELS

Chapter 12 defined the challenge facing Systems Engineers (SEs) translating a User's abstract vision into the physical realization of a system, product, or service. We highlighted the importance of developing an integrated, multi-level System Development Strategy as the solution. This presents a new challenge: *how do SEs plan, organize, and orchestrate implementation of the strategy*?

On initial inspection, traditional management methods say that SEs need to analytically "decompose" the system into smaller parts that can be assigned to multi-discipline teams to accomplish the work. However, you soon learn that traditional management "division of labor" methods of assigning tasks to everyone are *not always efficient and effective*. For example, the concept of "decomposing" a system or product always had such an odd connotation and is not an accurate representation of what SEs actually do. We (1) arbitrarily bound notional Problem Spaces that are often dynamic and partition them into one or more potential Solution Spaces, (2) readjust the boundaries to find the optimal solution, and (3) derive, allocate, and flow down requirements through successive lower levels. Approaches for what works for one system may not necessarily work for the next system. Such is the case for System Development Models.

If you study the spectrum of types of projects, you will find a variety of unique circumstances and opportunities that range from:

- A diverse set of Stakeholders that have *known* or *unknown* requirements that may be stable or dynamically change constantly.
- Exploratory, new, or existing technologies.
- Components that range from simple to highly complex.

So, the question for SEs becomes: *what approach do we employ to deal with these circumstances? Is there a one-size-fits-all approach?*

In response to the first question, over the past several decades, advancements in SE have evolved a series of System Development models to develop systems, products, or services. However, viewing a project organization as a "Development System" has its own levels of performance, effectiveness, and efficiency driven by factors such as technical leadership, culture, education and training, experience, tools, and facilities. Application of some System Development models works well where requirements are generally *well known and stable*, other models apply to situations where requirements are *unknown and possibly unstable*. As a result, a one type of model may be applicable for one project; multiple models may be required for other projects.

This chapter introduces and investigates the primary System Development Models that represent approaches for developing systems, products, or services. Models include the:

- Waterfall Development Model
- V-Model
- Spiral Development Model
- Evolutionary Development Model
- Incremental Development Model
- Agile Development Model

System Engineering Analysis, Design, and Development: Concepts, Principles, and Practices, Second Edition. Charles S. Wasson.

Companion Website: www.wiley.com/go/systemengineeringanalysis2e

Our discussions describe each model, identify how each evolved, highlight flaws, and provide illustrative real-world examples.

You may ask why topics such as these are worthy of discussion in an SE book. *Isn't this project management*? There are two reasons:

- First, SEs need a toolkit of System Development approaches that enables you to leverage a variety of requirements definition, risk, and maturity challenges.
- Second, you need to fully understand each of the models; their origins, attributes, and flaws enable you to select the *right* approach to specific types of System Development challenges.

As an SE, you need to fully understand the history of how some of the models evolved and apply to various types of System Development scenarios.

15.1 DEFINITIONS OF KEY TERMS

- **Agile Development**—A development approach that focuses on quick reaction responses to changing User requirements via a series of incremental product development cycles comprised of short, iterative development cycles referred to as Sprints. Agile Development consists of several supporting terms that include the following:
  - **Daily Scrum**—A daily, 15 minute stand-up meeting in which Agile Development or Scrum Team members account for 3 questions: (1) what did I accomplish the previous workday, (2) what issues remain to be solved, and (3) what do I intend to accomplish today.
  - **Exploration Factor (EF)**—An assessment of a software project or entity's requirements or technology uncertainty or risk on a scale of 1 (low) to 10 (high) (Highsmith 2013).
  - **Feature**—A noteworthy, distinguishing characteristic of a system, product, or service such as its usability, capability, performance, compatibility, interoperability, or non-functional attributes such as size or weight that are perceived by the User as benefits. For example, a key feature of an AC-powered digital clock might be its ability to maintain time during a power loss via an internal battery.
  - **Feature-Boxed**—A term representing a bounded set of User capabilities or features to be delivered without regard to their relative priorities within the set. *Uncompleted* features are returned to a Product Backlog for reassignment to the next or later Sprint. Contrast to Time-Boxed.
  - **Product Backlog**—A repository of User capability or feature requirements that have been prioritized by the User into categories such as mandatory, must-have desired, and nice-to-have.
  - **Product Backlog Burndown Chart**—A graph used to track the reduction in the quantity of User capability requirements registered in a Product Backlog over time.
  - **Product Increment (Build)**—An optional step representing *incremental* deliveries consisting of a prioritized set of User Stories, capability requirements, or features to be delivered within a specific timeframe. A Product Increment is sometimes referred to as a "build" in software development and consists of several Sprints performed in sequence or concurrently.
  - **Product Increment (Build) Backlog**—An optional repository of prioritized User Story or feature requirements assigned to a Product Increment or software "build."
  - **Scrum**—A structured framework governing the development and incremental delivery of User Stories or requirements based on a defined theory and set of practices and rules. Refer to Schwaber and Sutherland (2011).
  - **Sprint**—A short, iterative, and incremental development cycle within a Scrum that varies from 7 to 14 days in which Agile Development or Scrum Teams analyze a set of prioritized User Stories or requirements assigned from a Product Backlog; incrementally design, code, and test the software; conduct daily stand-up reviews or *scrums* to assess progress, status, and issues from the previous day's activities and plans for the day; review and approve Product Releases; demo each release to the Customer or User; and proceed to the next Product Increment Sprint.
  - **Sprint Backlog**—A repository of User Story, capability, or feature requirements that have been assigned to a Sprint. Reduction in the *quantity* of Sprint Backlog capability or feature requirements is tracked via a Spring Backlog Burndown metric.
  - **Sprint Iteration Cycle**—A sequence of *Iterative* and *Incremental* Development (IID) activities required to transform a set of User capabilities or features assigned to a Sprint into one or more deliverable work product releases.
  - **Sprint Release**—The delivery of a Sprint work product that complies with one or more User capability or feature requirements registered in a Sprint Backlog.

  - **Time-Boxed**—A term representing time constraints placed on a Sprint to deliver a set of prioritized User capabilities or features based on that order. Uncompleted capabilities or features with a lesser priority are returned to the Product Increment Backlog for assignment to the next or a later Sprint. Contrast with Feature-Boxed.
  - **User Story**—A brief statement of a Customer's, User's, or End User's role-based business or mission operational need typically using a syntax such as "As a (role), we need (what) … so that (reason) …." (Cohn, 2008) A constraining criterion for the scope of a User Story is that it must be accomplished within the time-box of a Sprint. Two or more User Stories may comprise an Epic.
  - **Theme**—Customer or User's vision or overarching objective for a class of systems, products, or services to satisfy a variety of operational needs.
  - **Epic**—Customer or User's abstraction representing a collection of User Stories.
- **Evolutionary Development Strategy**—A development strategy used to develop "a system in builds, but differs from the Incremental Strategy in acknowledging that the user need is not fully understood and all requirements cannot be defined up front. In this strategy, user needs and system requirements are partially defined up front, then are refined in each succeeding build" (MIL-STD-498, p. 37).
- **Full Operational Capability (FOC)**—The attainment of a planned set of *incremental* development capabilities to meet User operational needs and performance-based objectives within a specified timeframe. Contextually, an FOC could refer to a specific system, product, or service or the deployment of a Systems or Products to all elements of an organization.
- **Grand Design Development Strategy**—A development strategy that is "essentially a once-through, do-each-step-once' strategy. Simplistically: determine user needs, define requirements, design the system, implement the system, test, fix, and deliver" (MIL-STD-498, p. 37).
- **Incremental Approach**—"Determines user needs and defines the overall architecture, but then delivers the system in a series of increments ("software builds"). The first build incorporates a part of the total planned capabilities, the next build adds more capabilities, and so on, until the entire system is complete" (DAU 2012, p. B-205).
- **Initial Operational Capability (IOC)**—"In general, attained when some units and/or organizations in the force structure scheduled to receive a system have received it and have the ability to employ and maintain it. The specifics for any particular system IOC are defined in that system's Capability Development Document (CDD) and Capability Production Document (CPD)" (DAU 2012, p. B-107).
- **Proof of Concept**—Refer to Chapter 12's *Definition of Key Terms*.
- **Proof of Principle**—Refer to Chapter 12's *Definition of Key Terms*.
- **Proof of Technology**—Refer to Chapter 12's *Definition of Key Terms*.
- **Quality Function Deployment (QFD)**—An integrated business, marketing, and technical strategy that focuses on the Customer—Stakeholder Users and End Users—to capture and understand their operational needs, priorities, preferences, and desires as a basis for specifying the capability and performance characteristics of a system, product, or service that will deliver an expected level of *customer satisfaction*.
- **Spiral Approach**—"A risk-driven controlled prototyping approach that develops prototypes early in the development process to specifically address risk areas followed by assessment of prototyping results and further determination of risk areas to prototype. Areas that are prototyped frequently include user requirements and algorithm performance. Prototyping continues until high risk areas are resolved and mitigated to an acceptable level" (DAU 2012, p. B-205).
- **V-Model**—A graphical model that illustrates the time-based, multi-level strategy for (1) decomposing specification requirements, (2) procuring and developing physical components, and (3) integrating, testing, evaluating, and verifying each set of integrated components. The V-Model is one of the most commonly used System Development models. Forsberg and Mooz (1991, p. 4), the originators, refer to it as the "Vee" Model.
- **Waterfall Approach**—"Development activities are performed in order, with possibly minor overlap, but with little or no iteration between activities. User needs are determined, requirements are defined, and the full system is designed, built, and tested for ultimate delivery at one point in time. A document-driven approach best suited for highly precedented systems with stable requirements" (DAU 2012, p. B-205).

15.2 INTRODUCTION TO THE SYSTEM DEVELOPMENT MODELS

Heuristic 15.1 System Development Improvements

System Development Improvements requires four objectives: (1) *understanding* where you have been, (2) *knowing*

where you are now, (3) *deciding* where you want to go, and (4) *leveraging* your strengths to navigate and avoid obstacles.

The heuristic above serves as a compass heading for the structure of this chapter. People often think of new models as innovations. In general, this is true; however, new models encompass more than just an innovation; they correct deficiencies and mistakes of the past.

Most System Development models are *unprecedented* in the sense that each new model attempts to *correct* current *shortcomings* in System Development approaches while building on *key strengths* of previous models. For example, many people believe that Agile Development is a new creation in terms of its title and implementation. However, Agile Development builds on selective attributes of prior models such as Iterative and Incremental (IID), Evolutionary Development, V-Model, and Spiral Development discussed in this chapter. So, it is important to understand the background of these models as a basis for understanding and appreciating the state and application of current models to system or product development. Let's begin with a brief background that drives the need for System Development models.

Author's Note 15.1

System Development Approaches versus Models

Industry, academia, and government often refer to System Development *approaches* versus System Development *models*. What is the difference?

- An *approach* is a high-level, hypothesis-based, conceptual roadmap that describes sequences of actions to be performed for achieving a desired outcome.
- A *model* implements an *approach* using a logical and/or mathematical transfer function for transforming a set of Acceptable Inputs into a specific set of Acceptable Outputs–Outcomes (Figures 3.1, 3.2, and 20.4).

Since our discussion employs graphical models that impart an underlying approach, we will use both terms.

Chapter 15 addresses the primary types of models used for System Development. The models include the (1) Waterfall Development Model, (2) V-Model, (3) Evolutionary Development Model, (4) Iterative and Incremental Development (IID) Model, (5) Spiral Development Model, and (6) Agile Development Model. On inspection, these models may appear to be peers; however, they have specific System Development applications. In fact, System Development may require application of several different types of these models depending on the system or product, level of risk, and so forth.

In the mid- to late 1980s, as system complexity increased and became more software intensive, industry, academia, and government began to recognize that SE decision-making required tighter coupling in multi-discipline integration. Until that time, SE and Software Engineering restricted their—"stovepipe"—methodologies and processes to focus on their own activities. Boehm (2006, p. 8), one of today's Software Engineering thought leaders, made the following observation:

> "Traditionally (and even recently for some forms of agile methods), systems and software development processes were recipes for standalone "stovepipe" systems with high risks of *inadequate interoperability* with other stovepipe systems. Experience has shown that such collections of stovepipe systems cause *unacceptable delays* in service, *uncoordinated and conflicting* plans, *ineffective or dangerous* decisions, and *inability to cope* with rapid change."
>
> —(Wasson emphasis)

The history of System Development models exposes more than the need for multi-discipline SE. The models must be supported by common, multi-discipline problem-solving and solution development methodologies such as the Wasson SE Process Model (Figure 14.1).

Our first topic begins with the Waterfall Development Model, which is no longer used by most organizations due to its flawed approach. You may ask: *if it is no longer used, why is it addressed here, especially since education and knowledge are driven by learning new concepts and approaches*? This is true. However, education and knowledge require understanding how SE as a discipline evolved to its present state based on (1) *why* projects and systems fail and (2) fallacies of previous methods to avoid repeating those mistakes in the future. To understand where you need to go, you have to understand where you have been (Heuristic 15.1). The Waterfall Model serves as a mini case study for better understanding of how System Development Models have evolved.

15.3 WATERFALL DEVELOPMENT STRATEGY AND MODEL

The Waterfall Development Model shown in Figure 15.1 represents one of the early attempts to characterize software development in terms of a model. The model's name is based on its cascading appearance resembling a waterfall.

Bennington (1983, p. 1) states that the Waterfall Model was introduced in a presentation made "in Washington, D.C., in June 1956 at a symposium on advanced programming methods for digital computers, sponsored by the Navy Mathematical Computing Advisory Panel and the Office of Naval Research."

Some literature *incorrectly* attributes the Waterfall Model to Dr. Winston W. Royce. The fact is that Royce (1970) made a presentation at IEEE WESCON in August 1970 entitled "Managing Large Software Development Projects." In his

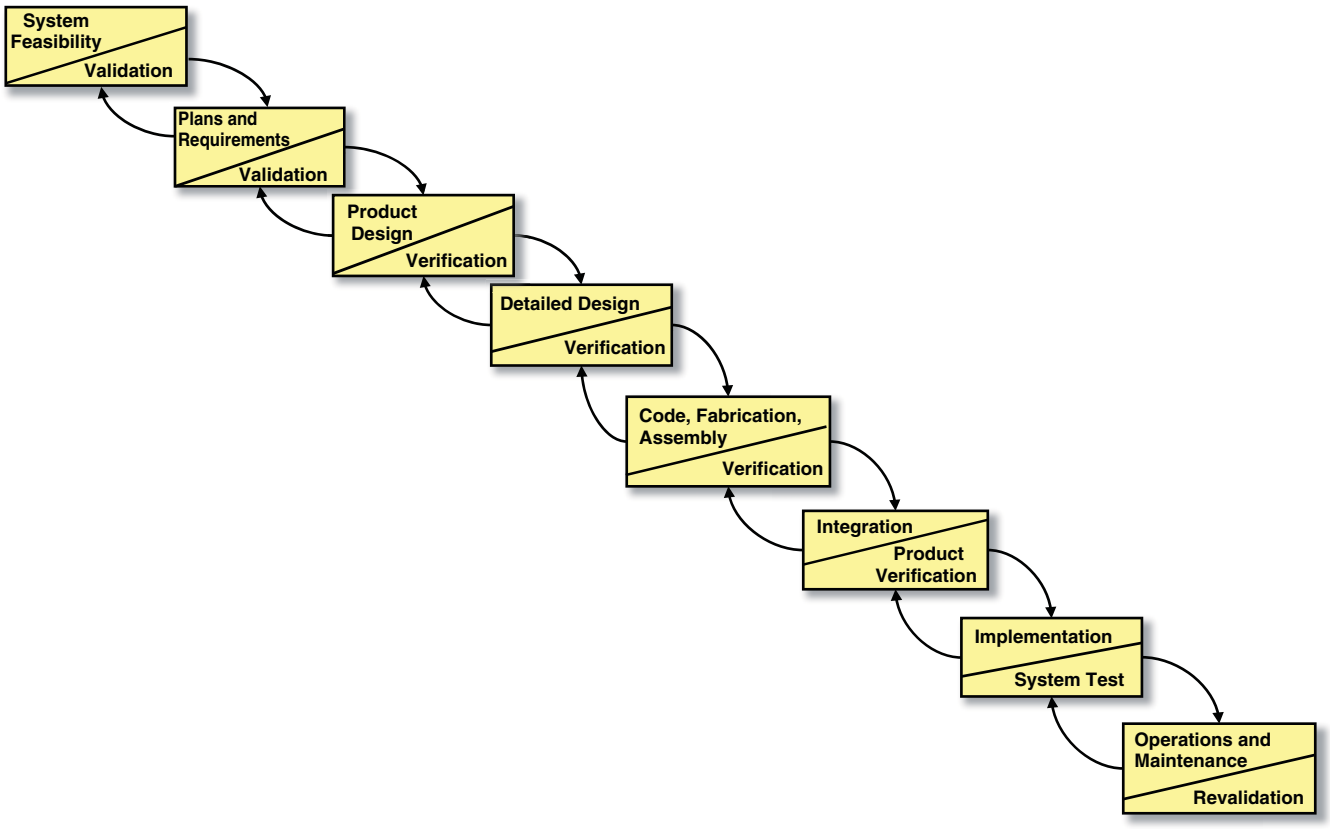


Figure 15.1 The Waterfall Model of the Software Lifecycle (Boehm, 1988, Figure 1, p. 62 adaptation of Royce (1970, Figure 3, p. 330)). Used with Permission

presentation, he presented a paper that later became associated with the Waterfall Model as the basis for pointing out the fallacies of the current software development paradigm (Royce 1970, pp. 328–329).

Unfortunately, the Waterfall Model is often *incorrectly* attributed to Royce as an innovation. It was simply an illustrative graphic representing Royce's observations of the state of SW development at the time. Larman and Basili (2003) note that Royce (1970) simply presented his opinions concerning government contracting at that time—Requirements Analysis → Design → Development in rigid sequence. In fact, Royce's approach was to "do it twice." His presentation eventually led to the designation of the graphic as the "Waterfall." Although, the term *waterfall* is never mentioned in Royce's paper, his name became incorrectly identified with the Waterfall. He simply presented his views as an observer, not its innovator (Larman and Basili 2003, p. 3).

The Waterfall Model is often described as a lockstep, *sequentially linear process* with essentially no overlaps or opportunities to return to a previous step. Therein lies its primary shortcoming in addition to other issues.

Boehm (1985, p. 63) described the Waterfall Model as "a highly influential 1970 refinement of the stagewise model." It provided two primary enhancements to the stagewise model:

- Recognition of the feedback loops between stages and a guideline to confine the feedback loops to successive stages to minimize the expensive rework involved in feedback across many stages
- An initial incorporation of prototyping in the software life cycle, via a "build-it-twice" step running in parallel with requirements analysis and design:

> "The waterfall model's approach helped eliminate many difficulties previously encountered on software projects. … A primary source of difficulty with the waterfall model has been its emphasis on fully elaborated documents as completion criteria for early requirements and design phases."

In comparing the Waterfall Model with IID, Larman and Basili (2003, p. 2) reference a paper by Walker Royce indicating his father, Dr. Winston W. Royce, described the

Waterfall Model as "… the simplest description, but that it would not work for all but the most straightforward projects …"

Referral Refer to Wikipedia (2014), Winston (1970), and Boehm (1988) for additional information about the history of the Waterfall Model.

15.4 "V" SYSTEM DEVELOPMENT STRATEGY AND MODEL

Chapter 12 established a sequence of highly interdependent processes that depict the workflow for transforming the User requirements into a System Design Solution. In general, the strategy provides an end-to-end framework for:

- Verifying compliance with the System Performance Specification (SPS) requirements
- Validating that the deliverable system, product, or service satisfies the User's validated operational needs

Figure 12.3 illustrates a "U"-shaped strategy for (1) decomposing and refining an abstract Problem Space into multiple levels of Solution Space systems and (2) integrating the multi-level systems. This infrastructure provides the foundation for the V-Model. Before we get into that discussion, let's shift to a higher-level discussion.

We introduced Figure 12.2 as a generalized System Development Process Workflow that included these primary processes. For the V-Model, the System Design, Component Procurement and Development, and System Integration, Test, and Evaluation (SITE) Processes form the foundation of Figure 12.3. Specifically:

- The System Design Process is implemented by the downward-facing left arrow.
- The Component Procurement and Development Process is represented at the bottom center by the Procure/Develop Components box.
- The SITE Process is implemented by the upward-facing arrow on the right side.

Despite the "U-shaped" pattern of the structure in Figure 12.3, these activities occur over time resulting in a V-shaped structure; Figure 15.2 illustrates a representation V-Model.

The V-Model was originally described in a formal presentation by Forsberg and Mooz (1991). By appearance, some

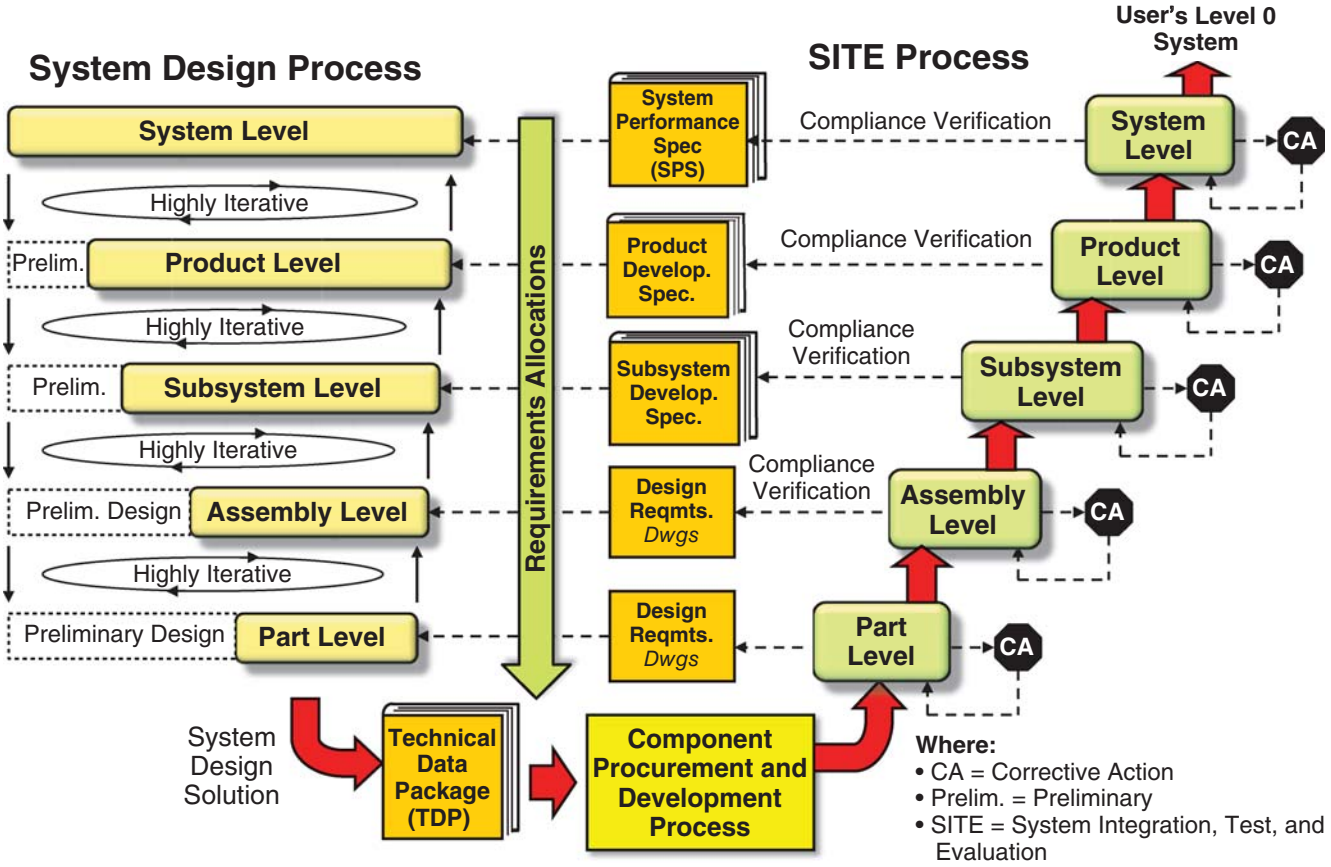


Figure 15.2 Wasson Adaptation of the V-Model for System Development

will say the V-Model is another instance of the Waterfall Model. Given its downward, stepwise form, it does infer a Waterfall shape. However, the graphic is *highly iterative*. To illustrate this point, let's describe how the V-Model is applied to implementing the System Development Process Workflow shown in Figure 12.2.

15.4.1 System Development Process: V-Model Implementation

The V-Model is a *highly iterative*, pseudo time-based, stage–gate model. In general, workflow progresses from left to right over time. However, the *highly iterative* characteristic of the SE Design Process (Chapter 14) and verification corrective action aspects of SITE allow loopbacks to a preceding step in time. Corrective actions may require reevaluation of lower-level Critical Operational Issues (COIs) or Critical Technical Issues (CTIs), specifications, designs, and components. So, as the corrective actions are implemented over time, workflow progresses from left to right to delivery and acceptance of the system, product, or service.

To better understand how the V-Model is applied, let's link key activities of the System Development in terms of the processes they represent.

15.4.2 System Design Strategy: V-Model Implementation

The System Design Process requires derivation, allocation, and flow down - decomposition - of high-level SPS requirements (System Developers Problem Space) into multiple levels of abstraction (Solution Spaces) over time to form an overall System Design Solution as shown in Figure 15.3.

Observe the initially staggered parallel tracks of development activities shown in Figure 14.9. The duration of each track extends to the right culminating in a CDR. The track duration symbology represents maturing design solutions at each level of abstraction. As stated in Principle 12.1, it is the Lead SE's (LSE's) job to bring *maturity* and *stability* to higher levels to enable lower levels to finalize their design decisions. Referring to Principle 12.3 concerning when a System Design Solution is complete, the track durations represent that the design solution at any level is subject to change if COIs or CTIs are discovered at lower levels.

Also observe the oval icons down the lower left side of Figure 14.9. The icons signify the application of the SE Process (Chapter 14) to each level of abstraction and Entities within each level. As a result, the SE Process creates the

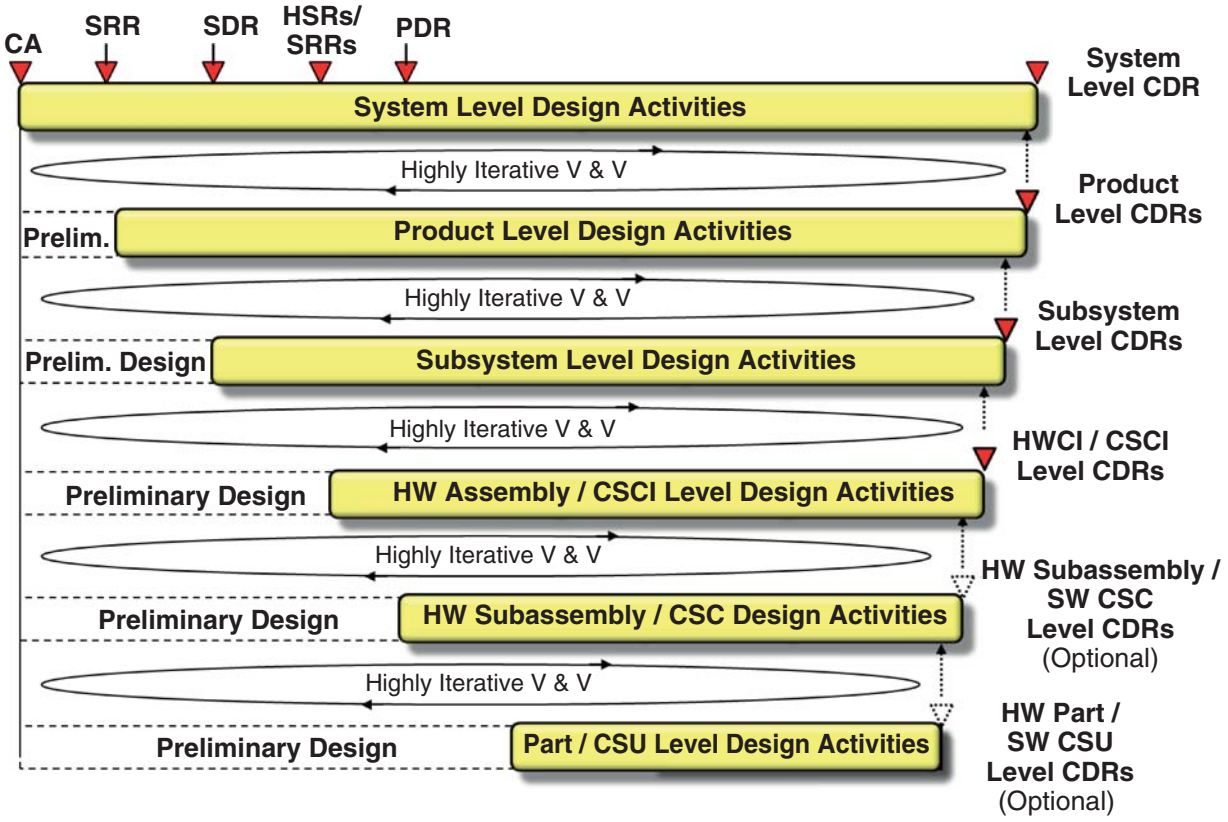


Figure 15.3 Wasson Adaptation of the V-Model System Design Activities

Requirements, Operations, Behavioral, and Physical Domain solutions for each ENTITY.

Now, consider the "V" shown in Figure 13.4. Enterprises and Engineers for simple convenience take liberties to depict the V-Model as slanted, mirror image facing bars connected at the bottom. *Uninformed* managers and Engineers often *erroneously* portray the left side of the V-Model as the Waterfall Model (Figure 15.1). That is factually incorrect! As illustrated in Figures 15.2 and 15.3, the V-Model *iterates* over time between each of its levels of abstraction to enable resolution of most, if not all, of the COIs and CTIs up until its CDRs.

15.4.3 Component Procurement and Development Process: V-Model Implementation

The V-Model Component Procurement and Development Process implements the TDP based on component *Make* versus *Buy* versus *Buy–Modify* decisions made during the System Design Process. Key activities include:

1. Procurement/acquisition of external hardware and software components from subcontractors or vendors.
2. Fabrication, Assembly, Integration, and Test (FAIT) as well as verification of hardware and software components developed in-house from new or legacy designs.
3. Receiving inspection of procured components accompanied by vendor Certificates of Compliance (CofCs).
4. Verified component transfers to the SITE Process as they become available.

15.4.4 SITE Strategy: V-Model Implementation

The V-Model SITE Process consists of key activities required to integrate and verify components *bottom-up* into a working SYSTEM that fully complies with its SPS and lower-level specification requirements as shown in Figure 15.2. Key activities include:

1. Integration of components into an ENTITY such as a SUBASSEMBLY, ASSEMBLY, SUBSYSTEM, and PRODUCT as shown in Figures 12.6 and 15.2.
2. Verification of integrated sets of ENTITIES at various levels of abstraction for compliance to its specification requirements using Test Cases (TCs) and Test Procedures.
3. Data capture and authentication of verification compliance results (Chapter 13).
4. Assessment of verification results for compliance with specification requirement(s) as a condition for achievement of each requirement.

15.4.5 V-Model Application and Software Development: Criticisms

Software development activities have attempted to develop SW components in lockstep with the V-Model and lament that it is simply *unworkable*. They contend that there are much better methods such as Agile Development discussed later in this chapter that can produce better software. So, *what is the problem with software development using the V-Model*?

The V-Model is essentially a hardware-driven, *stage–gate* paradigm process. For example, due to the expense and risk of developing physical Systems, the V-Model includes three high-level *stages* and *control gates* at the completion of the System Design, Component Procurement and Development, and SITE Processes (Figures 12.2 and 12.3). Everything in a System Development Project moves in lockstep through these stages. To illustrate this point, observe the example work products shown in Table 15.1 that are typically required for each of the System Development Strategy processes.

You may ask: *why is the V-Model staged*? The answer is risk. Each process stage assesses the status, progress, maturity, and risk of the evolving System Design Solution and its technical compliance, technology, budget, and schedule risk of committing resources before proceeding to the next process. So, the V-Model is staged as a risk mitigation process with a focus on multi-level integration of physical ENTITIES that comprise the System.

Engineering best practices say: (1) create designs that comply with specification requirements; (2) prototype technical risk areas; (3) document design decision artifacts via meeting notes, drawings, and parts lists for review and approval; (4) Fabrication, Assembly, Integration, and Test (FAIT) components in accordance with the drawings; and (5) test and verify levels of integrated components. PARTS have to be formed or machined, printed circuit boards have to be populated and soldered, complex wiring bundles have to be created, and all have to be integrated into very tight spaces. The implementation of these physical work products consumes valuable resources—time and money—and therefore poses some level of risk. *Did you observe the underlying hardware decision-making stage–gate flow of the V-Model*?

If we investigate the evolution of a hardware or software component, especially prototypes, consider the following:

- When a machined component is discovered to have a *defect*—design flaw, tolerance problem, material composition problem, or workmanship problem—it may have to be *reworked*, *scrapped*, or *replaced*.
- When a printed circuit board assembly fails compliance testing, has a long-lead item component failure, or has a workmanship problem, it is either *reworked*, awaits a long delivery of a replacement part, or *scrapped*.

TABLE 15.1 Example Hardware and Software Work Products in Lockstep: V-Model Process Implementation

Domain	System Design Process CDR Stage–Gate Example Work Products	Component Procurement and Development Process Inspection and Verification Stage–Gate Example Work Products	SITE SVT State–Gate Example Work Products
System	• SPS • System architecture • System Test Cases (TCs) • System test procedures • SYSTEM level drawings • System Design Descriptions (SDDs)		• System test and compliance results
Hardware	• Entity specifications • HW specifications • HW drawings • HW design descriptions • HW TCs • HW test procedures	• Material and Component Acquisition • Component Fabrication, Assembly, Integration, & Test (FAIT)	• HW test and compliance results • Verified and Integrated HW and SW components
Software	• Software specs • Software designs • SW TCs • SW test procedures • Software Test Descriptions (STDs)	• Computer SW Unit (CSU) Code and Unit Test (CUT) • Code Walkthroughs	• SW test and compliance results • Unit and component level testing and integration

CDR, Critical Design Review; SVT, System Verification Test.

- When software code fails or a defect is discovered, it is *recoded* and *compiled* "on the spot" assuming that there are no structural or logic issues.

So, when software developers suggest that the V-Model is *unworkable*, hardware paradigms respond with: *what is their problem*? They design software, prototype risk areas, code components, and integrate, test, and verify them just like hardware. Therein lies the problem.

Software does perform these activities. However, there are several reasons for the software response:

1. Whereas hardware laboratory prototypes are typically *non-deliverable* and *throwaway*, prototype algorithms and code are *reusable* and can be reused from legacy deigns, refined, verified, and delivered.
2. If the System Design Process duration is 8 months, *hypothetically*, is it really necessary to "park" prototype software code and wait until the completion of the scheduled SYSTEM Level CDR to begin coding the deliverable software as part of the Component Procurement and Development Process? There should be better alternatives.
3. Hardware and software, as work products of human decision-making and collaboration, inherently have *latent defects* such as design flaws, errors, and deficiencies, due to late discovery during the SITE Process. Development tools, V&V, and technical reviews facilitate defect discovery and reduction prior to the SITE Process.

Software developers tend to reject V-Model mandates for all System Development. The V-Model's stages *unwittingly* postpone coding of deliverable software components until the Component Procurement and Development Process where later discovery of *latent defects* increases risk resulting in schedule and costs impacts.

Since the late 1980s, software developers have led the way in innovating models that enable them to work more *efficiently* and *effectively*. The models and methods are more *conducive* to the nature of their environments—designing, coding and unit testing, integrating, testing, and verifying software. If you ask software developers today what model they would use, a common response is Spiral Development or Agile Development discussed later in this chapter.

To properly understand software development criticism of the V-Model and interest in Spiral and Agile Development, let's defer our continuation of this discussion until we introduce these development models.

Heading 15.1

The preceding V-Model discussion provided a general foundation in how systems, products, or services are developed from a project and technical perspective. System Development models have two contexts:

- **Context #1**—Models multi-discipline SEs employ to *transform* an abstract Stakeholder Problem Space into the physical realization of a system, product, or service to address that need. These models include the V-Model, Spiral Model, Agile Development Model, and Wasson SE Process Model (Figure 14.1).
- **Context #2**—Models multi-discipline SEs employ to enable Stakeholders to deal with *affordability* and *market-driven issues* such as the Iterative and Incremental Development (IID) Model and Evolutionary Development Model.

Since the IID and Evolutionary Development Models have influences on the Spiral and the Agile Development Models, let's investigate these in that sequence.

15.5 SPIRAL DEVELOPMENT STRATEGY AND MODEL

Because of inherent flaws in the Evolutionary Development Model, coupled with a lack of understanding, maturity, and risk in system requirements up front, Dr. Barry Boehm introduced the Spiral Development Model illustrated in Figure 15.4.

Although Dr. Boehm's original paper was published in 1985, Larman and Basili (2003) observe that most citations refer to the 1986 version (Larman and Basili 2003, p. 6).

Spiral Development employs a series of *highly iterative* development activities whereby the deliverable work product of each activity may not be the deliverable system. Instead, the evolving set of *knowledge* and subsequent *system requirements* that lead to the development of a deliverable SYSTEM or PRODUCT contribute to the maturing System Design Solution. The knowledge base evolves via *proof of concept* or *proof of technology* demonstrations to a level of maturity worthy of (1) introduction to the marketplace and (2) production investments from an acceptable risk perspective.

DSMC (2001) described Spiral Development as follows:

> "The spiral approach also develops and delivers a system in builds, but differs from the Incremental Approach by acknowledging that the user need is not fully formed at the beginning of development, so that all requirements are not initially defined. The initial build delivers a system based on the requirements, as they are known at the time development is initiated, and then succeeding builds are delivered that meet additional requirements as they become known. (Additional needs are usually identified and requirements defined as a result of user experience with the initial build)."

The development spiral consists of four quadrants as shown in Figure 15.4:

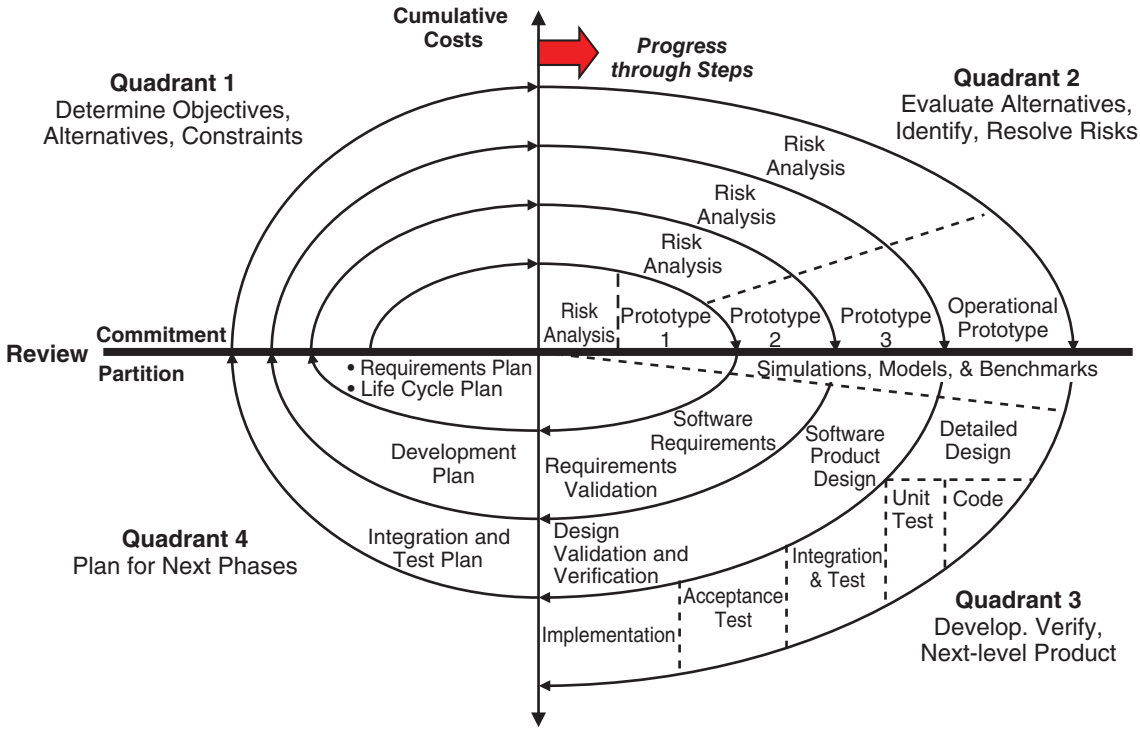


Figure 15.4 Spiral Development Model (Source: Boehm (1988), Figure 2, p. 62) Used with permission.

- **Quadrant 1:** Determine objectives, alternatives, and constraints.
- **Quadrant 2:** Evaluate alternatives, identify, and resolve risks.
- **Quadrant 3:** Develop and verify next-level product.
- **Quadrant 4:** Plan next phases.

Although Spiral Development originates from software development, the concept is equally applicable to systems, hardware, and training, for example. Consider the following example.

Example 15.1 **Discovery of Unprecedented System Requirements**

Assume that an Enterprise has an operational need for a commercial product that does not exist—an *unprecedented* SYSTEM. Requirements for the SYSTEM may be *unknown* or science may lack definitive characterizations of the OPERATING ENVIRONMENT. So, how do you solve the issue?

The answer may consist of a series of projects using Spiral Development. Each project develops prototype system that results in a set of specification requirements. Each set of project specification requirements evolves until the requirements finally mature and lead to the unprecedented system.

Given this example of *undefined* requirements, let's investigate how Spiral Development can be applied to driving out requirements for a system, product, or service that can be developed.

15.5.1 Quadrant 1: Determine Objectives, Alternatives, and Constraints

Activities performed in this quadrant include the following:

- Establish an understanding of the SYSTEM or PRODUCT objectives—namely, performance, functionality, and ability to accommodate change (Boehm 1988, p. 65).
- Investigate implementation alternatives—namely, design, reuse, procure, and procure/modify.
- Investigate constraints imposed on the alternatives—namely, technology, cost, schedule, support, and risk.

Once the SYSTEM or PRODUCT's objectives, alternatives, and constraints are understood, development proceeds to Quadrant 2 (Evaluate Alternatives, Identify, and Resolve Risks).

15.5.2 Quadrant 2: Evaluate Alternatives, Identify, and Resolve Risks

Engineering activities performed in this quadrant select an alternative approach that best satisfies technical, technology, cost, schedule, support, and risk constraints. The focus here is on *risk mitigation*. Each alternative is investigated and prototyped to reduce the risk associated with the development decisions. Boehm (1988, p. 65) describes these activities as follows:

> "… This may involve prototyping, simulation, benchmarking, reference checking, administering user questionnaires, analytic modeling, or combinations of these and other risk resolution techniques."

The outcome of the evaluation determines the next course of action. If COIs/CTIs such as performance and interoperability (i.e., external and internal) risks remain, more detailed prototyping may need to be added before progressing to the next quadrant. Boehm (1988, p. 65) adds that if the alternative chosen is "operationally useful and robust enough to serve as a low-risk base for future product evolution, the subsequent risk-driven steps would be the evolving series of evolutionary prototypes going toward the right (hand side of the graphic) … the option of writing specifications would be addressed but not exercised." This brings us to Quadrant 3.

15.5.3 Quadrant 3: Develop and Verify Next-Level Product

If a determination is made that the previous prototyping efforts have resolved the COIs/CTIs, activities to develop and verify the next-level product are performed. As a result, a basic approach such as the V-Model or other model may be employed. If appropriate, *incremental development* approaches may also be applicable.

15.5.4 Quadrant 4: Plan Next Phases

The Spiral Development Model has one characteristic that is common to all models—the need for advanced technical planning and multi-discipline reviews at critical *staging* or *control* points. Each cycle of the spiral culminates with a technical review that assesses the status, progress, maturity, merits, and risk of development efforts to date; resolves COIs/CTIs; and reviews plans and identifies COIs/CTIs to be resolved for the next iteration of the spiral.

Subsequent implementations of the spiral may involve lower-level spirals that follow the same quadrant paths and decision considerations.

Referral For a more detailed description of the Spiral Development Model, refer to Boehm (1988) or any of his textbooks.

15.6 ITERATIVE AND INCREMENTAL DEVELOPMENT MODEL

Sometimes, the development of SYSTEMS and PRODUCTS is constrained by a variety of reasons such as:

- Availability of resources (expertise)
- Lack of availability of interfacing systems
- Evolving interfaces
- Lack of funding resources
- Technology risk
- Logistical support

When confronted with these constraints, the Users, the Acquirer, and the System Developer may be confronted with formulating an IID Strategy. IID has origins that are traceable back to the 1930s to Walter Shewhart at Bell Labs as a method for quality improvement (Larman and Basili 2003, p. 2).

The strategy may require establishing an Initial Operational Capability (IOC) followed by a series of Incremental Development "builds" that enhance and refine a SYSTEM's or PRODUCT's capabilities to achieve an FOC by some future date. Figure 15.5 illustrates how Incremental Development is phased.

The DAU (2012, p. B-205) characterizes Incremental Development as follows:

> "The incremental approach determines user needs and defines the overall architecture, but then delivers the system in a series of increments - i.e., software builds. The first *build* incorporates a part of the total planned capabilities, the next build adds more capabilities, and so on, until the entire system is complete."

Boehm (1981, p. 41) characterized Incremental Development as simply "a refinement of the build-it-twice full prototype approach and of the level by level, top-down approach."

15.6.1 Implementation

Current implementation of the Incremental Development Model requires that a sound "build" strategy be established "up front." As each build cycle is initiated, development teams establish unique system requirements for each build—either by separate specification or by delineated portions of the SPS. Each build is designed, developed, integrated, tested, and verified via a *series* of overlapping development models such as the V-Model example illustrated in Figure 15.6.

Referral For additional information concerning the history of IID, refer to Larman and Basili (2003).

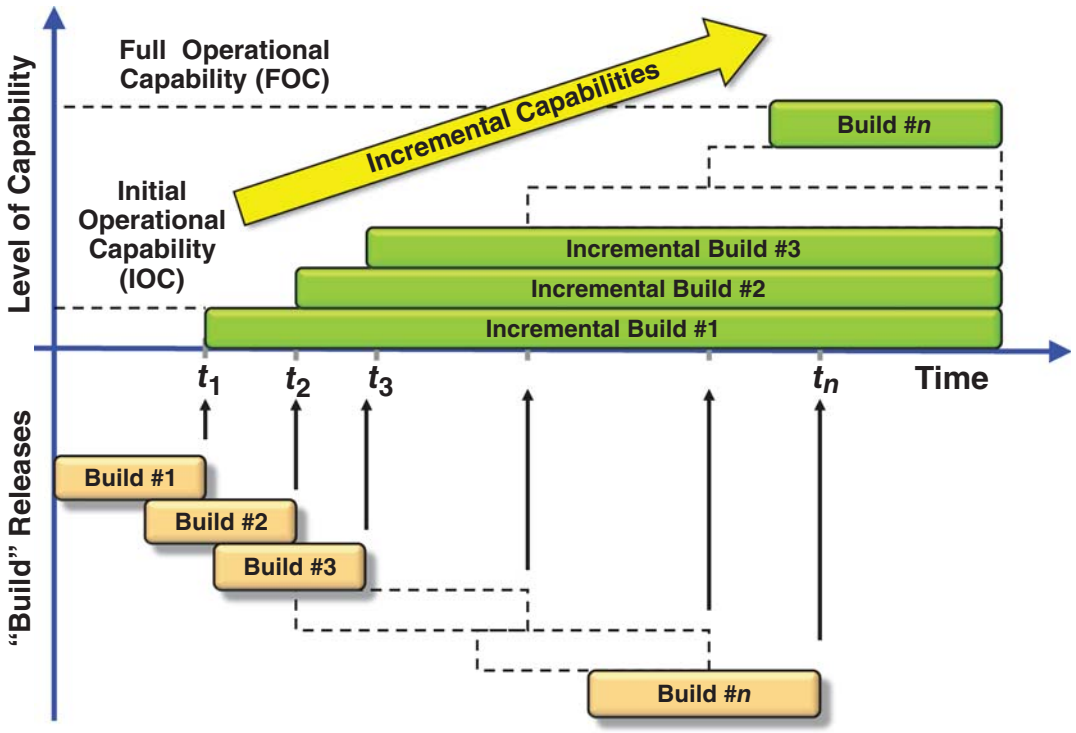


Figure 15.5 Incremental Development Model Concept

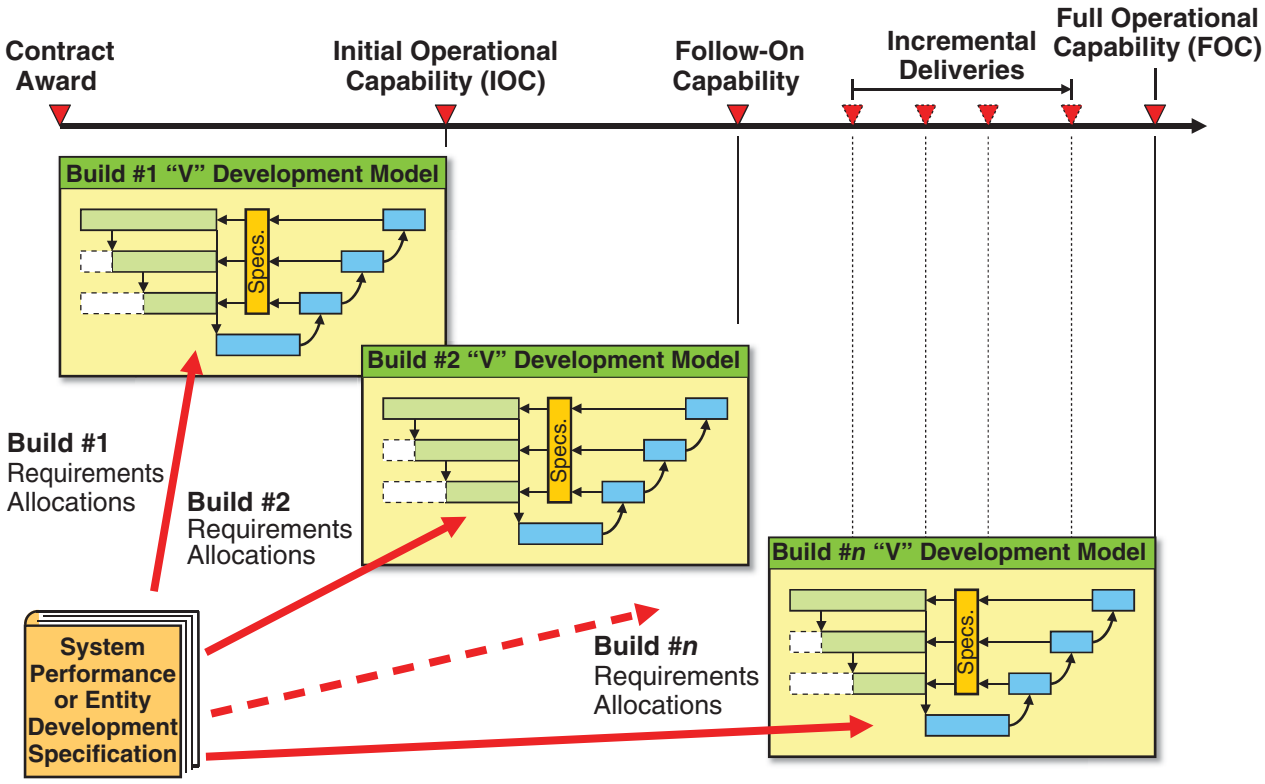


Figure 15.6 Application of the V-Model to Incremental Development

15.6.2 The Challenge for Systems Engineering (SE)

When implementing the Incremental Development Model approach, SEs, in collaboration with other disciplines, must:

1. Using the SPS, thoroughly analyze and partition the set of capabilities into "builds."
2. Schedule the "builds" over time to reflect User priorities such as capability gaps, degree of urgency, available resources, and schedules.
3. Flow down and allocate the requirements—System to PRODUCTS to SUBSYSTEMS to ASSEMBLIES.

Incremental "builds" may include integrating newer components into the system and *upgrading* existing components. The challenge for SEs is to determine how to establish and partition the initial set of capabilities and integrate other capabilities over time without disrupting existing system operations or degrading performance. Intensive interface analysis is required to ensure the "build" integration occurs in the proper sequence and the support tools are available.

15.7 EVOLUTIONARY DEVELOPMENT STRATEGY AND MODEL

Boehm (1988, p. 63) states that the Evolutionary Development Model is based on the premise that "stages consist of expanding increments of an operational software product, with the directions of evolution being determined by operational experience." This conception is based on an evolutionary strategy of a system or product development through a series of Preplanned Product Improvement (P3I) releases.

Evolutionary development provides a potential solution for Acquirers, Users, and System Developers to evolve a System Design Solution over time as requirements are refined. As discussed in Chapter 20, some SYSTEMS or PRODUCTS are single-use items; others are longer-term, multi-application items. For some mission and system applications, you generally know at system acquisition what the requirements are. In other applications, you may only be able to define a few "up-front" objectives and capabilities. Over time, the fielded SYSTEM/PRODUCT requires new capabilities as Enterprise Problem/Opportunity Spaces evolve due to competitive or security threats.

Some SYSTEMS, such as computers, become obsolete in a very short period of time and are retired and disposed. From a business perspective, the cost to upgrade and maintain the devices becomes prohibitive; for example, the marginal utility and Return on Investment (ROI) of upgrading an existing computer's hardware and software *versus* buying a new computer.

In contrast, some Users, driven by decreasing budgets and slow changes in the external environments, may use systems

and products far beyond their original intended service lives. Consider the following example.

Example 15.2 **B-52 Stratofortress Service Life**

The US Air Force B-52 aircraft is postured to achieve a service life well into the 21st century (Global Security—B-52 Stratofortress Service Life, 2011). During its lifetime, the aircraft's systems and missions have evolved from their IOC when the aircraft was first introduced in 1955 to the present via capability upgrades (Global Security—B-52 History, 2011). The projected service life span far exceeds what the aircraft's innovators envisioned.

15.7.1 Fallacies of the Evolutionary Development Model

Conceptually, the Evolutionary Development Model may be suited for some applications; however, it too has its fallacies. Boehm notes the following points:

> Fallacy 1: Evolutionary Development is "generally difficult to delineate from the old code-and-fix model, whose spaghetti code and lack of planning were the initial motivation for the waterfall model." (Boehm 1988, p. 63)
>
> Fallacy 2: Evolutionary Development is "also based on the often-unrealistic assumption that the user's operational system will be flexible enough to accommodate unplanned evolution paths." (Boehm 1988, p. 63)

Regarding Fallacy 2, Boehm (1988, p. 63) states that "This assumption is unjustified in three primary circumstances:

1. Circumstance in which several independently evolved applications must subsequently be closely integrated.
2. Information sclerosis cases, in which temporary work-around for software deficiencies increasingly solidify into unchangeable constraints on evolution, …
3. Bridging situations, in which the new software is incrementally replacing a large existing system. If the system is poorly modularized, it is difficult to provide a good sequence of 'bridges' between old software and the expanding increments of new software."

One of the challenges of any type of *iterative* development such as Spiral Development is the need for convergence and closure to a System Design Solution. That is, avoid the attributes associated with the ad hoc, *endless loop* SDBTF-DPM Engineering Paradigm that never seems to come to completion. This brings us to the Agile Development Model.

15.8 AGILE DEVELOPMENT STRATEGY AND MODEL

Principle 15.1 **Agile Incremental Product Releases Principle**

Incrementally develop and deliver Just-in-Time (JIT) product releases using short iterative development cycles called Sprints based on the value and priority of each to the User.

15.8.1 What Is Agile Development?

When people think of something as "agile," the connotation is of a human with the *flexibility* to make decisions and do whatever they please "just to get the job done." The reality is that everyone working on a project simply cannot do whatever they please. The result would be *chaos* and *bedlam*. However, the intent of the term stems from a need for an Enterprise or project to provide the *capability* and *flexibility* to *rapidly respond* to customer requirements that are often dynamic. In this context, consider the following definition:

- **Agile Development**—A highly *iterative*, methodology-based development process employed by a team to (1) respond to an environment of *changing* or *evolving* Customer, User, or marketplace requirements and priorities; (2) collaborate with the Customers to understand, clarify, and prioritize operational needs; (3) instill daily team member interactions and accountability; and (4) incrementally produce and deliver *high-value–high-return* system, product, or service releases JIT to meet those needs.

Author's Note 15.2 **Customers versus Users**

Observe the choice of terms, *Customer* and *User*. Agile software developers typically refer to the *Customer*. However, the term *Customer* is an abstract term that has numerous connotations. In SE, we differentiate in terms of roles—*User*, *End User*, and *System Acquirer* as the User's acquisition and technical representative. This will become a very important point of distinction in a later discussion in this section.

Author's Note 15.3 **Product Releases Context**

Note the operative term *release*. A *release* could have two contexts:

- Context #1—Internal distribution within the project
- Context #2—External distribution to existing customers such as an urgent security update due to a new computer virus

15.8.2 Why Does System, HW, or SW Development Need to Be "Agile"?

In 2004, IBM started conducting biannual surveys of "CEOs and public sector leaders in every part of the world to gauge their perspective on emerging trends and issues" (IBM 2012, p. 3). The 2012 IBM survey, for example, posed the following question:

> "How are Chief Executive Officers (CEOs) responding to the complexity of increasingly interconnected organizations, markets, societies and governments – what we call the connected economy?"
>
> —(IBM 2012, p. 13)

Since 2004, Market Factors were Priority #1 among CEOs every year through 2010. In contrast, Technology Factors progressed from Priority #6 in 2004 to #3 in 2006, 2008, and 2010. In the 2012 survey, Technology Factors replaced Market Factors, which fell to Priority #3 (IBM 2012, p. 13).

Recognizing that the focus of the IBM study was on the *connected economy*, there is an important lesson to be learned here. Observe the phrase "… responding to the complexity of…." Enterprises that plan to be in business in the years ahead have to shift their System Development paradigms in response to Customer, marketplace, and competition forces such as time to market. Otherwise, they may not survive or will be devoured by their competitors. *Responsiveness to change* is an integral part of the Agile Development methodology.

15.8.3 Agile Development Origins

Many people erroneously believe that the concept of Agile Development is a new innovation and originated in software development. This is *partially true* and has contributed to many popular *misconceptions*. The reality is that the Agile Software Development title is the convolution of two sources:

1. A 1991 Agile Manufacturing study by Nagel and Dove at the Lehigh University Iacocca Institute
2. The "packaging" (Highsmith 2013) of a series of *iterative* and *incremental* software development frameworks, especially from the 1980s through the 1990s, that had evolved since the 1960s and earlier

Let's explore each of these contributing origins.

15.8.4 1991 Agile Manufacturing Study

The concept and naming of Agile Development was influenced by a Lehigh University Iacocca Institute Study commissioned by the US Office of the Secretary of Defense Manufacturing Technology (MANTECH) Program. Results of the study were published by Nagel and Dove (1991) in a two-volume report entitled *21st Century Manufacturing Enterprise Strategy: An Industry-Led View* (Volume 1) and *Infrastructure* (Volume 2). Within the report, "Crossing the threshold: A Vision of the Agile Manufacturing Enterprise" establishes the vision for an *agile* manufacturing culture. The significance of this concept and the *agile* name provided the foundational name for the Agile Software Development as a philosophy.

15.8.5 The Origins of Agile Software Development

From an SW perspective, the "Agile" attributes originate from early models related to the evolution of IID methods. Larman and Basili (2003, p. 2) indicate that IID methods trace back to the 1950s. They also note that Agile Development practices existed in NASA's Project Mercury from the 1960s. The Mercury Project, which was "time-boxed," was similar to Extreme Programming practice. Software developers employed a "test-first" development approach in which they planned and wrote tests prior to each "micro-increment."

During the 1980s and 1990s, software development approaches continued to advance earlier IID methods with a focus on prototyping software as a risk mitigation method. Boehm's (1988) Spiral Model is an example.

In the 1990s, various thought leaders created a variety of software development prototyping approaches. Examples included Adaptive Software Development, Extreme Programming, Crystal Light, and Scrum. Many of these leaders authored books on these software development approaches.

Highsmith (2013) observes that beginning in the late 1990s, many of these innovators began to recognize similarities in their approaches and began a series of dialogs seeking ways to leverage their work. After several preliminary meetings, the group decided to convene a meeting in 2001 to create a *new philosophy* for software development. Seventeen representatives from organizations that created software concepts such as "Extreme Programming, SCRUM, DSDM, Adaptive Software Development, Crystal, Feature-Driven Development, Pragmatic Programming, and others sympathetic to the need for an alternative to documentation driven, heavyweight software development processes …" participated in the meeting (Agile Manifesto 2001a).

One of the decision outcomes from the meeting was a *framework* that became known as the Manifesto for Agile Software Development. As a new philosophy for software development, the Manifesto states:

> "We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

- **Individuals and interactions** over processes and tools.
- **Working software** over comprehensive documentation.
- **Customer collaboration** over contract negotiation.
- **Responding to change** over following a plan.

> That is, while there is value in the items on the right (plain text), we value the items on the left (Boldface) more."
>
> —(Webpage—Manifesto for Agile Software Development 2001c)

On inspection, most people instinctively—and incorrectly—interpret the paradox in each of the four statements above as being *mutually exclusive*. Highsmith (2013) acknowledges that this has been a communications challenge. However, reread the statements. First, observe the word "over." He adds that the statements *do not* state that Individuals and Interactions are *more important* than Processes and Tools. The statements are simply presented as paradoxical choices. Given a choice, which would you choose?

- (Competent) Individuals and interactions or (mediocre) processes and tools?
- Working software or comprehensive documentation?
- Customer collaboration or contract negotiation?
- Responding to change or following a plan?

The reality is that System Development projects require some level of each of the paradoxical choices. However, regardless of the development process used—V-Model, Spiral, and Agile, Customers expect to collaborate with developers who know how to *think* and *respond* to their needs without being overwhelmed by rigid plans, processes, and documentation practices.

Another key aspect of the meeting was the identification of a set of key principles by the participants. These became known as the Twelve Principles of Agile Software (Agile Manifesto, 2001b).

With the new philosophy for software development, the decision challenge became: *what shall we name it*? Highsmith (2013) says that the team identified a number of descriptive terms on individual notecards and placed them on a wall for collaborative review and discussion. During the collaboration, several of the team members that were familiar with Nagel and Dove's (1991) concept of Agile Manufacturing characterized it as one of the best descriptors of the approach expressed by the proposed software manifesto. As a result, the new *philosophy* became known as the Agile Manifesto for Software Development.

Given this background on the origins of Agile Development, let's investigate how Agile Software Development is performed. As an overview, the intent here is to provide an overview of the fundamental concepts of Agile Development.

Referral You are encouraged to read the works of authors listed at the Manifesto for Agile Software Development Web site (Agile Manifesto 2001c).

15.8.6 Agile Software Development Overview

The thrust of Agile Software Development focuses on delivering *incremental* releases of a system, product, or service that provide the *highest value–highest return* and priority to the Customer or User. Traditional System Development tasks focus on an *end deliverable* based on tasks that are often measured in weeks or months. In contrast, Agile Software Development focuses on short development cycles of *incremental releases* measured in days, not weeks, months, or years.

Whereas traditional V-Model and Spiral Development Model employing Earned Value Management (EVM) methods have task milestones that may be spaced a few weeks to a few months apart that lead to *inefficiencies*, daily Agile meetings referred to as Daily Scrums ensure a disciplined focus on *convergence* and *closure* for accomplishing specific performance-based results. Some may view this as micromanaging System Development; others view it improving the *efficiency* similar to Design for Six Sigma (DFSS) in driving out unnecessary documentation, steps, and waste.

Conceptually, if you focus and measure progress based on (1) accountability for daily task outcomes as opposed to weeks or months and (2) deliver *high-value–high-return* releases to the Customer, you should be more *efficient* and *effective* in achieving the same end result—a deliverable system, product, or service on time and within budget with acceptable risk.

Author's Note 15.4

Accountability, Efficiency, and Effectiveness

Note that we said *conceptually*. This last statement sounds great on paper but has serious contextual implications and ramifications for various types of marketplaces, business domains, and contracting environments.

To understand how Agile Software Development is performed, let's begin with an overview description followed by some of the details. Figure 15.7 provides a generalized example for the overview.

Agile Software Development begins with *collaboration* between the software developers and their Customers. The collaboration results in a series of User Stories, which are brief statements that express an operational need or requirement that is unique to each Customer's role. Software developers analyze the User Stories and follow up with the Customers to:

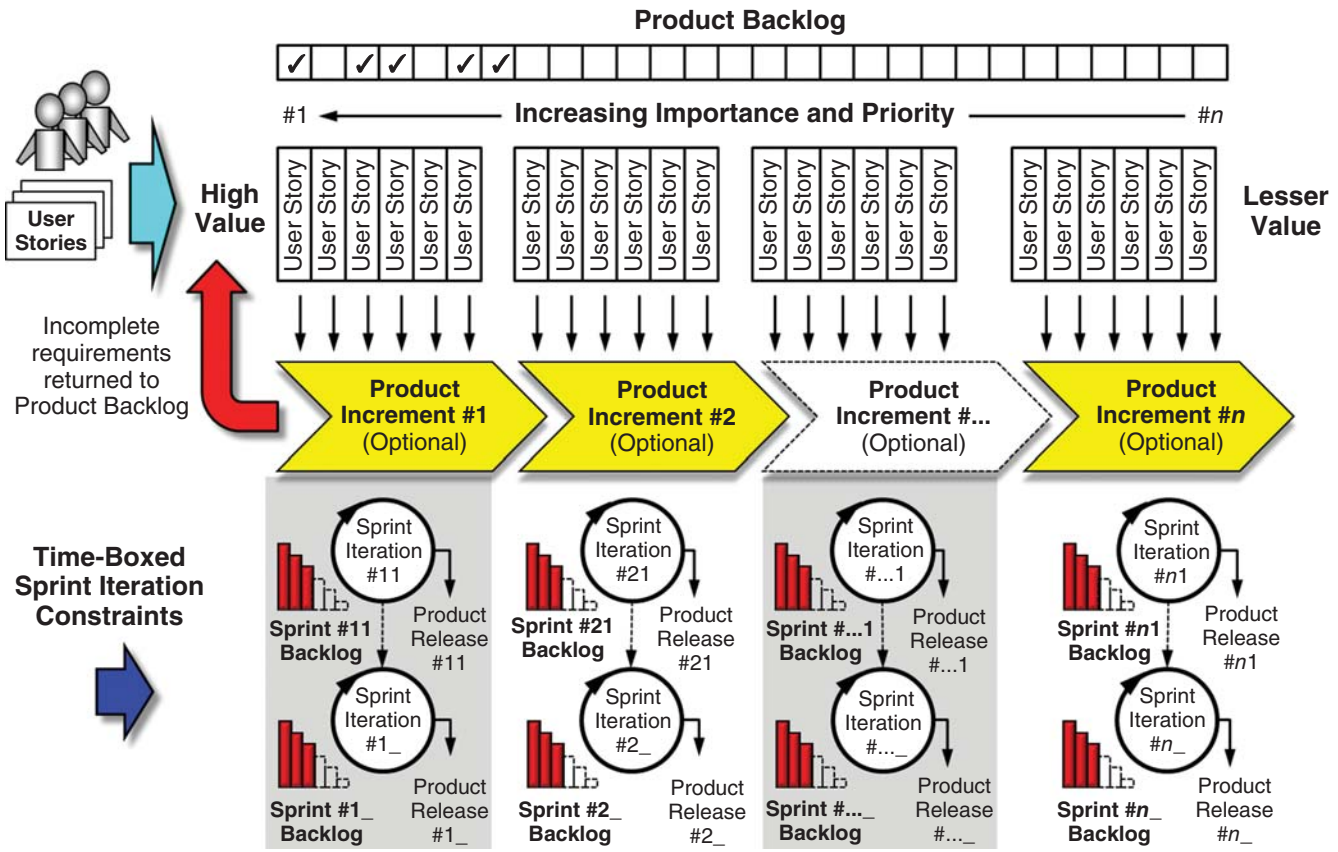


Figure 15.7 Agile Product Development Cycle (Scrum)

- Clarify their understanding of the operational needs and semantics.
- Establish Customer priorities in terms of high-value needs and delivery sequences.

The set of requirements is then *prioritized* and reviewed collaboratively with the Customer to finalize the set of priorities. The prioritized repository is referred to as the Product Backlog.

As a performance tracking tool, Customer requirements in the Product Backlog are tracked with a graphic referred to as a Burndown Chart such as the example shown in Figure 15.8. The state of the Product Backlog represents a repository of prioritized requirements remaining to be implemented via a series of Product Increment Development Cycles or Scrums.

15.8.7 Themes, Epics, and User Stories

Principle 15.2

User Story Authoring Principle

Every User Story should be written by the User in their own words from the perspective of a specific role-based use of a system, product, or service.

Principle 15.3

User Story Composition Principle

Each User Story consists of two parts: (1) a Statement of Need and (2) Conditions of Satisfaction (COS) (Cohn 2008).

Principle 15.4

User Story Priority Principle

Express each User Story in terms of the originator's role, need, value, and priority to their Enterprise or mission.

Agile Software Development employs terms such as themes, epics, and User Stories to characterize collaboration with their Customers. There are also differing views by Enterprises and developers as to which is relevant. User Stories tend to be the primary preference. To better understand the scope of each term, let's begin with User Stories and then relate User Stories to themes and epics.

Each User Story consists of two parts:

- **Statement of Need**—Customer or User's personal Statement of Need

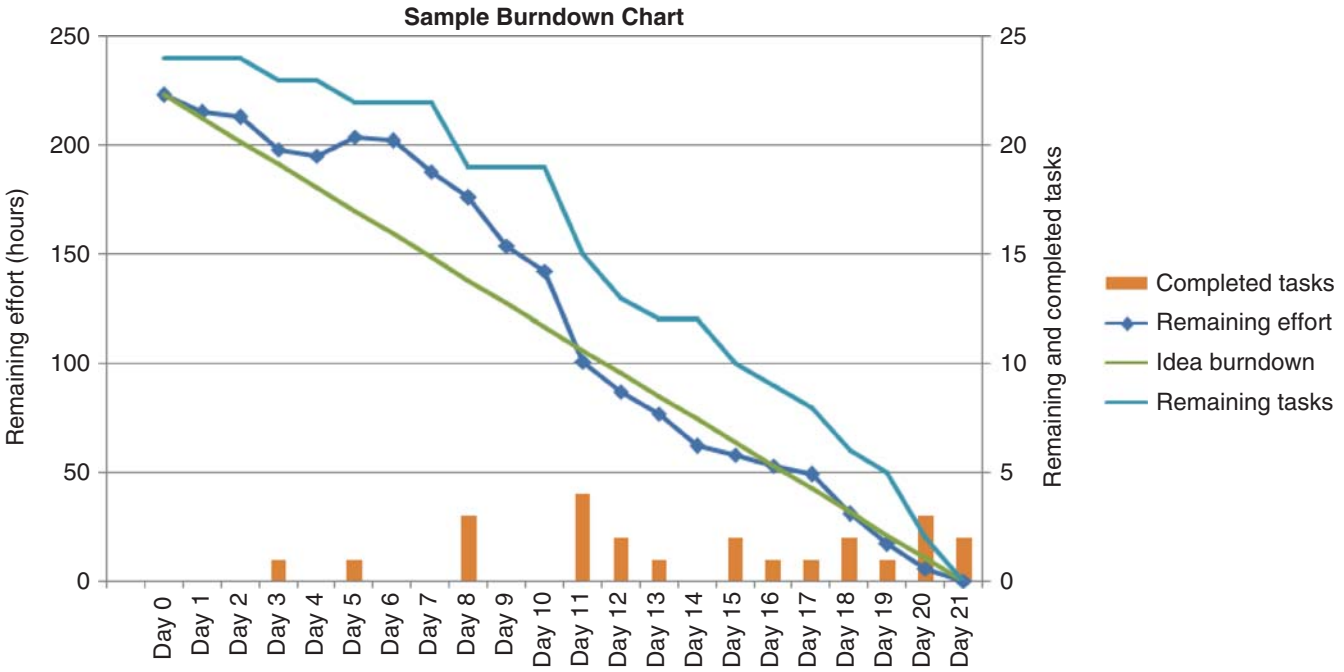


Figure 15.8 Product Burndown Chart Example (Source: Straub (2009), Wikimedia Commons—Public Domain Use)

- **COS** (Cohn 2008)—How the Customer or User intends to validate the deliverable product as satisfying the Statement of Need

Typically, each part of a User's Story is handwritten by the Customer or User on the front of an index card or sticky note that can be easily pinned or stuck to a wall. One side of the card or note expresses the operational need; the obverse side expresses the validation of the need. Since a key aspect of Agile Development is Customer or User determination of high-value–high-return needs, the wall exercise enables moving the notes or cards around. Let's address each of these two parts beginning with the Statement of Need.

15.8.7.1 Statement of Need

Principle 15.5

User Story Syntax

Format each User Story using an "As a <type of user>, I want <some goal> so that <some reason>" syntax (Cohn, 2008).

Cohn (2008) suggests using a standard format such as

"As a <type of user>, I want <some goal> so that <some reason>."

To illustrate Cohn's standard format, consider the example below.

Example 15.3

User Story Capture

As a business traveler (User Role), I want a portable device with wireless connectivity (goal) so that I can download current and future weather reports from the Internet for planning job tasks outdoors that are dependent on the weather (reason).

Cohn (2008) suggests that User Stories are often "more informative to write the story with the specific user most likely to perform the action." He also observes that people often question the value of the "reason" attribute in the User Story template.

Cohn (2008) cites two advantages of the "so that" construct: … it helps in identifying:

1. "the meaning and relative importance of a story even long after it has been written" and
2. "identify potential alternative solutions."

Others point out that the "so that" provides insights into the User's motivation.

Once the User Stories are collected, some organizations capture them in a spreadsheet or database with separate columns for the User Role, Goal, and Reason. This provides a means for quickly analyzing and mining the data set, especially in identifying commonalities across a set of User Stories, Customers, and Users.

Every User Story has a *personal context* that is unique to their role as a User—Administrator, Operator, Maintainer, Trainer, or End User—and expected outcomes that will provide some benefit or overcome some problem based on past experience. Therefore, every User Story should:

1. Be written in terms of the originator's role, not name, that establishes the context of the story. However, you still need to ID the User's name and contact information for follow-up collaboration, as required.
2. Be date stamped.
3. Express a level of importance (such as 1–10) to the User.

Here's another example.

Example 15.4

User Story: Computer Virus Checker Updates Date: XX/XX/XX

As a manager (User) accountable for the integrity of our computer network, I need an immediate quick fix (goal) so that we can quarantine viruses and prevent their further proliferation (reason).

Personal Value: 10 on a scale of 1 (low) to 10 (high).

Agile software developers value the User Stories as a means of documenting the User's perspective of a *real* or *perceived* operational need in their own words.

Once the initial set of User Stories has been reviewed and analyzed, it provides a basis for Agile software developers to follow up with each User Story's originator to discuss *requirements* implementation. Observe that the reference to *requirements* in the last sentence does not say *specification requirements* or *capability requirements* … only "requirements" at least in an Agile Software Development context.

15.8.7.2 Conditions of Satisfaction (COS) Since successful achievement of a system, product, or service, at least commercially, resides in the eyes, minds, and experiences of the Customer, it is equally important to have them describe *in their own words* how they expect to *validate* that the Statement of Need has been met. Cohn (2008) refers to these as COS. He notes that "The COS aren't executable tests but they do say what should be tested at a high level." Each COS should be written on the side of an index card or note opposite from the User Story. Here's how a COS for Example 15.4 might read for a skilled User familiar with the process.

Example 15.5

COS (Back of Index Card)

I would turn the device on, go out to the web, enter the URL for the weather Web site or select a bookmarked URL, enter in my location, and view; download, or printout current weather conditions or future weather reports.

Agile software developers often refer to a *User Story* as a form of contract with the Users (Customers). In principle, this is true figuratively speaking. However, in other business domains, this is a *misnomer* unless the requirements are formally documented in a specification incorporated by reference into a legally binding contract.

The concept of allowing User Stories to *evolve* and *change* over time *infers* that software development is in a *continual* state of change. Software updates include not only *incremental* updates but also *revisions* to previous deliveries of a product. In a commercial Agile Software Development environment where numerous *iterations* of a product are prototyped and test marketed at internal expense, this may be acceptable. However, this is not the case in other environments, especially development of larger systems based on Firm-Fixed Price (FFP) contracts. Where User Stories are allowed to evolve and change over time, Cost-Plus-Fixed-Fee (CPFF) contracts or a series of contracts that *evolve* and *mature* User requirements may be the appropriate solution to accommodate evolving User Stories.

Lastly, the relationship of Themes and Epics to User Stories simply stated:

- An *Epic* is an abstraction representing a collection of User Stories. Cohn (2008) suggests keeping large Epics that contain numerous User Stories with dependencies intact as long as possible.
- A *Theme* is a Customer's *overarching* description of a class of products that cover a variety of User Stories. Examples include medical infusion devices and radar technologies.

Views and usage of relevance and usage of the terms Themes and Epics vary among organizations.

Given a basic understanding of User Stories, Themes, and Epics, let's address how User Stories are implemented via Agile Product Development Cycles.

15.8.7.3 User Stories versus UCs: What's the Difference? The concepts of *User Stories* and *UCs* are sometimes confusing to engineers, analysts, and PMs, functional managers, and executives. User Stories are a key operative term in Agile software developer semantics. Agile software developers tend to become polarized in terms of preferences for one term versus the other. *What is the difference*?

- A *User Story* is simply a brief personal statement that expresses a User's role-based operational need in their own words. User Stories and other tools such as Quality Function Deployment (QFD) are useful in performing User Needs Analysis (Chapter 5).
- A *UC* simply expresses a performance-based capability—Print Report—the User expects a system, product, or service to produce in response to a set of stimuli, excitations, or cues.

A *User Story* documents an expressed need, want, or desire; a UC elaborates *how* the User in collaboration with the System Developer envisions a system, product, or service to satisfy that need. The UC is then elaborated into a sequence of operational tasks (a primary UC flow) and scenario-driven alternative tasks (alternative flows), each with a performance-based outcome to be accomplished. Despite differences, both are written in collaboration with the User using words they can easily understand, review, and feed back comments.

Author's Note 15.5

Misapplication of UCs

SEs, Engineers, and Analysts often *convolute* usage of terms such as User Stories and UCs. Comments such as *Use(less) Cases* are common as a result of *misapplication*. For example, SEs and others amateurishly attempt to employ UCs—an ounce of knowledge—to identify User operational needs, have difficulty, and then blame UCs. Once again:

- User Stories are useful in identifying *User operational "use-based" needs*.
- UCs simply elaborate a concept for *how* those needs enable us to define primary and alternate flow tasks—capabilities—that become the basis to bound and specify a system, product, or service's specification requirements.

Think about it! If User Stories identify a "use-based" need, the next logical step is to identify various UCs and scenarios of usage. Simply stated: (1) *what is the system, product, or service expected to accomplish to produce an outcome that satisfies the User's need and (2) what could go wrong*?

The polarization of Agile software developer preferences for User Stories versus UCs often centers around two key arguments that are dependent on personnel capabilities. For example, here are viewpoints arguments:

- **User Story Preferences Argument**—Why write a formal XX page System UCs Document when we can collaborate with the User based on their User Stories. If the story changes, we can change the software to accommodate their needs. No need for a lot of "paper pushing."
- **UC Preferences Argument**—User Stories are okay. However, I can't take a set of *User Stories* and turn them over to lesser experienced personnel. They keep coming back to me wanting clarification and an explanation of how to implement the high-level statements. That's why UC documents provide definitive information that can be used by System Developers and User Enterprises regardless of experience.

The argument, however, goes deeper than simply preferences for User Stories versus UCs. This last point illustrates a dichotomy among Agile software developers:

- Some say that User Stories are derived from UCs. They contend that User Stories represent segments or branches of UC SysML™ Activity Diagrams that model User interactions and System behavioral responses.
- Others contend that UCs are derived from User Stories. Cohn (2008), referring to Agile Software Development, observes that some people mix User Stories and UCs. They start with large Epics as User Stories and then employ UCs to provide additional detail. Eventually, the organization evolves to User Stories.

Given that some contend that User Stories are too abstract for lesser experienced personnel, the User Stories transformation into UCs may be reflective of that situation. It is unclear if the evolution to User Stories over time is the result of an Enterprise personnel maturing to the point that User Stories are adequate for their experience and skill levels.

So, *what is the relevance of this discussion to SE and how can Agile Development methods benefit SE*? User Stories and UCs both have relevance and significance to SE. That brings us to our next topic, Understanding User Stories, UCs, and Specification Requirements.

15.8.8 Agile Development (Scrum) Cycles

Agile Development Cycles are typically referred to as *Scrums*. The concept of scrums was developed by Schwaber and Sutherland (2011) in the 1990s. As a foundational concept, scrums were a key contribution to the new Agile Manifesto philosophy of software development.

Schwaber and Sutherland (2011, p. 5) describe a *scrum* as follows:

> "Scrum is a framework structured to support complex product development. Scrum consists of Scrum Teams and their associated roles, events, artifacts, and rules. Each component within the framework serves a specific purpose and is essential to Scrum's success and usage."

The term Scrum:

- Originates from Rugby football concerning a formation for restarting play after an infraction or game stoppage (Rugby IRB 2013, Law 20, p. 134)

- Is sometimes referred to as *development cycles* or *iterations*

An Agile Development Team or Scrum Team is comprised of a Product Owner, a Development Team, and a Scrum Master. A brief synopsis of these roles is provided below (Schwaber and Sutherland 2011, pp. 5–7):

- **Product Owner**—An individual accountable for completion of the work, serves as the Voice of the Customer (VOC), maximizes the Agile Development Team value, and maintains the priorities of User Stories and any changes
- **Development Team**—A self-organizing, possibly multi-discipline team accountable for performing the work to incrementally release work products
- **Scrum Master**—An individual that serves as a "servant-leader" to the Scrum Team to ensure compliance to "Scrum theory, practices, and rules"

Referral Refer to Schwaber and Sutherland (2011) for a detailed description of the Scrum framework and its implementation.

15.8.9 Agile Product Development Cycles or Scrums

Principle 15.6

User Stories Repository

Create a Product Backlog managed by a Product Owner to serve as *the* single repository for storing and managing all User Stories.

Principle 15.7

User Story Scope Principle

Scope the selected set of User Stories for completion within the time-boxed constraints of a Sprint.

One of the objectives of Agile Development is to *incrementally* deliver *high-value* product releases to the Customer in a series of short 7–14-day iterations referred to as Sprints. The endgame is to identify a set of higher-value requirements or User Stories that can be implemented by an Agile Development or Scrum Team(s) over a series of Sprints. Sprints are conducted until all User Stories in the Product Backlog are implemented within the time-boxed constraints. Cohn (2008) observes that a "good Product Backlog" will be comprised of 90% User Stories and the remaining 10% is "just stuff" people placed there that could potentially be converted into User Stories.

The challenge becomes: *how does an Agile Development or Scrum Team get from a large number of prioritized Customer User Stories or requirements in the Product Backlog to a small set of requirements that can be implemented and released in 7–14 days*? That is the focus of our remaining discussion.

Author's Note 15.6

Incremental Release Context

The context of an Agile Development *incremental release* is important here. A *release* can have one of two contexts:

- Context #1—Internal distribution within the project to other developers such as System Analysts developing models and simulations
- Context #2—External distribution to existing Customers such as an urgent security update due to a new computer virus

15.8.10 Product Increments (Optional)

Principle 15.8

Agile Product Increment Principle

Create optional Product Increments based on a grouping of User Stories that form a planned software "build."

Although the *Scrum Guide* (Schwaber and Sutherland 2011) does not address Product Increments per se, some Enterprises prefer to include these as an *optional step*, especially if multiple Sprints are being performed simultaneously on larger projects.

Referring to Figure 15.7, the Agile Development or Scrum Team partitions the highest-priority User Stories or requirements into a series of Product Increments such as Product Increments #1 and #2. Each Product Increment is scoped to be accomplished within a specified time period or time-box such as 30–45 days.

Principle 15.9

Product Backlog Completion Principle

Track *planned* versus *actual* performance of Product Backlog User Stories in terms of those completed, those in-process, and those remaining to be implemented.

In Agile Software Development, a Product Increment Backlog repository of requirements typically does not exist per se; however, it is recommended for performance tracking purposes. Ultimately, any Product Increment requirements that were not completed within the designated time-box must be returned to the higher-level Product Backlog for assignment to a subsequent Product Increment.

15.8.11 Sprint Development Cycles or Scrums

Principle 15.10

Sprint Backlog Completion Principle

Track *planned* versus *actual* performance of Sprint Backlog User Stories in terms of those completed, those in-process, and those remaining to be implemented.

Beginning with Product Increment #1, Customer requirements are analyzed and partitioned into a series of 7–14-day Sprint Scrums for implementation and incremental delivery. Product Increment requirements allocated to each Sprint are allocated and flowed down to a Sprint Backlog repository for performance tracking such as the Burndown Chart example shown in Figure 15.8.

Principle 15.11

Daily Scrums Principle

Conduct brief daily scrums to assess individual or team accountability for three questions:

1. What did you accomplish yesterday?
2. What issues need to be addressed today?
3. What do you plan to accomplish today?

Each Sprint consists of Daily Scrum Meetings at the start of each workday. Daily Scrums are staged as 15 minute "stand-up" accountability forums for the Agile Development Team members to answer three key questions (Principle 15.11):

1. *What did you accomplish yesterday*?
2. *What issues need to be addressed today*?
3. *What do you plan to accomplish today*?

Principle 15.12

Incomplete User Stories Principle

Return User Stories that are *incomplete* at the end of a time-boxed Sprint to the Product Increment Backlog (optional) or Product Backlog for *reprioritization* and *reassignment* to a subsequent Sprint.

As each Sprint incrementally releases product requirements and their implementations, the Sprint Backlog is decremented. When the Sprint *time-box* expires, residual requirements that have not been implemented are returned to the higher-level Product Increment Backlog, as applicable, for reallocation to the next or a subsequent Sprint.

Given this overview discussion, let's delve into an example of a Sprint methodology.

Each Sprint Iteration Cycle is an *iterative*, methodology-based process such as the example shown in Figure 15.9. As an example, key steps in a Sprint Methodology consist of:

1. Plan Sprint
2. Establish Sprint Task Backlog
3. Prioritize Sprint Backlog Tasks
4. Perform Sprint Backlog
5. Conduct Daily Scrum Meeting
6. Iterate Tasks to Completion
7. Review Incremental Release
8. Demonstrate and Deliver Incremental Release
9. Decrement Sprint Backlog
10. Cycle Back to Next Sprint Backlog Task
11. Conduct Sprint Retrospective
12. Exit to Next Sprint

15.8.12 Minimal Documentation Approach

Agile Development Teams partition User Stories into smaller increments to fit within a Sprint. It is important to note that Sprints focus on development of the product with *minimal* documentation. *Recall the paradoxical statements in the Agile Software Manifesto*. Solarte (2012) provides insights concerning considerations for "minimal" documentation.

Our discussions above focused on *prioritized* User Story implementation via the Sprints. The implementation of User Stories requires two concurrent efforts: (1) development of the product and (2) its documentation. With the focus on incremental product deliveries, documentation has a tendency to lag behind. Inquiries concerning both are often met with the paradoxical statement "*Which do you want me to produce? Product or documentation? Given our tight schedule, pick one or the other*; *you can't have both*." How far the documentation lags behind is an issue. Some suggest including the documentation as a User Story that goes into the Product Backlog. We will address this topic in more detail later in the chapter.

15.8.13 Agile Software Development: Risk Approaches

Principle 15.13

Development Process Model Principle

Select a development process model based on requirements, technology, team skills, and schedule risk constraints unique to the product, not the project.

In selecting an appropriate Development Process Model, Highsmith (2013) suggests consideration of what he refers to as Exploration Factors (EFs). In general, an EF represents the level of *uncertainty* or *risk* associated with a new

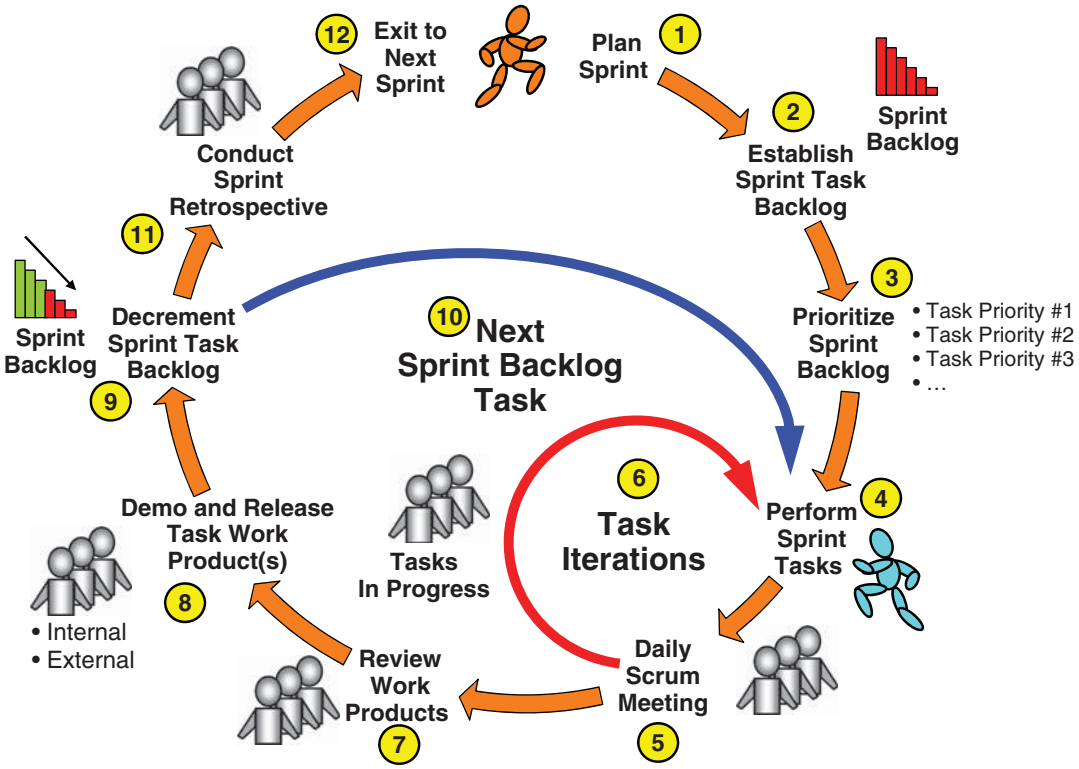


Figure 15.9 Sprint Iteration Workflow Cycle

development. For example, an EF = 1 represents a low-risk development in which the requirements are well known. An EF = 10 represents a *high risk* in which the requirements and/or technologies are *relatively unknown*. An EF 3–4 project, for example, might use any one of a number of development methods such as V-Model, Spiral, and Agile.

Most organizations perform R&D to meet future marketplace projections to ensure a competitive advantage. The objective is to initiate a "pipeline" of new technologies or technology applications to mature JIT for a key project's needs. This leads to key question: *if this is the Enterprise strategy, what was R&D doing or not doing that lead to the project having an EF 10 Agile Development effort on a customer deliverable*? Highsmith (2013) observes: "There are many flavors of Agile depending on the EF. If the EF is *low*, then iterations will mostly go as planned, even at the release plan level—6+ months or so. If the EF is *high*, iterations will be more R&D like. Short iterations are still valuable even when you think you know what the requirements are—because often it doesn't work out that way."

Principle 15.14

Risk–Success Expectations Principle

Adjust project expectations of success based on the level of risk (Highsmith, 2013).

Highsmith offers a word of *caution* about high EF projects. As you go higher from 1 to 10 along the EF scale, you have to change your expectations of success. Success on high-risk EF 10 requirements and technology projects will probably be different than on an EF 1 project.

15.8.14 Test-Driven Development (TDD)

One of the key Agile Development methods involves a concept referred to as TDD. TDD, which is commonly used for software development, consists of writing the software test for a capability or feature *before* developing the deliverable product's (production) code.

Conceptually, the intent of TDD is to use the requirement statement to establish a threshold that says "develop no more code than is required to pass the test." The underlying assumption here is that the requirement and test are both *valid* and *accurately* address the capability Problem Space to be solved. Some will argue that the focus shifts from Engineering design to an *endless loop* of the SBTF Paradigm to "pass a test." Ultimately, the result depends on the culture and discipline of the Agile Development Team.

Given that test procedures are typically written *after* a hardware or software component has been designed, the possibility exists that a Unit-Under-Test (UUT) may be *overdesigned*. Additionally, there is a risk that the specification

TABLE 15.2 Bridging SE and Agile Development Semantics

SE Semantics and Concepts	Agile Software Semantics and Concepts
Users, End Users, System Acquirer	Customers
Operational needs	Requirements
UCs and scenarios	User Stories, Epics, Themes
System levels of abstraction	Multi-level scrums
Capabilities	Features
Essential documentation	*Minimal* documentation
Multi-level component verification System acceptance and delivery	Incremental product releases

requirement used as the basis for the test was *poorly written*, *untestable*, or *expensive* (Chapter 22). So, the TDD concept has merit.

15.8.15 Special Topics: SE and Agile Software Development

The preceding overview of Agile Software Development leads to a key question: *how can SE leverage Agile Development concepts to improve overall SE and project efficiency and effectiveness*? At this point, you have probably noticed that Agile Software Development employs concepts and semantics that are different from others addressed in this text. Table 15.2 provides a general comparison of SE versus Agile Development concept and semantics differences.

15.8.15.1 Agile Application to Multi-discipline SE People often ask: *how does Agile Development apply to SE*? In general, for moderate to large, complex systems that require multi-discipline integration with various types of unique workflows, perhaps there might be a better question to ask. *Does SE have task-based activities for which Agile Development methods might improve performance*? The answer is yes. Before we list some examples, there are a couple of points we need to make:

- Agile Development produces incremental, high-value product releases in response to Customer requirements and priorities. Observe the generic usage of the term *product* or more appropriately *work product releases*. For software, the Agile Manifesto emphasizes *working software* over comprehensive documentation—*working software*. Working software may or may not be in the final form as the deliverable; it could be algorithmic prototypes.
- SEs and System Analysts *do not* design hardware or software products per se. SEs apply their multi-discipline experience to *orchestrate*, *integrate*, and *facilitate* technical decision-making of teams that develop hardware and software. SE and System Analyst *work products* capture these decision-making artifacts. Whereas the context of applying Agile Development methods to software development produces *working software*, the application of Agile Development to SE and System Analysis produces *work products—such as specifications, architectures, descriptions, and so forth. These work product releases* provide information that feeds a Sprint's Supply Chain (Figure 4.1).

Examples of multi-discipline SE activities include:

- Identification of Stakeholders—Users and End Users, their needs, and priorities
- Identification and development of SYSTEM or ENTITY UCs and scenarios
- System architecture selection, Analysis of Alternatives (AoA) selections, and so forth
- Concept of Operations (ConOps) and its Operational Concept Descriptions (OCDs)
- Specification requirements development
- Interface definition
- System TCs

In each of these cases:

- Multi-discipline SE activities would be performed based on Supply Chain (Figure 4.1) *internal* or *external* customer priorities.
- Requirements might be *unknown* and *fuzzy*—at least initially—and continue to *evolve* and *mature* over time.
- Details could be derived top-down *iteratively* through a series of multi-level Product Increments and Sprints.

You may ask: *what is different about Agile Development of these work products that is not already being done*? Consider the following example.

Example 15.6 **Traditional versus Agile Specification Development**

In some business domains, development, approval, and release of a specification could require up to three (3) calendar months. *Why three months*?

In general, a single SE is assigned a task as a specification developer. They spend from 2 weeks to a month collecting Stakeholder requirements and developing the initial draft of the document. Then, an additional two months are spent conducting reviews of the specification, resolving issues

and making corrective actions, and obtaining approval for release. *So, why 2 months*?

The multi-discipline stakeholders required to participate contend that they can only afford to spend a maximum of 1 hour preparing for and a maximum of 1 hour participating in a review per week due to "higher" priorities—sounds like a management performance problem. So, the process is *piecemeal*, *disruptive*, *inefficient*, and *ineffective* and requires a learning curve each week to reestablish their understanding of (1) the level of maturity and current issues and (2) changes that have been made. As a result, up to eight (8) weeks may be required to complete the reviews and obtain approvals unless the Customer or the PM or Project Engineer steps in and changes reviewer priorities.

Now, suppose an Agile Development Team could deliver a 90% complete version of the same document within a Sprint Iteration of 5 days instead of 3 months.

15.8.15.2 Agile Application to SE Proposal Development

Given Agile Development's focus on high-value–high-return strategy for incremental product development, another example application is proposal development. Enterprises sometimes have "must win" procurement opportunities that require leadership by external proposal consultants. Requests for Proposals (RFPs) typically have "time-boxed" constraints such as 30 or 45 days. The proposal strategy, in general, is dependent on fast-paced, multi-discipline teams that address specific topics within the technical, management, cost, and volumes with *highly aggressive* schedules.

Many of these efforts resemble Agile Development activities. Teams attend daily mandatory proposal meetings—Daily Scrums. The difference is that the proposal consultant leader addresses group level progress and status, issues, plans, communications, and corrective actions. Additionally, there are often impromptu Daily Scrums with individual teams to address team progress and status, issues, plans, communications, and corrective actions. The whole exercise is focused on meeting proposal development milestones—incremental development with highly iterative interactions among teams, proposal peer and independent reviews, and culminating with the delivery of the proposal.

15.8.15.3 User Stories, UCs, and Specification Requirements Dependencies

User Stories and UCs provide a logical pathway to specification requirements. Figure 15.10 provides an illustration of this point.

Observe the general left-to-right workflow over time from User Stories to System UCs to Specification Requirements in the lower portion of the graphic. Note that we said *general workflow*; these steps are *highly interactive*. Customers–Stakeholders, Users, and End Users consisting of MISSION SYSTEM and ENABLING SYSTEM operators, maintainers, trainers, and administrators are candidates for contributing one or more User Stories. The set of User Stories should represent the system, product, or service User community. SEs and System Analysts analyze the data from User Stories and conduct follow-up collaborations with the Users to *clarify* and *avoid* misinterpretation.

Author's Note 15.7

SE Involvement in Customer Operational Needs Meetings

The collection of User Stories reinforces that point that an SE should *always* accompany a Business Development or Marketing representative after the first contact.

User Stories provide the basis for developing a System UCs Document that *expands* or *refines* each UC into a set of User operational tasks and interactions that stimulate or excite a system, product, or service to produce the required behavioral outcomes.

15.8.15.4 Semantics: Linking Agile Customers to SE Stakeholder Users and End Users

Agile Software Development uses the term *Customers*; SE refers to role-based *Stakeholders* such as System Acquirers, Users (operators and maintainers), and End Users (Chapter 3). *What are the differences*?

Software developers have a number of types of prototyping tools including simulations they can employ to collaborate with "customers" concerning displays, layouts, transitions, and so forth as in interim step without having to have the deliverable product hardware. This enables them to have direct access to and collaboration with User operators and maintainers. However, the scope of "System" Development involves more than "software" development. SEs and Analysts must consider User operations such as deployment; operations, maintenance, and sustainment (OM&S); hardware, software, and courseware; Specialty Engineering Integration such as human factors, safety, Reliability, Maintainability, and Availability (RMA); and other considerations. Consider the following example.

Example 15.7

Enterprise Context: Internal and External Customers

From an Enterprise project perspective, a System Developer has *external* (Customer) Users and *internal* (System Developer) Users. For example, a System Developer has an internal Supply Chain such as the one shown in Figure 4.1. Contextually, *internal* Users (Customers) consist of higher-level System Development project teams at the SYSTEM, PRODUCT, AND SUBSYSTEM levels of abstraction that allocate and flow down requirements to the next lower level. A PRODUCT Level Team performing

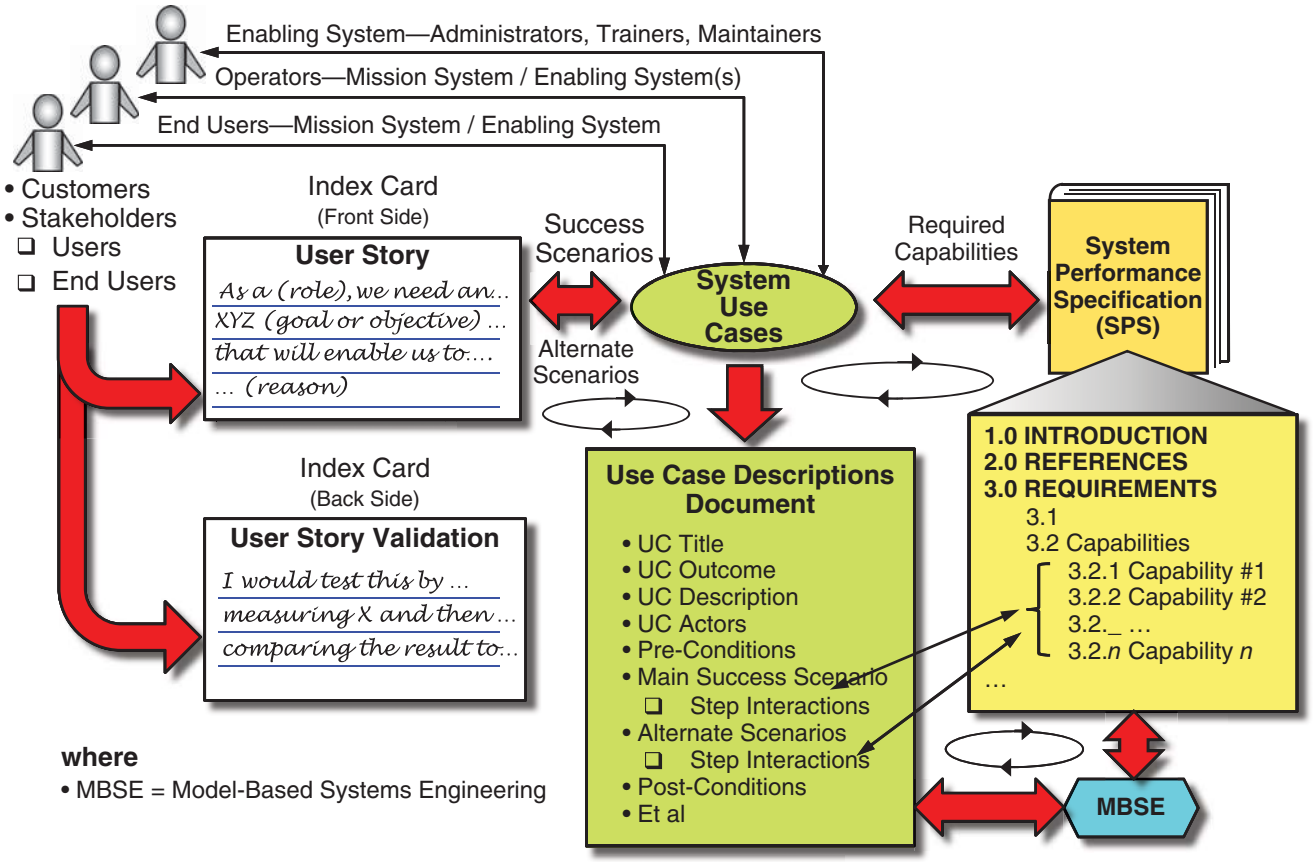


Figure 15.10 The Relationship between User Stories, UCs, and Specification Requirements

a System Acquirer (role) acquires, integrates, tests, and verifies SUBSYSTEMS. The SUBSYSTEM Level Team delivers its SUBSYSTEM to the high-level (internal customer) team for integration into their PRODUCT.

To better understand the *context* and importance of external and internal Users in Agile Development, Figure 15.11 provides an illustration.

15.8.15.4.1 External System Users Observe the line dividing a System Developer's External and Internal Customers. External Customers such as a SOI MISSION SYSTEM and ENABLING SYSTEM Users performing operator or maintainer roles provide the primary inputs for Agile Development User Stories.

As details of the System Design Solution evolve and mature over time at lower levels of abstraction, External Customer specialists collaborate with System Developers. Collaboration occurs throughout the development of PRODUCTS, SUBSYSTEMS, AND ASSEMBLIES, via informal conversations and technical reviews such as Preliminary Design Reviews (PDRs) and CDRs. External Customer Specialist examples include sensors, electronics, propulsion, algorithmic, and medical technologies; energy such as oil, gas, wind, solar, and nuclear; transportation; aerospace and defense (A&D) such as aircraft; ship and spacecraft; and domains.

Author's Note 15.8

User Priorities

For small to large, complex system development efforts, the System Acquirer representing the User's technical interests will state that contract requirements including specification requirements have the *same priority*. Unless specified otherwise in the contract, the notion of Product Increment releases (Figure 15.7) is strictly internal.

In contrast, commercial ventures are often characterized by highly open collaborative efforts in which the User(s), Business Development, or Marketing decides and drives Product Increment Backlog priorities.

15.8.15.4.2 Internal System Users Given a set of External Customer User Stories as inputs, System Developer SEs and Analysts derive SYSTEM Level UCs and the SPS (Figure 15.11). In theory, SEs formulate and select architectures from a set of viable candidates based on SYSTEM,

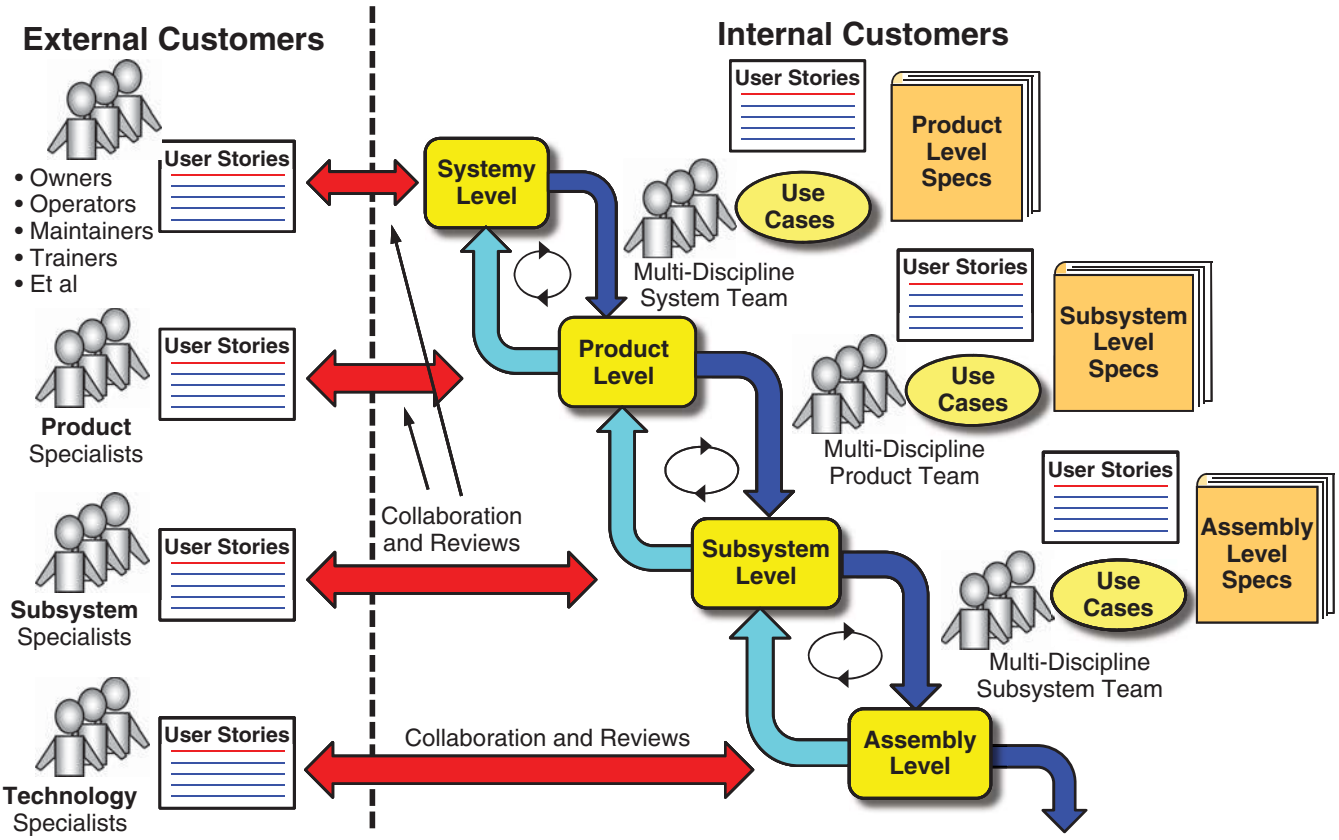


Figure 15.11 The Importance of External and Internal User Stories in System Development

PRODUCT, AND SUBSYSTEM, specification requirements as illustrated in Figure 20.4. Specification requirements become the hierarchical framework of capability *requirements* and *constraints* that will ultimately be used for verification of the specified ENTITIES such as the right side of the V-Model shown in Figure 15.2.

The reality is that specification requirements that are derived, allocated, and flowed down to lower levels are *influenced* by the User Stories, written or unwritten, of each higher-level team. Figure 15.11 provides an illustration. Each requirement statement is a reflection not only of the literal *capability* or *constraint* to be flowed down but also nuances of each team member's experiences, lessons learned, and skills in wording. Requirements are derived from UC capability threads such as Figures 21.6 and 21.7 introduced later. Each UC thread expands or refines a User Story.

Does the lower-level PRODUCT, SUBSYSTEM, AND ASSEMBLY *team that is the recipient of allocated and flowed down requirements know about the higher-level UC thread*? Typically, no, unless an informal request is made for clarification of a specific requirement. The clarification may be due to the (1) requirement being incomplete, ambiguous, or having a conflict with another requirement or (2) an attempt to understand the rationale for the requirement—namely, how the capability fits into the higher-level UC thread. This same process repeats for lower levels.

15.8.15.5 Agile Software Development: System Development Models

Principle 15.15

Non-prescriptive Methods Principle

As a process, Agile Development does not prescribe a specific development method or model to be used to create a system, product, or service.

Earlier in this chapter, we noted that customers often want to know what type of System Development Model a project is using—that is, one size fits all. The reality is that you select the model(s) that makes sense for a given system entity at any level of abstraction (Figure 8.7). The selection is based on the risk-dependent circumstances such as *unknown*, *fuzzy*, or *poorly* defined requirements, technologies, and the Enterprises competency in applying the model(s). The answer might be one model or several models, depending on the project. The overall project might decide

to use the V-Model for overall System Development, Spiral Development for developing SUBSYSTEM #1's hardware or software, and Agile Development for developing SUBSYSTEM #2's software.

Agile Development, as is the case with any model, is one of several *methods* that can be applied for application to any ENTITY such as a SUBSYSTEM, ASSEMBLY, AND SUBASSEMBLY or its HARDWARE/SOFTWARE. There are caveats, however, in which Agile Development methods may be better suited for specific system, HW, or SW development tasks, applications, and deliverables for which a dedicated team is assigned with *minimal* external distractions. In development environments that thrive on "multi-tasking" of personnel, Agile Development may not be practical.

Due to Agile Development's adoption in the software community, people often *erroneously perceive* it to be a method for coding software faster. This is factually incorrect! Agile Development is an *approach* for focusing software development activities on high-value–high-return customer priorities. Schwaber and Sutherland (2011) refer to it as a *framework*.

Remember that as stated in Principle 15.13, Agile Development *does not prescribe* a specific System Development Model. During our earlier discussions of the V-Model and Spiral Development in this chapter, we noted that SW developers contend that the *stage–gate* models such as the V-Model are impediments for producing deliverables efficiently. The irony here is that Software Developers still have to specify, design, build, integrate, and test each Sprint User Story to "Engineer" the product. As a result, they come full circle back to a form of scaled-down Development Process Model such as the V-Model or Spiral Model within the Sprint to produce incremental software deliveries as shown in Figures 15.7.

15.8.15.6 SE Essential versus Agile Minimalist Documentation

Principle 15.16

Minimal–Essential SE Documentation Principle

Produce only *minimal*, *essential* documentation that is *necessary* and *sufficient* for the:

1. Developers to specify, design, build, integrate, test, train, and maintain (optional) the product.
2. User's to operate, maintain, and sustain (where applicable) the system, product, or service to perform missions.

One of the Agile Software Manifesto's statements expresses the paradoxical choice of "working software over comprehensive documentation." An illustrative example of "comprehensive documentation" occurs in the A&D, medical, nuclear, and other domains for contracts issued by a government or subject to government oversight and regulations. Requirements originate in the interest of flight, ground, or medical safety for (1) the operators, the maintainers, and the public and (2) the need for maintainers with various levels of skills to easily understand and maintain the systems for many years.

With the advent of the Agile Software Manifesto, it is good to challenge ourselves to determine "where truth resides." *What is the least costly, minimal, and essential approach to documenting a System Design Solution*? The answer depends on the Users and their skill levels. Agile Developers contend that Agile principles do not say "no documentation": only the *minimal* amount—as abstract as the term is—of documentation needed to help someone understand how the SYSTEM OR PRODUCT is *designed*. The *fallacy* of this assertion is as follows:

1. What is easy for a software developer to understand a design decision within their skillset may be difficult for someone else with a different skillset.
2. The fielding of a system, product, or service requirements requires more than simply understanding its *design*. You must understand its deployment, operation, maintenance, and sustainability (OM&S) as well as retirement and disposal, which are contributors to the overall TCO.

A Word of Caution 15.1

The Minimal Documentation Conundrum

One of the shortcomings of the human condition is a general dislike for documentation. Where Product Owner and Agile Development or Scrum Team leadership is *weak* or *inexperienced*, a "minimum" documentation objective can evolve into a culture that rationalizes their decisions—"no documentation is great!"—contrary to sound engineering practices. Comments such as these express personal opinions, not SE User advocacy values. Insightful, experienced leadership with decision-making wisdom is critical to ensure the necessary level of essential documentation. Challenge yourself and your team: *if I (we) had to operate or maintain this system or software, would there be sufficient information to enable us to perform the task without having to spend hours searching for it or trying to figure out what the original developer intended*?

In SE, we emphasize the need for only *essential* documentation—as abstract as that term is. We can say that *essential* means that reasonably competent designers and maintainers should be able to read, understand, or analyze a SYSTEM or PRODUCT's design, understand what decisions and trade-offs were made and why, be able to train other Users, and maintain the system in the field.

Therefore, you might expect to see *decision artifacts* such as specification requirements; ConOps; architecture, design, and performance trade-offs and selections; and a design description of the SYSTEM including its EQUIPMENT (hardware and software); drawings, wiring lists, and parts lists; and TCs and procedures.

The challenge question is w*hat level of documentation is both necessary and sufficient*? Viewing the documentation for an ENTITY such as a SUBSYSTEM, ASSEMBLY, Computer Software Component (CSC), or Computer Software Unit (CSU) as a "system" requires understanding *who* its Stakeholder Users and End Users are and their role-based UCs. Then and only then can you determine what is *minimal* and *essential* to satisfy the component's *necessary* and *sufficiency* documentation criteria.

15.8.15.7 Agile Application to Integrated Hardware and Software Development Initial responses about applying Agile Development to a SYSTEM or integrated HW and SE PRODUCT raise eyebrows, especially for medium to large, complex systems. Some will argue that changes can be made in software in a matter of minutes. In contrast, mechanical and electrical/electronic HW that executes the SW changes may require more time such as hours, days, weeks, or months to modify.

Suppose we have a tightly packaged electromechanical device—SUBSYSTEM, ASSEMBLY, AND SUBASSEMBLY—with no space for modification without disassembling the whole package. Although software changes may be easily uploaded onto the device, electronic and mechanical hardware modifications may require days, weeks, or months. But also observe that if the device is physically assembled as a packaged electromechanical device, application of Agile Development may be more applicable to the software changes within the device. Remember that one of the motivations for moving to software-intensive systems is to accommodate *quick changes* to avoid expensive, time-consuming hardware changes, assuming they are not impacted.

Now, consider the application of Agile Development methods to the early design of the device.

Example 15.8

Application of Agile Development to Integrated HW and SW Designs

Assume that a multi-discipline project team is assigned accountability for developing a microprocessor-based controller device with Input/Output (I/O) ports. The team formulates and selects an architectural design and creates a laboratory breadboard model using wire-wrap or other technology.

As the design evolves and matures, from time to time, modifications may be made to accommodate changing User requirements. In that case, reconfiguration and modification of electronic connections, port addresses, as well as the I/O driver software might take 30 minutes to an hour. Remember that the breadboard prototype is an *unpackaged* working model. So, in this context, multi-discipline Agile Development methods apply to early integration of hardware and software designs and components such as ASSEMBLY or SUBASSEMBLY Level prototypes.

You may ask: *what about mechanical components*? Changes in today's mechanical technologies are also becoming responsive to changing requirements. Such is the case with 3-D printing for parts manufacturing that can employ lasers or depositions to create components that would otherwise take weeks or months to machine. Example technologies include Stereolithography (SLA), Fused Deposition Modeling (FDM), Selective Laser Sintering, and others.

15.8.16 Agile Development Summary

In summary, Agile Development methods offer opportunities for SE to improve its *efficiency* and *effectiveness*, especially where dedicated task teams are able to work undistracted or with minimal interruption. However, a word of caution!

A Word of Caution 15.2

Due Diligence Decisions

As with any business decision, it is crucial that an Enterprise perform due diligence before amateurishly deciding to employ the latest marketplace methods. What works for one Enterprise or project may not work for others depending on personnel education and training, in-grained cultures, management, and discipline. You need to "fit check" concepts such as Agile Development beginning with small pilot projects and teams. Deciding to employ Agile Development or any method for the first time on a moderate to large, complex System Development project is high risk and is a recipe for failure. There is a difference between making risk mitigated, *informed* decisions versus ad hoc, *uninformed* decisions.

15.9 SELECTION OF SYSTEM VERSUS COMPONENT DEVELOPMENT MODELS

Principle 15.17

Project Development Models Principle

Select System Development process models to uniquely satisfy SYSTEM, PRODUCT, SUBSYSTEM, ASSEMBLY, development requirements, technology, risk, team skills, and development constraints.

Principle 15.18

Development Model Rationale Principle

Document each development model selection including rationale for:

- Selection
- Rejection of other models

Recognize that system development projects may employ several different development strategies, depending on the system or entity. You may find instances where the system is developed using IID of which one or more of its "builds" may employ another development strategy such as Spiral Development.

If you analyze most systems, you will find SUBSYSTEMS, ASSEMBLIES, AND SUBASSEMBLIES that:

- Have well-defined requirements.
- Employ mature, off-the-shelf technologies and design methods that have matured over several years.
- Are developed by highly competent and experienced developers.

Other SUBSYSTEMS and ASSEMBLIES may have the opposite situation. They may have:

- Ill-defined or immature requirements.
- Immature technologies and design methods.
- Inexperienced developers.
- All of these.

Where this is the case, you may have to select a specific development strategy that enables you to reduce the development risk. Consider the following example.

Example 15.9 **Multiple Development Process Models**

Over the years, automobile technology has evolved and increased in complexity. Today, we enjoy the benefits of new technologies such as fuel injection systems, Antilock Braking Systems (ABS), front-wheel drive, airbag Safety Restraint Systems (SRS), crumple zones, GPS mapping, and so on. All facets of automobile design have evolved over the years. However, for illustration purposes, imagine for a moment that the fundamental automobile at higher levels of abstraction did not change drastically. Most still have four doors, a passenger compartment, trunk, windshield, and steering. However, the maturation of the major technologies noted earlier required strategies such as Spiral Development that enabled them to mature and productize technologies such as hybrid engine or ABS for application and integration into an Evolutionary Development Model of the basic automobile.

15.10 CHAPTER SUMMARY

Our discussion in this chapter provided an overview of the various System Development Strategy practices. SYSTEM and PRODUCT development approaches require implementing a smart strategy that enables you to meet technical, cost, and schedule requirements with acceptable risk as well as User operational needs. Selecting the *right* system or product development strategy is a key competitive step. From an SE perspective, familiarize yourself and team with the basic attributes of each type of model and understand how to apply it to meet your specific application needs.

15.11 CHAPTER EXERCISES

15.11.1 Level 1: Chapter Knowledge Exercises

1. What is a system development process model?
2. What are the six primary system development process models?
3. Describe the "V"-Model, its characteristics, and shortcomings.
4. Describe the Waterfall Development Model, its characteristics, and shortcomings.
5. Describe the Evolutionary Development Model, its characteristics, and shortcomings.
6. Describe the Incremental Development Model, its characteristics, and shortcomings.
7. Describe the Spiral Development Model, its characteristics, and shortcomings.
8. Describe the Agile Development Model, its characteristics, and shortcomings.
9. How is the SE Process Model applied to these development process models?
10. Does the Agile Development Method prescribe a specific Development Model? If so, which one(s)?

15.11.2 Level 2: Chapter Knowledge Application Exercises

Refer to www.wiley.com/go/systemengineeringanalysis2e.

15.12 REFERENCES

Agile Manifesto (2001a), *History: The Agile Manifesto* webpage. Retrieved on 7/19/13 from http://www.agilemanifesto.org/history.html.

Agile Manifesto (2001b), *Principles Behind the Agile Manifesto* webpage. Retrieved on 7/19/13 from http://www.agilemanifesto.org/principles.html.

Agile Manifesto (2001c), *Manifesto for Software Development* webpage. Retrieved on 7/19/13 from http://www.agilemanifesto.org.

Benington, Herbert D. (1956). "*Production of Large Computer Programs*," ACR-15 Proceedings of the Office of Naval Research (ONR), Symposium on Advanced Programming Methods for Digital Computers, U. S. Navy Mathematical Computing Advisory Panel, Washington, D.C: Office of Naval Research, Dept. of Navy, June 1956, pp. 15-27. (Also available in the *Annals of the History of Computing*, Oct. 1983, pp. 350-361, and *Proceedings of Ninth Int'l Conf. Software Engineering*, Computer Society Press, 1987.)

Benington, Herbert D. (1983), *Production of Large Computer Programs*. *IEEE Annals of the History of Computing*, New York: Institute of Electrical and Electronic Engineers (IEEE) Educational Activities Department. Retrieved on 6/20/13 from http://sunset.usc.edu/csse/TECHRPTS/1983/usccse83-501 /usccse83-501.pdf.

Boehm, Barry (1981), *Software Engineering Economics*, New York: Prentice-Hall.

Boehm, Barry (1985), "A Spiral Model of Software Development and Enhancement," *Proceedings of the International Workshop Software Process and Software Environments*, ACM Press. (Also in ACM *Software Engineering Notes*, August 1986, pp. 22-42.)

Boehm, Barry (1988), *A Spiral Model of Software Development and Enhancement*, Figure 2, *IEEE Computer*, New York: Institute of Electrical and Electronic Engineers (IEEE).

Boehm, Barry (2006), "Some Future Trends and Implications for Systems and Software Engineering Processes", *Systems Engineering*, Vol. 9, No. 1, New York: Wiley.

Cohn, Mike (2008), *Advantages of the "As a user, I want" User Story Template*, Blog Post, Broomfield, CO: Mountain Goat Software. Retrieved on 9/12/13 from http://www.mountaingo atsoftware.com/blog/advantages-of-the-as-a-user-i-want-user-story-template.

DAU (2012), *Glossary: Defense Acquisition Acronyms and Terms*, 15th Edition Ft. Belvoir, VA: Defense Acquisition University Press. Retrieved on 3/27/13 from http://www.dau.mil/pubscats /PubsCats/Glossary%2014th%20edition%20July%202011.pdf.

DSMC (2001), *Glossary: Defense Acquisition Acronyms and Terms*, 10th Edition Ft. Belvoir, VA: Defense Acquisition University (DAU) Press.

Forsberg, Kevin and Mooz, Hal (1991), *The Relationship of Systems Engineering to the Project Cycle*, Chattanooga, TN: National Council on Systems Engineering (NCOSE)/American Society for Engineering Management (ASEM).

Global Security (2011), *B-52 History*, Global Security Organization, Retrieved on 7/22/13 from http://www.globalsecurity.org /wmd/systems/b-52-history.htm.

Global Security (2011), *B-52 Stratofortress Service Life*, Global Security Organization, Retrieved on 7/22/13 from http://www.globalsecurity.org/wmd/systems/b-52-life.htm.

Highsmith, Jim (2013), *Interview discussion* on 7/8/13.

IBM (2012), *Leading Through Connections: Insights from the Global Chief Executive Officer Study*, CEO C-Suite Studies, Armonk, NY: IBM Corporation, Accessed on 7/16/13 from http://www-935.ibm.com/services/us/en/c-suite/ceostudy2012/.

IRB (2013) *Laws of the Game – Rugby Union*, Dublin, Ireland: International Rugby Board (IRB), Retrieved on 6/17/13 from http://www.irblaws.com/downloads/IRB_Laws_2013_EN.pdf.

Larman, Craig and Basili, Victor R. (2003), "Iterative and Incremental Development: A Brief History," *Computer*, New York: Institute of Electrical and Electronic Engineers (IEEE) Computer Society.

MIL-STD-498B (1994), *Military Standard: Software Development and Documentation*. Washington, DC: Department of Defense (DoD).

Nagel, Roger and Dove, Rick and (Principle Investigators) (1991), *21st Century Manufacturing Enterprise Strategy – An Industry-Led View* (Volume 1) and – Infrastructure (Volume 2). Eds: S. Goldman and K. Preiss. Darby, PA: Diane Publishing Company.

Royce, Winston W. (1970), "Managing the Development of Large Software Systems: Concepts and Techniques," *ICSE '87 Proceedings of the 9th International Conference on Software Engineering*, Los Alamitos, CA, USA: IEEE Computer Society Press.

Schwaber, Ken and Sutherland, Jeff (2011), *The Scrum Guide – The Definitive Guide to Scrum: the Rules of the Game*, Retrieved on 6/1/13 from http://www.scrum.org/Portals/0/Documents /Scrum%20Guides/Scrum_Guide.pdf.

Shewhart, Walter A. (1986 reprint from 1939), *Statistical Method from the Viewpoint of Quality Control*, Mineola, NY: Dover Publications.

Solarte, Kurt (2012), *Is "agile documentation" an oxymoron? Understanding the role of documentation in an agile development environment*, Retrieved on 9/16/13 from http://www.ibm.com /developerworks/rational/agile/agile-documentation-oxymoron /index.html.

Straub, Pablo (2009), *Sample Burn Down Chart webpage*, Wikimedia Commons, San Francisco, CA: Wikimedia Foundation, Inc. Wikipedia, Retrieved on 6/17/13 from http://en.wikipedia.org /wiki/File:SampleBurndownChart.

Wikipedia (2014), *Waterfall Model* webpage, San Francisco, CA: Wikimedia Foundation, Inc. Wikipedia, Retrieved on 7/17/13 from http://en.wikipedia.org/wiki/Waterfall_model.

16

SYSTEM CONFIGURATION IDENTIFICATION AND COMPONENT SELECTION STRATEGY

From the moment of system inception, System Users, Acquirers, Developers, Maintainers, and others begin making system development decisions that evolve over time. Such is the case for System Engineering and Development … deliver the verified physical system within the triple project constraints of compliance to technical requirements, on schedule, and within cost. This requires a highly structured, organized, and orchestrated approach to progressively measure and report technical status and maturity of the evolving system design solution. From a programmatic perspective, the accountability, corrective action, and periodic progress reporting of the status and maturity of the evolving system design solution resides with the project engineer or Lead SE (LSE) at key technical program events such as reviews, technical status reports, etc.

As part of their accountability, the project engineer or LSE must capture the state of the evolving system design solution that represents a "snapshot in time" of technical decisions. Examples include stakeholder operational needs, use cases and most likely or probable scenarios, required operational capabilities, operations, behaviors, physical implementation, and others. This chain of topics also represents dependencies that evolve and mature at key staging points throughout the System/Product Life Cycle phases. Since humans must have physical objective evidence to review, documentation such as specifications, architectures, designs, drawings, etc.; prototypes; demonstrations; and models and simulations become the basis for assessment. As a "snapshot in time," they represent arrangements and time-dependent interactions of operational, behavioral, or physical entities or components that require unique configuration identification, version, and time stamp—that is, Configuration A, Version 1.1, Month/Day/Year.

Collectively, the set of configurations become part of what is referred to as the Developmental Configuration of the system, product, or service. As technical reviews are conducted throughout the project, the evolving Developmental Configuration becomes the framework for measuring and reporting the progress, status, and maturity of the evolving system design solution.

One of the frailties of the human condition is the ability to: (1) make and commit to timely, informed decisions when required and (2) to build on those decisions that lead to other decisions and subsequently to delivery of the system, product, or service. Dr. William McCumber and Sloan (2002, p. 4) observed that the job of SEs (and Project Engineer) is to "maintain *intellectual control* of the problem solution" (Principle 1.3). Otherwise, *ad hoc*, Plug and Chug … Specify-Build-Test-Fix (SBTF) - Design Process Model (DPM) Paradigm Engineering will prevail and lead to confusion and chaos.

The LSE or Project Engineer in collaboration with the System Architect, appropriate team leads or engineers, the Configuration Manager, and others must define a strategy that identifies when and how the Developmental Configuration of the evolving System Design Solution and configurations of its lower level hierarchical entities will be identified, captured, reviewed, approved, baselined, released, and controlled. The planning for these decisions and their criteria are synchronized and documented in the

System Engineering Analysis, Design, and Development: Concepts, Principles, and Practices, Second Edition. Charles S. Wasson.

Companion Website: www.wiley.com/go/systemengineeringanalysis2e

project's *Systems Engineering Management Plan (SEMP) and Configuration Management Plan (CMP).*

This chapter describes how elements of a system's architectural configuration are identified and designated for configuration control and tracking. Our discussions begin with establishing Configuration Management (CM) semantics and explain why these terms are often confusing. We explore how architectural items are selected from external vendor Commercial-Off-the-Shelf (COTS) items or Non-Developmental Items (NDIs), or Acquirer Furnished Property (AFP), or developed in-house from legacy designs or new development. We provide illustrations of how items are assigned to engineering development teams. We conclude with a discussion of when to establish Developmental Configuration baselines and contrast SE and CM viewpoints.

16.1 DEFINITIONS OF KEY TERMS

- **Allocated Baseline**—"Documentation that designates the Configuration Items (CIs) making up a system and then allocates the system function and performance requirements across the CIs (hence the term - allocated baseline). It includes all functional and interface characteristics that are allocated from those of a higher level CI or from the system itself, derived requirements, interface requirements with other CIs, design restraints, and the verification required to demonstrate the achievement of specified functional and interface characteristics. The performance of each CI in the allocated baseline is described in its item performance specification" (DAU, 2012, p. B-10).
- **Acquirer Furnished Property (AFP)**—"Physical assets such as EQUIPMENT, MISSION RESOURCES, HARDWARE, SOFTWARE, and FACILITIES provided by the User or other organizations via the Acquirer contract to the System Developer for modification and/or integration into a deliverable system, product, or service."
- **Baseline**—A collection of configuration-controlled, specification and design requirements documentation for a Configuration Item (CI) or set of CIs that represents the current, approved, and released version of a document or data item available for decision making.
- **Baseline Management**—"In configuration management, the application of technical and administrative direction to designate the documents and changes to those documents that formally identify and establish baselines at specific times during the life cycle of a configuration item" (SEVOCAB, 2014, p. 28 - Copyright 2012 ISO/IEC. Used with Permission) (ISO/IEC/IEEE 24765:2010).
- **Change Request (CR)**—A formal document submitted to Configuration Management (CM) requesting and specifying a corrective action to a Developmental Configuration Baseline document that contains a latent defect such as an error, inaccuracy, deficiency, design flaw, and so forth. CRs are typically: (1) initiated as a result of an investigative analysis triggered by a Problem Report (PR); (2) reviewed by a Configuration Control Board (CCB); (3) reviewed and approved, rejected, or placed on hold by the CCB; and (4) tracked to completion by CM based on successful verification of the corrective action. CRs for software corrective actions are referred to as Software Change Requests (SCRs); SCRs follow the same process as CRs with the exception of review and approval by a Software Configuration Control Board (SCCB).
- **Commercially Available Off-the-Shelf (COTS)**—"A Commercial Item (CI) sold in substantial quantities in the commercial marketplace and offered to the government under a contract or subcontract at any tier, without modification, in the same form in which it was sold in the marketplace. This definition does not include bulk cargo such as agricultural products or petroleum" (FAR, Subpart 2.101) DAU, 2012, p. B-34.
- **Commercial-Off-the-Shelf (COTS) Product**—A product available via a catalog or web site that is designed for ambient or industrial operating environments.
- **Computer Software Configuration Item (CSCI)**—"An aggregation of software that satisfies an end use function and is designated for separate configuration management by the acquirer. CSCIs are selected based on trade-offs among software function, size, host or target computers, developer, support concept, plans for reuse, criticality, interface considerations, need to be separately documented and controlled, and other factors" (MIL-STD-498, 1994, p. 5).
- **Configuration**—"A collection of an item's descriptive and governing characteristics, which can be expressed in functional terms, i.e., what performance the item is expected to achieve; and in physical terms, i.e., what the item should look like and consist of when it is built" (DAU, 2012, p. B-39).
- **Configuration Baseline**—A snapshot at a specific instant in time representing the current state of a system, product, or service's Developmental Configuration technical work products such as specifications, designs, drawings, schematics, parts lists, analyses, trade studies, and so forth that are new or have been revised, reviewed, approved by an approval authority, and released for project decision making.
- **Configuration Control**—"(1) A systematic process that ensures that changes to released configuration documentation are properly identified, documented, evaluated for impact, approved by an appropriate level of authority, incorporated, and verified. (2) The configuration management activity concerning: the systematic proposal, justification, evaluation, coordination, and disposition of proposed changes; and the implementation of all approved and released changes

into (a) the applicable configurations of a product, (b) associated product information, and (c) supporting and interfacing products and their associated product information." (MIL-STD-61A (SE), p. 3–4 and 3-5)

- **Configuration Change Management** - The CM process of administering formal change management procedures. Formal process activities include receipt and logging of new document work products or Change Requests (CRs) to current baselines, coordination and conduct of reviews of proposed changes, approval of changes by an approval authority, follow-up corrective actions, incorporation of changes into current baseline, and formal release for project decision making.
- **Configuration Identification**—"(1) The systematic process of selecting the product attributes, organizing associated information about the attributes, and stating the attributes. (2) Unique identifiers for a product and its configuration documents. (3) The configuration management activity that encompasses selecting configuration documents; assigning and applying unique identifiers to a product, its components, and associated documents; and maintaining document revision relationships to product configurations" (MIL-STD-61A, p. 3–5).
- **Configuration Item**—A System or Entity at any level of abstraction such as Equipment, Hardware, Software, or Courseware item that has been separately identified and designated for: (1) development under formal Configuration Control Management procedures and version control and (2) assignment of model numbers and serial numbers.
- **Configuration Management**—"A management process for establishing and maintaining consistency of a product's performance, functional, and physical attributes with its requirements, design and operational information throughout its life. (MIL-STD-61A, p. 3–5)
- **Configuration Status Accounting (CSA)**— A CM process that: (1) implements one of the four CM functions, (2) formally documents the state of the Developmental Configuration of a system or product at a specific instance in time, (3) maintains and accounts for current configuration status of the Developmental Configuration and any of its baselined documents, (4) records the current disposition of Acquirer Furnished Equipment (AFP) and contract documents, and (5) archives and tracks approved Change Requests (CRs).
- **Data**—Information documented in any form of media such as documents, audio and video recordings, photographs, drawings, personal notes, measurements, and so forth that contain Enterprise, contractual, security, programmatic, technical, and personnel information that can be read, deciphered, aggregated, transferred, and analyzed for decision making.
- Developmental Configuration: Refer to the Chapter 12 *Definition of Key Terms*.
- **Effectivity**—Designation defining the point in time, an event, or a product range (e.g., serial, lot number, model, date) at which changes or variances to specific products are to be effected. The authorized and documented point of usage for a specific configuration of a part/assembly/installation, etc. (FAA SEM, 2006, Vol. 3, p. B-3).
- **Firmware**—"The combination of a hardware device and computer instructions or computer data that reside as read-only software on the hardware device. The software cannot be readily modified under program control" (DAU, 2012, p. B-85.).
- **Hardware Configuration Item (HWCI)**—"An aggregation of hardware that satisfies an end use function and is designated for separate configuration management by the Acquirer" (MIL-STD-498, p. 5).
- **Item**—"A nonspecific term used to denote any product, including systems, materiels, parts, subassemblies, sets, accessories, etc." (MIL-HDBK-61A (SE), 2001, p. 3–8).
- **Legacy System**—An existing system design that has been verified and subjected to field use and may or may not be operational.
- **Line Replaceable Unit (LRU)**—"An essential support item removed and replaced at field level to restore an end item to an operationally ready condition" (DAU, 2012, p. B-144).
- **Make–Buy–Modify Decisions**—Technical decisions that determine whether to develop an item or lower level item internally, procure as a COTS/NDI or subcontract item from an external vendor, or procure an item from an external vendor and modify internally to meet Entity Development Specification (EDS) requirements.
- **Non-conformance**—"The failure of a unit or product to meet a specified requirement" (MIL-HDBK-61A (SE), 2001, p. 3–8).
- **Non-Developmental Item (NDI)**—A COTS product that has been modified or adapted (i.e., customized or tailored) by its developer to meet procurement specification requirements for application in a prescribed operating environment.
- **Out-of-the-Box Functionality**—Specified capabilities and levels of performance inherent to a COTS product or NDI as specified in a vendor catalog or Production Specification.

- **Outsourcing**—A business decision to procure systems, products, or services from external organizations based on cost avoidance resource availability, or other factors.
- **Product Breakdown Structure (PBS)**—The physical hierarchical structure of a system, product, or service.
- **Release**—A formal CM event announced by a formal letter and/or electronic notification stating that an item has been formally baselined and placed under formal CM Control or an updated version is now approved for decision making.
- **Technical Data Package (TDP)** "A technical description of an item adequate for supporting an acquisition strategy, production, engineering, and logistics support. The description defines the required design configuration and procedures required to ensure adequacy of item performance. It consists of all applicable technical data such as drawings and associated lists, specifications, standards, performance requirements, quality assurance provisions, and packaging details." (MIL-HDBK-61A (SE), 2001, p. 3-10)
- **Variance**—A non-compliance such as an *error* or *inaccuracy* in information; physical items such as EQUIPMENT, HARDWARE, SOFTWARE, or COURSEWARE; material composition; or workmanship practices that deviates from a standard reference for compliance such as a specification, drawing, and so forth.

Author's Note 16.1

In previous discussions, our semantics have used terms such as *entities* and *components* that have contextual definitions. The CM community uses the term *item* to connote physical entities in the same manner SEs refer to *components*. In this chapter, we will use *item* due to the chapter's context. When you read *item*, equate it to *entities* or *components*.

16.2 ITEMS: BUILDING BLOCKS OF SYSTEMS

Principle 16.1

Configuration Items (CIs) Principle

Every physical component within a system as an *item* by CM. Some *items* are designated as CIs for internal development; others as COTS products or NDIs available for procurement from external vendors for procurement.

Depending on the size and complexity of the system or item being developed, Product Development Teams (PDTs) such as Integrated Product Teams (IPTs) are assigned roles and responsibilities for specifying, designing, developing, integrating, and verifying various components within the system. This presents some significant challenges:

- How do you partition the architecture of the EQUIPMENT Element into a PBS with multiple levels of Physical CIs?
- How can projects establish semantics that enable them to communicate with others about the types of items being developed internally or procured from external vendors?
- How can project teams communicate about the evolution of system items that go through various stages from abstract system specification entities into physical components used to build the system?

The solution resides in integrated "building blocks" referred to as items, CIs, HWCIs, and CSCIs. Each of these building blocks represents *semantics* used to identify abstract entities within a system or product and their evolution from the System Performance Specification (SPS) or Entity Development Specification (EDS) to deliverables.

16.3 UNDERSTANDING CONFIGURATION IDENTIFICATION SEMANTICS

Configuration identification knowledge for most SEs typically comes from informal exposure via verbal discussions in meetings and On-the-Job Training (OJT) over many years. Most engineers:

- Have little or no formal training in the four functions of CM: Configuration Identification, Configuration Control, CSA, and Configuration Audits.
- Are self-taught through observation and experiential knowledge of general CM standards and concepts.

As a result, some may perceive themselves to be configuration experts.

These rudimentary skills provide basic insights about system architectural configuration decisions. Instead of seeking *insightful* guidance from a competent CM, technical leads exercise their authority, make decisions, and wander down self-directed decision-making paths leaving the project engineer, LSE, or CM and others to needlessly expend valuable time and resources contending with the Law of Unintended Consequences and cleaning up the after effects of their poor decisions. So, to *minimize* the confusion and chaos, let's begin by introducing some key terms. Figure 16.1 depicts entity relationships to support our discussions.

16.3.1 Configuration Items (CIs)

When a System Developer decides to develop a major item such as a PRODUCT, SUBSYSTEM, OR ASSEMBLY, or SUBASSEMBLY in-house, the project designates the *item* – entity or component - as a CI. Since the item will be new,

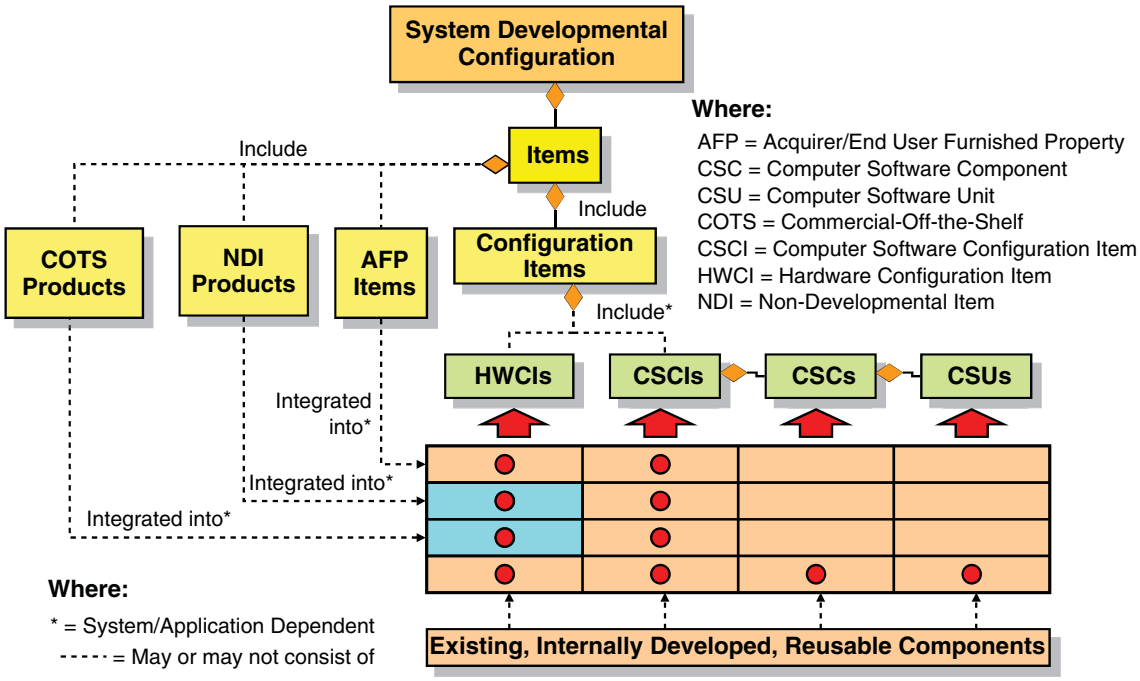


Figure 16.1 System Configuration Identification Elements

has an *unproven* design solution, and requires verification, CIs typically require a specification that *specifies* and *bounds* its capabilities and performance.

A CI, as a major item such as a PRODUCT, SUBSYSTEM, OR ASSEMBLY, integrates lower level items - entities or components - that may consist of combinations of AFP, COTS products, NDIs, and two other types developed by the System Developer in-house:

- Hardware Configuration Items (HWCIs)
- Computer Software Configuration Items (CSCIs)

16.3.2 HWCIs and CSCIs

HWCIs are major hardware items and CSCIs are major software applications designated for formal configuration control. Using an automobile as an example:

- HWCIs might consist of end use Systems, Products, or Assemblies such as engine, mission computer system, entertainment system, and so forth.
- CSCIs might consist of the end use software operating on computers such as the mission computer, Anti-Lock Braking System (ABS) distributed computers, and so forth.

Referring to Figure 16.1, an HWCI or CSCI may consist of COTS products, NDIs, AFP internally developed or legacy items, or combinations of these.

- For items designated as an HWCI, its requirements are documented in an HWCI Requirements Specification (HRS). Typically, the scope of each HRS addresses only one HWCI.
- For items designated CSCI, its requirements are typically documented in a CSCI Software Requirements Specification (SRS). Typically, the scope of each SRS addresses only one CSCI.

These guidelines for HWCIs and CSCIs may evolve over time. Always consult your contract and Project Engineer for guidance.

A Word of Caution 16.1

Specification of CSCIs and HWCIs

Traditionally, one specification documents an HWCI or CSCI. In that context, a System Acquirer can easily acquire a single HWCI or CSCI. For odd reasons, some Enterprises today document multiple HWCIs or CSCIs within a single specification such as an HRS or SRS. That may be acceptable for strictly internal development, especially software. However, you certainly would not send an SRS specifying all your CSCIs to your competitor when requirements for one and only one CSCI are required – yes, competitors do team together. Think smartly before you decide to document multiple CSCIs within a single SRS. Traditional Engineering practice says avoid it!

16.3.3 CI Boundaries

CI Boundary Principle

Principle 16.2 CIs are constrained to the confines of a physical boundary such as a SUBSYSTEM, ASSEMBLY, SUBASSEMBLY, HWCI, CSCI, or PART; they do not physically exist beyond that boundary.

Items and CIs should not be partitioned across physical device boundaries; this is a violation of Principle 16.2 and addressed later in Figure 27.8. In general, CIs:

- Are bounded by an SPS, EDS, HRS or CSCI SRS.
- Reside within the boundaries of a physical item—such as SYSTEM, SUBSYSTEM, ASSEMBLY, or SUBASSEMBLY such as a computer system, printed circuit board, or software application.
- Must be verified for compliance against their respective HRS or SRS.

To illustrate this point, consider the hypothetical example below.

Computer Word Processing Software Application Example

Example 16.1 Assume you are tasked to develop a word processing software application for an EQUIPMENT Element desktop computer system (HWCI) and a printer (HWCI) being developed by your project. The team designates the word processor application as a CSCI. When installed properly, the Word Processing CSCI resides on the Desktop Computer System (HWCI), not the Printer (HWCI). When the word processor User decides to print a document, the Word Processor CSCI invokes the Desktop Computer System's (HWCI) operating system services to transfer the document to the the Printer CSCI for printing.

16.3.4 Firmware

Some processor-based applications such as Single-Board Computers (SBC) employ software encoded into an Integrated Circuit (IC) referred to as *firmware*. Firmware, which is nonvolatile memory devices, may be implemented as single-use, read-only, or reprogrammable devices as shown in Figure 16.2.

Initially, a software program to be executed by an SBC is developed as a CSCI software application and debugged on laboratory prototype SBC hardware using emulators and other devices. Emulator hardware on software development devices usually have a cable with connector that plugs into the socket where a microprocessor will reside on the PC board. As a result the emulator – emulating the actual processor - can exercise the SBC's I/O, memory, and so forth as part of the software debugging process.

When the software application reaches *maturity* and is ready for final installation on the SBC, the CSCI's code is electronically programmed into the firmware device, either separately or once installed on the SBC. Once programmed, the firmware is:

- Designated as an HWCI.
- Assigned a part number, serial number, version, and date.

Both the CSCI and HWCI are controlled in accordance with the formal CM change management procedures.

Heading 16.1 The preceding discussions introduced the semantics of configuration identification. Our follow-on discussions illustrate why the referential nature of Configuration Identification semantics when applied to levels of abstraction sometimes result in confusion.

16.3.5 Configuration Semantics Synthesis

To understand how Configuration Identification semantics relate to multi-level system architectures, Figures 16.1 and 16.3 depict the Entity Relationships (ERs). Table 16.1 provides a listing of ER rules that govern the implementation of this graphic.

Heading 16.2 At this juncture, we have established the semantics framework for understanding CIs. The question is: *How do we determine which items should be designated as CIs*? This brings us to our next topic on the Selection of CIs.

16.3.6 Selection of CIs

The preceding discussions established that CIs are typically items, especially major items, developed internal to the System Developer. While this is an important criterion, CIs often require additional considerations. The best approach for selecting CIs is to simply establish a set of selection criteria. Then, perform a *reasonableness check* to ensure that the selection:

1. Is logical.
2. Provides the proper visibility concerning technical, cost, and schedule performance; risk; and configuration control.
3. Exposes development activities at a level that can be used for assessing risk.

Some organizations establish specific criteria for selecting CIs that go beyond simply deciding to develop an item internally. These decisions should be made in collaboration with a project's CM.

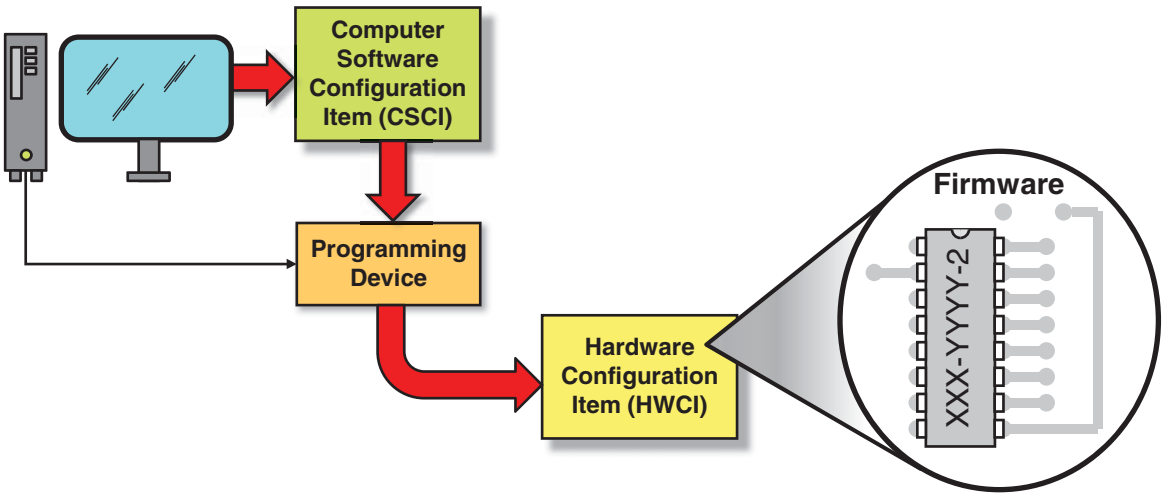


Figure 16.2 Evolution of Firmware from Software to Hardware

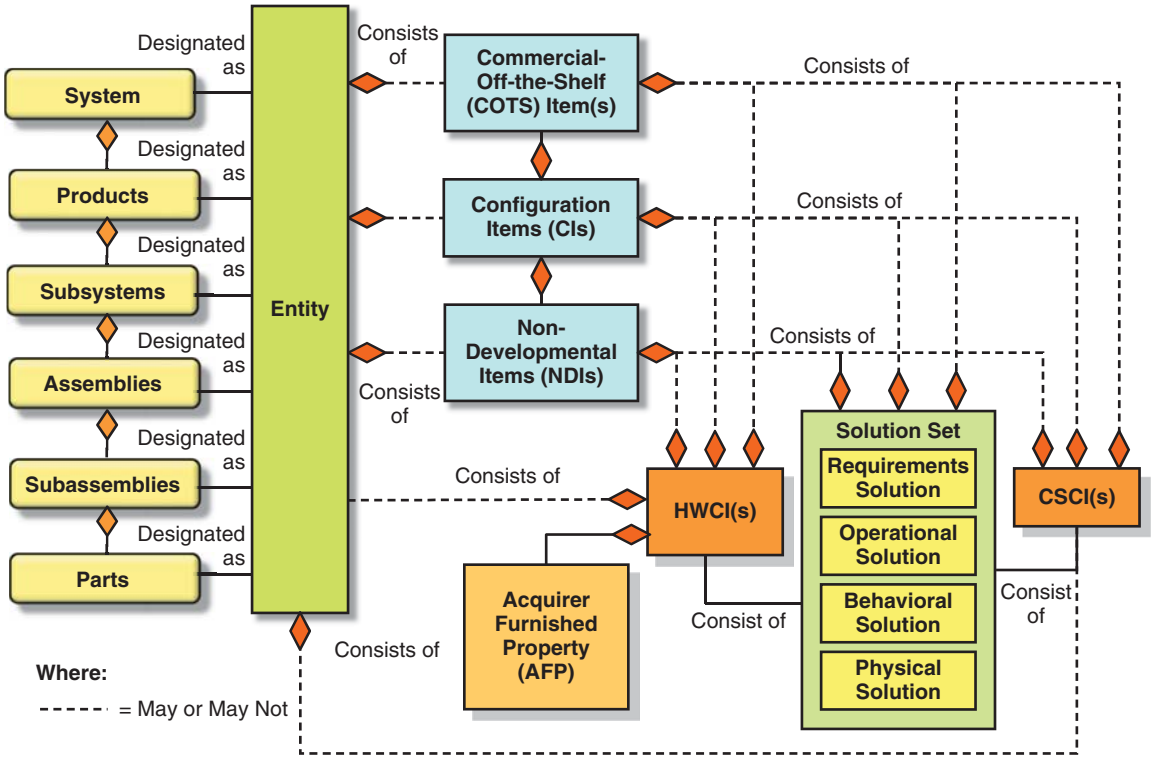


Figure 16.3 Item/CIs Compositional Entity Relationships (ERs)

The selection of CIs often varies from one organization or business domain to another. To standardize thinking about selecting CIs, MIL-HDBK-61A, for example, offers the following guidance in selecting CIs:

1. Does the item implement critical capabilities (e.g., security protection, collision avoidance, human safety, nuclear safety)?
2. Would CI designation enhance the required level of control and verification of these capabilities?
3. Will the item require development of a new design or a significant modification to an existing design?
4. Is the item computer hardware or software?
5. Does the item incorporate unproven technologies?
6. Does the item have an interface with a CI developed under another contract?

TABLE 16.1 Configuration Semantics Rules

Rule	Title	Configuration Identification and Development Rule
16-1	Items	Every *physical component* within a system, regardless of level of abstraction, is referred to as an *item*
16-2	CIs	*Items* at all levels of abstraction selected for *internal* development are referred to as CIs
16-3	CIs	*Items* originate from several types of sources: 1. Replicated from existing legacy component designs 2. Acquired as COTS, NDI, or AFP components or materials 3. Acquired as COTS, NDI, or AFP components or materials and modified in-house 4. Developed as new designs such as an HWCI(s) or CSCI(s)
16-4	CI Composition	A CI's composition may consist of one or more COTS products, NDIs, HWCIs, CSCIs, AFPs, or combinations thereof
16-5	HWCIs and CSCIs	Develop an HRS for each HWCI; develop a CSCI SRS for each CSCI
16-6	CI Solution Domains	Each CI, HWCI, and CSCI is characterized by its Four Domain Solutions (Chapters 11 and 14) Requirements Domain, an Operations Domain, a Behavioral Domain, and a Physical Domain Solution
16-7	CSCIs	The Product Structure for each CSCI consists of at least two or more Computer Software Components (CSCs), each of which consists of at least two or more Computer Software Units (CSUs)
16-8	CI Ownership	Each CI, HWCI, and CSCI must be assigned to an accountable individual or team responsible for its design, development, and integration and verification

7. Can the item be readily marked to identify it as a separate, controlled item?
8. Does the item interface with a CI controlled by another design activity?
9. Will it be necessary to have an accurate record of the item's exact configuration and the status of changes to it during its life cycle?
10. Can (or must) the item be independently tested?
11. Is the item required for logistic support?
12. Is it, or does it have the potential to be designated for separate procurement?
13. Have different activities have been identified to logistically support various parts of the system?
14. Is the item at an appropriate level for Government configuration control? (MIL-HDBK-61A , Table 5.2. p. 5–8)

16.3.7 Configuration Identification Responsibility

Configuration Identification, as an informed, multi-discipline, decision-making process, requires collaboration with stakeholders. Contrary to what many people believe, it is not a decision by one individual exercising their discretionary authority without inputs from the key decision stakeholders. As the multi-level system or entity architectures evolve, the CM, Software Configuration Manager (SCM), LSE, development team leads, and others collaborate to establish criteria for identifying items and CIs and avoid costly corrective actions.

At this point, we have established the basic set of Configuration Identification semantics and how they are applied to multi-level system architectures. These discussions highlighted the need during internal development to prepare an EDS for each CI, HWCI, and CSCI. For the first article produced as part of the Developmental Configuration, this is a straightforward process. However, two key questions emerge:

- How are these specifications maintained for production systems that evolve over time as new capabilities and refinements are added to established designs as new models or versions?
- How do these impacts affect systems or products that are already fielded but may require retrofitting?

This brings us to our next topic, Configuration Effectivity.

16.3.8 Configuration Effectivity

Production systems or products may evolve over several years as newer technologies, capabilities, and improvements are incorporated into the evolutionary System Design Solution. As such, the capabilities and CIs may change. The question becomes: *How do we delineate the changes in configuration to a given CI, HWCI, or CSCI?* CM addresses these configurations via a concept referred to as *configuration effectivity*?

Every CI, HWCI, and CSCI is formally assigned a *unique* identifier that delineates it from all others. Examples include (1) model number and (2) serial number. As a SYSTEM or

PRODUCT's physical configurations evolve over the years requiring changes to the SPS, HRS, or SRS, there is a need to delineate those items from their predecessors. In general, organizations simply reference the effectivity beginning with Serial Number XXX; others append a "dash number" to the model number such as Model 123456-1 to indicate a specific version. Most Enterprises affix barcode labels to CIs, HWCIs, and CSCIs to facilitate automated scanner version or configuration tracking.

Versioning provides the System Developer a couple of options: (1) it allows *evolutionary* tracking of a product line over its life span and (2) it provides a means to account for special customizations delivered to an Acquirer. In lieu of model numbers and versioning, some vendors may be required to designate and label the item with contract numbers, serial numbers, and other information. Here's an example.

Example 16.2 **SYSTEM or ENTITY CI Labeling**

Developers of military systems may be required to install CI labels in accordance with the current version of MIL-STD-130. Commercial product labeling may be required in accordance with government or industry standards such as Underwriters' Laboratories (UL).

Changes are versioned via new drawings, part lists, etc. and labeled accordingly. *But what happens to the SPS, HRS, or SRS that serves as the source or originating requirements for a product line?* This brings us to our next topic, Effectivity-Based Specifications.

16.3.9 Effectivity-Based Specifications

During the development of a SYSTEM or PRODUCT, multi-discipline SEs prepare EDSs for PRODUCTS or SUBSYSTEMS, HRSs for HWCIs, and SRSs for CSCIs that form the Developmental Configuration. Although cost is a key constraint, most first article systems or products may not represent the most cost-effective solution due to schedule and other constraints. *First articles* are simply an Engineering solution that complies with specification requirements. Typically, each deliverable is assigned contract, model, and serial numbers.

If a system is planned for production, Product Engineering focuses on reducing the recurring per unit production cost via design improvements, component and material selection and procurement, manufacturing methods, and so forth. Subsequently, the improvements culminate in a revised EDS with effectivity beginning with S/N XXX forward.

After production starts, CIs, HWCIs, and CSCIs evolve over time. Whereas during the original system development, revisions to the Developmental Configuration specifications were issued when changes occurred. So, when production item changes occur, you have to revise not only the specification level but also the scope of the S/N effectivity.

When this occurs, SEs are confronted with a decision. *Do we (1) create a new specification unique to a specific configuration effectivity - models and S/Ns or (2) update the original specification to delineate which requirements apply to specific models and serial number effectivity within the document*? The update approach enables requirements to be kept in a single document but may become confusing and difficult to manage over time.

16.4 CONFIGURATION ITEM (CI) IMPLEMENTATION

16.4.1 Aligning CIs with the Specification Tree

Once CIs are identified, the key question is: *How do you designate their location within the system's product structure?* The answer is: CIs should be *explicitly* identified in the Specification Tree based on the component it specifies within the System Architecture's hierarchical PBS as shown in Figure 16.4.

Here, the System Development Team (SDT) analyzes the SPS requirements to create the SYSTEM Level architecture, as shown in the lower left corner of the graphic. The architecture consists of PRODUCTS A and B. PRODUCT B, which will be developed internally, is designated as a CI consisting of Items B_1 through B_3.

As the system architecture evolves, the specification tree changes accordingly as shown on the right side of the graphic. Based on the designation of CIs and items, EDSs are identified:

- Requirements for PRODUCT A are specified in the PRODUCT A Development Specification.
- Requirements for PRODUCT B are specified in the Product B Development Specification.

As the PRODUCT B's architecture evolves, Items B_1–B_3 are identified as architectural components. Trade studies are conducted for each item to determine if they are available as COTS products, developed internally, procured externally, and so forth:

- Since specific PRODUCT B Development Specification requirements can be met by two separate COTS products, Items B_1 and B_2 are identified and will be procured as COTS products from different vendors.
- The remaining PRODUCT B Development Specification requirements will be tentatively allocated to Item B_3. The requirements are analyzed and a decision is made to develop Item B_3 internally as a CI. Thus, the requirements allocated and flowed down from the Product B Development Specification to Item B_3

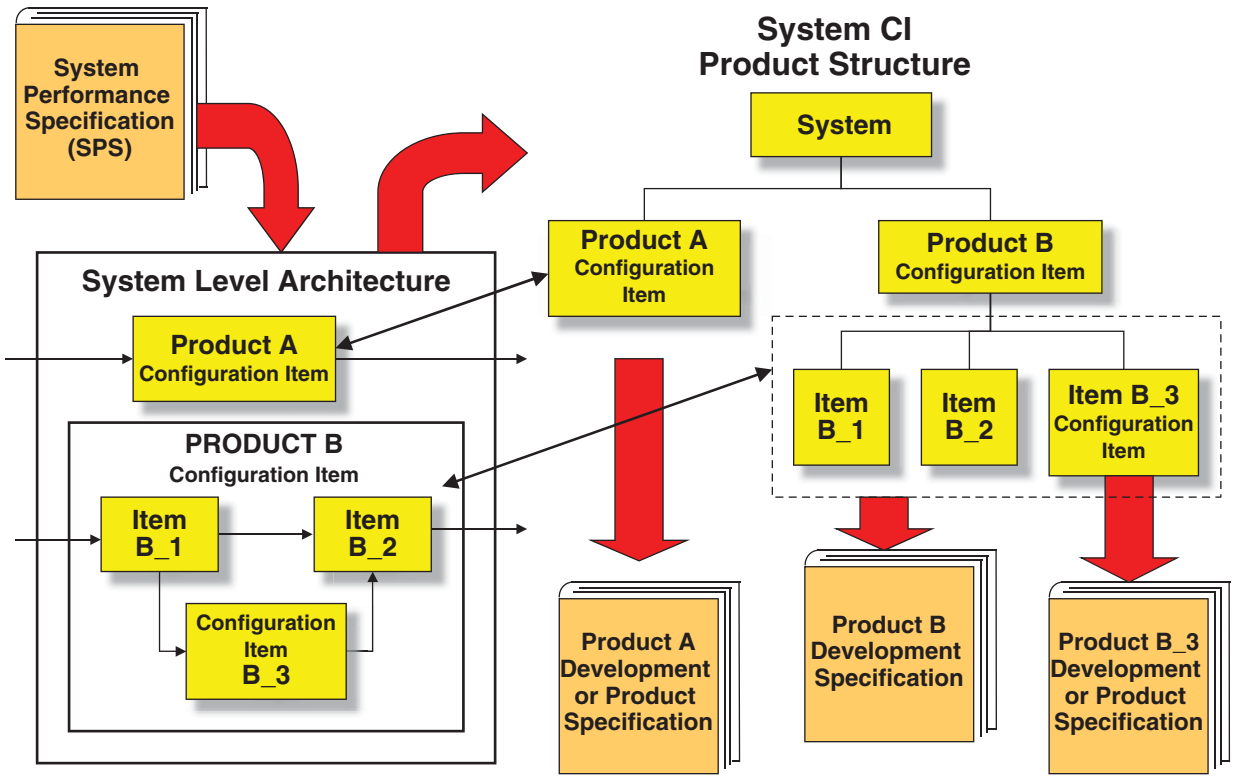


Figure 16.4 System Architecture and its Product Breakdown Structure (PBS) Specification Tree Configuration Documentation

will be incorporated into the Item B_3 Development Specification.

16.4.2 Assigning Ownership of CIs and Items

CI Ownership and Accountability Principle

Principle 16.3 Every Configuration Item (CI) should be assigned to an owner accountable for its design, implementation, integration & test, verification, and validation.

As the Specification Tree evolves, accountability for CI development should be assigned to owners such as PDTs or IPTs. Figure 16.5 provides an illustrative example. Note how the SYSTEM's architecture decomposes along PBS lines. This is a key point, especially the operative word *product*.

For programs that employ PDTs or IPTs, each IPT focuses on "product" development and collaborates with IPTs developing items that interface to their assigned product. For example, IPT #1 *collaborates* with IPT #2 concerning mutual interface definition and design compatibility and interoperability issues.

Accountability for developing one product is assigned to one and only one PDT or IPT. Depending on the size, complexity, and risk of the multi-level items, an IPT may be assigned accountability for *items* as illustrated in Figure 16.5. Accountability for developing PRODUCTS A and B, which have a moderate degree of complexity and risk, is assigned to IPT #1. In contrast, accountability for PRODUCT C is assigned to IPT #2 due to its *complexity* and *risk*. This brings us to a final point.

Projects often get into trouble because they establish a *functional* project organizational team structure – EE, ME, SwE, and so forth - *instead* of using the System Architecture as the basis to derive the project team structure. With the exception of SE, project functional organizations became outdated decades ago.

Project Organizations

Example 16.3 If a project's organizational structure were developed first, PRODUCTS A and B, which have totally separate "end uses," may be *arbitrarily* and *amateurishly* bundled together and labeled as a PRODUCT or SUBSYSTEM. Net result—A team is assigned to develop the specification requirements for two entities that share no interfaces or have any relevance to one another.

Remember that SE principles require a PRODUCT, SUBSYSTEM, ASSEMBLY, etc. to consist of an integrated set of

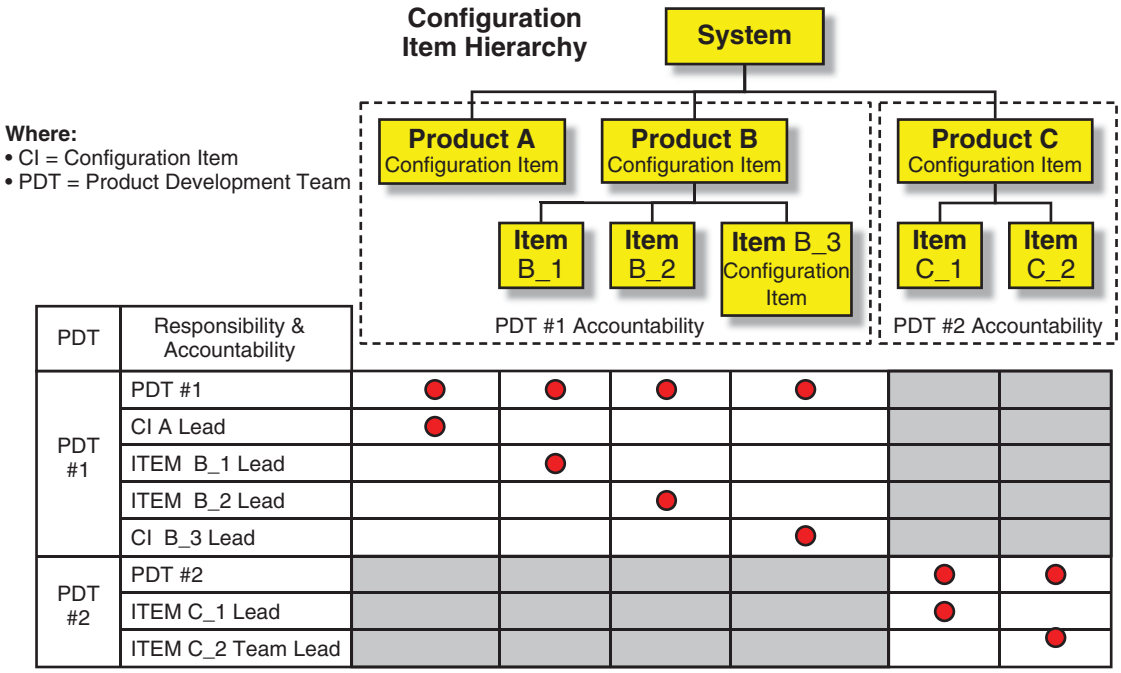


Figure 16.5 Configuration Item (CI) Accountability Assignments Matrix

components to produce an outcome that is greater than the individual components can achieve—for example, *emergence* (Chapter 3).

The reality is projects develop "end use" SYSTEMS, PRODUCTS, SUBSYSTEMS, and so forth that require multi-discipline Engineering, not "stovepipe" disciplines. As a result, projects organizational structures are typically established along deliverable product lines. Consider the following example.

16.4.3 Recognizing Types of Architectural Item Boundaries

The Industrial Revolution introduced new concepts for standardizing and reproducing modular and interchangeable components via *repeatable* and *predictable* methods. *Standardization* enables us to leverage the benefits of *economies of scale* by simplifying the number of components and, in turn, reducing inventory costs. Our discussions about SYSTEM, PRODUCT, SUBSYSTEM, ASSEMBLY, SUBASSEMBLY, and PART levels of abstraction expound on these themes.

The concept of *modularity* can easily lead to an SE mind-set that *all* items and CIs are constructed as modular *plug and play* boxes. However, there are systems that require integration across the traditional "box" boundaries. We tend to think in terms of pure hierarchical structures consisting of peer level SUBSYSTEMS, ASSEMBLIES, and so forth. Hierarchically, this is correct. However, some of the peer level SUBSYSTEMS or ASSEMBLIES are ENABLING SYSTEMS providing capabilities that support other MISSION SYSTEM peers that perform specific missions.

In general, systems often consist of two classes of PRODUCTS or SUBSYSTEMS:

- *Mission-specific* PRODUCTS or SUBSYSTEMS – MISSION SYSTEMS.
- *Infrastructure* PRODUCTS or SUBSYSTEMS – ENABLING SYSTEMS

Figure 16.6 illustrates this type of architecture. Consider the following example.

Example 16.4 **SUBSYSTEMS Versus ENABLING Subsystems**

Office building systems have well-defined architectural "box" boundaries comprised of floors and office areas unique to Enterprises supported by plumbing and electrical; Heating, Ventilation, and Air Conditioning (HVAC); communications; and others represent the building's *infrastructure* systems as shown in Figure 16.6. As illustrated, the *infrastructure* or ENABLING SUBSYSTEMS transcend and support each of the MISSION Subsystems. Fuel, electrical, communications, and air subsystems traverse the entire structure—for example, propulsion, passenger cabin, and storage subsystems—of aircraft and automobiles.

16.4.4 Multiple Instances of CI Implementation

Although we tend to think that every *item* in a system is unique, SYSTEMS AND PRODUCTS often have multiple

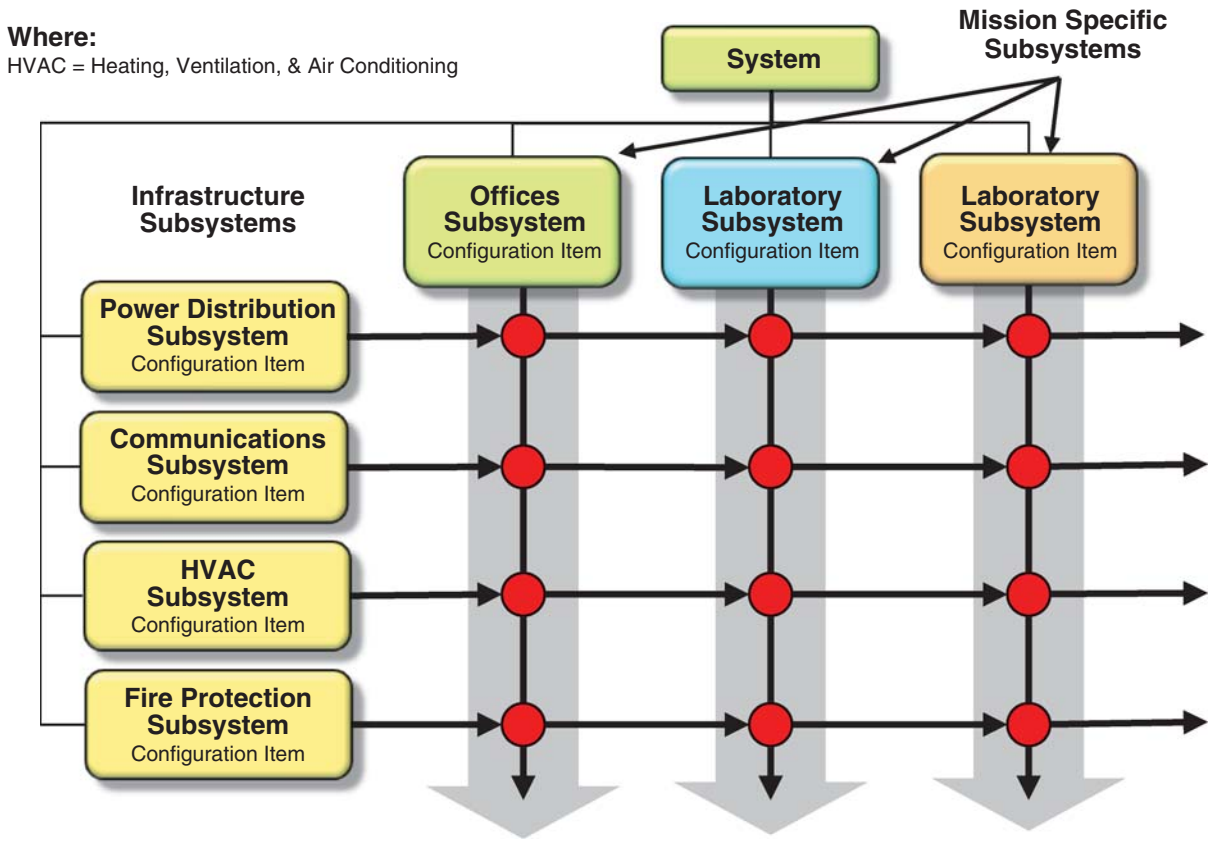


Figure 16.6 System Architecture Illustrating Line versus Crosscutting Configuration Items (CIs)

instances of a single CI throughout the system. One of the roles of SE and the SEIT is to reduce the Total Cost of Ownership (TCO) – development and life cycle costs, schedule, and risk. You do this by investigating the *evolving* System Design Solution and searching for opportunities to *standardize* components and interfaces. The bottom line is: avoid "reinventing the wheel" by creating specialized CI designs that can be satisfied by one common CI design. *How can this be accomplished?*

One method for standardizing for *design reuse* is to simply perform a Domain Analysis of the SYSTEM or PRODUCT to identify similar design needs as opportunities for leveraging a single, multi-purpose design solution in hardware or software. A common example occurs with standardizing interfaces such as communications data protocols, electrical signals, and mechanical layouts types.

16.5 DEVELOPMENTAL CONFIGURATION BASELINES

System development requires translation of *abstract* SPS requirements into a deliverable physical solution. The translation decomposes or refines abstract complexity into more manageable lower levels of detail. Ultimately, a detailed design *emerges* and *matures* to a point whereby design requirements such as drawings and software designs are *sufficient* in detail to support procurement, fabrication, and coding of all items in the system.

Lower levels of design decisions, as refinements of higher level decisions, are totally dependent on the *maturation* and *stability* of the higher level design decisions (Principle 14.2). Otherwise, the entire System Design Solution evolves into multi-level *confusion* and *chaos*. So, as higher level decisions stabilize, it is important to capture and control the state of the evolving system design solution referred to as the Developmental Configuration.

Principle 16.4

Primary Configuration Baselines Principle

CM views every System as having three primary Developmental Configuration Baselines:

1. System Requirements or Functional Baseline
2. Allocated Baseline
3. Product Baseline

Systems in production have a Production Baseline.

The Developmental Configuration is characterized by a series of configuration "snapshots" that capture the evolving System Design Solution at strategic stages of maturity. Baseline management of the Developmental Configuration provides a mechanism for SEs to maintain "*intellectual control*" (Principle 1.3 - McCumber and Sloan, 2002, p. 4) of the *evolving* and *maturing* System Design Solution via configuration change management. Therefore, the scope of the Development Configuration spans the time period from Contract Award until SYSTEM or PRODUCT delivery and acceptance or is subsequently committed to a Production Baseline.

Principle 16.5 **SE Design Configurations Principle**

Every SYSTEM or ENTITY at any level of abstraction has eight SE design configurations: As Specified, As Allocated, As Designed, As Built, As Verified, As Validated, As Maintained, and As Produced that must be current and consistent with each other.

From a System Development perspective, there are a series of System/Product Life Cycle Phase technical reviews or events that assess the progress, status, and maturity of the evolving System Developmental Configuration as a condition for committing resources to begin the next stage of development. Table 16.2 provides a listing of key *staging* or *control points*.

Let's briefly synopsize each of these SE configurations. Each configuration is reviewed, approved, baselined, released, and controlled via formal CM procedures.

16.5.1 "As-Specified" Developmental Configuration

Principle 16.6 **SPS Ownership Principle**

As a contract element, the System Acquirer *owns* and *controls* the contract SPS; the System Developer maintains the SPS in accordance with the contract direction *authorized* and *provided* by the Acquirer Contracting Officer (ACO).

The "As-Specified" Configuration represents the *incremental* state of the evolving SYSTEM, SUBSYSTEM, ASSEMBLY, and others' specification requirements, *collectively* and *individually*, that comprise the System Requirements Baseline of the Developmental Configuration as it evolves over time. The initial instance of the "As-Specified" Configuration is established as the System Requirements Baseline by some organizations based on formal review and approval of the SPS by the System Acquirer and System Developer at the System Specification Review (SSR) (Chapter 18).

16.5.2 "As-Allocated" Configuration

The "As-Allocated" Configuration represents the incremental state of the evolving *incremental* state of the evolving SYSTEM, SUBSYSTEM, ASSEMBLY, and others' specification requirements allocations to items within their respective architecture. For example, allocation of:

- SPS requirements to SUBSYSTEMS in the SYSTEM Level architecture
- SUBSYSTEM specification requirements to ASSEMBLIES in the SUBSYSTEM Level architecture

16.5.3 "As-Designed" Developmental Configuration

The "As-Designed" Configuration represents the incremental state of evolving SYSTEM, SUBSYSTEM, ASSEMBLY, and others' detailed design documentation derived from their respective specifications in the System Requirements Baseline. The initial instance of the "As-Designed" Configuration begins with formal review and approval of the SYSTEM Level Architecture by some organizations at the System Design Review (SDR) (Chapter 18).

TABLE 16.2 Key Developmental Configuration Staging Points as a Function of System/Product Life Cycle Phase

System Life Cycle Phase	Process	Developmental Configuration SE Baselines	Technical Review
System Development	System Design	"As Specified"	System Specification Review (SSR)
	System Design	"As Allocated"	System Design Review (SDR)
	System Design	"As Designed"	Critical Design Review (CDR)
	Component Procurement and Development	"As Built"	Component Verification
	System Verification	"As Verified"	System Verification Review (SVR)
	System Validation	"As Validated"	
System Production	Production Process	"As Produced"	Production Readiness Review (PRR)
System Operations, Maintenance and Sustainment (OM&S)		"As Maintained"	Fielded System

16.5.4 "As-Built" Developmental Configuration

The "As-Built" Configuration represents the state of the Developmental Configuration of any *physical* entity such as SYSTEM, SUBSYSTEM, ASSEMBLY, and others that have been:

1. Procured, modified, and/or developed internally or externally.
2. Physically labeled with Model, Serial Number (S/N), and version designations.
3. Formally verified for compliance to its SPS, EDS, or design requirements such as engineering drawings, etc.

16.5.5 "As-Verified" Developmental Configuration

The "As-Verified" Configuration represents the incremental state of the multi-level Developmental Configuration documentation and physical system at completion of its formal System Verification Review (SVR) (Chapter 18). At this stage, the "As-Specified," "As-Designed," and "As-Built" Configurations should be *identically consistent* with each other, *complete*, and *compliant* with the multi-level System Requirements Baseline specification requirements.

16.5.6 "As-Validated" Developmental Configuration

The "As-Validated" Configuration represents the state of the Developmental Configuration documentation and physical system as validated by the User or an Independent Test Agency (ITA) representing the User during the Operational Test and Evaluation (OT&E) under prescribed field operating environments and conditions.

16.5.7 "As-Maintained" Developmental or Production Configuration

The "As-Maintained" Configuration represents the state of the Developmental Configuration or Production Configuration of the fielded system or product as maintained by the User or their designated representative. From a User, SE, and a CM perspective, *failure* to keep the "As-Maintained" Configuration documentation current and synchronized with the physical system or entity is *a major risk* area, especially for developmental systems planned for production.

Author's Note 16.2

Maintenance of Fielded System Configuration Documentation

The "As-Maintained" Configuration is a crucial staging point for SE. It is not *uncommon* for the User to forgo maintaining SYSTEM or PRODUCT documentation due to the lack of discipline, management, or budgets. The result is a physical System or Product that does not *identically match* its configuration documentation, which is a major risk factor for the System Developer.

16.5.8 "As-Produced" Production Configuration

The "As-Produced" Configuration represents the state of the Production Baseline used to manufacture the system or product in production quantities. The initial instance of the "As-Produced" Configuration is established at the Production Readiness Review (PRR) (Chapter 18).

16.5.9 Developmental Configuration Staging or Control Points

As the Developmental Configuration evolves through a series of design and development stages, it is important to establish agreement among the Acquirer, User, and System Developer about the evolving and maturing System Design Solution. Depending on the type of contract and what authority each party has in the decision-making process, we do this via *staging* or *control points*.

Staging or control points, which consist of major technical review events, are intended to represent stages of maturity of the evolving system design solution as it advances into lower levels of abstraction or detail over time. We do this via a series of Configuration Baselines.

From a CM perspective, there are four configuration baselines:

- System Requirements or Functional Baseline
- Allocated Baseline
- Product Baseline
- Production Baseline

Note that our discussions identified two perspectives of the evolving and maturing Developmental Configuration: (1) an SE configuration perspective and (2) a CM perspective. Despite the semantics, both sets correlate as illustrated in Table 16.2.

16.5.10 Configuration Identification and Baseline Management Summary

During our Configuration Identification discussions, we:

- Introduced a set of CM terms such as item, CI, COTS, NDI, and AFP.
- Identified options available for developing items and CIs.
- Noted that CIs are constrained to the boundaries of itself as a physical entity.
- Addressed issues related to Configuration Effectivity.

This brings us to our next topics concerning how components are selected for Items and CIs.

16.6 COMPONENT SELECTION AND DEVELOPMENT

The allocation of SPS and EDS requirements to lower level items is a *highly iterative* process driven by component selection decisions. In general, System Developers have to answer the question: *What is the best value, lowest cost, and acceptable risk approach to selecting components to meet contract requirements?*

1. Are there in-house, reusable component designs already available?
2. Do we procure commercially available components from external vendors?
3. Are there commercially available components that require only minor modifications to meet our requirements?
4. Do we procure commercially available components from external vendors and modify them in-house or have the vendor modify them?
5. Do we obtain components from the User as AFP?
6. Do we create the component in-house as new development?

Depending on the outcome of these questions and the *item's* specified requirements, the initial requirements may have to be reallocated to reconcile capabilities provided by commercial components.

Our discussion in this section focuses on Component Selection and Development Practices that drive SE decision-making. After a brief discussion of options available for system development, we introduce the concepts of COTS and NDIs. Next, we define a methodology that describes a decision-making method for selecting the strategy for component development. We conclude with a summary of driving issues that influence COTS/NDI selection.

16.6.1 Reducing System Costs and Risk

One of the key objectives of SE is to *minimize* TCO - development and life cycle - costs as well as risk. Achievement of these objectives requires insightful strategies that include selection of components at all levels of abstraction.

Most Engineers enter the workforce with noble aspirations to innovate and create *elegant* designs. Although this is partially true, it is also a reflection of misunderstood priorities. New design should be a resort after attempts to find existing components that meet specification requirements have been exhausted (Principle 4.19).

Design as a Last Resort Principle

Principle 16.7 System design of a new CI should be a last resort option after all practical efforts have been exhausted to identify legacy designs that are reusable COTS, NDI, or AFP components to meet entity requirements.

So, what should the priorities be? This brings us to Item Design Implementation Options and Priorities.

16.6.2 Item Design Implementation Options and Priorities

From a CM perspective, every item or entity within a SYSTEM or PRODUCT, regardless of level of abstraction, is generically referred to as an *item*. Items originate from at least seven different options available for Engineering the development of *physical components*:

Option #1: Procure Component(s) from an external vendor's catalog.

Option #2: Procure Vendor Component(s) and Customize In-House
Procure a component from a vendor's catalog and customize or adapt it in-house to comply with Engineering drawings.

Option #3: Procure Customized Component(s) from External Vendor
Procure a component from a vendor's catalog and pay them to modify or customize it in accordance with a procurement contract and specification, assuming the service is available.

Option #4: Reuse Legacy Component Design(s)
Procure parts, components, or raw materials from external suppliers. Then, Fabricate, Assemble, Integrate, and Test (FAIT) in-house by reusing existing, legacy designs.

Option #5: Design and Develop New Component(s) In-House
Design a new component in-house; procure parts, components, or raw materials from external suppliers; and FAIT in accordance with new component design requirements specified in engineering drawings.

Option #6: Subcontract New Component(s) Development
Design a component in-house and procure from an external vendor.

Option #7: Subcontract New Component Development
Procure component design and development from

an external System Developer in accordance with a performance specification.

A Word of Caution 16.2

Legacy Design Reuse Issues

Be advised that reuse of *existing* or *legacy* designs may present *legal* and *contractual* issues regarding the type of funding used to develop the product, data rights, and so forth. *Always* consult with your Enterprise's program, contracts, legal, and export control organizations for guidance in these areas.

Based on these design/procurement options, the System Design Solution for any *item* at any level of abstraction within the SYSTEM or PRODUCT may consist of one or more combination of these implementations. As a result, the COTS/NDI/new development composition of any design implementation depends on the application as illustrated in Figure 16.7. Observe that design implementation ranges from two extremes: 100% COTS to 100% new development. So, the challenge for the SEIT and component selection decision-makers is to determine the *right* combination of options that reduce the TCO - development and life cycle costs - as well as risk.[1]

Items procured from a vendor's catalog are referred to as COTS products. When COTS products are contracted to be *customized* or *modified* by the vendor for a specific application, they are referred to as Non-Developmental Items (NDIs). When the System Acquirer provides items – property - that may be available for consideration as a design option or required by contract to be modified and/or integrated into the deliverable system, they are referred to as Acquirer Furnished Property (AFP).

Author's Note 16.3

Acquirer Furnished Property (AFP)

When the System Acquirer serving as the User's contract and technical representative delivers AFP property to the System Developer, each item must be formally logged, tagged, tracked, and controlled in accordance with the contract Terms and Conditions (Ts & Cs). Planned modifications to AFP typically *require written authorization* by the ACO. Ensure that the Ts & Cs clearly delineate who is accountable for:

- Providing the complete set of AFP documentation
- Dealing with AFP maintenance and failures while in the possession of the System Developer

16.7 VENDOR PRODUCT SEMANTICS

In general, there are two basic classes of commercial vendor products: COTS products and NDIs. Let's scope each of these classes.

16.7.1 COTS Products

COTS products represent a class of products that can be procured from a vendor's catalog by part number. Procurement of COTS products is accomplished via a purchase order or

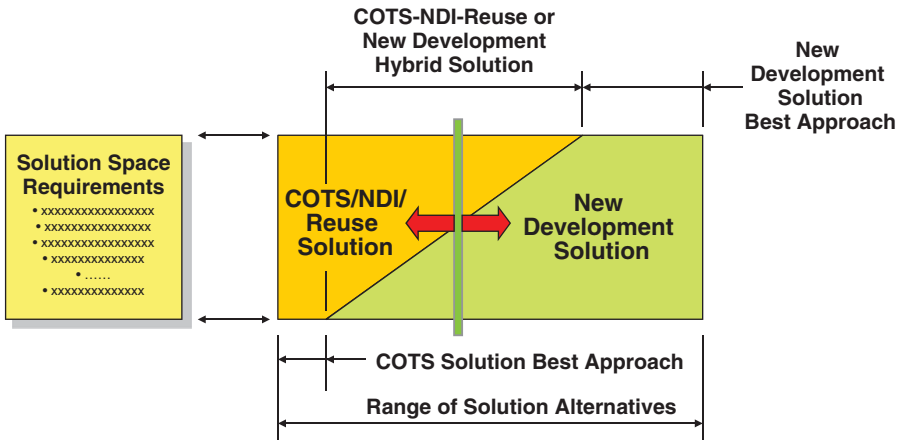


Figure 16.7 Decision Making to Find the Optimal Mix of COTS/NDI/Reuse/New Development Items to Meet Configuration Item (CI) Technical, Total Cost of Ownership (TCO), Schedule, and Risk Factors.

[1]Referral—For additional information about Make versus Buy versus Buy–Modify decisions, refer to the most recent edition of the Project Management Institute (PMI) *A Guide to the Project Management Body of Knowledge* (PMBOK®).

other procurement mechanism. Generally, vendors provide *a Certificate of Compliance (C of C)* verifying that the product meets its published Product Specification requirements.

16.7.2 NDIs

NDIs represent COTS products that have been modified or customized to meet a set of Procurement Specification requirements. Procurement of NDIs is accomplished via a purchase order that references a Procurement Specification that specifies and bounds the capabilities and performance of the modified COTS product. Prior to delivery, the System Developer as Acquirer (role) verifies the NDI, in the presence of Acquirer (role) witnesses, typically for compliance to its procurement specification.

16.8 COMPONENT SELECTION METHODOLOGY

The selection of components to fulfill EDS requirements requires a methodology that enables the item's development team to minimize technical, cost, schedule, and risk. In general, the selection process involves answering the questions posed during the introduction to this practice. *So, how do we answer these questions?*

One solution is to establish a basic methodology that facilitates component selection based on an Analysis of Alternatives (AoA) (Chapter 32). Although there are a number of ways the methodology can be created, Figure 16.8 illustrates one example.

16.8.1 COTS Selection Methodology

The methodology used to select item components can be described in a six-step, highly iterative process as illustrated in Figure 16.8.

- Step 1: Identify candidate components.
  - Step 1.1. Identify potential in-house legacy of reusable design solutions.
  - Step 1.2. Assess in-house solution(s) feasibility, capabilities, and performance.
  - Step 1.3. Identify potential COTS/NDI product solution(s).
  - Step 1.4. Assess COTS/NDI solution(s) feasibility, capabilities, and performance.
  - Step 1.5. Investigate the feasibility of modifying COTS products in-house.

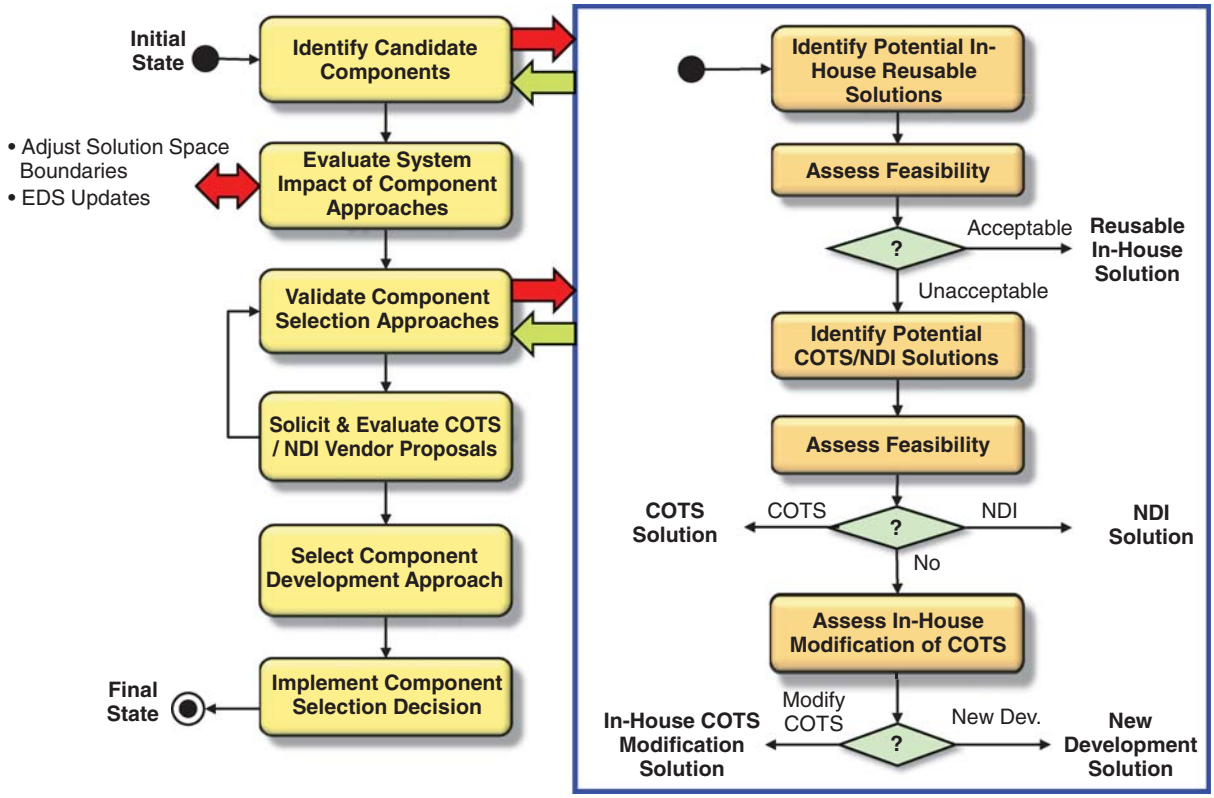


Figure 16.8 Example Component Development Methodology

- Step 2: Evaluate the overall impact of component approaches to item and system performance.
- Step 3: Validate component selection approaches (repeat Step 1).
- Step 4: Solicit and evaluate vendor COTS/NDI proposals.
- Step 5: Select component development approach.
- Step 6: Implement component selection decision.

A Word of Caution 16.3

COTS Products Meeting Specification Requirements

Avoid the notion that you can simply find a COTS solution "off-the-shelf" that meets specification requirements. You may discover you have a *paradox*: select a COTS product that is reasonably or somewhat complaint to specification requirements versus an NDI product or new development that may be cost and/or schedule prohibitive. The solution may be to reassess the driving requirements, if reasonable and practical, to accommodate the COTS product's capabilities and levels of performance.

Boehm (2006, p. 10) observes that "COTS economics generally makes sequential waterfall processes (in which the pre-specified system requirements determine the capabilities) incompatible with COTS-based solutions (in which the COTS capabilities largely determine the requirements; a desired capability is not a requirement if you can't afford the non-COTS solution that would provide it)."

16.8.2 Component Selection Summary

When the component selection and development decision-making process is complete, each item within the multi-level System Architecture hierarchy and the Product Breakdown Structure (PBS) may have a combination of various types of components such as in-house, COTS, NDI, or AFP that satisfy specification requirements allocated to the item.

Now that we have established a basic methodology, let's examine some of the driving issues that influence COTS/NDI selection.

16.9 DRIVING ISSUES THAT INFLUENCE COTS/NDI SELECTION

COTS/NDI products may or may not be applicable to your contract application. Only you, your Enterprise or project and the Acquirer/User, if appropriate, can make that determination. Let's look at examples of some of the types of questions that you should consider asking when selecting COTS/NDI products.

A Word of Caution 16.4

Acquisition Questions COTS/NDI Vendors

The lists of questions that follow are illustrative examples for each category. Every set of Acquirer requirements, applications, COTS/NDI products, and perspectives is *unique*. Consult with Subject Matter Experts (SMEs) in your Enterprise or employ the services of a qualified, credible consultant of your choice and expense to assist you in formulating the list of questions. Thoroughly investigate potential COTS products solutions and perform due diligence *before* you commit to a decision.

COTS Product Line Example Questions

1. What is the heritage, maturity, and future of the COTS product line and family?
2. What is the vendor and parent company's commitment to the COTS product line and family?
3. What is the size of the COTS product user base?
4. What Enterprises or industries are the primary users of the COTS product?
5. What are the current technology trends relative to the product line directions?
6. At what stage of maturity is the COTS product in its maturity cycle as well as the marketplace?
7. How long has this version of the COTS product been produced?
8. Is the COTS product in Alpha and Beta testing? If not, how long ago did this occur?

Customer Satisfaction Example Questions

1. Is the vendor willing to provide a list of customer references to discuss their experiences with the product?
2. What is the degree of customer satisfaction with current COTS product and prior versions?
3. Do customer testimonials, complaints, and applications of usage correlate?
4. Is the customer satisfaction based on using the product in accordance with the vendor's prescribed operating environment?

16.9.1 Corporate Commitment to and Stability of the COTS Product Example Questions

1. How long will the vendor commit on paper to (1) producing and (2) supporting the version the COTS product under consideration?
2. How financially stable is the vendor and parent company?

3. How stable is the vendor's workforce (i.e., turnover) that develops and supports the COTS product.
4. For future reference, who are the vendor's SMEs and how long they have been with the organization and worked on the COTS product?
5. What are their roles relative to the COTS product?

COTS Product Design Example Questions

1. Assuming you have some level of access to the vendor's documentation, what is the quality and depth of COTS product documentation?
2. What degree of verification and validation (V&V) has been performed on the COTS product?
3. Was the Product Verification and Validation (V&V) performed internally, by an independent laboratory, or did the vendor depend on the User community to "find the defects"?
4. Does the COTS product have documented an accessible test points, test hooks, entry/exit points in the design? Are these available to the developers and maintainers?
5. Does the COTS product use an industry standard interface that fully complies with the standard or just a subset of the standard? What are the compliance exceptions?
6. What liberties such as interpretations and assumptions has the vendor taken with implementing the standard?
7. Is the vendor willing to modify the COTS product interface to meet your system or product's interface? To what degree?
8. What systems or products does the vendor certify that the COTS product is compatible and interoperable with? For example, Microsoft Windows, MAC OS, and others.
9. What are the current known *defects* in the existing COTS product? Are there plans and priorities to correct them?
10. How many known and *latent defects* remain in the current "new and improved" product version?
11. How many *undocumented* and *untested* "features" such as latent defects cannot be or will not be corrected remain in the COTS product?
12. What detailed design information and "on call" support is available to support System Developer integration of the COTS product into their system or product? Are there fees required for this support?
13. Will future versions of the COTS product be forward and backward compatible with the current version being considered?

COTS Product Production Example Questions

1. What is the *quality* and *discipline* of the COTS vendor's quality assurance (QA) organization as well as its CM system and version control, assuming it exists?
2. Are the COTS products *serialized* and *tracked* for upgrades and recalls?

COTS Product Support Example Questions

1. What degree of 24/7 support (i.e., 24 hours per day/7 days per week) is the vendor willing to support the COTS product? From what country, time zone, and hours of availability; Internet on-line support and documentation; and "live support" via 800 numbers or "read what is on-line"?
2. What *level of responsiveness* to technical support is the vendor willing to provide (4 hours, one week, etc.)?
3. What *level of accessibility* to COTS product SMEs is the vendor willing to provide?
4. Does the COTS product have Operations and Support (O&S) and technical manuals?
5. Are Operations and Maintenance (O&M) and technical manuals delivered with the COTS product, available freely on-line, or do you have to purchase them? If purchase, how readily available are they?
6. What plans does the vendor have for sustainment of the COTS product? How long?
7. Are *alignment* and *calibration* procedures and data documented and available to System Developers?
8. Does the vendor provide *field service support*? How responsive? What constraints (days of the week, holidays, hours, etc.)?
9. If you purchase the COTS/NDI product, are you required to contact the vendor to perform field service on site to "remove and replace, align, and calibrate," or can the System Developer perform this?
10. Who pays for field service calls and expenses?

COTS Product Warranty Example Questions

1. Is each COTS product covered by an *expressed* or *implied warranty*? Will the vendor provide a copy?
2. What actions or physical *modifications* by the Acquirer may *invalidate* or *void* the manufacturer's warranty?
3. What are acceptable System Developer modifications to the COTS product that do not void the warranty?
4. Is the vendor *willing to modify* the COTS product or do they recommend third parties?

5. Do third-party modifications to the product impact the warranty? What authorizations are required to modify the product and maintain a warranty

COTS Product Procurement Example Questions

1. Does the COTS product require *usage licenses* for software, hardware, and export control (Chapter 18)?
2. Are licenses based on a per platform basis or is a site license available subject to a maximum number of "floating" users?
3. If a site license is available, how is the number of users *restricted* (simultaneous, total population, number of available "keys," etc.)?
4. Are there "other bundled" products included in the COTS product or must be procured for a small additional cost? Do they require licenses?
5. Is there a *minimum buy* quantity requirement for the COTS product?
6. If there is a *minimum buy*, is this on a per purchase basis, cumulative over the year, or cumulative over several years?
7. What quantities are the price breakpoint thresholds?
8. What product quality assurance processes are in place to ensure the quality of the COTS product?
9. What *certifications* is the vendor willing to provide regarding the integrity of the product and its materials?
10. Are COTS product specifications, processes, and test procedures available for review and inspection?

Principle 16.8

Caveat Emptor Principle

When procuring COTS/NDI, thoroughly *investigate* and *understand* the TCO - development and life cycle - and support; otherwise, *caveat emptor*—buyer beware!

- Since vendors will only answer the question you ask, always ask what you *need-to-know* about the product that you may not have asked.
- Finally, *caveat emptor*—let the buyer beware!

In summary, you, your project, and your Enterprise are fully and solely accountable for the decisions you make. Thoroughly conduct a 360° search of all factors *surrounding* a COTS/NDI product or any type of system, product, or service utilization including Stakeholders – Users and End Users … *before* … you make a decision.

16.10 CHAPTER SUMMARY

In summary, our discussions in Chapter 16 focused on System Configuration Identification and Component Selection. Although addressed as a chapter, they are rules that bind the development of System Architecting and Architectures addressed in Chapter 26.

How does this relate to SE?

System Configuration Identification and Component Selection can be distilled into three words by Dr. William McCumber. Your job as an SE is to "maintain intellectual control" (McCumber and Sloan, 2002, p. 4) (Principle 1.3 - McCumber and Sloan, 2002, p. 4) of the evolving and maturing System Design Solution. via baseline management oversight. If you fail to do this, you have lost control of the project.

Your other mission is to ensure that the System Design Solution's physical components are selected, procured, or developed to not only to comply with specification requirements but also to *minimize* the Total Cost of Ownership (TCO) – development and life cycle costs – and achieve acceptable risk within budgetary, schedule, and technology risks. Selection of those components may come from a variety of sources.

COTS/NDI solutions *may* be a *right choice* for you and your application; in other cases, they *may not* be. As an SE leading the selection process, you must make informed decisions based on:

1. The facts corroborated by field evidence.
2. Trade-offs to meet requirements, *minimize* development and life cycle costs, and reduce risk to an acceptable level.
3. Other User's application experiences.

COTS products can be very powerful tool to reducing development and life cycle costs; they can also become a major problem as well. Perhaps the best way to *think* of COTS is to use a mirage analogy: make sure that what you *find* when you implement the product matches the *virtual image* you perceived. Whatever decision you make, you will have to live with your action(s) and consequences. Research component selections and vendors carefully and thoroughly incorporate flexibility for contingency planning and make wise choices. The bottom line is *caveat emptor*—let the buyer beware!

16.11 CHAPTER EXERCISES

16.11.1 Level 1: Chapter Knowledge Exercises

Answer each of the What You Should Learn from This Chapter questions identified in the chapter's Introduction.

1. What is Configuration Management (CM)?

2. What are the four functions of CM?

3. What is Configuration Identification? Give examples.

4. What is Configuration Status Accounting (CSA)? Give examples.

5. What is Configuration Change Management?
6. If you wanted to know the current status of a drawing, which of the four CM functions would provide the information?
7. What is a Developmental Configuration, when and how does it evolve, and who is accountable for its establishment and maintenance?
8. Delineate the differences between components, entities, items, and CIs and their Entity Relationships (ERs)?
9. Who is responsible for identifying CIs?
10. What decision-making criteria are used to select CIs?
11. How do CIs relate to the specification tree?
12. What are COTS products and NDIs and how do they relate to items and CIs?
13. What is a configuration baseline?
14. What is the relationship between baselines and the Developmental Configuration?
15. What is meant by configuration effectivity?
16. Describe the evolution of the Developmental Configuration and its baseline.
17. What is the "As-Specified" Configuration, and when is it established?
18. What is the "As-Designed" Configuration, and when is it established?
19. What is the "As-Built" Configuration, and when is it established?
20. What is the "As-Verified" Configuration, and when is it established?
21. What is the "As-Validated" Configuration, and when is it established?
22. What is the "As-Maintained" Configuration, and when is it established?
23. What is the "As-Produced" Configuration, and when is it established?
24. What is the difference between MISSION SYSTEM versus ENABLING SYSTEM CIs? What are their relationships. Using an office building, automobile, and an aircraft, illustrate the relationships.
25. What is the difference between baseline management and CM?
26. What are the six primary approaches for developing components?
27. What is a COTS product?
28. What is a NDI?
29. What is AFP?
30. Who is the source of AFP?
31. How are COTS products procured?
32. How are NDIs procured?
33. What are the steps of the Component Selection Methodology?
34. What questions should you ask vendors that may have products or services you are considering selecting?

16.11.2 Level 2: Knowledge Application Exercises

Refer to www.wiley.com/go/systemengineeringanalysis2e

16.12 REFERENCES

Boehm, Barry, (2006), "Some Future Trends and Implications for Systems and Software Engineering," International Council on Systems Engineering (INCOSE), *Journal of Systems Engineering*, Vol. 9, No. 1, Malden, MA: John Wiley & Sons, Inc.

DAU (2012), *Glossary: Defense Acquisition Acronyms and Terms*, 15th ed. Ft. Belvoir, VA: Defense Acquisition University (DAU) Press. Retrieved on 6/1/15 from http://www.dau.mil/publications/publicationsDocs/Glossary_15th_ed.pdf

FAA SEM (2006), *System Engineering Manual*, Version 3.1, Vol. 3, National Airspace System (NAS), Washington, DC: Federal Aviation Administration (FAA).

McCumber, William H. and Sloan Crystal (2002), *Educating Systems Engineers: Encouraging Divergent Thinking*, Rockwood, TN: EagleRidge Technologies, Inc.

MIL-HDBK-61A(SE) (2001), *Military Handbook: Configuration Management Guidance*, Washington, DC: US Department of Defense (DoD).

MIL-STD-498 (1994), *Software Development and Documentation*, Washington, DC: US Department of Defense (DoD).

SEVOCAB (2014), *Software and Systems Engineering Vocabulary*, New York, NY: IEEE Computer Society. Accessed on 5/19/14 from www.computer.org/sevocab

17

SYSTEM DOCUMENTATION STRATEGY

When systems, products, or services are developed, documentation of key decision-making events and their artifacts becomes very important for "maintaining intellectual control" (Principle 1.3 – McCumber and Sloan, 2002, p. 4) of the technical program. Documentation, especially *formal* documentation, can become very costly and consume valuable project resources to produce. If you are the System Acquirer, inevitably, every document has a value to the User with Return on Investment (ROI) that must be traded off for System capabilities. *Do we purchase documentation or system capabilities and where is the balance*?

Unfortunately, Enterprises often view that the role of the System Development and Systems Engineering (SE) Processes is to produce documentation for delivery. The reality is that the focus of the System Development and SE Processes is on producing systems, products, or services that can be *verified* as compliant with the specified requirements and *validated* as satisfying the User's operational needs. In sharp contrast, documentation is simply a mechanism to capture and describe key technical decision-making *artifacts*.

Hypothetically, you can *informally* document essential decision artifacts on every square inch of the "back of a napkin." However, it would lack organization, substantive content, coherent meaning, be subject to interpretation, and so forth. Configuration Management (CM) version control (Chapter 16) and Change Request (CR) reviews and approvals would be a disaster!

When a technical project Request for Proposal (RFP) is released and proposed, there are several key questions SEs need to be able to answer:

- Question #1: What is the *minimum, essential* set of System Development documentation (Principal 15.16) that should be produced during the development of SYSTEMS, PRODUCTS, or SERVICES to document key technical decisions and satisfy development, deployment, operations, maintenance, sustainment retirement, and disposal operational needs?
- Question #2: How can we tailor documentation requirements to obtain the *essential* documentation we need?
- Question #3: Is there a way to reduce the level of formality for specific documents to capture *essential* information?
- Question #4 emerges: Of all the decision documentation produced during the development of a SYSTEM, PRODUCT, or SERVICE, how can we get an inventory listing for future reference to purchase? This last question presents potential legal issues.

Chapter 17 explores the types of documentation commonly produced for system design and development projects to support development of the overall System Design Documentation Strategy. Our discussions introduce key contracting terms such as Contract Data Requirements Lists (CDRLs), Data Accession List (DAL), and Design Criteria List (DCL) and describe each one and its relationship to the overall system development contract. We identify four types of documents such as plans, specifications, design, and test documentation and provide a graphical schedule of their release points.

System Engineering Analysis, Design, and Development: Concepts, Principles, and Practices, Second Edition. Charles S. Wasson.

Companion Website: www.wiley.com/go/systemengineeringanalysis2e

Our discussions also provide commonly applied rules for developing SE and development documentation. We emphasize the importance and criticality of assessing documentation for export control of sensitive data, proprietary information, and technology. Our final discussion topic identifies several types of documentation issues SEs need to be prepared to address.

17.1 DEFINITIONS OF KEY TERMS

- **Authorized Access**—A formal approval issued by an Acquirer or Enterprise project signifying that an internal or external individual or Enterprise with a need to know is authorized for limited access rights to specific types of data for a restricted period of time subject to handling and control procedures established by the contract or project.
- **Contract Data Requirements List (CDRL)** —An attachment to a contract that identifies a list of documents to be delivered under the Terms and Conditions (Ts&Cs) of the contract. Each CDRL item should reference delivery instructions including (1) *when* the documents are to be delivered; (2) *what* outline, format, and media to be used; (3) to *whom* and in *what* quantities; (4) *level of maturity* such as outline, draft, and final; (5) *requirements* for corrective action; and (6) approvals.
- **Data Accession List (DAL)** —"An index of the generated data that is made available upon request" (DI-MGMT-81453, 2007).
- **Design Criteria List (DCL)**—A listing of design data that characterize the capabilities, performance, and Operating Environment boundary conditions of an external system or entity.
- **Engineering Release Record**—A record prepared by a project Configuration Manager publicizing the official release of an engineering document such as a plan, specification, design, and drawing that has been formally approved, baselined, and authorized for *internal* technical decision-making use.
- **Released Data**—The collection of project documentation including revisions that have been formally approved, baselined, and authorized for *internal* technical decision-making use since contract award, beginning of a phase of development, or new version of a System or Product.
- **Subcontract Data Requirements List (SDRL)**—A listing of data deliverables required by a subcontract. Refer to CDRL definition above.
- **Technical Data**—"Technical data is recorded information (regardless of the form or method of recording) of a scientific or technical nature (including computer software documentation)" (MIL-HDBK-61A, p. 3-10).
- **Technical Data Package**—"A technical description that is adequate to support acquisition of an item, including engineering, production, and logistics support. The technical description defines the design configuration and procedures required to ensure adequacy of item performance. It consists of all applicable technical data, such as engineering drawings, associated lists, product and process specifications and standards, performance requirements, quality assurance provisions, and packaging details" (MIL-HDBK-61A, p. 3-10).
- **Working Data**—Data such as informal working papers (e.g., draft, preliminary, etc.), decisions, designs, analyses, trade studies, etc. that may or may not be deliverable as formal documents and are considered to be works in progress with levels of maturity. Some Enterprises loosely refer to documentation from external Enterprises required for technical decision-making as *working data*.

17.2 QUALITY SYSTEM AND ENGINEERING DATA RECORDS

System development, as an incremental decision-making process, requires that technical specifications, plans, design documentation, test procedures, and test results be documented for deliverables and decision-making, and be archived for historical purposes. The *integrity* of the documentation data requires Verification and Validation (V&V) reviews to ensure accuracy, preciseness, consistency, and completeness.

For ISO 9001-based Quality Systems, each step of the documentation process produces *decision artifacts* as data records that provide *objective evidence* as proof that you accomplished what you planned to do. When you plan your technical program, you must establish and communicate the strategy for creating and capturing data records for key decisions, technical meeting and review conference minutes, development of design documentation, and results from tests, analyses, and trade studies. The strategy will be documented across a variety of plans beginning with the Program/Project Management Plan (PMP). Discipline specific implementations are provided in Technical Management Plans (TMPs) such as an Engineering Management Plan (EMP) or Systems Engineering Management Plan (SEMP), Configuration and Data Management Plan (DMP), Quality Assurance Plan (QAP), Risk Management Plan (RMP), Hardware Development Plans (HDPs), Software Development Plans (SDPs), Test Plans, and others.

Based on this introduction, let's begin with a high-level perspective of documentation typically produced by a project.

Author's Note 17.1

Documentation Titles

Documents introduced in this chapter identify types of documents commonly used in SE. Your Enterprise or industry may employ different names. Regardless of what titles your discipline or industry employs, the *concepts*, *principles*, or *practices* presented apply universally to the development of systems, products, or services developed in business domains such as medical, transportation, aerospace and defense, energy, telecommunications, and others.

17.3 SYSTEM DESIGN AND DEVELOPMENT DATA

System data consist of six basic types:

- Contract deliverable data
- Subcontract, vendor, supplier-required data
- Operation and support data
- Working data
- Organizational command media
- Engineering personnel data records

Let's explore each of these.

17.3.1 Contract Deliverable Data

The Acquirer of a system levies documentation requirements on a System Developer via Ts&Cs of the contract. Enterprises issue contracts containing a *CDRL* that lists specific documents such as plans, analyses, conference minutes, etc. to be delivered throughout the contract. CDRL items are assigned a unique item number such as A001 and provide delivery instructions concerning documentation format and media, and conditions for submittal, submittal dates, and distribution lists. Regarding documentation format, some Enterprises use Data Item Descriptions (DIDs) that provide detailed instructions for a specific type of document that includes the CDRL item's outline and contents.

17.3.1.1 CDRL Items CDRL items typically employ a form that identifies the unique item number, title, quantity, documentation format, instructions for preparation, conditions for submittal, submittal dates, distribution lists, etc.

17.3.1.2 DIDs Most contracts require CDRLs to be delivered in a specific format. The DoD, for example, employs DIDs that provide preparation instructions regarding the deliverable document's outline and contents. In general, a DID is simply a description of a specific type of document to be delivered. Contents include an outline annotated with specific content instructions.

17.3.2 Subcontractor, Vendor, and Supplier Data

Acquirers (role) levy data requirements on contractors via contracts and subcontracts. These data apply to:

1. Configuration Items (CIs) and items such as Commercial Off-the-Shelf (COTS) items and Non-Developmental Items (NDIs) procured under subcontract.
2. Certificates of Compliance (CofC) that purchased items are compliant with a vendor's product specification.
3. Design and management reporting data to support SE design activities.

Subcontracts include an SDRL that identifies the set of data deliverables. In some cases, the System Developer may directly flow down CDRL requirements via SDRLs to subcontractors or extract specific CDRL requirements and incorporate into SDRLs to serve as inputs to support System Developer CDRL documents delivered to the System Acquirer.

Vendor and supplier deliverables and data are often procured as part of a purchase order, for example. The order may require the vendor to supply a CofC certifying that the delivered item *meets* or *exceeds* the vendor or supplier's product specification requirements.

17.3.3 Operations, Maintenance, and Sustainment (OM&S) Data

Operations and support data consist of the data required to *operate* and *maintain* the system, product, or service. Examples include Standard Operating Practices and Procedures (SOPPs), installation guides, and operator's manuals. OM&S data for a system may be delivered as part of the System Developer's contract or support contract or developed internally by the User.

17.3.4 Working Data

Working data represent data such as analyses, trade studies, modeling and simulation results, test data, technical decisions and rationale generated during the course of designing, developing, integrating, and testing the system. Unless these data are *explicitly* identified as CDRL items under the Terms & Conditions (Ts&Cs) of the contract, they are for internal usage only by the System Developer's personnel.

Documentation as viewed by the System Acquirer and User is often expensive. Confronted with a decision to purchase *formal* documentation versus system capabilities, they typically go with additional capabilities based on operational needs and affordability. As a result, the Acquirer and User may request the opportunity to view these data on the System Developer's premises with the understanding that they

are not deliverables. An inventory of data produced on the project is documented via a Data Accession List (DAL).

Occasionally, the Acquirer may request the opportunity to view these data on the System Developer's premises with the understanding that they are not deliverables but may be available for purchase.

Once the System is fielded, if it is determined that specific types of data are needed as *formal* documentation, the DAL enables the Acquirer and User to procure it within a reasonable timeframe subject to retention of data records requirements specified by the contract's Ts&Cs.

17.3.5 Enterprise Command Media Required Data

Another type of system development data are work products required by Enterprise command media such as policies, processes, and procedures. Data required by Enterprise command media may include plans, briefings, drawings, wiring lists, analyses, reports, models, and simulations. Unless explicitly specified by the contract CDRL or SDRL, these data are considered to be *non-deliverables*.

You may ask: *If a document is not required by contract, why consume resources producing it?* In general, the answer is Enterprises that possess levels of SE capability know and understand that specific data such as plans, specifications, and test procedures are crucial for success, regardless of CDRL or SDRL data requirements. CDRL and SDRL data requirements may be inadvertently overlooked or unaffordable by the System Acquirer during the formal solicitation process or, as is often the case, the Acquirer decides not to procure the data. If you determine that specific data items are missing during the formal RFP solicitation process, confer with the proposal leader regarding how to address the missing data items.

Improper Use of Acquirer Contract Funding

Warning 17.1 Be aware that if you use Acquirer funding to develop a document required by Enterprise command media but not required by the Acquirer's contract, the Acquirer effectively "owns" the document. Even worse is naively using Acquirer project funding to produce documents required by Enterprise command media and then listing the document in the DAL as being "available for purchase." *Always* confer with your Enterprise's project and legal organizations.

17.3.6 Engineering Personnel Data Records

The final type of SE data includes engineering personnel data records as part of their normal tasking. Personnel data records should be maintained in an electronic Engineering Notebook or an assigned folder on a project network drive. Personnel data records include plans, schedules, analyses, sketches, reports, conferences, meeting minutes, action items, tests conducted, and test results. These data types are summarized in technical reports and progress and status reports.

17.4 DATA ACCESSION LIST (DAL) AND DATA CRITERIA LIST (DCL)

Contracts and subcontracts often require two types of documentation as part of the CDRL or SDRL deliveries:

1. DAL
2. DCL

Let's describe each of these documents.

17.4.1 DAL

The development of most systems and products involves two types of documentation: *deliverable* data and *non-deliverable* data. An Acquirer (role) specifies and negotiates the documentation to be delivered as part of the contract delivery work products. During the contract, the System Developer or subcontractor may produce additional documentation that is not a contract/subcontract deliverable but may be needed later by the Acquirer or User.

Where this is the case, Acquirer (role) contracts/subcontracts often require the System Developer (role) to prepare and maintain a DAL that lists all documentation produced under the contract or subcontract for *deliverables* and *non-deliverables*. DI-MGMT-81453A (2007) states, "The purpose of the DAL is to provide a medium for identifying contractor internal data which has been generated by the contractor in compliance with the work effort described in the Statement of Work (SOW) . . . ." This enables the Acquirer to determine if additional data are available for procurement to support system maintenance. If so, the Acquirer may negotiate with the contractor or subcontractor and modify the original contract to procure that data.

Each contract or subcontract should include a requirement for a System Developer to provide a DAL as a CDRL/SDRL item to identify all CDRL/SDRL and non-deliverable documentation produced under the contract/subcontract for review and an opportunity to procure the data at a later date.

A Word of Caution 17.1

Always confer with your project's Configuration Manager (CM), Project Engineer, Project Manager, and legal organization for specific guidance concerning deliverable documentation, *who* pays for it, *who* owns the data, *what* data rights are associated with

delivery, etc. Different funding sources create significant legal issues.

17.4.2 DCL

Systems, products, and services are often required to be integrated into HIGHER ORDER SYSTEMS. For physical systems, interoperable interfaces or models are crucial. For simulations and emulations, each model must provide the precise *form, fit*, and *function* of the simulated or emulated device.

In each of these cases, the physical system, model, simulation, or emulation must comply with specific design criteria. Source data authentication, verification, and validation are critical to ensure the integrity of the SE decision-making efforts.

How do you identify the design data that will be required to support the SE design effort? The process of procuring this data begins with a Design Criteria List (DCL). The DCL is developed and evolves to identify specific design documents required to support the System Development effort. Generally, SEs and Product Development Teams (PDTs) are responsible for submitting a detailed list of external documents such as title and document ID to the Data Manager for acquisition. On receipt of the requested documents, the Data Manager:

1. Forwards the documents to CM for processing and archival storage.
2. Notifies the design data requestors regarding receipt of the documents.

Each contract or subcontract should require publication of a DCL by the System Developer identifying specific documentation sources.

17.5 SE AND DEVELOPMENT DOCUMENTATION SEQUENCING

One of the challenges for SEs is determining when various documents should be prepared, reviewed, approved, baselined, and released. Although every contract and project requirements vary by Acquirer, there are some general schedules that can be used to prepare SE and development documentation. In general, we can categorize most SE documentation into four classes:

- Planning documentation
- Specification documentation
- System design documentation
- Test documentation

17.5.1 Planning Documentation

Figure 17.1 illustrates a basic schedule as general guidance for preparing and releasing various types of *technical plans.* These documents include:

1. Key Technical Management Plans (TMPs) such as Hardware Development Plans (HDPs), Software Development Plans (SDPs), Configuration and Data Management Plans (DMPs), and Risk Management Plans (RMPs).
2. Test plans such as System Integration, Test, and Verification Plans as well as Hardware and Software Test Plans (HTPs/STPs).
3. Supporting technical plans such as System Safety Plans, Procurement Plans, Manufacturing Plans, and System Sustainment Plans.

17.5.2 Specification Documentation

Figure 17.2 illustrates a basic schedule as general guidance for preparing and releasing various types of specifications. These documents include:

1. System Performance Specification (SPS)
2. Product/Subsystem Entity Development Specifications (EDS)
3. HWCI/CSCI Requirements Specifications (HRS/SRS)

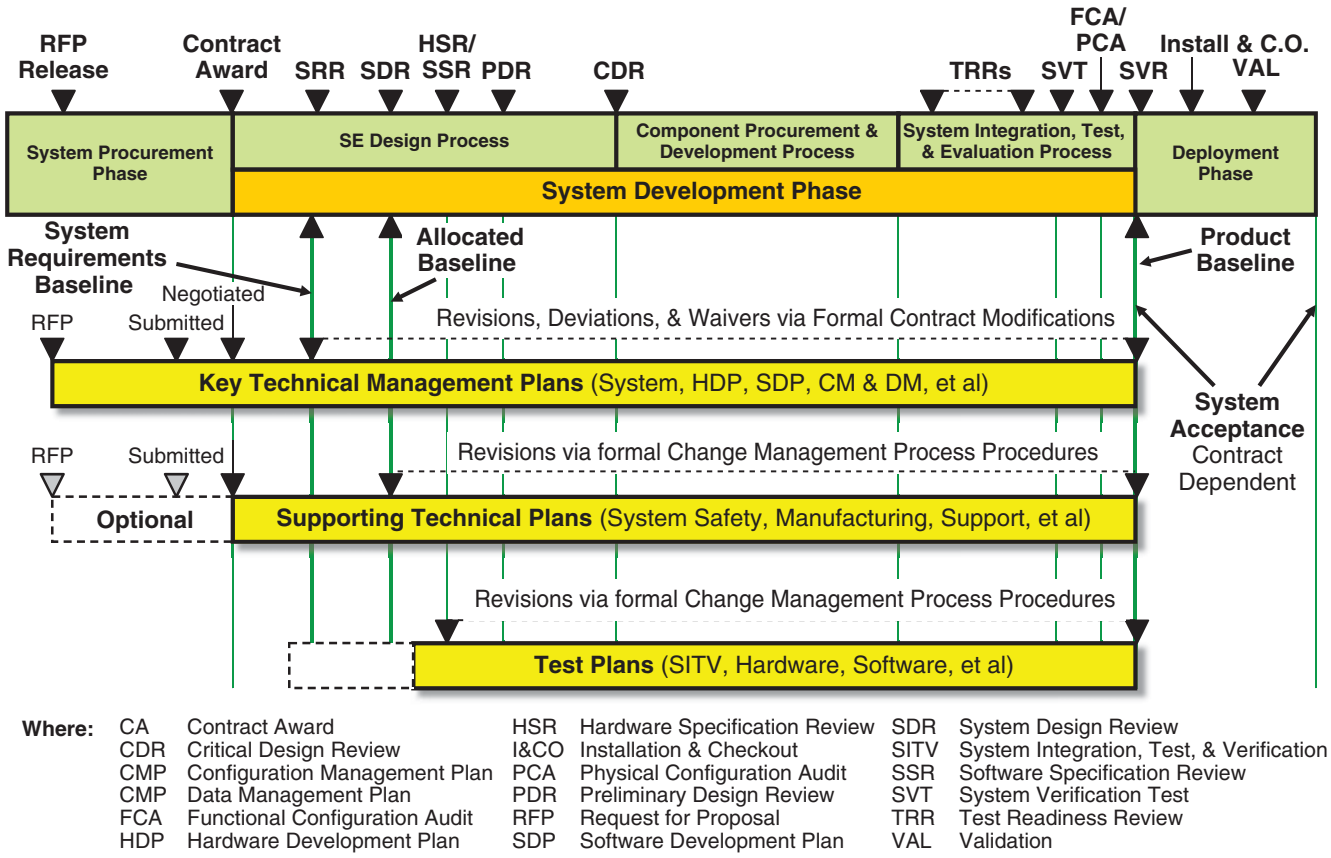
17.5.3 System Design Documentation

Figure 17.3 illustrates a basic schedule as general guidance for preparing and releasing various types of technical design documentation. These documents include:

1. Concept of Operations (ConOps)
2. System/Segment Design Description (SSDD)
3. Interface Control Documents (ICDs) and drawings for hardware items
4. Interface Design Descriptions (IDDs) for software
5. Software Design Description (SDD)
6. Database Design Description (DBDD)

17.5.4 Test Documentation

Figure 17.4 illustrates a basic schedule as general guidance for preparing and releasing *test* documentation. These documents include various levels of test procedures and test quality records.



Where:

CA	Contract Award	HSR	Hardware Specification Review	SDR	System Design Review
CDR	Critical Design Review	I&CO	Installation & Checkout	SITV	System Integration, Test, & Verification
CMP	Configuration Management Plan	PCA	Physical Configuration Audit	SSR	Software Specification Review
CMP	Data Management Plan	PDR	Preliminary Design Review	SVT	System Verification Test
FCA	Functional Configuration Audit	RFP	Request for Proposal	TRR	Test Readiness Review
HDP	Hardware Development Plan	SDP	Software Development Plan	VAL	Validation

Figure 17.1 Example—Planning Documentation Development and Release Strategy

17.6 DOCUMENTATION LEVELS OF FORMALITY

Most people avoid documentation tasks. The general viewpoint is that documentation is *non-value-added bureaucracy*. Engineers, in particular, rationalize that if they had wanted to specialize in documentation, they would have pursued it as a course of study. This attitude is counter to a professional engineering development environment that is so dependent on documented data *maturity* and *integrity* to make *informed* decisions.

SE development documentation involves two key decisions:

1. What must be documented?
2. To what degree of formality do you document the details?

The preceding discussions addressed what you should document. Let's explore the second point further.

If you ask Engineers and Analysts to prepare a document, they lament about having an impossible task, regardless of timeframe to complete. The *knowledge* and *maturity* of a professional is reflected in the individual's ability to sift rapidly through a large amount of data, identify the key points, and articulate the results in summary form.

Principle 17.1 **Task Expectations Principle**

A 40-hour task with an 8 hour limitation produces an 8 hour summary of key points. The same task with a 40-hour limitation produces a summary with key points and elaborated levels of details. Recognize the difference!

When you document plans, specifications, and reports, there are key points relevant to the subject that must be presented in the document outline. These points, in turn, require various levels of details. The key points are:

1. Engineers need to learn to identify what information - major points - is required to be communicated.
2. Apply common sense to scale the level of detail to fit the outcome based on the time and resources available.

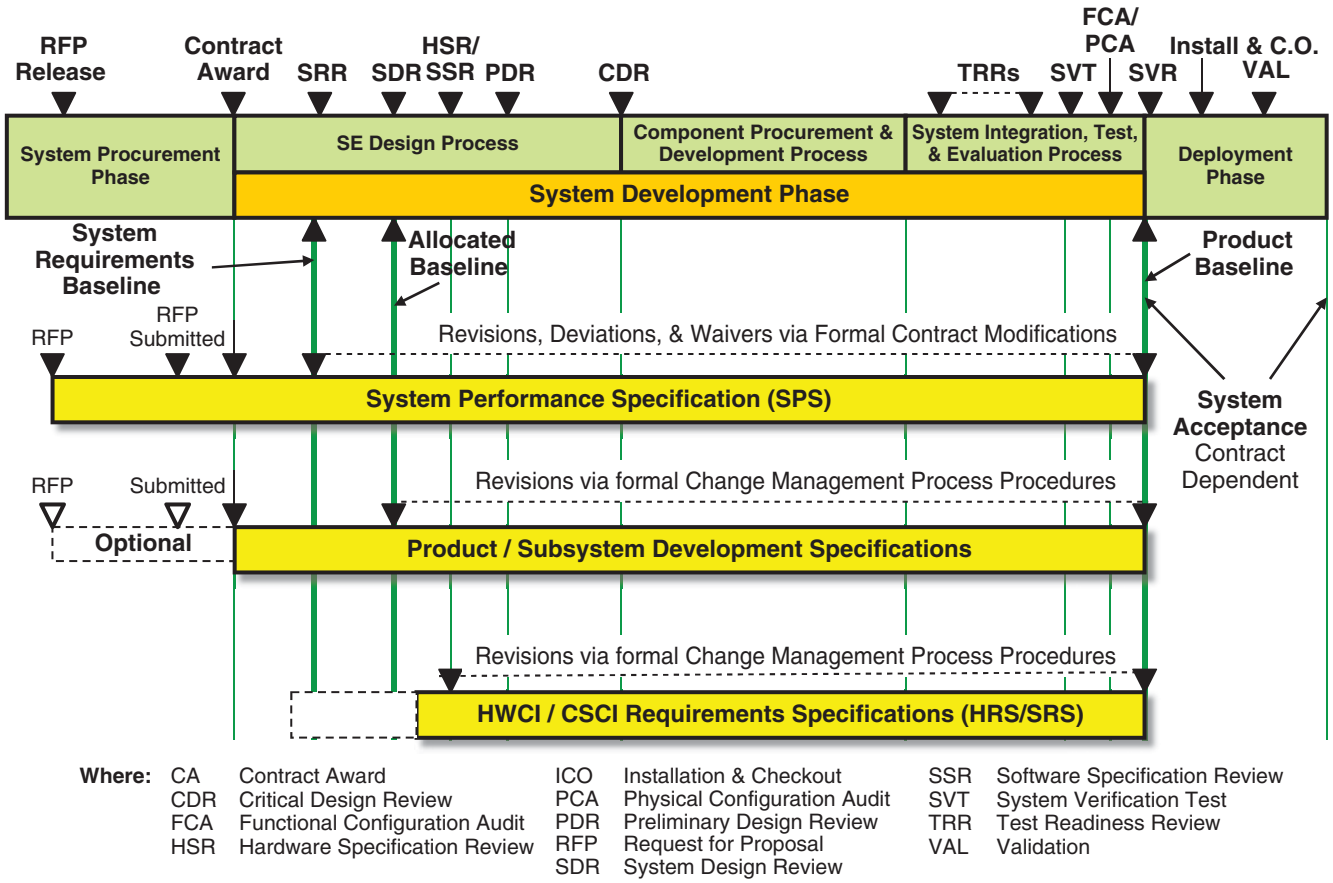


Figure 17.2 Example—Specification Development and Release Strategy

Author's Note 17.2

Responding to Management Quick Reaction Tasks

If you have addressed the critical points, anything else should be just supporting detail. The ability to scale these levels of detail and still address the key points in a short period of time depends on an SE's competence—personal knowledge, experience, and seasoned decision-making maturity. Work to achieve this level of performance and become recognized as a competent professional.

17.7 EXPORT CONTROL OF SENSITIVE DATA AND TECHNOLOGY

Today, the Internet provides a mechanism for immediate data communications and access between Enterprises in-country as well as internationally. As a result, the Internet offers tremendous opportunities for contract programs to exploit the technology by establishing Web sites that enable authorized Enterprises and individuals who are geographically dispersed to post and access technical program information. This environment, when uncontrolled, however, adds new dimensions and threats to information access.

People often confuse Export Control information with security classification systems. Although both are certainly interrelated, Export Control information includes *sensitive* but *unclassified* data. Classified data handling and procedures are yet another issue.

Warning 17.2

Export Control and Security Information Protection

Always consult with your Enterprise's Export Control Officer and Security, Legal, and Contracts organizations for specific requirements before initiating an action that may potentially *violate* Export Control or security laws, regulations, and procedures. The laws carry *severe* penalties for non-compliance.

Many people *erroneously* believe that a technology or information export to a foreign national, organization, or country occurs when that information is physically transferred outside the country. ABSOLUTELY NOT TRUE!!! The US

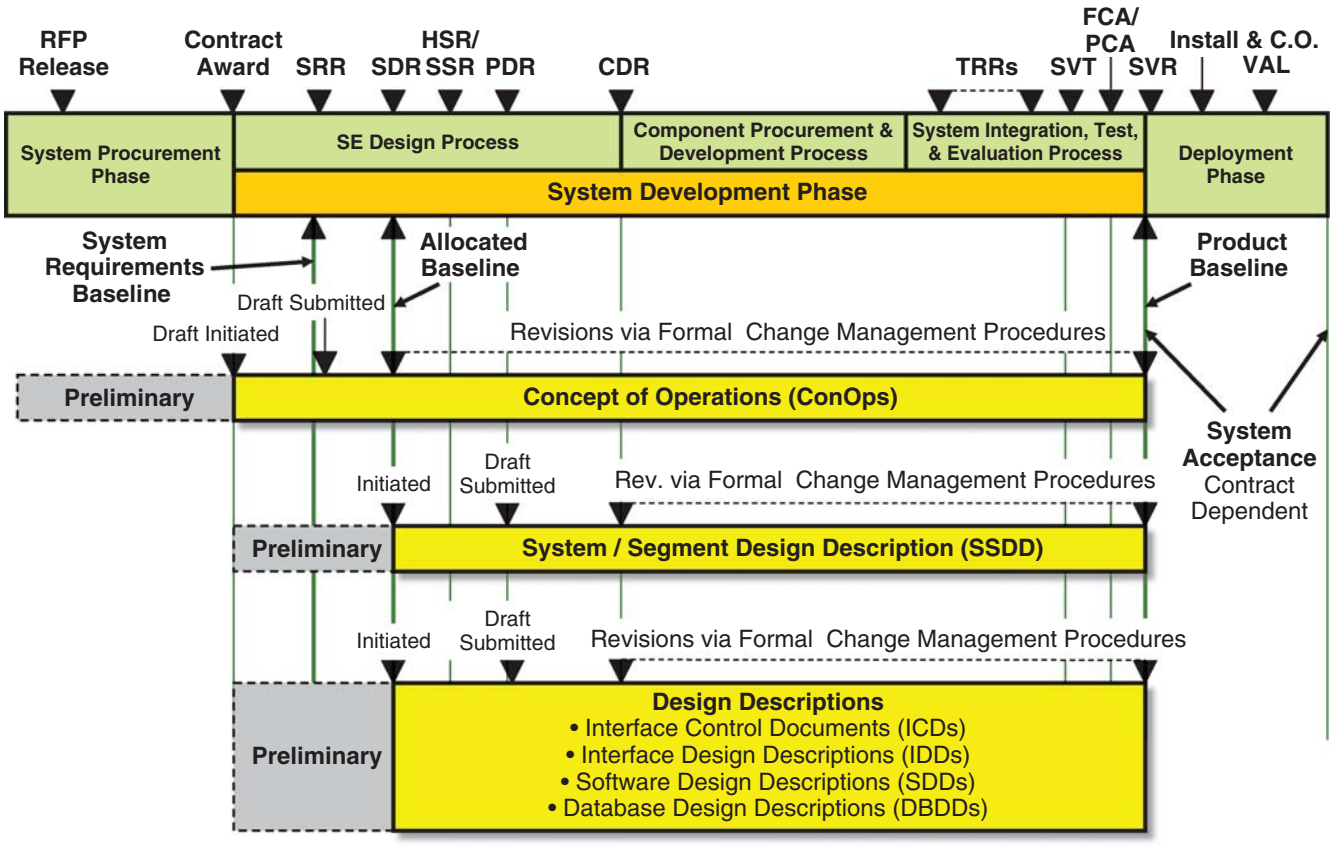
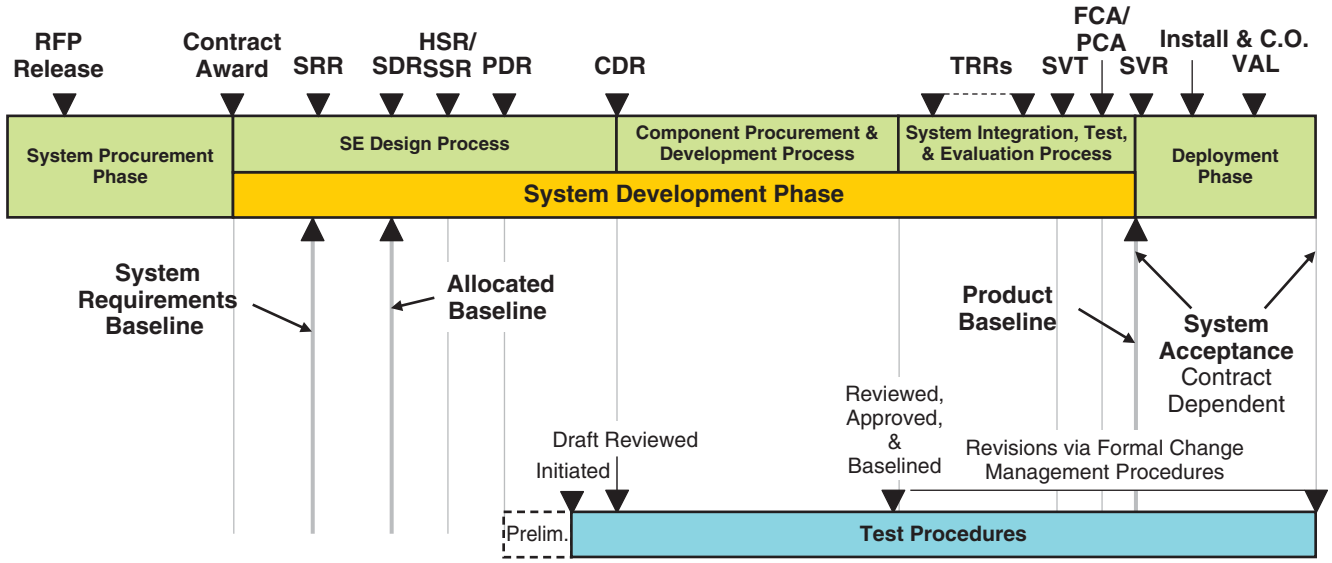


Figure 17.3 Example—System Design Documentation Development and Release Strategy



Where:

CA	Contract Award	I&CO	Installation & Checkout	SSR	Software Specification Review
CDR	Critical Design Review	PCA	Physical Configuration Audit	SVT	System Verification Test
FCA	Functional Configuration Audit	PDR	Preliminary Design Review	TRR	Test Readiness Review
HSR	Hardware Specification Review	RFP	Request for Proposal		
		SDR	System Design Review		

Figure 17.4 Example—Test Documentation Development and Release Strategy

government and other countries have explicit laws and regulations that govern the export of technology and data to foreign nationals, organizations, and countries. The laws carry *severe* penalties for non-compliance.

The fact is technology transfer also occurs in-country with foreign nationals, whether direct or via the Internet. In the United States, technology transfer is governed by Export Control laws and regulations such as the *most current* International Traffic and Arms Regulations (ITAR) (US Department of State).

In general, you have *violated* the export control governance if you:

1. Post Export Control technology or information to an *unsecure* Web site.
2. E-mail or transfer Export Control technology or information via the Internet.
3. Simply transfer Export Control technology or information to foreign nationals, organizations, or governments, irrespective of location, without being *licensed* and have taken *reasonable measures* to restrict access to only authorized users.

17.8 SYSTEM DOCUMENTATION ISSUES

Sensitive Information Protection Principle

Principle 17.2 Establish a sensitive information protection program within your Enterprise, identify a central Point of Contact (POC), develop and disseminate guidelines and protocols; and train personnel to implement them properly.

System documentation has a number of issues that SEs must address. Here are several issues that commonly challenge programs:

- Issue 1: Data validation and authentication.
- Issue 2: Posting acquirer and vendor documentation on the Internet.
- Issue 3: Proprietary Information (PI), Intellectual Property (IP), and Non-Disclosure Agreements (NDAs).
- Issue 4: Vendor-owned data.
- Issue 5: Electronic signatures.

Issue 1: Data Validation and Authentication

Since SE technical decision-making is predicated on data *integrity*, the challenge is determining if external and internal data are *valid* and *authentic* and what restrictions govern their use. This includes DCL items for models and simulations, interfaces, and test data. Additionally, production contracts may require using SE documentation created by the original System Developer or vendor. *How do you determine the validity and authenticity of the data?*

Data *validity* and *authenticity* require rigorous investigation. Data integrity is particularly troublesome when the original System Developer delivered the Product Baseline and transferred CM control via the Acquirer to the User that did not maintain the documentation. The challenge is: *Does the "As-Maintained" System or Product reflect the "As-Designed, Built, Verified, and Validated" documentation (Chapter 16)?* Thoroughly investigate the User's change history for the System and its documentation for the period following delivery by the original System Developer.

Issue 2: Posting Acquirer and Vendor Documentation on the Internet

People, in general, often *naively* believe that they can arbitrarily copy and post Acquirer and vendor documentation on a project Web site.

If you need to provide project personnel access to this data, simply provide links from the project's Web site to the Acquirer or vendor Web site, assuming they approve. This avoids ownership, proprietary data, concurrency, copyright, and other issues.

Issue 3: Proprietary Information (PI) and Non-Disclosure Agreements (NDAs)

Finally, before any data exchange can occur between any Enterprise and the Acquirer, User, subcontractors, and vendors, execute Proprietary Information Agreements (PIAs) and/or NDAs between the parties.

Issue 4: Vendor-Owned Data

When Acquirers procure a system, they are often willing to trade off documentation funds for system capabilities, especially when they have limited budgets. Later, if they decide to procure system upgrades that are dependent on these data, the System Acquirer shifts the burden and risk of obtaining the data to the System Developer. If the original developer and new System Developer are competitors, this creates a very challenging problem for projects and SEs to solve.

The challenge is one of *cost* versus *schedule*. In a highly competitive marketplace, projects inevitably *underestimate* the amount of resources required to acquire documentation owned by other Enterprises. Thoroughly investigate these issues during the proposal phase and establish data exchange agreements and Ts&Cs prior to Contract Award. As an Acquirer, if you wait until after Contract Award to help the System Developer you have selected to acquire this information, *guess who is in the power position to control the negotiation*? The supplier is and you can rest assured that the procrastination will cost you and the System

Developer significantly, especially if the supplier was a competitor that lost the acquisition!

Issue 5: Electronic Signatures

Paper systems have been replaced by electronically networked Enterprises sometimes referred to as Integrated Data Environment (IDE) based on Enterprise Resource Planning (ERP) tools. In general, IDEs enable projects to create virtual development environments in which geographically dispersed Enterprises and their personnel with an authorized "need to know" can have desktop access to specific information. This includes being able to open data files and view the information.

When implementing an IDE, one of the key issues is establishing a secure method for SE documentation reviewers to electronically access, review, comment, and approve the documents. Establish "electronic signature" standards, protocols, methods, and tools to ensure that only authorized reviewers can approve documentation for decision-making implementation.

Unauthorized Posting of System Acquirer Documentation

Warning 17.3 Although Acquirer Request for Information or Proposal (RFI/RFP) solicitation and other data may be posted for public access, do not copy this material and post it unless your Enterprise has *prior written authorization* from the Acquirer or Vendor. Posting creates configuration management control, proprietary data, data concurrency, and copyright issues.

A Word of Caution 17.2

Proprietary Information Agreements (PIAs) and/or NDAs

Always consult with the Project Manager and/or Technical Director first for proper protocol and procedures for:

1. Communicating and exchanging data with external organizations
2. Establishing Proprietary Information Agreements and/or NDAs

Lastly, obtain approval from your Enterprise's Contracts, Legal, and Export Control Officers.

17.9 CHAPTER SUMMARY

Principle 17.3

Contract Issues Principle

Rule #1: *Always* read the contract!

Rule #2: When contract issues arise, go back to Rule #1!

This concludes our overview discussion of the System Documentation Strategy. In summary, establish the project's SE design and documentation strategy "up front" during the proposal phase concerning

- Who pays for what documentation?
- Who requires what types of documentation?
- Who is authorized to have "need to know" access to specific types of information?
- Who has what data rights?
- What *Quality Records* (QRs) will be maintained by the project concerning who is responsible and accountable, levels of formality, etc. Where are these documented?
- *How* will sensitive information will be protected.
- *When* will documentation at various levels of maturity will be released.
- What are the ground rules for what goes onto the project's DCL and DAL.
- What ownership, data rights, etc. will be established for CDRLs, SDRLs, etc.

In general, if System Acquirer contract funds were expended to document decision artifacts, the information belongs to the Acquirer. Information, in this context, includes (1) *formal* documentation such as plans, conference minutes, specifications, analyses, trade studies, and drawings and (2) *less formal* documentation such as meeting notes, analyses, trade studies, models, simulations, etc. The difference resides in what the contract requires as formal deliverable documentation. Always read the contract; when all else fails … read the contract!

The System Developer may be contractually obligated to deliver all documentation, except for *proprietary* information developed internally or by third parties such as subcontractors, vendors, etc. After delivery, System Acquirer may or may not find reading someone's notes, sketches, etc. easy to read or useful. The conversion of *less formal* data by a System Developer into coherent, meaningful documentation requires *more formality* that translates into affordability trade-offs such as cost and time.

Finally, fully understand: (1) all federal and state laws and regulations and (2) Enterprise command media, contract, and protocol requirements that govern Export Control, Proprietary Information (PI), Intellectual Property (IP), and copyrights.

17.10 CHAPTER EXERCISES

17.10.1 Level 1: Chapter Knowledge Exercises

1. What is project data?

2. What is technical data?

3. What is a CDRL?
4. What is the nominal sequencing and release of SE documentation?
5. What is a Subcontract Data Requirements List (SDRL)?
6. What is a DID?
7. What is a DAL?
8. What is a Design Criteria List (DCL)?
9. What are product data?
10. What are product description data?
11. What are product design data?
12. What are product support data?
13. What is the difference between approved data and released data?
14. What mechanisms are used to officially release SE documentation?
15. How should sensitive information data exchanges between Enterprises be orchestrated?
16. Why is Export Control so critically important in An enterprise?

17.10.2 Level 2: Chapter Knowledge Application Exercises

Refer to www.wiley.com/go/systemengineeringanalysis2e

17.11 REFERENCES

DID DI-MGMT-81453 (2007), *Data Item Description (DID): Data Accession List (DAL)*, Washington, DC: US Department of Defense (DoD).

McCumber, William H. and Sloan Crystal (2002), *Educating Systems Engineers: Encouraging Divergent Thinking*, Rockwood, TN: Eagle Ridge Technologies, Inc.

18

TECHNICAL REVIEWS STRATEGY

Successful system development requires incremental assessments of the status, progress, maturity, and risk of the evolving System Design Solution at *critical staging* or *control points*. These milestone events serve as decision *gates* to authorize and commit project resources to the next stage of System Development. At various stages of the System Development Processes Workflow (Figure 12.3), decision gates are intended to answer the following types of questions:

- Is there an agreement between the key Stakeholders—the User, End User, Acquirer, and System Developer—concerning the *adequacy*, *validity*, and *completeness* of requirements for the system, product, or service? Do the Stakeholders share a common understanding and interpretation of the requirements? Have all requirements issues been resolved? Are the requirements realistically achievable, measurable, testable, and verifiable within the planned technical, technology, cost, and schedule constraints and acceptable risk?
- Are we confident that the System Developer has a System Design Solution selected from a set of *viable* candidate solutions that represent the best balance of technical, technology, cost, schedule, and support performance and risks?
- Have System Performance Specification (SPS) requirements been derived, allocated, and flowed down to entities at the PRODUCT, SUBSYSTEM, ASSEMBLY, and SUBASSEMBLY levels of abstraction?
- Do we have a Preliminary System Design Solution that is *necessary* and *sufficient* to commit to detailed design with *acceptable* risk in terms of meeting technical, technology, cost, and schedule performance requirements?
- Does the System Design Solution provide the right level of detail that is *necessary* and *sufficient* to commit to the Component Procurement and Development Process (Figure 12.2) with *acceptable* risk in terms of meeting technical, cost, and schedule performance requirements?
- Are components that were procured and/or developed completed Fabrication, Assembly, Integration, & Test (FAIT) ready to enter System Integration, Test, and Evaluation (SITE)?
- Is the SYSTEM or PRODUCT ready to undergo formal SYSTEM Level verification and acceptance by the System Acquirer, as the User/End User's contractual and technical representative?
- Does the SYSTEM or PRODUCT meet the User's *validated* operational needs—its operational utility, suitability, usability availability, efficiency, and effectiveness (Principle 3.11)?

To address these questions, technical reviews enable the System Acquirer to *verify* that the *evolving* and *maturing* System Development Solution *complies* with contract or task specification requirements and is progressing with *acceptable* risk toward delivery on schedule and within budget.

18.1 DEFINITIONS OF KEY TERMS

- **Accomplishment** "An accomplishment is the desired result(s) prior to or at completion of an event that

System Engineering Analysis, Design, and Development: Concepts, Principles, and Practices, Second Edition. Charles S. Wasson.

Companion Website: www.wiley.com/go/systemengineeringanalysis2e

indicates a level of the program's progress" (DoD IMP-IMS Guide, 2005, p. 4).

- **Conference Minutes** A *quality record* that serves as a written summary of a formal project event and documents the attendees, agenda, discussion topics, decisions, actions items, and handouts.
- **Critical Design Review (CDR)** A major technical project event conducted by a System Developer with the System Acquirer and User to assess the progress, status, maturity, plans, and risks of each Configuration Item's (CI) detailed design solution. The event serves as a *critical staging point* for *authorizing* and *committing* resources for the Component Procurement and Development Process of the System Development Phase.
- **Criteria** "Criteria provide definitive evidence that a specific (Integrated Master Plan) accomplishment has been completed. Entry criteria reflect what must be done to be ready to initiate a review, demonstration, or test. Exit criteria reflect what must be done to clearly ascertain the event has been successfully completed" (DoD IMP-IMS Guide, 2005, p. 4).
- **Event** "An event is a program assessment point that occurs at the culmination of significant program activities: accomplishments and criteria" (DoD IMP-IMS Guide, 2005, p. 4).
- **Gate** "A checkpoint, or point in time, where a decision is made to 'Go Forward' with a project. 'Return' to a previous stage to obtain more information or 'Stop' it permanently" (DOE, 2007, p. 21).
- **Gate Decision** "Project decisions made by gatekeepers at each gate; may include Go Forward, Stop, Hold, or Return" (DOE, 2007, p. 21).
- **Gate Review** "A meeting with gatekeepers and project team at each gate to review results, assess them against criteria, and make project decisions" (DOE, 2007, p. 21).
- **Hardware/Software Specification Review (HSR/SSR** An assessment of each unique Hardware Configuration Item (HWCI) or Computer Software Configuration Item (CSCI) requirements specifications to determine their adequacy to authorize preliminary hardware design or preliminary software design and commit resources to support those activities.
- **In-Process Review (IPR)** An *interim* or *incremental* assessment of a work product such as a document or design solution during its development to provide "early" Stakeholder feedback as an independent, peer level, assessment including guidance and recommendations for resolving COIs and CTIs.
- **Integrated Master Plan (IMP)** "The IMP is an event-based plan consisting of a hierarchy of program events, with each event being supported by specific accomplishments, and each accomplishment associated with specific criteria to be satisfied for its completion" (DoD IMP-IMS Guide, 2005, p. 4).
- **Integrated Master Schedule (IMS)** "The IMS is an integrated, networked schedule containing all the detailed discrete work packages and planning packages (or lower-level tasks or activities) necessary to support the events, accomplishments, and criteria of the IMP (if applicable). The IMP events, accomplishments, and criteria are duplicated in the IMS … " (DoD IMP-IMS Guide, 2005, p. 5).
- **Master Program Schedule (MPS)** An MPS is a Gantt chart project schedule depicting major program and technical activities, key milestones, and events.
- **Major Technical Review** A formal review mandated by contract, task, or Enterprise command media that requires System Acquirer and System Developer stakeholder participation to achieve review objectives and outcomes.
- **Peer Review** A formal or informal review of an SE's or Integrated Product Team's (IPT) *work products* by knowledgeable Subject Matter Experts (SMEs).
- **Performance Measurement Baseline (PMB)** A PMB represents a baseline that documents the current project resource budget allocations, schedule, and technical requirements that have been evaluated to be reasonable to achieve and measure using Earned Value Measurement (EVM) methods.
- **Preliminary Design Review (PDR)** A major technical project event conducted by a System Developer with the System Acquirer and User to review a system's HWCI or CSCI designs with intent to authorize and commit resources to detailed design.
- **Production Readiness Review (PRR)** "A formal assessment by system stakeholders to authenticate the current Product Baseline and the readiness of the Technical Data Package (TDP) for production" (Former MIL-STD-1521B, 1992 p. 7/8).
- **Ready-to-Ship Review (RTSR)** *A formal* assessment by system stakeholders to determine the *readiness* of the system to be disassembled and shipped to a User's designated location.
- **System Design Review (SDR)** An assessment of the evolving system design solution to evaluate the completeness and maturity of the system architecture and its interfaces, identification of PRODUCTS/SUBSYSTEMS, allocation of SPS requirements to products/subsystems, and risks.
- **System Requirements Review (SRR)** "An assessment of the conciseness, completeness, accuracy, reasonableness, and risk of the SPS requirements to permit

the development of a system with an intent to avoid misinterpretations, inconsistencies, and errors, especially during *system verification*, *validation*, and *acceptance*."

- **System Verification Review (SVR)** A formal assessment by system stakeholders to authenticate the results of the Functional Configuration Audit (FCA) and Physical Configuration Audits (PCA) relative to *SPS* requirements and verification methods.
- **Technical Reviews** "A series of system engineering activities by which the technical progress on a project is assessed relative to its technical or contractual requirements. The reviews are conducted at logical transition points in the development effort to identify and correct problems resulting from the work completed thus far before the problems can disrupt or delay the technical progress. The reviews provide a method for the performing activity and tasking activity to determine that the development of a configuration item and its documentation have a high probability of meeting contract requirements." (MIL-HDBK-61A, 2001, p. 3-10).
- **Test Readiness Review (TRR)** An assessment of the maturity of all aspects of a multi-item/CI to determine (1) its *state of readiness* to proceed with testing with a critical focus on Environmental, Safety, and Health (ES&H) concerns and (2) authorize initiation of the tests.

18.2 APPROACH TO THIS CHAPTER

Technical reviews are one area in which there is a diversity of opinions and implementations that are Enterprise dependent. Commercial industry, which operates from a speculative business model that projects and anticipates marketplace needs, employs a Stage-Gate Process (Figure 3.4) to validate, authorize, and commit resources for system or product development. Governmental agencies acquire systems, products, or services typically via solicitation-based procurements processes that employ contract-driven decision stage gates.

If you investigate these processes, they do share commonalities from a Systems Engineering and Development (SE&D) perspective. They require:

- Assessment of the completeness, consistency, coordination, and risk of technical plans and resources.
- Validation of the *necessity* and *sufficiency* of specification requirements for system boundaries, interfaces, capabilities, performance levels, and design and construction constraints.
- A conceptual view such as a Concept of Operations (ConOps) that describes how the Stakeholders - Users and End Users - envision integrating the system, product, or service into their Enterprise assets, deploying the asset to the field or marketplace, operating and supporting the asset, and disposing of the asset.
- An assessment of the *status, progress, maturity,* and *risk* of an evolving multi-level System Design Solution that includes architectures, interfaces, designs, Specialty Engineering disciplines—Safety, Reliability, Maintainability, Sustainment, Human Factors.
- Establishment and continual assessment of the integrity of the Developmental Configuration and Production Baselines (Chapter 16) and corrective actions.
- Validation of the *integrity, completeness,* and *consistency* of a TDP to commit to component procurement and development and corrective action evaluation throughout procurement.
- Advance planning, coordination, and resource adequacy for SITE.
- An assessment of the readiness for SYSTEM or PRODUCT acceptance and delivery.

Beyond these words, the scope of actual implementations differs by country and business domain—medical, aerospace and defense, transportation, energy, and so forth. As a result, our approach is to highlight what technical reviews need to accomplish from a Systems Engineering (SE) perspective. You should refer to your Enterprise command media for your business domain for specific policy and implementation guidance. To facilitate our discussion, we will use technical reviews employed by the US Department of Defense (DoD) as a frame of reference to illustrate some of the types of technical decision-making to be accomplished. The contexts of these reviews are universal for system development and equally apply to medical, energy, transportation, communications, and other business domains.

18.3 TECHNICAL REVIEWS OVERVIEW

System development is highly dependent on a proactive, efficient, technical decision-making throughout the System Development Phase that involves the System Acquirer and User. The *timing* of constructive technical assessment and feedback by stakeholders at *critical control* or *staging points* enables the System Engineering and Integration Team (SEIT) to evolve the System Design Solution to maturity to ensure it meets their needs within resource constraints. The mechanisms for staging these assessments and making key technical decisions consist of a series of technical reviews.

Technical Review Scheduling Principle

Principle 18.1 Schedule technical reviews at critical *control* or *staging* points in system development to assess the status, progress, maturity, compliance, and risk of the evolving System Design Solution in meeting its contract and specification requirements; no sooner, no later.

The objective of a technical review is to enable *key stakeholders* to assess the evolving system design solution at critical *staging* or *control points* to determine the *progress*, *status*, *maturity*, *integrity*, *plans*, and *risk* as a condition of a System, CI, or Non-Developmental Item (NDI) for committing resources for the next segment or phase of project activities.

18.3.1 Categories of Technical Reviews

Technical reviews consist of *major reviews* conducted formally and *internal reviews* that tend to be less formal. In general, the *major reviews* are typically required by contract and involve the System Acquirer and User *stakeholder* representatives as participants.

18.3.2 Formal Contract Technical Reviews

Contract reviews, which are referred to as project *events*, are formally conducted in accordance with the contract's *Terms and Conditions* (Ts&Cs) of the contract. Generally, the contract identifies the reviews to be conducted and specifies guidance for preparing, conducting, and completing the review. In addition to the contract guidance that defines how the reviews are to be conducted, there is also a *protocol* associated with providing directions and guidance during the review, attendees to be invited and by whom, and so forth.

Conference Invitations Protocol Principle

Principle 18.2 Under contract protocol, System Developers invite the System Acquirer to project events; the System Acquirer extends invitations to the User and End Users unless the System Acquirer has authorized the System Developer to send invitations.

18.3.3 Traditional Contract Technical Reviews

Years ago, contracts used technical reviews as a limited, but important, window for the customer (System Acquirer) to look into the contractor's operations and assess how well the effort was progressing. Depending on contract size, complexity, and priority, the customer assigned and deployed an on-site representative to the contractor's facility. This individual's task was to monitor day-to-day operations and communicate to the "home organization" views on how well the contractor efforts were progressing. In general, the reviews enable customer project managers to ask themselves "*does the System Developer's design solution review materials correlate with the progress depicted in prior phone conversations with the contractor—'glowing' contractor status and progress reports and so forth?*"

Several weeks prior to a technical review, the System Developer prepared large documentation packages for distribution to the customer and User for review. During this review, the contents of the Technical Data Package (TDP) were discussed over a period of several days in agonizing detail. As a result, technical reviews consumed large amounts of time, were costly, and provided "feedback" too late, which caused *rework* and *impacted* customer schedules. These issues, in conjunction with escalating contract costs and process inefficiencies, prompted the need for current System Development status information in terms of hours or days, not weeks or months.

18.3.4 Integrated Process and Product Development (IPPD)

The need for acquisition reform, streamlining, process improvement, and reduced rework influenced a move toward IPPD environments. IPPD environments, which include the System Acquirer as an integral part of the "team," provide on-site access to the details and nuances of the product development effort.

As a result of the DoD Acquisition Reform initiatives in the DoD, for example, the technical reviews paradigm began to shift. Whereas technical reviews consumed several days agonizing over documentation details, the paradigm shifted to simply spending a few hours resolving Critical Operational Issue (COI) and Critical Technical Issue (CTI) decisions. *Why?* If the User and System Acquirer are participants in the IPPD team processes, either on-site or virtually, they should be *intimately* familiar with the design details. The only agenda topics remaining should focus on *resolution of issues* between the System Acquirer and the System Developer.

Finally, another aspect was the shift from *date-driven* to *event-driven* contract reviews.

18.3.5 Date-Driven versus Event-Driven Reviews

Technical reviews, specified by contract, are typically of two types: *date-driven* or *event-driven*.

- *Date-driven reviews* mandate that a review be conducted "X" days After Contract Award (ACA)—on a specific calendar date.
- *Event-driven reviews* are conducted when the development efforts reach specific maturity levels, usually within a general timeframe. The timeframe may be specified as "within XX days or months ACA."

18.4 CONDUCT OF TECHNICAL REVIEWS

Technical reviews are conducted in accordance with the Ts&Cs of the contract, subcontract, or associated agreements. Every contract should specify the *who*, *what*, *when*, *where*, *how*, and *why* technical and programmatic reviews are to be accomplished. If not, do not sign until all of the contract's Ts&Cs of the technical reviews have been clearly and explicitly delineated, mutually understood, and agreed to by all parties.

Technical Review Entry/Exit Criteria Principle

Principle 18.3 Avoid signing contracts with ambiguous technical review *entry* and *exit* criteria language that may be *open* to interpretation, especially when it comes to getting paid.

Technical reviews are more than simply forums to make nice and orderly presentations. The reviews are an opportunity to brief the System Acquirer and User "for the record" on what progress has been made in maturing the System Design Solution since the last review. Reviews provide an opportunity for the System Acquirer to validate what has been documented in the monthly contract progress reports.

18.4.1 Staging and Conduct of Technical Reviews

In their truest technical form, professionally objective and constructive technical reviews strive to ensure that the evolving System Design Solution is maturing as planned, will yield a final system, product, or service that satisfies the Stakeholder - User and End User - operational needs with no or limited *latent defects*, and makes corrective actions to achieve these results. This requires serious, positive, and open collaboration between the System Acquirer and System Developer. Unfortunately, technical reviews sometimes become "dog and pony shows" as a perfunctory "check the box" event to satisfy contract payment criteria and so forth. In these situations, the System Developer showcases a parade of presentations—parade floats—for the System Acquirer and then asks how they liked the System Design Solution.

Inviting a System Acquirer to review moderate to highly complex System Design Solutions *without* adequate prior review is *problematic* for all parties in terms of achieving professional objective stated previously. In all fairness to the System Acquirer, comprehending the breadth and depth of the System Design Solution is often *impractical*, especially if the System Acquirer review team *lacks* sufficient staffing and technically qualified and experienced reviewers.

One solution to this problem was instituted in the late 1980s with the introduction of the IPT concept. In this concept, the System Developer creates a project organizational framework of IPTs that are uniquely assigned to focusing on developing a specific System, Product, Subsystem Assembly, and so forth. The System Acquirer also has IPTs related to system planning for system deployment, operations and support, and disposal. In some cases, these IPT discussions may be germane to the scope of System Developer activities, in other cases not due to Acquirer unique Enterprise coordination and planning.

Conceptually, the intent is to provide open access to System Developer IPTs for the Acquirer to review work products, status, progress, and risk. Having System Acquirer personnel intimately familiar with the respective System, Product, Subsystem, Assembly, and Subassembly Level solutions avoids the need for days of consuming project resources agonizing over unfamiliar detail design presentations. As a result, major technical reviews shifted to a focus on resolving COIs/CTIs thereby requiring less meeting time. Ideally, the IPT approach yields positive results, assuming all parties are able to work together in a positive, constructive manner.

To work successfully, System Developer IPT environments do require constraints such as:

- System Acquirer personnel attending IPT meetings are not permitted to provide technical decisions or direction; only the System Acquirer Contracting Officer (ACO) is officially authorized to provide contract direction.
- Both parties have to recognize that meeting work products, discussions, and decisions are limited to the meeting environments and should not enter the "water cooler" gossip chain to become unplanned distortions of the facts, hearsay, etc.—the Law of Unintended Consequences. The project resources required to manage these situations trump System Acquirer open participation in IPT meetings. When both parties professionally respect the meeting constraints, open participation can be very productive for all parties and yield mutual benefits.

18.4.1.1 Checks and Balances Benefits Technical reviews provide checks and balances for the System Developer, System Acquirer, and User with inherent benefits for all.

18.4.1.2 System Acquirer and User Perspectives For the System Acquirer and User, technical reviews provide the opportunity to:

1. Assess the status, progress, maturity, and risks of the product development efforts.
2. Factor project technical review results into the realism of User deployment schedules.
3. Express priorities and preferences.
4. Provide legal technical direction, depending on contract type Ts&Cs.

18.4.1.3 System Developer Perspective For the System Developer or Service Provider, the technical reviews provide the opportunity to:

1. Clearly demonstrate product development maturity and progress.
2. Address and resolve COIs/CTIs.
3. Validate System Acquirer priorities and preferences.
4. Obtain System Acquirer agreement, if appropriate for the type of contract, baseline system documentation to serve as a reference for future discussion, and scope for technical guidance and direction.

18.4.1.4 Type of Contract In general, the type of contract determines how a review is accomplished and the degree of *influence* or *approval* the customer has over the information presented. In the case of Firm-Fixed Price (FFPs) contracts, the System Developer conducts technical reviews to inform the System Acquirer and User about progress to date, current status, and risks.

Depending on the type of the contract,[1] the System Acquirer may be limited in the *degree* to which they can *approve* or *reject* the System Developer's solution without contract modification. In contrast, Cost Plus Fixed Fee (CPFF) contracts typically enable the System Acquirer to exert a large amount of *control* over the contractor's decision-making and make adjustments in cost and schedule to *accommodate* changes in the contract's technical direction.

Principle 18.4 **Contract Protocol Principle**

Read and *thoroughly understand* your contract. If unclear, consult with your Enterprise's Project Management Team (PMT). The PMT, in turn, consults with internal Contract and Legal organizations for specific guidance and expertise in interpreting the contract's Ts&Cs.

18.5 CONTRACT REVIEW REQUIREMENTS

Major technical reviews are conducted by the System Developer as a contract event in accordance with the Master Project Schedule (MPS), IMP, or IMS, as appropriate.

18.5.1 Technical Reviews Location

Technical reviews are conducted at locations specified in accordance with the contract's Ts&Cs. Generally, the reviews are performed at the item's System Developer's facility due to close proximity to the documentation and actual hardware and software for demonstrations.

[1]Refer to the Project Management Institute (PMI) *A Guide to the Project Management Body of Knowledge* (PMBOK®) for additional information about project management contract types.

Author's Note 18.1 **"Item" Context**

Remember, "item" in the previous context represents a SYSTEM, PRODUCT, SUBSYSTEM, and so forth (Chapter 16). For example, if a subcontractor is developing a SUBSYSTEM, reviews are conducted at the subcontractor's facility and include invitations to the prime System Development contractor, who may elect to invite their customer, the System Acquirer. Under contract protocols (Principle 18.2), the System Acquirer always invites the User *unless* prior authorization has been made with the System Developer.

18.5.2 Planning and Orchestrating the Review Conferences

Major technical reviews are often referred to as *conferences*. Each conference consists of three phases of activities to ensure successful completion of the review: Pre-Conference, Conference, and Post-Conference.

- *Pre-Conference Phase* activities include coordination between the System Developer and System Acquirer to set the date, time, and location of the review, as well as to set the agenda, invitees, special facility access, and arrangements such as security, parking, and protocols.
- *Conference Phase* activities include conducting the conference in accordance with the planned agenda, classification level, rules of conduct/engagement, recording of *conference notes*, and *action items*.
- Post-*Conference Phase* activities include resolving conference action items, preparing and approving the conference minutes, incorporating corrections into documentation, and establishing baselines, where applicable.

Principle 18.5 **Conference Documentation Principle**

Conference minutes document attendees, agenda, discussion topics, meeting, and action items. Perform the task well and obtain System Acquirer acceptance via the ACO.

Principle 18.6 **Conference Minutes Principle**

Technical review agendas, attendees, discussions, decisions, and action items are documented via conference minutes and reviewed, approved, and released via contract protocol.

18.5.3 Contract Review Completion Exit Criteria

Some contracts require completion of technical events, such as reviews, as a prerequisite for obtaining contract progress payments. *Exit criteria* are employed to *explicitly* identify what must be accomplished as a contract *condition* for System Acquirer *acceptance*.

Some formal solicitations such as Request for Proposals (RFPs) require identification of *exit criteria* as a requirement. Where this is the case, the Offeror's proposal may become part of the contract. If contract progress payments to the System Developer are linked to project events, such as reviews, make sure that *exit criteria* are explicitly stated in language that *does not* require interpretation when time comes to contractually close the review and pay the contractor.

A Word of Caution 18.1

Review Entry and Exit Criteria

Review *entry* and *exit criteria* may appear simple on *initial inspection*. However, they can easily become *major showstoppers* due to misinterpretation(s), especially where contract progress payments are "on the line."

When you state that the software design is complete, be *very explicit* as to what you mean by "Software Design Complete." It bears emphasizing that the System Acquirer interprets "Software Design Complete" as *everything*:

- Think about these statements!
- Recognize the scope of your intent *before* you write them into a proposal and sign a contract, especially if progress payments are at risk. This includes the *IMP* and associated dictionaries.

Why? Simply stated, if you *do not* follow the contract words, you do not get PAID! And guess who negotiated the contract words? Your Enterprise and project did!

18.5.4 Posting and Distribution of Technical Review Materials

Today's contracting environment often involves contractor teams across the country and around the globe to integrate their efforts via collaborative development and review environments. For technical reviews, World Wide Web (WWW)-based reviews provide opportunities to perform IPRs without the expense of travel or disruption of work. However, keep in mind that this media also presents major data security issues related to proprietary data, copyright law, security classification, and export control.

Warning 18.1

Export Control of Technology

The US ITAR governs the export of critical technologies and data by any means including the Internet. Refer to the Export Control Warning (Warning 17.2). *Always refer to your Legal and Contracts organizations as well as the Export Control Officer for guidance in these areas.*

18.5.5 Technical Review Contract Direction

Principle 18.7

Official Contract Direction Principle

Under contract protocol, only the System Acquirer's Contracting Officer (ACO) is *officially authorized* to issue technical direction to the System Developer in response to technical review comments, conference minutes, action items, contract modifications, and acceptance of contract documents.

The conduct of contract technical reviews is often viewed as a causal event. Beware! Despite the suave coolness of the System Developer's "*we're glad you are here*" and the System Acquirer's "*we're glad to be here*" pleasantries, the activity carries some very *serious* political and legal implications that demand your full attention and awareness.

Remember, only the ACO is officially authorized to provide contract direction to the System Developer regarding the technical review. The System Acquirer's Project Manager (PM) provides project and technical guidance to the ACO to convey officially to the System Developer. As a result, you will often hear an System Acquirer Project Manager make introductory remarks at the beginning of the review and begin with a disclaimer: "*we are here to listen … anything we say or ask does not infer or should be construed as technical direction … Our (*System *Acquirer) Contracting Officer is the only one officially authorized to provide contract direction.*"

18.5.6 Technical Issue Resolution

One of the greatest challenges of technical reviews is making sure all showstopper COIs/CTIs are resolved to the mutual satisfaction of the System Acquirer and System Developer Enterprises.

COIs/CTIs often have far-reaching impacts that range from achieving the mission objectives to simply printing out a report. COIs impact one of more of the integrated set of SE System Elements—EQUIPMENT, PERSONNEL, FACILITIES, and so forth. COI and CTI impacts span the spectrum from SPS requirements to PART Level design requirements, and vice versa. Therefore, it is *imperative* that COI/CTI issues be introduced, understood, and resolved at the technical reviews to avoid schedule impacts.

Projects often refuse to acknowledge or publicize COIs/CTIs. Ironically, these issues are often ignored until the project finally has to confront them. Humans have a natural propensity to pretend and believe that problems—such as COIs/CTIs—will simply "just go away."

Sometimes this happens; however, in most cases this is a *panacea*. Although there are reasonable and practical bounds to issue resolution, sooner or later you will learn to confront these issues "early on." Historically, you either pay the price now or pay an even *greater* price farther downstream (Chapter 13 cost-to-correct) to resolve an outstanding issue, assuming the set of potential solutions are viable at that point in time in terms of cost and schedule performance.

The realities are Users, System Acquirers, and System Developers procrastinate on resolving COIs/CTIs. Enterprises spend millions of dollars every year publicizing how they have cut back expenses by reducing the number of pencils and paper people use. Yet, these expenses are often insignificant when compared to the dollars wasted procrastinating to resolve the inefficiencies of COIs/CTIs decision-making. There *are no* easy solutions or "quick fixes" other than to say that all parties—Users, System Acquirers, System Developers, and subcontractors among these—collectively need to do a better job confronting these challenges. So, *how does resolution of COIs/CTIs relate to technical reviews*?

Technical reviews, in general, provide the forum for time-constrained "open technical debate" to resolve any outstanding or lingering COIs/CTIs. Granted, some people "debate" just to hear themselves talk. The context here is the need for an environment in which the participants remove their Enterprise "hats and badges" and focus all energies on alternative paths to bring an issue to consensual closure to the mutual satisfaction of all parties. *Tough to accomplish*? Yes! *Are there alternatives*? Yes, there is a "programmatic direction approach" whereby the project managers for the System Acquirer or System Developer PMs may dictate a solution to meet schedule. This is not a desired path *technically* or *programmatically*. If you don't like the "programmatic direction approach," take the necessary steps to ensure that the "technical approach" works.

Obviously, you do not want to consume valuable time and resources in a technical review "debating" COIs and CTIs, unless they surface *unexpectedly* from the *unknown* – unknown-unknowns. If COIs/CTIs are known prior to a review, special arrangements should be made before a review to schedule some form of "working group" meeting of all parties. The session must be constrained with the understanding that a group must bring the issue to closure and employ the review to present the recommendations and finalize resolution.

One of the most *overlooked* aspects of technical reviews is the *psychology* of managing customer perceptions. From a System Developer's perspective, the reviews are opportunities to manage customer *expectations*. Likewise, System Acquirers and Users formulate *perceptions* during the review as well as throughout contract performance. Perceptions of System Developer contract performance influence *future interactions*, be it future business, follow-on business, or negotiable items of the contract. While the primary focus on a technical review is the current contract or task, the User and System Acquirer may be subconsciously asking themselves, "*do we want to do business with this System Developer again?*"

Although not discussed openly, the reviews enable System Acquirers to *validate* levels of *confidence* that they made the right choice in selecting your Enterprise to perform on this contract as opposed to your competition. Remember, successful contract performance reflects on System Acquirers and how their Stakeholders, the User, and executive management view your Enterprise.

18.6 IN-PROCESS REVIEWS (IPRs)

IPRs, which represent another class of technical reviews, occur in two forms:

1. As a System Developer's incremental assessment of its evolving *work products*.
2. As an Acquirer's readiness assessment for conducting a major technical review for a contract.

Let's describe each of these types.

18.6.1 System Developer IPRs or Peer Reviews[2]

System Developer personnel conduct peer level IPRs for their evolving work products with internal Stakeholders. IPRs serve three primary purposes:

1. To assess compliance of an evolving and maturing work product relative to SPS or EDS requirements, contract requirements, and Enterprise OSPs.
2. To identify COIs and CTIs and provide collaborative recommendations for corrective action.
3. To identify Action Items (AIs) for corrective action and address completion of previous IPR AIs.

IPRs review evolving and maturing work products such as plans, specifications, designs, test procedures, and so forth. Where project organizations are based on Product

[2]For additional information about Peer Reviews, refer to the NASA *System Engineering Handbook* NASA/SP-2007-6105 Rev1 Appendix N: "Guidance on Technical Peer Reviews/Inspections," pp. 312–314, as an example.

Development Teams (PDTs), the System Acquirer and User may have standing invitations to participate in the reviews.

IPRs, like any type of review, should be prescribed by Enterprise command media such as Organizational Standard Processes (OSPs). OSPs define how IPRs are to be conducted, who is to be invited, conditions for cancellations, meeting minutes, action items and tracking, and so forth. Projects typically issue internal OSPs such as a project memo that defines "up front" role-based personnel such as the LSE, QA, Safety, and others that are required to be invited to IPRs. Since IPRs are a critical staging point for efficient usage of resources, peer level participation is important. As a result, OSPs typically have ground rules for IPR cancellation if specific roles and levels of participation are not met thereby resulting in rescheduling.IPR participants primarily include project personnel. However, due to the need for an independent assessment, Enterprise Subject Matter Experts (SMEs) external to a project as well as System Acquirer personnel may be invited to attend, especially if they have field offices within the System Developer's facility.

Author's Note 18.2

System Acquirer Participation in Informal IPRs

System Acquirer participation in System Developer project IPRs has a lot of serious implications. System Acquirer personnel sometimes unwittingly attempt to provide direction, which is a serious violation of contract protocol. Only the ACO can legally provide System Acquirer contract direction (Principle 18.7).

Given this constraint, System Acquirer participation becomes an assignment of "reporting back," which is fine. However, any *misperceptions* or *misinterpretations* gyrate into an issue that *unnecessarily spirals out of control* for political purposes.

Conceptually, in the "spirit of openness," System Acquirer participation in IPRs should be fine. However, given the risk of *unintended* and *unnecessary* political ramifications, the decision needs to be made on an individual contract basis.

18.6.2 Contract IPRs

Major technical reviews can be very costly, especially if conducted prematurely. Some contracts may require the System Developer to conduct Contract IPRs to assess project readiness for conducting the formal technical review. Contract IPRs are conducted 30 to 60 days prior to a major technical review to assess the readiness to "go-ahead" with SDR, PDR, CDR, and so forth. Due to the increasing expenses of travel to attend review events, this approach increases the *level of confidence* that the System Developer will be prepared on the required date and minimize the cost inefficiencies of rescheduling immature or poorly prepared project reviews.

18.7 CONTRACT TECHNICAL REVIEWS

Technical review agendas consist of two types of discussion topics:

1. Programmatic topics
2. Technical topics unique to the type of review

In general, more than 90% of the review time should be devoted to technical agenda items. Sometimes, this is not the case and the System Acquirer and System Developer have to resolve programmatic issues that may be stagnating as a barrier to technical progress.

18.7.1 Success Criteria: Technical Reviews

Technical reviews, in general, should be an *uneventful* assessment of the technical *progress*, *status*, and *maturity* of the evolving System Design Solution and Developmental Configuration. Unfortunately, technical reviews become *emotional* events when the technical review *fails* to meet System Acquirer of System Developer expectations. Technical review success is dependent on managing expectations of:

- *Entry criteria*—What performance-based outcomes and objectives have the System Acquirer and System Developer agreed via the contract or other agreements to be *necessary* and *sufficient* for conducting a technical review?
- *Exit criteria*—What performance-based outcomes and objectives have the System Acquirer and System Developer agreed via the contract or other agreements to be *necessary* and *sufficient* to declare a technical review as "complete" or "closed?"

18.7.2 Entry Criteria: Technical Reviews

Principle 18.8

Technical Review Entry Criteria Principle

At a *minimum*, establish explicit *entry criteria* for technical reviews based on:

- *Level of maturity* of work products to be reviewed.
- COIs or CTIs to be resolved.
- Closure status of outstanding AIs.
- Assessment of System Design Solution supporting data—analyses, trade studies, etc.
- Configuration Status Accounting (CSA) of the Development Configuration.
- Other criterion, as required.

One way to manage System Acquirer, User, and System Developer technical review expectations is to establish a set of *entry criteria* as part of contract Statement of Work (SOW). Since technical reviews as conferences have a cost in terms of budget and schedule, both parties negotiate expectations. This does not preclude the System Acquirer and System Developer Project Managers from coordinating via contract protocol on other topics for discussion.

One of the challenges of technical reviews is their *efficiency* and *effectiveness* in conducting business. Often, discussions become bogged down in esoteric topics that needlessly consume time that is better spent on more important topics that may not get addressed. Therefore, it is imperative that the System Acquirer and System Developer PMs jointly manage this time.

18.7.3 Conduct of the Technical Review

Effective technical reviews should be of short duration and focus on the key issues to be resolved, not a line-by-line or presentation chart educational exercise for the System Acquirer and User personnel. There are always exceptions when a System Acquirer or User decision-maker decides they need to attend. *How should we solve these issues*?

- First, assuming the project employs IPTs that have open, on-site access for System Acquirer and User personnel, those personnel should be intimately familiar with the evolving System Design Solution and issues prior to the review including keeping their management informed.
- Second, System Developers should provide advance review material to the System Acquirer for distribution to their personnel and the User with appropriate time to prepare for the review. Typically, advance review materials are required in many SOWs.

18.7.4 Exit Criteria: Technical Reviews

Technical Review Exit Criteria Principle

Principle 18.9

At a *minimum*, establish explicit *exit criteria* for technical reviews based on:

- Closure of outstanding and new AIs.
- Resolution of COIs or CTIs.
- Acceptable assessment of System Design Solution maturity and risk.
- Approval of conference minutes.
- Other criterion, as required.

Technical review *exit criteria* simply posture the review for success; however, they do not guarantee success. This requires establishing objectives for successful outcomes to be achieved. The results are technical review *exit criteria* documented in the SOW as well coordinated by the System Developer and System Acquirer PMs prior to the review. Tables 18.1–18.10 provide examples of types of outcomes expected from various types of technical reviews.

18.7.5 Standard Review Work Products and Quality Records (QR)

The preceding discussions highlighted what is to be accomplished at reviews. In a decision-making event, it is important to document the review materials and results for *historical* and *reference* purposes. The following is a list of example *work products* and *QRs* that, at a *minimum*, should be produced for each of the major technical reviews:

Technical Review Work Products and QR Examples

Example 18.1

- Conference agenda (e.g., meeting minutes and action items)
- Attendee lists
- Presentation materials
- Handouts (analyses, trade studies, modeling and simulation results, etc.)
- Conference notes
- Conference minutes
- Action items (AIs) Open/On-Hold/Closed
- Other supporting documentation, as required

18.7.6 Standard Technical Review Items

Standard technical review topics include the following examples:

Technical Review Topic Examples

Example 18.2

- Review updates to technical plans, approaches, or procedures.
- Review risk mitigation plans and approaches.
- Review of current schedule status and progress.
- Review Contract Data Requirements List (CDRL) item status.
- Review Contract Line Item Number (CLIN) status.
- Review results of the Requirements Traceability Audit (RTA).
- Provide authority, if applicable to the contract, to proceed and commit resources to the next System Development Phase Process segment (Figure 12.2).

TABLE 18.1 Example IBR Supporting Objectives/Exit Criteria and Expected Decisions

Item	Example IBR Objectives/Exit Criteria	Expected Decision(s)
IBR-1	Assess the *adequacy*, *completeness*, *consistency*, and risk of the *SPS*	• Action item(s) or • Concurrence/approval
IBR-2	Assess the adequacy, completeness, consistency, and risk of the contract and program schedule elements: • MPS • IMP • IMS	• Action item(s) or • Concurrence/approval
IBR-3	Assess the adequacy, completeness, consistency, and risk of the contract and program cost elements: • CSOW • CWBS • CDRL items • CLINs • Control accounts • Work packages	• Action item(s) or • Concurrence/approval
IBR-4	• Establish the PMB: • Technical performance baseline elements • Schedule performance baseline elements • Cost performance baseline elements	• Action item(s) or • Concurrence/approval

TABLE 18.2 Example SRR Supporting Objectives/Exit Criteria and Expected Decisions

Item	Example SRR Objectives/Exit Criteria	Expected Decision(s)
SRR-1	Verify that the User's *problem space* and solution(s) space(s) are properly understood and bounded (e.g., what IS/IS NOT part of the *solution space*)	• Action item(s) or • Concurrence/approval
SRR-2	Verify that the requirements *completely*, *consistently*, *accurately*, and *concisely* articulate and bound the User's *solution space (s)* for the SYSTEM in a manner that *avoids misinterpretation* by multiple reviewers	• Action item(s) or • Concurrence/approval
SRR-3	Verify that any ambiguous, overlapping, incomplete, inconsistent, or unbounded requirements are eliminated	• Action item(s) or • Concurrence/approval
SRR-4	Evaluate the quality (e.g., bounded, measurable, testable, and verifiable) of the requirements in terms of required capabilities and performance	• Action item(s) or • Concurrence/approval
SRR-5	Determine if all stakeholder requirements have been adequately addressed subject to contract cost schedule constraints	• Action item(s) or • Concurrence/approval
SRR-6	Verify each SPS Section 3.X requirement (Table 20.1) has at least one or more Section 4.X verification methods such as Inspection, Analysis, Demonstration, Test, or combinations of these	• Action item(s) or • Concurrence/approval
SRR-7	Verify that the Section 4.0 Qualification Provision (Table 20.1) verification methods *represent* the least cost, schedule, and technical risk approach to proving the requirements compliance	• Action item(s) or • Concurrence/approval
SRR-8	Where appropriate, obtain consensus of *SPS* requirements, interpretations and clarifications, modifications, etc. subject to ACO approval	• Action item(s) or • Concurrence/approval
SRR-9	Where applicable, obtain authorization to establish the System Requirements Baseline	• Action item(s) or • Concurrence/approval

TABLE 18.3 Example SDR Supporting Objectives/Exit Criteria and Decisions Expected

Item	Example SDR Objectives/Exit Criteria	Expected Decision(s)
SDR-1	Assess the progress, status, maturity, and risk of the System Level design solution—architecture, interfaces, etc.	• Action item(s) or • Concurrence/approval
SDR-2	Review of the Preliminary ConOps	• Action item(s) or • Concurrence/approval
SDR-3	Review Mission Event Timeline (MET) allocations	• Action item(s) or • Concurrence/approval
SDR-4	Review and approve, if appropriate, SPS requirements allocations to Products or Subsystems and other components	• Action item(s) or • Concurrence/approval
SDR-5	Review any supporting analyses and trade studies relevant to SDR decision-making	• Action item(s) or • Concurrence/approval
SDR-6	Review *Preliminary* Product or SUBSYSTEM Level EDS	• Action item(s) or • Concurrence/approval
SDR-7	Resolve any critical *operational* or *technical* issues related to system capabilities, performance, interfaces, and design criteria (e.g. data)	• Mutual resolution • Closure
SDR-8	Review the system Life Cycle Cost analysis	• Action item(s) or • Concurrence/approval
SDR-9	Establish the Allocated Baseline plus any corrective actions required as criteria for acceptance	• Action item(s) or • Concurrence/approval

These topics serve as standard agenda topics for every review and are presented in summary form as part of the introductory remarks. In addition to the standard review topics, review unique topics are provided in the discussions of each review that follow.

Author's Note 18.3

DoD Technical Reviews

As a very rigorous and disciplined technical review process, US DoD technical review process provides insights into the types of reviews and topics required to assess the status, progress, maturity, and risk of the evolving System Design Solution. In the discussions that follow, *we will use the DoD process as an example reference model*. The important take-away here is that if you develop *commercial* systems, products, or services, you and your Enterprise should have your own technical review process appropriate for your line of business and industry.

18.7.7 Common Types of Technical Reviews

Common types of major system level technical reviews include:

- Integrated Baseline Review (IBR)
- System Requirements Review (SRR)
- System Design Review (SDR)
- Software Specification Reviews (SSRs)
- Hardware Specification Reviews (HSRs)
- Preliminary Design Review (PDR)
- Critical Design Review (CDR)
- Test Readiness Reviews (TRRs)
- System Verification Review (SVR)
- Production Readiness Review (PRR)

The sequencing of these reviews is illustrated in Figure 18.1.

Each of the SYSTEM Level technical reviews is supported by PRODUCT, SUBSYSTEM, AND ASSEMBLY Level assessments that culminate in a baseline for each item or CI. Because of the expense of travel, some reviews may be conducted virtually via audio teleconferences, Audio and Video Teleconferences (ATCs / VTCs), or on-site, depending on item maturity. In some instances, several PRODUCT, SUBSYSTEM, OR ASSEMBLY Level reviews may be conducted sequentially during the same timeframe at a location, typically the System Developer's facility. Consider the following example:

TABLE 18.4 Example HSR/SSR Objectives/Exit Criteria and Decisions Expected

Item	Example HSR/SSR Objectives/Exit Criteria	Expected Decision(s)
HSR/SSR-1	Assess the *adequacy* and *completeness* of requirements allocations and traceability of an HWCI or CSCI to its *higher-level item development specification*	• Action items or • Concurrence/approval
HSR/SSR-2	Resolve any COIs or CTIs related to HWCI or CSCI capabilities, performance, interfaces, and design criteria (e.g. data)	• Action items or • Concurrence/approval
HSR/SSR-3	Establish the criteria and corrective actions required to baseline the HWCI or CSCI requirements	• Action items or • Concurrence/approval
HSR/SSR-4	Review each HWCI/CSCI, including use cases and scenarios, inputs, processing capabilities, and outputs	• Action items or • Concurrence/approval
HSR/SSR-5	Review HWCI/CSCI performance requirements, including those for execution time, storage requirements, and similar constraints	• Action items or • Concurrence/approval
HSR/SSR-6	Review *control flow* and *data flow interactions* between each of the entities that comprise the HWCI/CSCI	• Action items or • Concurrence/approval
HSR/SSR-7	Review interface requirements between the HWCI/CSCI and all other CIs both *internal* and *external* to the HWCI/CSCI	• Action items or • Concurrence/approval
HSR/SSR-8	Review *qualification* or *verification* requirements that identify applicable levels and methods of testing for the software requirements that comprise the HWCI/CSCI	• Action items or • Concurrence/approval
HSR/SSR-9	Review any special delivery requirements for the HWCI/CSCI	• Action items or • Concurrence/approval
HSR/SSR-10	Review Quality Factor requirements: • Reliability, Maintainability, Availability (RMA), • Usability, • Testability, • Flexibility, • Portability, • Reusability, • Security, • Interoperability	• Action items or • Concurrence/approval
HSR/SSR-11	Review mission requirements of the System and its associated operational and support environments related to the HWCI/CSCI	• Action items or • Concurrence/approval
HSR/SSR-12	Review HWCI/CSCI capabilities and characteristics within the overall system	• Action items or • Concurrence/approval
HSR/SSR-13	Identify any HSR/SSR corrective actions required to establish the HWCI/CSCI Requirements baseline	• Action items or • Concurrence/approval

Example 18.3

Successful completion of SUBSYSTEM CDRs serve as *entry criteria* to a SYSTEM Level CDR, which then determines if the system design is *mature* enough to proceed with the Component Procurement and Development Process of the System Development Phase.

18.7.8 Integrated Baseline Review (IBR)

The IBR is typically the first review event conducted for a System Development project. Whereas most reviews are one-time events, the IBR is a process that occurs continuously throughout the contract period of performance. IBRs are commonly found in government contracts; however, the IBR is a sound concept that applies to any type of System Development or Services contract.

TABLE 18.5 Example PDR Objectives/Exit Criteria and Expected Decisions

Item	Example PDR Objectives/Exit Criteria	Expected Decision(s)
PDR-1	Briefly review any updates to the System Level architecture	None
PDR-2	Briefly review any updates to the Product or Subsystem Architectures	None
PDR-3	Review HWCI Design Solutions: • HWCI specification requirements and traceability • HWCI use cases • HWCI theory of operations • HWCI architecture • HWCI requirements allocations • HWCI support of system phases, modes, and states of operation • HWCI performance budgets and margins • HWCI technical performance measures (TPMs) • HWCI analyses and trade studies • HWCI-to-HWCI interoperability • HWCI CTIs • HWCI-to-CSCI(s) integration	• Action items or • Concurrence/approval
PDR-4	Review CSCI Design Solutions: • CSCI specification requirements and traceability • CSCI use cases • CSCI theory of operations • CSCI architecture • CSCI requirements allocations • CSCI support of SYSTEM Level phases, modes, and states of operation • CSCI analyses and trade studies • CSCI performance budgets and margins • CSCI-to-CSCI interoperability • CSCI-to-HWCI integration • CSCI CTIs • CSCI critical technology issues	• Action items or • Concurrence/approval
PDR-5	Review Specialty Engineering considerations such as: • Human Factors Engineering (HFE), • Logistics, • Reliability, • Availability, • Maintainability, • Supportability, • Sustainability, • Security, • Producibility, • Environmental, • Training, • Vulnerability, • Survivability, • Susceptibility, • And so on	• Action items or • Concurrence/approval
PDR-6	Review Hardware/Software/Human System Integration (HSI)issues	• Action items or • Concurrence/approval

TABLE 18.6 Example CDR Objectives/Exit Criteria and Expected Decisions

Item	Example CDR Objectives/Exit Criteria	Decision(s) Expected
CDR-1	Determine if the detailed design satisfies the performance and engineering requirements specified in an *item's development specification*	• Action items or • Concurrence/approval
CDR-2	Assess the detailed design *compatibility* and *interoperability* internal to the item and externally to other System Elements: • Equipment (Hardware and Software) • Facilities • Personnel • Mission Resources • Procedural Data • System Responses (i.e., behavior, products, by-products, or services)	• Action items or • Concurrence/approval
CDR-3	Assess item achievement of allocated technical *performance budgets and safety margins*	• Action items or • Concurrence/approval
CDR-4	Assess *specialty engineering* considerations such as: • Reliability, Maintainability and Availability (RMA) • Producibility • Logistics • Security • Survivability • Vulnerability • Environmental • Susceptibility • Human Factors Engineering	• Action items or • Concurrence/approval
CDR-5	Assess any detailed analyses, trade studies, modeling and simulation, or demonstration results, etc. that support decision-making	• Action items or • Concurrence/approval
CDR-6	Assess the adequacy of verification test plans for each item	• Action items or • Concurrence/approval
CDR-7	Review *preliminary* test cases and procedures	• Action items or • Concurrence/approval
CDR-8	Freeze the Developmental Configuration	• Action items or • Concurrence/approval

In general, the objective of the IBR is to establish a mutual understanding between the System Acquirer and System Developer that contract and technical requirements can be *reasonably accomplished* within the contract period of performance and delivery schedule with the resources—personnel, facilities, etc.—available. The IBR consists of an integrated assessment of all key contract technical, cost, and schedule documents. As such, the IBR seeks to assess and answer the *who*, *what*, *when*, *where*, and *how* for establishing the PMB:

1. *What* work is to be performed as specified in the Contract CSOW, Work Breakdown Structure (WBS), IMP, SPS, CDRL, CLINs, Enterprise command media, and so forth.
2. *Who* is accountable for performing the work—for example, project organization and IPT charters.
3. *When* the work is to be accomplished—for example, MPS and IMS.
4. *Where* the work will be performed—System Developer's facility, subcontractor or vendor facility, etc.
5. *How* the work will be resourced and controlled—for example, control accounts and work packages linked to the *Contract Work Breakdown Statement (CWBS).*

On completion of the IBR, a PMB is established that will serve as the frame of reference for assessing work progress, performance, and risk.[3,4]

[3]Refer to DAU (2015) and NDIA (2010) for additional information.
[4]Refer to the ANSI/PMI 99-001-2013 (2013) Project Management Institute (PMI) A Guide to the Project Management Body of Knowledge (PMBOK®) for additional information.

TABLE 18.7 Example TRR Objectives/Exit Criteria and Expected Decisions

Item	Example TRR Objectives/Exit Criteria	Expected Decision(s)
TRR-1	Assess the readiness of the *test article* to undergo testing (e.g., destructive or non-destructive)	• Action item(s) or • Concurrence/approval
TRR-2	Coordinate and assess the readiness of all test article interfaces and resources	• Action item(s) or • Concurrence/approval
TRR-3	Verify that test plans and procedures are approved and communicated	• Action item(s) or • Concurrence/approval
TRR-4	Identify and resolve all critical technical, test, statutory, and regulatory issues	• Action item(s) or • Concurrence/approval
TRR-5	Verify all safety, health, and environmental concerns are resolved and adequate *emergency* processes, services, and equipment are in place to support all aspects of the test	• Action item(s) or • Concurrence/approval
TRR-6	Verify all lower-level *Discrepancy Report (DR) corrective actions* have been completed and *verified* for HWCIs and CSCIs	• Action item(s) or • Concurrence/approval
TRR-7	Verify that the "As Built" test article identically matches its "As Designed" documentation	• Action item(s) or • Concurrence/approval
TRR-8	Coordination of Responsibilities for Test Conduct, Measurement, and Reporting	• Action item(s) or • Concurrence/approval
TRR-9	Designation of Test Safety Officers (RSOs), Range Safety Officers (RSOs), and Security Personnel, as appropriate	• Action item(s) or • Concurrence/approval
TRR-10	Obtain Authority to Proceed with specific tests	• Action item(s) or • Concurrence/approval

TABLE 18.8 Example SVR Objectives/Exit Criteria and Expected Decisions

Objective	Example SVR Objective/Exit Criteria	Expected Decision(s)
SVR-1	Audit and certify the results of the Functional Configuration Audit (FCA)	• Action item(s) or • Concurrence/approval
SVR-2	Audit and certify the results of the Physical Configuration Audit (PCA)	• Action item(s) or • Concurrence/approval
SVR-3	Identify any outstanding inconsistencies, latent defects such as design errors, deficiencies, flaws, etc.	• Action item(s) or • Concurrence/approval
SVR-4	Verify all approved *Engineering Change Proposals (ECPs)*, *Discrepancy Reports (DRs)*, etc. have been incorporated and verified	• Action item(s) or • Concurrence/approval
SVR-5	Authorization to establish the Product Baseline for the As Specified, As Designed, As Built, and As Verified configurations	• Action item(s) or • Concurrence/approval

TABLE 18.9 Example RTSR Objectives/Exit Criteria and Expected Decisions

Item	Example RTSR Objectives/Exit Criteria	Expected Decision(s)
RTSR-1	Verify that all compliance test data have been collected, documented, and certified	• Action item(s) or • Concurrence/approval
RTSR-2	Verify all cabling and equipment items are inventoried and properly identified	• Action item(s) or • Concurrence/approval
RTSR-3	Verify that anything related to the configuration installation is documented	• Action item(s) or • Concurrence/approval
RTSR-4	Assess verification data completeness to process with system disassembly, reconfiguration, packaging, packing, crating, and shipping	• Action item(s) or • Concurrence/approval
RTSR-5	Assess storage or deployment site facility readiness to accept system delivery for installation and integration: • Environmental conditions • Interfaces • "Gate keeper" decision authority approvals	• Action item(s) or • Concurrence/approval
RTSR-6	Verify coordination of enroute system transportation support including: • Licenses and permits • Route selection • Security • Resources	• Action item(s) or • Concurrence/approval

TABLE 18.10 Example PRR Review Objectives/Exit Criteria and Decisions Expected

Item	Example PRR Objectives/Exit Criteria	Decision(s) Expected
PRR-1	Authenticate the *integrity*, *adequacy*, and *completeness* of the current Product Baseline and place it under configuration control	• Action item(s) or • Concurrence/approval
PRR-2	Verify that design improvements (e.g., approved Engineering Change Proposals (ECPs)) to facilitate production have been incorporated, verified, and validated	• Action item(s) or • Concurrence/approval
PRR-3	Resolve any vendor, production, materials, or process issues	• Action item(s) or • Concurrence/approval
PRR-4	Make a Production "Go-Ahead" Decision and determine the scope of production	• Action item(s) or • Concurrence/approval
PRR-5	Establish the Production Baseline	• Action item(s) or • Concurrence/approval

Accomplishment of the primary IBR objectives is supported by secondary objectives and *exit criteria* that culminate in key decisions. Table 18.1 provides example objectives, exit criteria, and expected decisions.

Author's Note 18.4

The tables of examples that follow were originally derived from former MIL-STD-1521B that was canceled in 1995. The relevance of this document as a valued source remains today as evidenced by a DAU Web site note indicating "that if we were to bring back a Mil-Standard, this would be the first." (DAU, 2013a).

On successful completion of the IBR *exit criteria*, project control accounts and work packages are *activated*, and work is initiated for the SE Design Process (Figures 12.3, 15.2, and 15.3) of the System Development Phase.

18.7.9 System Requirements Review (SRR)

The SRR is typically the first opportunity for technical representatives from the User, System Acquirer, and System

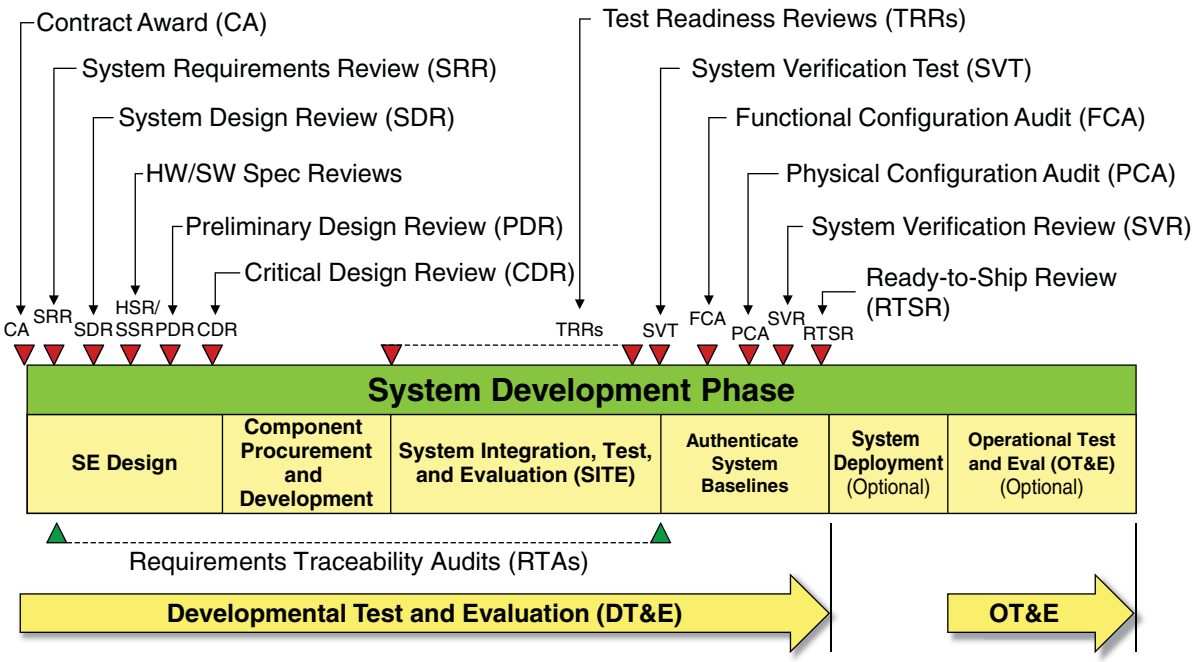


Figure 18.1 Technical Review Sequencing

Developer Enterprises to get together in a common forum to review, interpret, clarify, and correct, if appropriate, the system requirements.

The primary objectives of the SRR are to:

- Review, clarify, correct, and baseline the set of SPS requirements to ensure a common interpretation and understanding among decision-makers.
- Establish the System Requirements Baseline (SRB).

Accomplishment of the primary SRR objectives is supported by secondary objectives and *exit criteria* that culminate in key decisions. Table 18.2 provides example objectives/exit criteria and expected decisions.

18.7.10 System Design Review (SDR)

The SDR follows the SRR and is conducted in accordance with the contract Ts&Cs of the Contract Statement of Work (CSOW), SDR *entry* criteria, and project schedule. Following the SRR, the System Developer matures the system design solution derived from the System Requirements Baseline. At a *minimum*, the System Design Solution includes development and maturation of the System Architecture and interface requirements, ConOps, and preliminary allocations of SPS requirements to the product or subsystem levels of abstraction.

The primary objective of the SDR is to establish the SYSTEM Level design solution and Allocated Baseline as part of the evolving Developmental Configuration. Accomplishment of the primary SDR objectives is supported by secondary objectives and *exit criteria* that culminate in key decisions. Table 18.3 provides example objectives/exit criteria and expected decisions.

On successful completion of the SDR exit criteria, emphasis shifts to formulating and maturing HWCI-unique and CSCI-unique requirements specifications (HRS/SRS).

18.7.11 Hardware/Software Specification Reviews (HSRs/SSRs)

Once the PRODUCT or SUBSYSTEM Level requirements allocations have been established as the Allocated Baseline at the PDR, the architectures for product or subsystem level solutions are developed and matured. Each solution evolves from analyses of allocated requirements.

Trade studies are conducted as an Analysis of Alternatives (AoA) (Chapter 32) to select a preferred PRODUCT or SUBSYSTEM Architecture consisting of items, HWCIs, and CSCIs from a set of viable candidate solutions. PRODUCT or SUBSYSTEM Development Specification requirements are next allocated to items such as HWCIs, and CSCIs, as applicable. Requirements allocated to HWCIs and CSCIs are documented respectively in Preliminary HWCI Requirements Specifications (HRSs) and Preliminary CSCI Software Requirements Specifications (SRSs). The culmination of this activity results is an HSR or SSR, as applicable.

In addition to the standard review items, the primary objective of the HSR/SSR is to establish an HWCI's or

CSCI's requirements specification (HRS/SRS) baseline for its Developmental Configuration to provide a basis for peer level and lower-level decision-making (Principle 14.2).

Accomplishment of the primary HSR/SSR objectives is supported by secondary objectives and *exit criteria* that culminate in key decisions. Table 18.4 provides example objectives/exit criteria and expected decisions.

On successful completion of each HWCI's HSR and each CSCI's SSR exit criteria for their emphasis shifts to formulating and maturing the *Preliminary* System Design Solution. This includes development of HWCI and CSCI Level designs that respond to their respective HRS and SRS.

18.7.12 Preliminary Design Review (PDR)

The PDR represents the fourth major technical review in the design of a SYSTEM, PRODUCT, SUBSYSTEM, ASSEMBLY, and so forth. The review is conducted as a project event to assess the adequacy, maturity, completeness, consistency, and risk of the evolving System Design Solution down to the HWCI and CSCI design levels.

Following the HSR/SSR, the architectural solution of each HWCI and CSCI evolves and matures. Analyses and trade studies are performed to select a preferred HWCI or CSCI architecture that represents the best solution as determined by a set of pre-defined evaluation criteria. As each HWCI and CSCI solution reaches a level of maturity, PDRs are conducted for each HWCI and CSCI. On successful completion of all HWCIs and CSCIs, the System Level PDR is scheduled and conducted.

In addition to the standard review objectives, the primary objective of the PDR is to review and concur/approve the *Preliminary* System Design Solution down to HWCI/CSCI architecture levels. Accomplishment of the primary PDR objectives is supported by secondary objectives and *exit criteria* that culminate in key decisions. Table 18.5 provides example objectives/exit criteria and expected decisions.

Author's Note 18.5

Producibility

The term *producibility* is seldom used; however, the term has an *explicit connotation* most people understand. You can create the most *elegant* design but if it cannot be feasibly produced; it has little value. As a result, we use *producibility* here.

On successful completion of the PDR exit criteria, emphasis shifts to formulating and maturing HWCI-unique and CSCI-unique detailed designs for presentation at the CDR.

18.7.13 Critical Design Review (CDR)

The CDR is the fifth major technical review in the development of a PRODUCT, SUBSYSTEM, ASSEMBLY, and so forth. The review is conducted as a project event to assess the adequacy, maturity, completeness, consistency, and risk of the evolving System Design Solution down to the HWCI, ASSEMBLY, and PART Levels and CSCI Computer Software Component (CSC) and Computer Software Unit (CSU) levels.

In addition to the standard review objectives, the primary objectives of the system level CDR are to:

1. Review and concur/approve the System/CI design solution.
2. Make a decision to *authorize* and *commit* resources to the Component Procurement and Development Process (Figure 12.2) of the System Development Phase.

Accomplishment of the primary CDR objectives is supported by secondary objectives and *exit criteria* that culminate in key decisions. Table 18.6 provides example objectives/exit criteria and expected decisions.

On successful completion of the CDR exit criteria, emphasis shifts to procurement and development of physical components that implement the detailed design requirements. This includes new development or selection and acquisition of Commercial Off-the-Shelf (COTS) and NDIs (Figures 16.7 and 16.8).

Author's Note 18.6

CDR: Long Lead Item Scheduling Conflicts

For some projects, the timing of the CDR may be *incompatible* with long lead item procurement required to meet contract deliveries. Where this is the case, the System Developer may have to assume a risk by procuring long lead items early recognizing the Acquirer has not approved the CDR design.

18.7.14 Test Readiness Reviews (TRRs)

At each level of abstraction—PART, SUBASSEMBLY, ASSEMBLY, or SUBSYSTEM—some systems require TRRs to be conducted. TRRs range from major project events for large complex systems to simple team coordination meetings among development team members.

The primary objectives of a TRR are to:

1. Assess the readiness and risks of the test article, environment, and team to conduct a test or series of tests.
2. Ensure that all test roles are identified, assigned to personnel, and allocated responsibilities.
3. Authorize initiation of test activities.

Accomplishment of the primary TRR objectives is supported by secondary objectives and *exit criteria* that culminate in key decisions. Table 18.7 provides example objectives/exit criteria and expected decisions.

On successful completion of a TRR exit criteria, an *authorization to proceed* with specific tests is granted.

18.7.15 System Verification Review (SVR)

When System Level verification testing is completed, an SVR is conducted. The primary objectives of the SVR are:

1. Authenticate the results of System Verification Test (SVT), including the FCA and PCA results.
2. Establish the Product Baseline.

Accomplishment of the primary SVR objectives is supported by secondary objectives and *exit criteria* that culminate in key decisions. Table 18.8 provides example objectives/exit criteria and expected decisions.

On successful completion of the SVR exit criteria, the workflow progresses to a RTSR decision.

18.7.16 Ready-to-Ship Review (RTSR)

Following the SVR, an RTSR is conducted. The primary objective of the RTSR is to determine the system's *state of readiness* for disassembly and deployment to the designated delivery site.

Accomplishment of the primary RTSR objectives is supported by secondary objectives and *exit criteria* that culminate in key decisions. Table 18.9 provides example objectives/exit criteria and expected decisions.

On successful completion of the RTSR exit criteria, the system may require disassembly or reconfiguration, packaging, packing, crating, and shipping to the designated deployment or job site in accordance with the contract or direction from the ACO (Principle 18.7).

18.7.17 Production Readiness Review (PRR)

For SYSTEMS or PRODUCTS planned for production, a PRR is conducted shortly after production contract award. The primary objectives of the PRR are to:

1. Authenticate the Production Baseline configuration.
2. Make a production "go-ahead" decision to commit to Low-Rate Initial Production (LRIP) or Full-Scale Production (FSP).

Accomplishment of the primary PRR objectives is supported by secondary objectives and *exit criteria* that culminate in key decisions. Table 18.10 provides example objectives/exit criteria and expected decisions.

On successful completion of the PRR exit criteria, emphasis shifts to LRIP. Subsequent PRRs address readiness for FSP.

18.8 CHAPTER SUMMARY

In our discussion of technical reviews, we introduced the various types of reviews and their occurrence as critical control or staging points in a system's development. For each review, we identified the key objectives and referenced checklists for conducting the review. We also highlighted the importance of conducting maturity-based event reviews for each stage of development. Finally, we provided guiding principles to consider when conducting the review.

18.9 CHAPTER EXERCISES

18.9.1 Level 1: Chapter Knowledge Exercises

1. What is a technical review?

2. Why should you conduct technical reviews?

3. Who is responsible for conducting technical reviews?

4. What types of technical reviews should be conducted?

5. How are technical reviews results documented?

6. What is the relationship between technical reviews and System Development Processes (Figure 12.2)?

7. What is an IPR and what is it intended to accomplish?

8. What is a PMB?

9. What is an IBR and what is it intended to accomplish?

10. What is an SRR and what is it intended to accomplish?

11. What is an SDR and what is it intended to accomplish?

12. What is an HSR and SSR and what is it intended to accomplish?

13. What is an SSR and what is it intended to accomplish?

14. What is a PDR and what is it intended to accomplish?

15. What is a CDR and what is it intended to accomplish?

16. What is a TRR and what is it intended to accomplish?

17. What is a PRR and what is it intended to accomplish?

18.9.2 Level 2: Chapter Knowledge Application Exercises

Refer to www.wiley.com/go/systemengineeringanalysis2e

18.10 REFERENCES

ANSI/PMI 99-001-2013 (2013), *A Guide to the Project Management Body of Knowledge (PMBOK®)*, 5th Edition, Newton Square, PA: Project Management Institute (PMI).

DAU (2015) *Integrated Baseline Review (IBR)*, ACQuipedia Website, Ft. Belvoir, VA: Defense Acquisition University (DAU). Retrieved on 7/31/15 from https://dap.dau.mil/acquipedia/Pages/ArticleDetails.aspx?aid=cf5eb839-0881-4044-9f23-2c675726b481.

DoD IMP-IMS Guide (2005), *Integrated Master Plan and Integrated Master Schedule Preparation and Use Guide*, Version 0.9, Washington, DC: U.S. Department of Defense (DoD).

DOE (2007), *Stage-Gate Innovation Management Guidelines*, Industrial Technologies Program, Version 1.3, Washington, DC: U.S. Department of Energy (DOE).

MIL-HDBK-61A (SE) (2001), *Military Handbook: Configuration Management Guidance*, Washington, DC: Department of Defense (DoD).

MIL-STD-973 (1992, Cancelled 2000), *Military Standard: Configuration Management*, Washington, DC: Department of Defense (DoD).

MIL-STD-1521B (1992 - Cancelled 1995), *Military Standard: Technical Reviews and Audits for Systems, Equipments, and Computer Software*, Washington, DC: Department of Defense (DoD).

NASA SP-2007-6105 (2007), *System Engineering Handbook, Rev. 1*, Washington, DC: National Aeronautics and Space Administration (NASA). Retrieved on 6/2/15 from http://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/20080008301.pdf.

NDIA (2010), *Integrated Baseline Review (IBR) Guide, Revision 1, Integrated Management Division*, Arlington, VA: National Defense Industrial Association (NDIA). Retrieved on 7/31/15 from http://www.ndia.org/Divisions/Divisions/IPMD/Documents/ComplementsANSI/NDIA_IPMD_IBR_Guide_Rev_090110-b.pdf.

19

SYSTEM SPECIFICATION CONCEPTS

The formal mechanism for specifying *what* capabilities a system is required to provide, and *how well* the capabilities are to be performed is the System Performance Specification (SPS). The SPS establishes the formal technical requirements of the contract between the System Acquirer, as the User's contract and technical representative, and the System Developer.

Many people *erroneously* believe specifications are documents used on the "front end" of a program to design the system; this is only partially true. Specifications serve as the basis for decision-making throughout the System Development, Component Procurement and Development, and System Integration, Test, and Evaluation (SITE) Phases. They:

- Represent human attempts to translate, bound, and communicate the User's prescribed *solution space* into a language of text and graphics for capability (i.e., functional and performance) requirements to produce a physical system, product, or service that satisfies the intended operational need.
- Serve as a frame of reference for decision-making by establishing the thresholds for evaluating and verifying technical compliance as a precursor for final SYSTEM or PRODUCT acceptance and delivery.

Specification development requires support from a multi-level System Analysis process. The analysis decomposes bounded *solution space* capabilities into manageable, lower level specifications for the SUBSYSTEMS, PRODUCTS, and ASSEMBLIES that ultimately form the totality of the system.

Chapter 19 introduces System Specification Practices that establish the multi-level, integrated framework of specifications required to develop a system, product, or service. Our discussions introduce the various types of specifications, their contents, and relationships to other specifications, standards, and regulations. The discussion includes a general specification outline that can be used as reference model.

19.1 DEFINITIONS OF KEY TERMS

- **Deviation**—"A specific written authorization to depart from a particular requirement(s) of an item's current approved configuration documentation for a specific number of units or a specified period of time, and to accept an item which is found to depart from specified requirements, but nevertheless is considered suitable for use 'as is' or after repair by an approved method. (A deviation differs from an engineering change in that an approved engineering change requires corresponding revision of the item's current approved configuration documentation, whereas a deviation does not)" (MIL-HDBK-61A, p. 3–6).
- **Requirement**—"Any condition, characteristic, or capability that must be achieved and is essential to the end item's ability to perform its mission in the environment in which it must operate is a requirement. Requirements must be verifiable" (SD-15, 1995, p. 9).
- **Requirement**—An essential characteristic, condition or capability that shall be met or exceeded by a system or a component to satisfy a contract, standard,

System Engineering Analysis, Design, and Development: Concepts, Principles, and Practices, Second Edition. Charles S. Wasson.

Companion Website: www.wiley.com/go/systemengineeringanalysis2e

specification, or other formally imposed document (FAA SEM, 2006, Vol. 3, p. B-11).

- **Essential Requirement**—A statement that specifies and bounds a key system, product, or service capability required for mission success without unnecessarily constraining the solution set.
- **Performance Requirement**—The *quantitative* magnitude of an action-based outcome to be accomplished.
- **Source or Originating Requirements**—The set of requirements that serve as the publicly released requirements used as the basis to acquire a system, product, or service. In general, a formal Request for Proposal (RFP), solicitation's Statement of Objectives (SOO) or System Requirements Document (SRD) are viewed as User *source* or *originating requirements* (Chapter 12).
- **Specification Requirements**—A collection of essential capability requirements statements that specify and bound the performance-based capabilities of a system, product, or service entity such as a SYSTEM, PRODUCT, SUBSYSTEM, and ASSEMBLY.

- **Requirements Creep**—"The tendency of the user (or developer) to add to the original mission responsibilities and/or performance requirements for a system while it is still in development" (DAU, 2012, p. B-192).
- **Requirements Elicitation**—The process of identifying and collecting stakeholder requirements through understanding of the Problem and Solution Spaces (Chapter 4) such as via personal interviews and observation.
- **Requirement Owner**—An individual or team assigned with accountability for implementing a specific requirement in a specification or drawing.
- **Requirement Stakeholder**—Anyone who has a stake or vested interest in identifying, defining, specifying, prioritizing, verifying, and validating system capability and performance requirements. Requirements stakeholders include all personnel responsible for system definition, procurement, development, production, operations, maintenance, sustainment, retirement, and disposal of a system, product, or service.
- **Specification**—A document that describes the *essential* requirements for items, materials, processes, or services of a prescribed *solution space,* data required to implement the requirements, and methods of verification to satisfy specific criteria for formal acceptance.
- **Specification Owner**—An individual or team assigned with *accountability* for developing and controlling all requirements within a specification or drawing. Specification ownership resides with the individual or team that *owns* the SYSTEM / ENTITY Architecture in which the Entity being specified by the specification is represented including its interfaces.
- **Specification Tree**—"The hierarchical depiction of all the specifications needed to control the development, manufacture, and integration of items in the transition from customer needs to the complete set of system solutions that satisfy those needs" (MIL-STD-499B Draft (canceled), Appendix A, Glossary, p. 39).
- **Tailoring**—"The process by which individual requirements (sections, paragraphs, or sentences) of the selected specifications, standards, and related documents are evaluated to determine the extent to which they are most suitable for a specific system and equipment acquisition, and the modification of these requirements to ensure that each achieves an optimal balance between operational needs and cost" (MIL-STD-961E, p. 7).
- **Waiver**—"A written authorization to accept a Configuration Item (CI) or other designated item, which, during production or after having been submitted for inspection, is found to depart from specified requirements, but nevertheless is considered suitable 'as is' or after rework by an approved method" (DAU, 2012, p. B-239).

19.2 WHAT IS A SPECIFICATION?

Development of any type of requirements requires that you establish a firm understanding of:

1. What is a specification?
2. What is the purpose of a specification?
3. How does a specification accomplish a specific objective?

If you analyze the definition of a *specification* provided in this chapter's *Definition of Key Terms*, there are three key parts of this definition. Let's briefly examine each part.

- First, "… *essential* requirements for items, materials, processes, or services." Specifications are also written for services and multi-level components, materials that compose those components, and procedural processes required to convert those materials into a usable component.
- Second, "*data* required to implement the requirements …." System development is often constrained by the need to adhere to and comply with other contract or task documents such as: SOO, design criteria, specifications, standards, and regulations.

- Third, "… *methods of verification* to satisfy specific criteria for formal acceptance." Specifications establish the formal technical agreement between the System Acquirer and System Developer concerning how each requirement will be formally verified to demonstrate the physical SYSTEM / ENTITY has fully achieved compliance to the specified capability and associated level of performance. Note that verifying the achievement of a requirement may only satisfy incremental criteria required by contract for formal Acquirer acceptance. The contract may require other criteria such as installation and checkout in the field; field trails and demonstrations; Operational Test and Evaluation (OT&E); and resolution and corrective action of outstanding *latent defects* - errors, design flaws, and deficiencies. Remember, an SPS is a subordinate elements of the contract and represent only a portion of criteria for deliverable system acceptance and subsequent contract completion.

Given these comments concerning the definition of a specification, let's establish the objective of a specification.

19.2.1 Specification Objectives

The objective of a specification is to document and communicate:

1. *What* essential operational capabilities are required of an item (SYSTEM, PRODUCT, AND SUBSYSTEM).
2. *How well* the capabilities must be performed.
3. *When* the capabilities are to be performed.
4. *What* external interfaces must be provided and with whom.
5. Under *what* prescribed OPERATING ENVIRONMENT.
6. What design and construction constraints are levied.
7. How capability compliance will be verified.

19.2.2 Why Do We Need Specifications?

Objective Evidence Principle

Principle 19.1 If an analysis, review, decision, result, or event is not corraborated by a Quality Record (QR) as *objective evidence* of accomplishment, it is nothing more than hearsay.

A common question is: *why do we need specifications*? The best response is to begin with an old adage: If you do not tell people what you want, you can't complain about what they deliver. "But that's not what we asked for" is a frequent response, which may bring a retort "we delivered what you put on a piece of paper … [nothing!]."

In any case, if the technical wrangling continues, legal remedies may be pursued. The legal community seeks to unravel and enlighten via *who, what when, where*, and *how* discovery process:

1. *What* did you specify via the contract, meeting minutes, official correspondence, and conversations?
2. *Who* did you talk to?
3. *When* did you discuss it?
4. *Where* did you discuss the matter?
5. *How* did you document what both parties agreed to?

The best way to *avoid* these conflicts is to establish a mechanism "up front" prior to Contract Award (CA). The mechanism should capture the technical agreements between the Acquirer and the System Developer to the mutual satisfaction and understanding of both parties. From a technical perspective, that mechanism is the SPS.

The conflicts discussed previously, which characterize the Acquirer and System Developer/Services Provider or Subcontractor interfaces, are not limited to Enterprises. In fact, the same issues occur vertically within the System Developer's program organization between the SYSTEM, PRODUCT, SUBSYSTEM, ASSEMBLY, SUBASSEMBLY, and PART levels of abstraction. As a result, the allocation and flow down of requirements from the SPS to lower levels requires similar technical agreements between system development teams. This occurs as each allocated and assigned *problem space* is partitioned (Figure 4.7) into lower level Solution Spaces bounded by specification performance-based capability requirements.

In summary, *why do we need specifications*? To explicitly articulate and communicate in a language that:

- Employs terms that are simple and easy for the Acquirer, User, and System Developer Stakeholders to understand.
- Expresses *essential* features and characteristics of the deliverable system, product, or service.
- Avoids the need for "open" interpretation or further clarification that may lead to potential conflict at final system, product, or service acceptance.

19.2.3 What Makes a Specification "Good?"

People commonly ask: What constitutes a "good" specification? "Good" is a colloquial term. What is *good* to one User may be judged as "poor" by another User. Others refer to "well-written" specifications. But what does this mean? Proper grammar? Proper grammar does not mean that the document has substantive content. A better term may be "well defined." This leads to the question: *What makes a well-defined specification "good" or "well written"?*

SEs often respond to this question with comments such as "It's easy to work with," … "We didn't have to make too many changes to get it *right*," "we didn't have any problems with the Acquirer at acceptance." However, there are attributes of well-defined specifications that distinguish them from others as models. So let's propose some general attributes of well-defined specifications.

19.3 ATTRIBUTES OF A WELL-DEFINED SPECIFICATION

A specification, as a potentially legally binding document, is a formal document that requires development in accordance with professional standards based on lessons learned, best practices, and so forth. As such, there are specific technical attributes that form the basis for its development. The attributes list in the following text represents the *minimum* level for a specification. You are advised to supplement this list with additional attributes that represent technical performance standards required by your contract, project, Enterprise, business domain, and profession.

19.3.1 Attribute 1: Specification Ownership and Accountability

Principle 19.2

Specification Developer Principle

Every specification requires assignment of a developer accountable for its: development, approval by the next higher-level architecture owner, implementation, verification, compliance, and configuration controlled baseline maintenance.

Well-defined specifications are assigned to and owned by an System or Product Development Team (SDT/PDT) that is accountable for the implementation of the specification requirements, maintenance, verification, and final acceptance of the deliverable system, product, or service.

In general, there is a belief that a specification is "owned" by the PDT developing an Entity. The reality is: a specification should be "owned" by the team that "owns" and is accountable for the architecture in which the ENTITY being specified appears as a component. For example, a SUBSYSTEM EDS is "owned" by the SDT that is accountable for the SYSTEM Level architecture in which the SUBSYSTEM appears. *Why*? If PDT #1 that implements the Subsystem #1 EDS were to decide unilaterally to change the requirements that impact an external interface with Subsystem #2 without informing PDT #2, major problems can occur. Best practices dictate that the SDT owns SUBSYSTEM #1's EDS and SUBSYSTEM #2's EDS and their interfaces to avoid these situations. Refer to the discussion of Figure 27.1 for additional information.

19.3.2 Attribute 2: Standard Outline

Principle 19.3

Specification Outline Principle

Every specification should be based on an Enterprise standard topical outline that provides continuity and consistency with other specifications.

Well-defined specifications are based on standard outline topics that are recognized by industry as best practices and derived from lessons learned. The outline should:

- Be based on an Organizational Standard Process (OSP).
- Cover the spectrum of Stakeholder—User and End User—Engineering topics to ensure all aspects of technical performance are addressed.

A Word of Caution 19.1

Contract Requirements Guidance

Always consult your contract for guidance about performance specification formats and your Contract Data Requirements List (CDRL) for ENTITY Development Specification (EDS) formats. If guidance is not provided, confer with the Project's Project Engineer (PE). (Principal 18.7).

19.3.3 Attribute 3: Feasibility and Affordability

Principle 19.4

Specification Feasibility and Affordability Principle

Every specification should specify and bound a *solution space* that is feasible to implement, affordable, and has acceptable risk to the User.

A well-defined specification must be feasible and affordable to implement within realistically achievable technologies, skills, processes, tools, and resources with acceptable technical, cost, schedule, and support risk to the System Acquirer and the System Developer or Services Provider.

19.3.4 Attribute 4: Specification Uniqueness

Principle 19.5

Specification Uniqueness Principle

Each specification documents capability requirements that are *unique* to *one and only one* SYSTEM / ENTITY with no other instances within the SYSTEM.

Each SYSTEM / ENTITY should be specified and bounded by a set of specification requirements that are *unique* and exist without duplication or confliction.

Author's Note 19.1

A word about the context of *uniqueness*. In the case of Safety, Human Factors, and Security requirements that have a direct flow down to lower level entities (Chapter 21), only the subject of the requirement statement may be the *only* difference in the wording of the requirement. For example, (Entity) **shall** be designed in accordance with Para. X.X.X (Title) of MIL-STD-XXX. Even though the general statement has identical wording in other specifications, the ENTITY *subject* makes the statement unique, which establishes the *contextual uniqueness* of the requirement.

19.3.5 Attribute 5: Solution Independence

Principle 19.6

Solution Independence Principle

A specification specifies *what* is to be accomplished and *how well*, not how to design the system, product, or service.

Well-defined specifications specify: (1) *what* capabilities are required to accomplish missions and (2) *how well* they are to be performed. Specifying how to design the system imposes constraints and reduces the System Developer's flexibility in achieving at an optimal System Design Solution.

19.3.6 Attribute 6: Essential Requirements

Principle 19.7

Essential Requirements Principle

A specification contains only *essential* requirements that are *necessary* and *sufficient* to develop a system, product, or service without constraining the set of viable solution options.

Well-defined specifications state only the *essential* requirements that are *necessary* and *sufficient* to design or procure a system, product, or service without the need for additional clarification requirements or language.

19.3.7 Attribute 7: Specification and Requirements Accountability

Principle 19.8

Specification and Requirement Accountability Principle

Assign: (1) specification level and (2) individual requirement level responsibility to an individual who is accountable for its analysis, implementation, verification, traceability, and proposed updates.

One of the first steps in achieving System Development success is to ensure that every specification and each of its requirements will be implemented in the System Design Solution. This begins with assignment of responsibility and accountability. Enterprises assign accountability for the specification level; however, individual requirement accountability is often overlooked. Even when accountability is assigned, it is often unclear to the assignee what the scope of their accountability is - analysis, implementation, verification, traceability, and proposed updates.

19.3.8 Attribute 8: Explicit, Unambiguous, Outcome-Based Requirements

Principle 19.9

Single Interpretation Principle

Every specification requirement must be stated in a brief, clear, and concise manner that results in one and only one interpretation.

Well-defined specifications explicitly specify capability requirements using brief, concise, outcome-based, capability requirements statements.

One of the challenges of our educational system is the notion that specifications are written like novels. Requirements statements are "open to interpretation." During System Integration, Test, & Evaluation (SITE), verification becomes conflict in which the System Acquirer has one interpretation of compliance to specification requirements while the System Developer has another interpretation.

Well-defined specifications require every requirement statement to be articulated in a brief, clear, and concise manner that results in one and only one interpretation.

19.3.9 Attribute 9: Requirements Coverage

Principle 19.10

Specification Requirements Coverage Principle

Every specification must specify the set of essential capabilities required to perform User-defined missions with no deficiencies.

Well-defined specifications are:

- Complete, contain no voids, and require no further clarification.
- Do not contain any *undefined* performance values such as To Be Determined (TBD) or To Be Supplied (TBS) performance values.
- Specify one or more verification methods for each requirement statement that will serve as the basis for proving compliance.

19.3.10 Attribute 10: Requirements Consistency

Specification Requirements Consistency Principle

Principle 19.11 Specification requirements must be consistent in language, meanings, semantics, terminology, and standards of units with all project System/Entity specifications.

Well-defined specifications read as if they were written by a single developer. Therefore, to ensure *readability* and *avoid ambiguities*, all specifications within a System should be *consistent* in language, meanings, semantics, terminology, and standards of units.

19.3.11 Attribute 11: Semantics and Terminology

Semantics and Terminology Principle

Principle 19.12 Specifications should employ semantics and terminology that are familiar to its readers.

Well-defined specifications are written in a language that uses terms that are *easy to understand*. In general, many qualified people should independently read any requirement statement and emerge with the same interpretation and understanding of technical performance requirements. For example, the SPS should be written using semantics and terminology that has meaning to the System Acquirer/User and System Developer. Lower level EDS specifications should be written with semantics and terminology that is familiar to the System Developers.

19.3.12 Attribute 12: Requirements Traceability

Requirements Traceability Principle

Principle 19.13 Every specification and requirement must be traceable to the User's *source* or *originating* requirements.

Specifications should not be *random, ad hoc* "wish lists" of requirements. The validity and integrity of a specification requires that every requirement exists to accomplish a specific outcome that ultimately contributes to achievement of mission objectives. Since every requirement has a cost and risk for implementation (Principle 19.4), any specification requirement that is not traceable to the User's *source* or *originating* requirements should be eliminated as non-value-added waste.

19.3.13 Attribute 13: Requirement Performance Measure Traceability

Requirement MOP Traceability Principle

Principle 19.14 Where required, specification requirement performance measures must be traceable to a documented analysis that is baselined and controlled.

Well-defined specifications are traceable to a structured analysis—approach, methodology, and behavioral model such as Model-Based Systems Engineering (MBSE) (Chapters 10 and 33). One of the challenges in developing specifications is that requirements statement performance measures are often derived informally and discarded. This places the entire project at risk when months later the Project Manager (PM) and Project Engineer (PE) discover that no one can *authenticate* the *validity* of a specification performance measure … since it's source analysis was discarded!

19.3.14 Attribute 14: Stakeholder Community Requirements Priorities

Requirement Value-Based Priority Principle

Principle 19.15 Every specification requirement has a *value*-based *priority* to the Stakeholder community.

When requirements are elicited, each stakeholder places a value on the importance of the requirement to enable them to achieve their Enterprise missions. We refer to the value as a priority. The realities of system development are that *every* requirement has a cost to implement and deliver. Given limited resources and stakeholder values, bounding the *solution space* requires reconciling the cost of the *desired* requirements with the *available* resources. As a result, requirements should be prioritized for implementation, especially for Agile Development (Chapter 15) and Commercial Product Development (Figure 5.1). In contrast, Contract System Development (Figure 5.1) typically views all specification requirements as having the same priority.

19.3.15 Attribute 15: Acceptable Stakeholder Risk

Specification Risk Principle

Principle 19.16 Every specification has a level of cost, technical, technology, and schedule risk to develop and maintain that must be *acceptable* to the User.

Well-defined specifications represent a level of cost, technical, technology, or schedule risk that is acceptable to the stakeholders—System Acquirer or System Developer—and does not impose additional financial or schedule risks.

19.3.16 Attribute 16: Maturity and Stability

Specification Baseline Principle

Principle 19.17 Every specification should be reviewed, approved, and placed under Configuration Control as soon as it reaches a *level of maturity* and *stability* to enable technical decisions at lower levels to be made with acceptable risk.

Specifications are baselined at strategic *staging* or *control points* when all stakeholders are in mutual agreement with its contents.

Once the baseline is established, specifications are updated and verified through a Stakeholder review agreement process - Configuration Control (Chapter 16) - that ensures that the document properly reflects the current consensus of stakeholder requirements.

19.4 TYPES OF SPECIFICATIONS

When SEs specify the items, materials, and processes required to support system, product, or service development, *how is this accomplished*? These work products are specified via a hierarchical set of specifications that focus on bounding and specifying requirements for:

- Individual entities at various levels of abstraction—for example, Specification Tree.
- Materials and processes to support development of those items.

The hierarchical set of specifications is documented via a framework referred to as the Specification Tree (Figure 19.2). To understand the hierarchical structure within the specification tree, we need to first establish the types of specifications that may appear in the framework.

The classes of specifications include the following:

- **Detail Specification**—"A specification that specifies design requirements, such as materials to be used, how a requirement is to be achieved, or how an item is to be fabricated or constructed. A specification that contains both performance and detail requirements is still considered a detail specification. Both defense specifications and program-unique specifications may be designated as a detail specification" (MIL-STD-961E, p. 4).
- **Development Specification**—A document that specifies and bounds the capabilities of an *entity* or *item* below the System Level such as a Product, Subsystem, Assembly, or Subassembly.
- **Material Specification**—"A type of program-unique specification that describes such raw or processed materials as metals, plastics, chemicals, synthetics, fabrics, and any other material that has not been fabricated into a finished part or item (MIL-STD-961E, p. 5)."
- **Performance Specification**—"A specification that states requirements in terms of the required results with criteria for verifying compliance, but without stating the methods for achieving the required results .... A performance specification defines the functional requirements for the item, the environment in which it must operate, and interface and interchangeability characteristics (MIL-STD-961E, p. 6)."
- **Process Specification**—"A type of program-unique specification that describes the procedures for fabricating or treating materials and items (MIL-STD-961E, p. 6)."
- **Procurement Specification**—A statement that specifies and bounds a set of capability requirements for an entity such as a System, Subsystem, Assembly being procured as part of a contract, subcontract, or purchase order.
- **Product Specification**—A document that specifies and bounds performance characteristics for an entity such as a Subsystem, Assembly, or Subassembly after development and verification. For example, a commercial product specification for a computer system highlights key features and performance.

Descriptive definitions of each of these types of specifications are provided in the *Definitions of Key Terms* to this chapter.

19.4.1 Specification Types

To place all of these various types of specifications into perspective, let's use the example illustrated in Figure 19.1. Requirements for a system, as documented in the SPS, are *allocated* and *flowed down* one or more levels to one or more items such as Products, Subsystems, or Assemblies. Requirements for these items are captured in their respective development or procurement specifications that document the "As-Specified" Developmental Configuration (Chapter 16) for the Subsystem, Assembly, and Subassembly.

As the *highly iterative*, multi-level, and *recursive* SE process and design effort evolves, SEs develop one or more *design* or *fabrication specifications* to capture the attributes

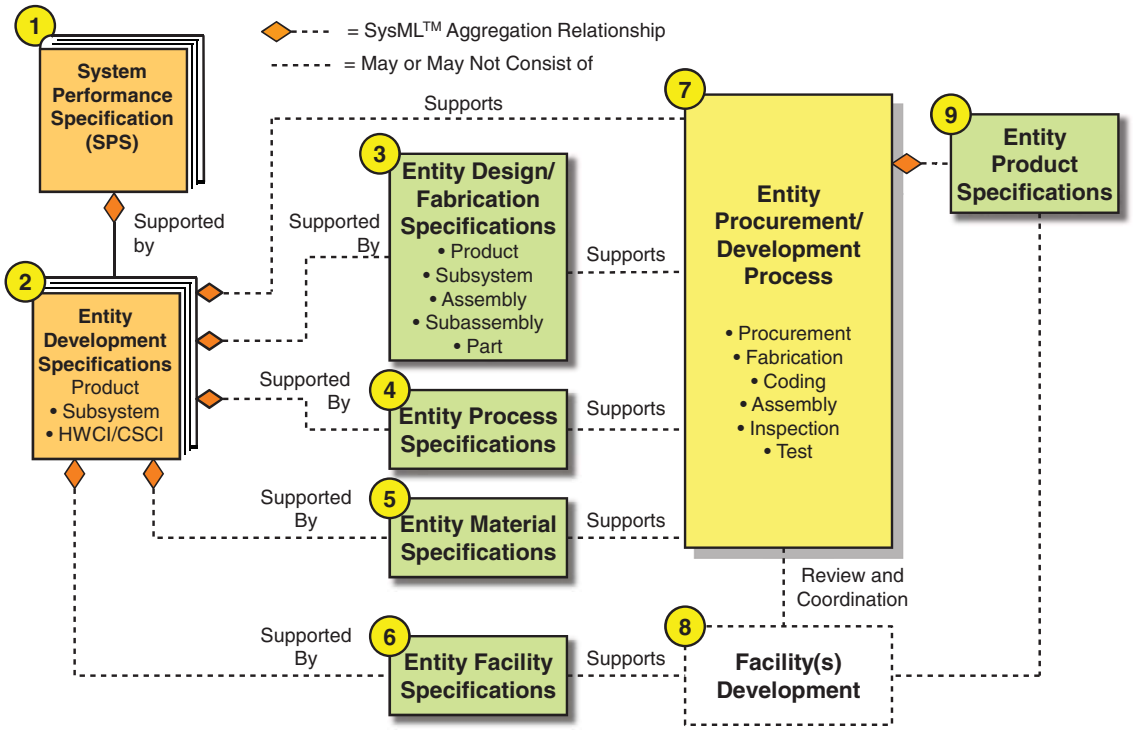


Figure 19.1 Application of Various Types of Specifications to SYSTEM/ENTITY Development

and characteristics of the physical parts to be developed. In the design effort, one or more *process specifications* and *material specifications* are developed to aid in the procurement, fabrication, coding, assembly, inspection, and test of the item. The collective set of baseline specifications represent the "As-Designed" Developmental Configuration (Chapter 16) that documents how to develop or procure the item.

The set of specifications generated for each item or CI serve as the basis for its development, as well as facilities, either internally or via external subcontractors or vendors. When the CI completes the SITE that includes system verification, the "As-Verified" Developmental Configuration is documented as the Product Baseline (Chapter 16) and captured as the CI's product specification.

19.4.2 The Specification Tree

Specification Tree Principle

Principle 19.18 Every system, product, or service should include a Specification Tree based on the System Architecture that serves as a hierarchy of linkable requirements.

The multi-level allocation and flow down of requirements employ a hierarchical framework that logically links system entities vertically into a structure referred to as the specification tree. The right side of Figure 19.2 provides an example of a Specification Tree.

19.4.2.1 Specification Tree Ownership and Control

The Specification Tree is typically *owned* and *controlled* by a project's Technical Director, Project Engineer, System Development Team (SDT), or a System Engineering and Integration Team (SEIT). The SEIT, as the highest-level technical team, also functions as a Configuration Control Board (CCB) to manage changes to the current baseline of a specification.

19.4.2.2 Linking the Specification Tree to the CWBS and System Architecture

People often mistakenly develop the Specification Tree as an independent activity unrelated to the system architecture and *Contract Work Breakdown structure* (CWBS). *This is a serious error!* The Specification Tree and the CWBS should reflect the primary structure of the System Architecture and be linked accordingly. To illustrate this point, consider the graphic shown in Figure 19.2.

Heading 19.1 Now that we have established the Specification Tree as the framework for linking System/Entity specifications, let's shift our focus to understanding the content of specifications.

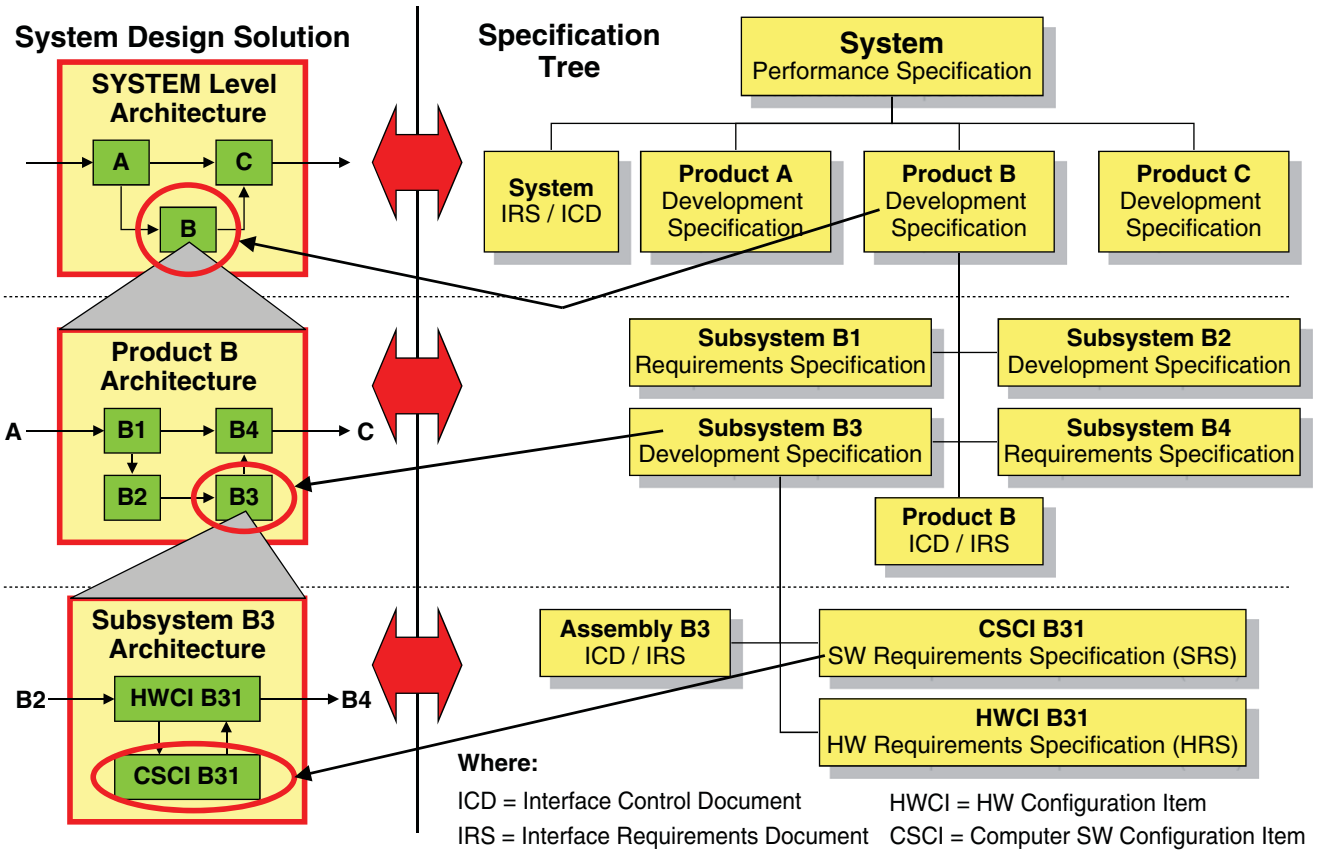


Figure 19.2 Correlating a System's Architecture to its Specification Tree

19.4.3 Specification Development Evolution and Sequencing

To better understand the development and sequencing of specifications, we use the System Development Process workflow as illustrated in Figure 12.2. If we generalize the System Development Process, Figure 19.3 illustrates how the multi-level specification development occurs over time from Contract Award (CA) through System Verification Test (SVT) (Chapter 18).

19.5 KEY ELEMENTS OF A SPECIFICATION

The specification development learning process for most Engineers begins with a specification outline. However, most Engineers lack exposure and understanding as to how the specification outline was derived. To eliminate the gap in this learning process, let's back away from the details of specification outlines and address what specifications should specify. Let's begin by graphically depicting a SYSTEM / ENTITY and the key factors that drive its *implementation*. Figure 19.4 serves as a reference for our discussion.

Author's Note 19.2

Remember, the term SYSTEM/ENTITY is used here in a generic sense. By definition, SEGMENTS, PRODUCTS, SUBSYSTEMS, ASSEMBLIES, Hardware Configuration Items (HWCIs) and Computer Software Configuration Items (CSCIs) are systems. Thus, the discussion here applies to any level of abstraction.

The SYSTEM / ENTITY, for which the specification is to be written, is shown in the central part of the figure. If we analyze systems, we will find that there are several key factors that characterize a system entity or item. The intent here is to characterize the entity's behavioral and physical characteristics and properties. The challenge question is, however, *how do we arrive at this set of attributes*? The answer resides in a variety of external factors that influence and constrain the entity. Let's investigate these factors.

19.5.1 Factor #1: Context

A specification should begin with an introductory statement of its context within the Level 1 System (Figure 8.4).

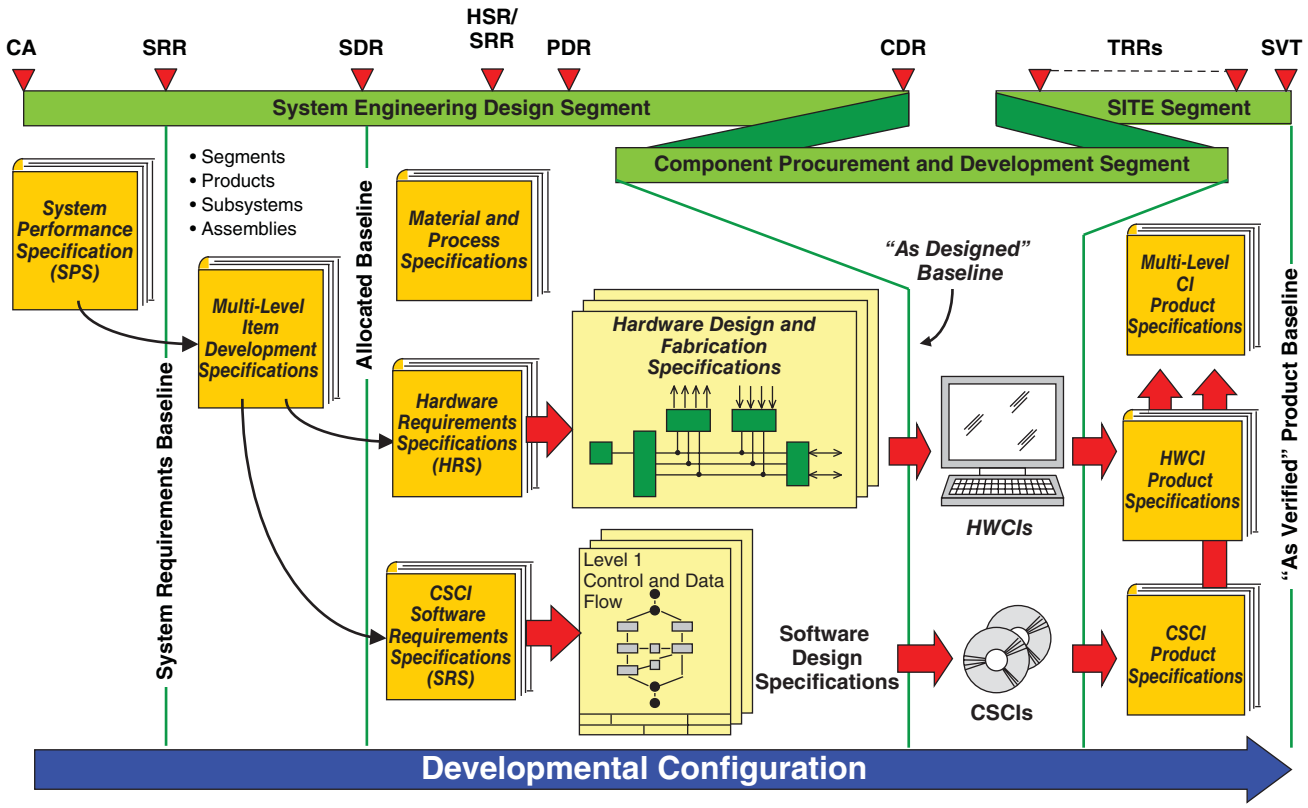


Figure 19.3 Multi-Level Specification Development Sequencing

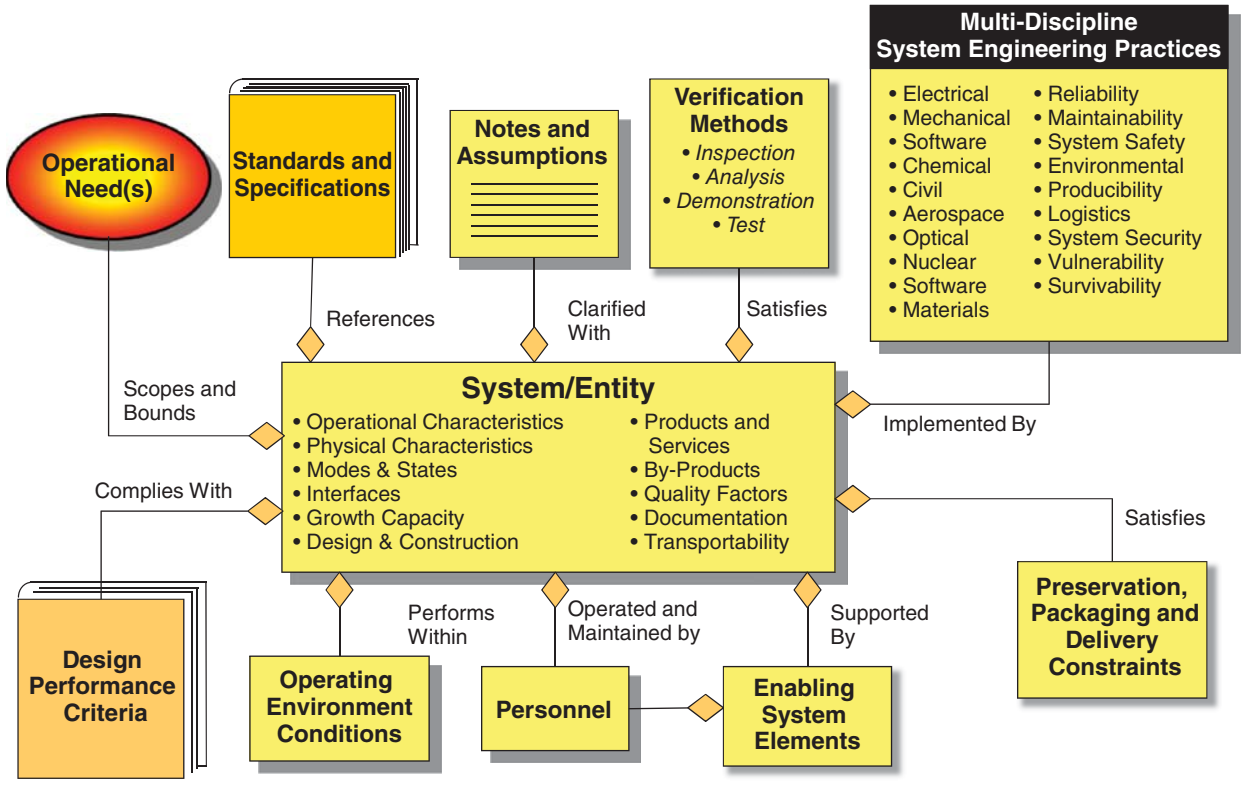


Figure 19.4 Key Elements of a Specification

19.5.2 Factor #2: Specifications, Standards, and Statutory Constraints

The design of any SYSTEM / ENTITY often requires strict compliance with existing specifications, standards, statutory, and regulatory constraints that may include interfacing systems, workmanship, and materials.

19.5.3 Factor #3: Notes and Assumptions

Since requirements are specified with text and graphics, they often call for contextual clarifications. Where some requirements are *unknown*, assumptions may be required though *undesirable*. Additionally, graphical conventions may have to be clarified. Therefore, specifications may require a section of Notes and Assumptions to provide a definition, context and usage information, terms, and conventions.

19.5.4 Factor #4: Verification Methods

Requirements that specify key attributes of a system entity are often written in language and terms the system designers understand. The challenge is, however, *how do you verify and validate that the physical entity complies with the requirements?* To solve this dilemma, specifications include a section that specifies requirements for SYSTEM / ENTITY Verification and Validation (V&V) (User option).

19.5.5 Factor #5: System Engineering Practices

Successful system development requires that best practices derived from lessons learned and Engineering discipline be consistently applied to minimize risk. Therefore, specifications invoke SE practices to ensure that the deliverable product will achieve SPS requirements.

19.5.6 Factor #6: Design and Construction Constraints

Design and Construction Constraints Principle

Principle 19.19 Every specification should specify design and construction requirements that constrain the development of System/Entity's capabilities.

Specifications do more than communicate *what* and *how well* a SYSTEM / ENTITY must accomplish. They communicate the constraints that are levied on the SYSTEM and System Development decisions related to system operations and capabilities. We refer to these as Design and Construction Constraints. In general, design and construction constraints consist of *non-functional* requirements such as size, weight, color, mass properties, maintenance, safety restrictions, human factors, and workmanship. Specifications also specify verification methods (Chapter 13) to be used to verify compliance to a requirement.

19.5.7 Factor #7: Preservation, Packaging, and Delivery

When the SYSTEM / ENTITY is to be delivered, care must be taken to ensure that it arrives fully capable and available to support operational missions. Preservation, packaging, and delivery requirements specify how the deliverable system/entity is to be prepared, shipped, and delivered.

19.5.8 Factor #8: Enabling System Element Requirements

MISSION SYSTEMS require sustainable Pre-Mission, Mission, and Post-Mission Support Test Equipment (STE) at *critical staging events* and *areas*. This may require the use of existing ENABLING SYSTEM EQUIPMENT such as Common Support Equipment (CSE) or Peculiar Support Equipment (PSE) and FACILITIES (Chapter 8) or the need to develop those items. Therefore, specification outlines include ENABLING SYSTEM requirements.

19.5.9 Factor #9: Personnel Element Requirements

Systems typically require the PERSONNEL Element for "hands-on" Command and Control (C2) of the SYSTEM during Pre-Mission, Mission, and Post-Mission operations. Additionally, human–machine trade-offs must be made to optimize system performance. This requires delineating and specifying *what humans do best* versus *what the* EQUIPMENT *Element does best* (Figure 24.14). Therefore, specifications identify the skill and training requirements to be levied on the PERSONNEL Element to ensure Human System Integration (HSI) success (Chapter 24).

19.5.10 Factor #10: OPERATING ENVIRONMENT Conditions

Environmental Conditions Principle

Principle 19.20 Every specification should specify the Environmental Conditions a System/Entity must operate and survive.

Every PRODUCT, SUBSYSTEM, and ASSEMBLY must be capable of performing missions in a prescribed OPERATING ENVIRONMENT at a level of performance that will enable achievement of mission success. Therefore, specifications must define and constrain the OPERATING ENVIRONMENT conditions that drive and bound entity capabilities and levels of performance.

19.5.11 Factor #11: Design Performance Criteria

System entities are often required to operate within performance envelopes, especially when simulating the performance of or interfacing with the physical systems. When this occurs, the specification must invoke external performance requirements that are characterized as design criteria. In these cases, a Design Criteria List (DCL) (Chapter 17) for the interfacing systems or system to be simulated is produced to serve as a reference.

19.5.12 Summary

In our discussion of the specification practices, we identified key challenges, issues, and methods related to developing system specifications. As the final part of the concept, we introduced the structure for a general specification. Given a fundamental understanding of specifications, we are now ready to explore types of requirements documented in a specification via the next topic, Understanding Specification Requirements.

19.6 SPECIFICATION REQUIREMENTS

Specifications establish the agreement of the technical capabilities and levels of performance required for a system, product, or service to achieve its mission and objectives within a prescribed *solution space*. As such, specifications represent human efforts to specify and bound the prescribed *solution space* that will enable the User to accomplish their Enterprise mission and objectives.

This section provides the foundation for specifying system, product, or service requirements. We explore the various categories and types of stakeholder and specification requirements—such as operational, capability, non-functional, interface, verification, and validation requirements. We expand the discussion of operational requirements and link them to the four types of such as Normal, Abnormal, Emergency, and Catastrophic as shown in Figure 19.5.

Specification requirements that bound a *solution space* are hierarchical and interrelated. We discuss the hierarchical structure and relationships among the various types of requirements. We illustrate why specifications prepared by the *ad hoc, endless loop*, Specify-Design-Build-Test-Fix (SDBTF)–Design Process Model (DPM) Paradigm Enterprise are prone to problems of *missing, misplaced, conflicting,* and *duplicated* requirements (Figure 20.3). These problems represent risk areas that SEs need to understand and recognize.

19.6.1 What is a Requirement?

The heart of a specification resides in its requirements. Each requirement statement serves to *specify* and bound the deliverable system, product, or service capability and level

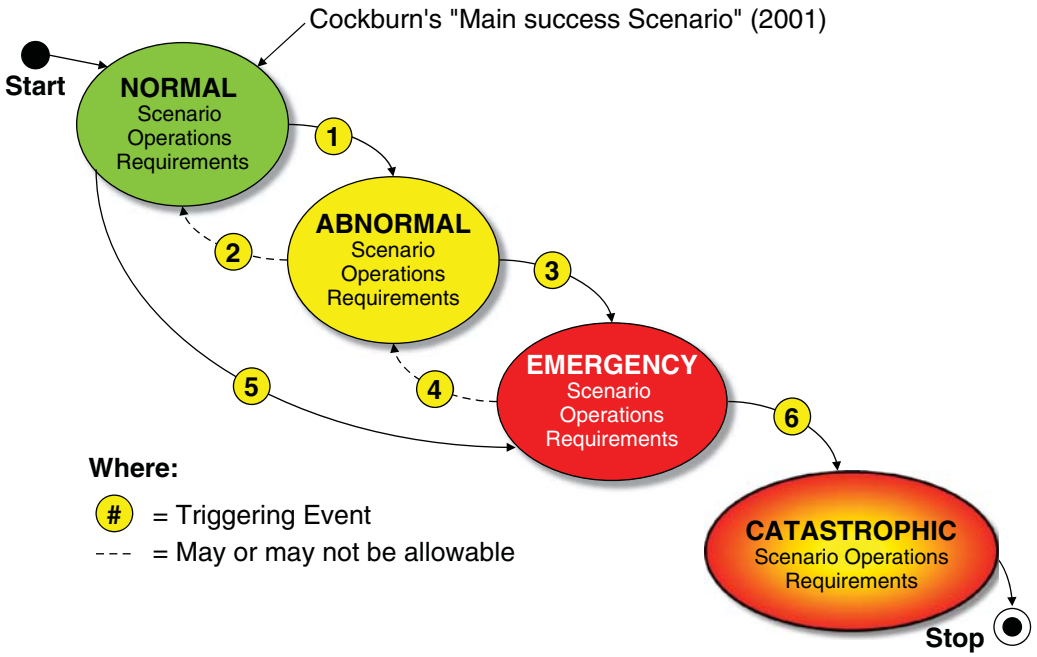


Figure 19.5 Specification Coverage—Four Types of Operations: Normal, Abnormal, Emergency, and Catastrophic Operations

of performance to be developed or modified. The prerequisite to developing requirements is to first identity what capability is required. Therefore, you need to understand how requirements are categorized.

19.6.2 Types of Specification Requirements

People often think of requirements in a generic sense and fail to recognize that requirements also have specific missions and objectives. If you analyze the requirements stated in the specifications, you will discover that requirements can be grouped into various categories. This discussion identifies the types of requirements and delineates usage of terms that sometimes complicate the application.

19.6.2.1 Source or Originating Requirements When a System Acquirer formally releases a requirements document that specifies and bounds the SYSTEM / ENTITY for procurement, the specifications are often referred to as the *source* or *originating requirements*. These requirements can encompass multiple categories of requirements in a single document or several documents, such as a SRD and a SOO.

One of the problems in applying the term *source* or *originating* requirements is that it is relative. *Relative to whom*? From a System Acquirer's perspective, the User's *source* or *originating* requirements should be traceable to the User's *validated* and documented operational need. These needs may evolve through a chain of decision documents that culminate in a procurement specification.

As a paradigm, we typically expect a document called a *specification* to *specify* requirements and it does. However, there are other documents such as contracts, Ts & Cs, and SOW that serve as source or originating requirements documents.

Another type of document, the SOO, specifies requirements but in the form of performance-based outcome objectives. The SOO consists of very brief and concise objectives that express the performance-based outcomes to be achieved. Typically, the scope of the SOO focuses on critical operational and technical objectives characterized by Measures of Effectiveness (MOEs) and Measures of Suitability (MOSs) (Chapter 5) but may include cost-effectiveness objectives. In short form:

- Land two astronauts on Mars by the Year 20XX.
- Develop an unmanned aircraft that can safely transport (TBD) passengers between Point A and Point B in less than Y hours.

Whereas an SPS bounds and specifies the *solution space*, the SOO is a higher level of abstraction that effectively defines the User's *problem space* and elicits qualified System Developers to analyze the SOO's *problem space* and propose a cost-effective Solution that may include a draft SPS. When Users have an abstract *problem space* to be solved, the SOO may become the source or originating requirements.

In summary, a System Acquirer might release RFP solicitation that consists of a SOO or a SRD. Based on the User's *sources* or *originating* requirements, the System Developer responds to the RFP with a proposed SPS.

19.6.2.2 Stakeholder Requirements A specification captures all *essential* and *prioritized* stakeholder requirements that *fit* within User's technical, technology, budget, development schedule, and risk constraints.

The primary objectives of the requirements stakeholders are to ensure that:

- All requirements *essential* to their task domain are identified, analyzed, and documented in an SPS.
- Performance requirements are *accurately* and *precisely* specified and given an equitable priority relative to other Stakeholder requirements.

Principle 19.21

SE User–End User Advocacy Principle

One of Systems Engineering's project roles is to serve as a User and End User advocate to preserve the *intent* and *integrity* of their requirements.

You will often hear Enterprises and Engineers comment "if a requirement is 'open to interpretation,' we will interpret it our way." First, based on the concepts, principles, and practices discussed in this text, you should avoid getting into a need to "interpret a requirement your way." Conversely, requesting clarification can also be potentially problematic and not turn out the way you desire; it certainly will not be any different during SITE. The best solution is deal with these matters *before* the contract is signed.

19.6.2.3 Requirements Stakeholder Elicitation and Documentation Since Stakeholder requirements reside on both sides of the contract interface boundary, each Enterprise ENTITY—User, Acquirer, and System Developer—is accountable for stakeholder requirements identification. So, *how does this occur*?

The System Acquirer is typically *accountable* for establishing a consensus of agreement among the User Community, preferably before the contract is awarded. This includes Stakeholders – Users and End user - with *accountability* for the SYSTEM outcomes and performance during the System Production, System Deployment; System Operations, Maintenance, and Sustainment (OM&S); and System Disposal Phases (Figure 3.3).

System Developer Stakeholder Requirements

Author's Note 19.3 The System Developer also has contract Stakeholder performance and financial interests in clarifying any internal Stakeholder requirements that relate to the System Development and System Production Phases, if applicable, of the contract. Thus, the System Developer Stakeholder requirements must be addressed prior to submitting the SPS as part of a proposal as well as during contract negotiations. This includes agreement on the relevance to the Enterprise and its long term strategies, Return on Investment (ROI), profitability, and resource availability.

Fully Understand Contract Requirements Before Signing!

Warning 19.1 Read, fully understand, and comprehend *what* commitments you are making when you sign a contract. System Acquirer and Users are often *very reluctant* to make contract modifications even when justified.

19.6.2.4 Threshold and Objective Requirements One method of expressing and delineating requirements priorities is to establish *threshold* and *objective* requirements. Definitions include:

- **Threshold Requirements**—Requirements that have "A minimum acceptable operational value below, which the utility of the system becomes questionable" (DAU, 2011, p. B-274).
- **Objective Requirement**—"The objective value represents an incremental, operationally meaningful, time-critical, and cost-effective improvement to the threshold value of each program parameter." (DoD 5000.2R, 2002, Section C1.2.2.2, p. 19)

Threshold and Objective Requirements Example

Example 19.1 A User may require a minimum level of acceptable system performance—*threshold requirement*—with an expressed desire to achieve a specified higher level. Achievement of a higher-level *objective requirement* may be dependent on the maturity of the technology and a vendor's ability to *consistently* and *reliably* produce or implement the technology.

Threshold and objective requirements must be consistent with the Operational Requirements Document (ORD) or Capability Development Document (CDD), SRD, and a Test & Evaluation Master Plan (TEMP).

When you specify threshold and objective requirements, you should explicitly identify as such. There are several ways of doing this.

- **Option 1:** Explicitly label each threshold or objective requirement using parenthesis marks such as XYZ Capability (Threshold) and ABC Capability (Objective).
- **Option 2:** Specify "up front" in the specification that unless specified otherwise, all requirements are "threshold requirements." Later in the document when objective requirements are stated, the following label might be used "… ABC Capability (Objective) …"

Additionally, you should always define the terms (i.e., threshold and objective) in the requirements section—for example, Section 3.0—of the specification. Option 3 might involve a summary matrix of threshold and objective requirements. A shortcoming of the matrix approach is that it is physically located away from the stated requirement. This approach, which may be confusing to the reader, creates extra work flipping back and forth between text and the matrix. Keep it simple and tag the statement (threshold or objective) with the appropriate label.

19.6.2.5 Requirements versus Specification versus Design Requirements Some people refer to the "requirements" while others refer to "specification requirements." *What is the difference*? The answer depends on the context.

The contract, as the overarching document, establishes requirements that encompass, among other things specification requirements, schedule requirements, compliance requirements, and cost requirements. For brevity, people often avoid saying "specification requirements" and shorten the form to simply "requirements." As we will see, for systems with multiple levels of specifications and specifications within levels, general usage of the term "requirements" requires identification of the specification that is the frame of reference.

19.6.3 Specification Requirements Categories

Requirements can be organized into various categories to best capture their *intended* use and objectives. Typical categories include: (1) operational requirements, (2) capability requirements, (3) non-functional requirements, (4) interface requirements, (5) design and construction constraints requirements, (6) verification requirements, and (7) validation requirements. Let's briefly describe each of these categories of requirements.

19.6.3.1 Operational Requirements

Operational requirements specify high-level requirements required to achieve system mission objectives and behavioral interactions and responses within a prescribed OPERATING ENVIRONMENT and conditions. These requirements answer the question: *what is the operational need solution space the SYSTEM/ENTITY is expected to satisfy*?

19.6.3.1.1 Types of Operational Requirements Systems, products, and services perform in operational phases. As such, SEs must ensure that the requirements that express stakeholder expectations for each phase are adequately addressed. At a *minimum*, this includes Pre-Mission Phase, Mission Phase, and Post-Mission Phases of Operations (Figure 5.5).

Each phase of operation includes at least one or more modes of operation that require a specified set of capabilities to support achievement of mission phases of operation objectives. If you investigate and analyze most MISSION SYSTEM operations, you will discover that the SYSTEM must be prepared to cope with four types of OPERATING ENVIRONMENT scenarios and conditions: (1) Normal, (2) Abnormal, (3) Emergency, and (4) Catastrophic as shown in Figure 19.5. Let's define the context of each of these conditions.

19.6.3.1.2 Normal Operations Requirements
Normal operations consist of a set of MISSION SYSTEM activities and tasks that apply to system capabilities and performance operating within their specified performance limits and resources.

When specification developers derive and develop requirements, human tendencies naturally focus on the SYSTEM/PRODUCT operating *ideally*—meaning as specified. Cockburn, for example, referring to UCs calls this condition a "main success scenario" (Cockburn, 2001, p. 87) or the "happy day case" (Cockburn, 2005, p. 49). Most specifications are written for a mission success result and fail to accommodate UC scenario requirements.

In the *ideal* world, this may be true, but in the real world systems, products, and services have finite reliabilities and life cycles. As such, Systems may not always operate normally. As a result, *UCs* and *scenarios* become very important in specifying SPS requirements that cover a prescribed set of OPERATING ENVIRONMENT scenarios and conditions such as Abnormal, Emergency, and Catastrophic Operations requirements.

19.6.3.1.3 Abnormal Operations Requirements

Principle 19.22

Condition-Based Requirements Principle

Every specification must specify requirements that address three types of Abnormal Operations:

- External system failures
- Degraded operations
- Internal system failures such as components and interfaces

Abnormal operations consist of a set of system operations and tasks that focus on *detecting*, *troubleshooting, identifying, isolating,* and *correcting* a physical capability condition. These conditions represent levels of performance that are *outside* the tolerance band for nominal mission performance but may not be *critical* to the safety of humans, property, or the environment.

Abnormal Operations encompass capabilities required to address External system failures, degraded operations, and internal system failures. For example:

- Figure 10.17 identifies the need for Abnormal Recovery Operations as part of exception Handling.
- Figure 26.8 addresses fault detection and containment.

19.6.3.1.4 Emergency Operations Requirements
Emergency operations consist of an urgent set of system operations and tasks that focus exclusively on *correcting, terminating,* or *eliminating* a life threatening or hazardous situation. This includes: safety, physical capability condition that has the potential to pose a major health, safety, financial, or security risk to humans, Enterprise, property, or the environment.

19.6.3.1.5 Catastrophic Operations Requirements
Catastrophic operations consist of a set of operations or tasks performed following a major system *malfunction* event that resulted in system failure and adversely affected the health, safety, financial, and security of humans, Enterprise, property, and the OPERATING ENVIRONMENT conditions in the immediate area.

You may ask: *how can you have catastrophic operations requirements if the System or Entity is destroyed*? These requirements, where appropriate, would address Enterprise level operations for an ENABLING SYSTEM to recover the SYSTEM/ENTITY. In that context, the SPS is written for the System of Interest (SOI), which includes the MISSION SYSTEM and its ENABLING SYSTEM(S).

19.6.3.1.6 Relationships between Operating Condition Categories Requirements for SYSTEM / ENTITY OPERATING ENVIRONMENT conditions must not only scope and bound the UC scenario or the condition but also the transitory modes and states that contributed to the conditions. To better understand this statement, refer to Figure 19.5.

As indicated in the figure, a system performs Normal Operations. The challenge for SEs is to derive Reliability, Maintainability, Availability (RMA) (Chapter 34), and spares requirements to support on-board *preventive* and *corrective* maintenance operations to sustain Normal operations. If a condition or event occurs prior to, during, or after a mission, the System may be forced to transition to Abnormal Operations. Normal Operations may encounter a situation

requiring an immediate transition to Emergency Operations. Analytically, Figure 19.5 assumes some Abnormal Conditions that precede the Emergency Operations.

During Abnormal Operations, a SYSTEM/ENTITY or its operators institute recovery operations, correct or eliminate the condition, and hopefully return to Normal Operations. During Abnormal Operations, an emergency condition or event may occur forcing the system into a state of Emergency Operations. Depending on the system and its post-emergency health condition, the system may be able to revert to Abnormal Operations.

If Emergency Operation corrective actions are *unsuccessful*, the system may encounter a *catastrophic* event, thereby requiring Catastrophic Operations.

19.6.3.1.7 How Do These Categories Relate to System Requirements? Depending on the system and its mission applications, the SPS must specify operational and capability requirements that cover these conditions to ensure the safety of humans, property, and the environment. *Will you find a section in the SPS titled Normal, Abnormal, Emergency, or Catastrophic operations*? Generally, no. Instead, requirements for these types of conditions are distributed throughout the SPS due to a general lack of knowledge about how to organize requirements. As a matter of good practice, however, it is recommended that Acquirer and System Developers maintain some type of documentation that links SPS requirements to these conditions to ensure proper coverage and consideration.

19.6.3.1.8 Failure to Define Requirements for These Categories History is filled with events that exemplify a SYSTEM's or PRODUCT's inability to cope with physical conditions in its OPERATING ENVIRONMENT. These occur simply because the conditional and scenario requirements were ignored, overlooked, or assumed to have only a remote probability of occurrence.

19.6.3.1.9 Allocation of Operational Requirements to System Elements When you develop the SPS, recognize that an SOI, MISSION SYSTEM, or ENABLING SYSTEM must be capable of supporting all types of scenarios and conditions - Normal, Abnormal, Emergency, or Catastrophic - within the practicality of budgets. Remember, a SYSTEM encompasses all System Elements (Chapter 8). Ultimately, when the EQUIPMENT Element is specified, SEs also allocate conditional requirements to the PROCEDURAL DATA, PERSONNEL, MISSION RESOURCES, or ENABLING SYSTEM Elements that satisfy safety requirements.

19.6.3.2 Capability Requirements

Capability requirements specify and bound a *solution space* with functional/logical performance actions each SYSTEM/ENTITY or item must be capable of producing such as outcome(s), products, by-products, or services. Traditionally, these requirements were often referred to as *functional* requirements and focused on the function to be performed. Capability requirements, however, encompass both the action and level of performance associated with *how well* the action must be performed.

19.6.3.3 Non-functional Requirements

Non-functional requirements relate to physical constraints such as SYSTEM/ENTITY attributes and characteristics - color, weight, safety, and so forth. Non-functional requirements *do not* perform any behavioral actions but may influence a specific operational outcome or effect. For example, does a *non-functional* requirement for bright yellow paint improve a system's capability or safety? No, but it improves system safety in terms of its "capability to be seen." If paint color is considered "non-functional," why then do camouflage paint patterns or coverings provide a capability of deception?

19.6.3.4 Interface Requirements

Interface requirements consist of those statements that specify and bound a system's direct or indirect connectivity or logical relationships with external systems beyond its own physical boundary.

19.6.3.5 Environmental Conditions Requirements

Environmental Conditions requirements specify and bound a system, product, or service's Operating Environment conditions such as temperature, humidity, altitude, wind, ice, snow, salt spray, and so forth. It is important to note that a specification requires a system, product, or service to perform to requirements *before, during,* AND *after* exposure to these Environmental Conditions.

19.6.3.6 Design and Construction Constraints Requirements

Design and Construction Constraints requirements levy constraints on SYSTEM/ENTITY design such as manufacturing, Human Factors (HF), Safety, security, and so forth.

Design requirements consist of any requirement specified on a drawing, wiring list, parts list, or standard concerning the implementation of a specification requirement, design and construction constraint, and manufacturing methods.

Please be advised that Engineers in various business domains such as aerospace and defense, commercial, have different usage of specification requirements.

- In most business domains such as Aerospace and Defense (A&D), a "specification" is a document for an entity such as a SYSTEM, PRODUCT, SUBSYSTEM, and ASSEMBLY that contains "shall" based syntactical statements that express a capability to be provided

including its level of performance and verification methods for proving compliance. Specification requirements are then translated into multi-level designs that consist of drawings, wiring lists, parts lists, and documents containing information is referred to as "design requirements."

- In some commercial domains, a *drawing* is viewed as a "specification," and its contents such as notes, part numbers, resistor, or capacitor values are referred to as "specification requirements."

19.6.3.7 Verification Requirements

Verification requirements specify methods to be employed to assess SYSTEM/ENTITY compliance with an operational, capability, interface, or non-functional requirement. Verification requirements are typically stated in terms of *verification methods* such as inspection, analysis, demonstration, test, and similarity (if allowable).

19.6.3.8 Validation Requirements

Validation requirements consist of mission-oriented, Use Case (UC) scenario statements intended to describe what must be performed to clearly demonstrate that the *right* system has been built to satisfy the User's intended operational needs. Validation requirements are typically documented in a Test and Evaluation Master Plan (TEMP) or OT&E Plan prepared by the User or an Independent Test Agency (ITA) representing the User. Since UCs represent how the User envisions using the system, a UCs Document would serve a comparable purpose. In general, validation should demonstrate that Critical Operational or Technical Issues (COIs/CTIs) or concerns have been *resolved* or *minimized*.

19.7 CHAPTER SUMMARY

In summary, Chapter 19 provides an introduction to specification Development. Key points include:

- A discussion of what a specification is, its objectives, and need.
- What is meant by a "Good" Specification versus a "Well-Defined" Specification.
- Attributes of Well-Defined Specifications.
- Types of specifications that can be used for various types of specification applications and when they are developed.
- An overview of specification topical contents.
- What is a requirement and various types of requirements within a specification.
- The need to address four types of System operational requirements in an SPS: Normal Operations, Abnormal Operations, Emergency Operations, and Catastrophic Operations.
- We noted that abnormal Operations should address three types of Abnormal Operations (1) external system failures, (2) degraded operations, and (3) internal failures such as components and interfaces and their possible recovery (Figure 10.17).

19.8 CHAPTER EXERCISES

19.8.1 Level 1: Chapter Knowledge Exercises

Answer each of the What You Should Learn from This Chapter questions identified in the Introduction.

1. What is a specification?
2. What is a well-defined specification?
3. Describe the evolution of specifications from initial System Concept to SPS.
4. What are the basic types of specifications?
5. How does each type of specification apply to system development?
6. What is a Specification Tree and how is it structured?
7. Who "owns" a specification and what is their authority to implement changes?
8. What is the generalized format for most specifications?
9. What is a requirement?
10. What is a source or originating requirement?
11. What is a stakeholder requirement?
12. What is an objective requirement?
13. What is a threshold requirement?
14. What are the categories of specification requirements?
15. What are operational requirements?
16. What are capability requirements?
17. What are non-functional requirements?
18. What are design requirements?
19. What are interface requirements?
20. What are verification requirements?
21. What are validation requirements?
22. What are requirement priorities?
23. What are the four types of operational requirements?
24. What are four common problems with requirements?

19.8.2 Level 2: Chapter Knowledge Application Exercises

Refer to www.wiley.com/go/systemengineeringanalysis2e

19.9 REFERENCES

Cockburn, Alistair (2001), *Writing Effective Use Cases*, Boston, MA: Addison-Wesley.

Cockburn, Alastair (2005), *Presentation–"Writing Effective Use Cases"* meets "*Agile Development*". Retrieved on 3/25/14 from http://Alistair.Cockburn.us.

DAU (2011), *Glossary: Defense Acquisition Acronyms and Terms*, 14th ed. Ft. Belvoir, VA: Defense Acquisition University (DAU) Press. Retrieved on 3/27/13 from http://www.dau.mil/pubscats/PubsCats/Glossary%2014th%20edition%20July%202011.pdf.

DAU (2012), *Glossary: Defense Acquisition Acronyms and Terms*, 15th ed. Ft. Belvoir, VA: Defense Acquisition University (DAU) Press. Retrieved on 3/27/13 from http://www.dau.mil/publications/publicationsDocs/Glossary_15th_ed.pdf.

DoD 5000.2-R (2002), *Mandatory Procedures for Major Defense Acquisition Programs (MDAPS) and Major Automation Information (MAIS) Acquisition Programs*, Washington, DC: Department of Defense (DoD).

FAA SEM (2006), *System Engineering Manual*, Version 3.1, Vol. 3, National Airspace System (NAS), Washington, DC: Federal Aviation Administration (FAA).

MIL-STD-499B Draft (1994), *Military Standard: Systems Engineering*, Washington, DC: Department of Defense (DoD).

MIL-STD-961E (2003), *DoD Standard Practice for Defense Specifications*. Washington, DC: Department of Defense (DoD).

SD-15 (1995), *Performance Specification Guide*, Defense Standardization Program, Washington, DC: Department of Defense, Office of the Assistant Secretary of Defense for Economic Security.

20

SPECIFICATION DEVELOPMENT APPROACHES

System Performance Specifications (SPS) and Entity Development Specifications (EDSs) provide the formal vehicle for a System Acquirer (role), System Developer (role), or Services Provider to:

1. Specify *what* capabilities a deliverable SYSTEM/ENTITY is required to provide.
2. Bound *how well* the capabilities are to be performed.
3. Identify external interfaces the SYSTEM/ENTITY must accommodate.
4. Levy constraints on the solution set.
5. Establish criteria concerning how the System Developer is to demonstrate proof of compliance as a prerequisite for delivery and acceptance.

At the highest level, the Statement of Objectives (S00) or an SPS or establish a contract's technical agreement between the System Acquirer, as the User's technical representative, and the System Developer.

20.1 DEFINITIONS OF KEY TERMS

- **Architecture-Based Approach**—A structured, analytical approach that employs (1) a multi-level, logical capabilities architecture framework and (2) behavioral modeling to specify system capabilities and performance requirements.
- **Feature**—A key capability *expected* or *perceived* by Stakeholders - Users and End Users – as a benefit that *motivates* them to purchase a system, product, or service. For example, battery operated and Internet ready.
- **Feature-Based Approach**—An *ad hoc*, brainstormed approach to specification requirements development. This approach, by virtue of its feature-based nature, is subject to omissions in the hierarchy of requirements and is highly dependent on reviewer assimilation and recognition of the omissions to correct them.
- **Performance-Based Approach**—A specification development approach that analytically treats a SYSTEM or ENTITY as a "black box" that is bounded and specified by (1) acceptable and unacceptable inputs, (2) external interface constraints, (3) acceptable and unacceptable performance-based outcomes, (4) constraint requirements, and (5) verification requirements.
- **Reuse-Based Approach**—An approach that (1) *exploits* or *plagiarizes* an existing specification text from one domain that *may or may not* to be applicable to a specific application in another domain and (2) essentially replaces semantics to create a new specification. This approach, which is typically prone to *errors* and *omissions*, is highly dependent on the specification writer's and reviewer's knowledge and expertise to identify and correct errors and omissions.
- **Specification Review**—A technical review by Stakeholders, peers, and Subject Matter Experts (SMEs) to assess the completeness, accuracy, validity, testability, verifiability, producibility, and risk of a specification and its requirements.

System Engineering Analysis, Design, and Development: Concepts, Principles, and Practices, Second Edition. Charles S. Wasson.

Companion Website: www.wiley.com/go/systemengineeringanalysis2e

20.2 APPROACH TO THIS CHAPTER

Chapter 20 introduces Specification Development Practices emphasizes the need for Enterprises to establish Organizational Standard Processes (OSPs) that establish a standard outline for use on projects. We introduce a standard outline that serves as an example for our discussions.

Establishing a standard specification outline, however, does not mean that it will be developed in a way that covers essential requirements for developing a system, product, or service. Our discussion introduces four common specification development approaches and highlights advantages and disadvantages.

Next we address special topics in specification development.

We conclude with a discussion of specification reviews, how to perform them, identify challenges, and address various approaches for reviewing specifications.

20.3 INTRODUCTION TO SPECIFICATION DEVELOPMENT

There are numerous ways of structuring a specification outline. Rather than elaborate on commonly available specification outlines, let's apply some *systems thinking* to what specifications are intended to communicate. Then, based on that discussion, translate the information into a meaningful outline structure that best fits your Enterprise or application.

20.3.1 Specification Outlines

As stated in Chapter 19, a *well-defined* specification begins with a standard outline (Principle 19.3). In general, specification outlines originate from System Acquirer contract requirements or internally from Enterprise command media such as Standard Engineering Practices. If you do not have one, consider establishing one in your Enterprise command media.

Heuristic 20.1 Specification Writing Versus Development

Structure a specification to present the highest-level requirements by the third level of the outline.

If you analyze and implement most specification outlines, you will discover two things:

1. Development of specification outlines are sometimes assigned to personnel with limited experience or may be between projects and need a task.
2. Specification outlines tend to be organized around major headings. You can organize a specification with many levels of abstract, esoteric topics resulting in the first meaningful requirement appearing at the fourth, fifth, or sixth level. Imagine having the highest-level requirement statement appear at the 5th level in Section 3.X.X.X.X. Since specifications for large, complex systems often require four to ten levels of detail, placing the first requirements at the 5th level makes the document *unwieldy* and *impractical* to read and reference.

20.3.2 Example Specification Outline

Figure 19.4 identified the key elements of a specification representing what needs to be communicated. Table 20.1 provides an example outline.

20.3.3 Specification Outline Notes

The example specification outline is simply one approach. You should tailor this outline to best meet the needs of your Enterprise and project.

Topics in this outline are similar to those such provided in DI-IPSC-81431A (FAA, 2000). The contents of that document have been proven over many decades. However, the outline above differs in several key areas for valid reasons.

20.3.3.1 Section 3.0 Requirements The introduction to Section 3.0 REQUIREMENTS should state that "The SYSTEM/ENTITY shall: (1) comply with the requirements specified herein *before, during,* and *after exposure* to the Environmental conditions specified in Section 3.5 Operating Environment Conditions, (2) be designed in accordance with Section 3.6 Design and Construction Constraints, and be (3) be verified in accordance with Section 4.0 Qualification Provisions.

20.3.3.2 Section 3.1 Missions Users acquire systems, products, and services as physical assets for personnel to use to perform their Enterprise missions. In that context, it is important to acquire the right tool suitable for performing specific types of missions. The Measure of Suitability (MOS) metric introduced in Chapter 5 *quantifies* this point. Therefore, it is useful for the System Developer to understand what types of User missions the system, product, or service is expected to perform.

Referring to Figure 20.1 - As a caveat to the preceding point, some Users create Operational Requirements Documents (ORDs) or Capability Development Documents (CDDs) that employ MOEs and MOSs as a conceptual basis for defining and acquiring a system, product, or service. System Acquirer derives capability requirements from the ORD or CDD for insertion into in a System Requirements Document (SRD) as part of a Request for Proposal (RFP). In that case, these topics may not be relevant in the specification.

TABLE 20.1 Example Specification Outline

1.0 INTRODUCTION

1.1. Scope
1.2. System Overview
1.3. Definitions of Key Terms (optional)

2.0 REFERENCED DOCUMENTS

2.1. User Documents
2.2. System Acquirer Documents
2.3. Project Documents
2.4. Specifications, Standards, and Handbooks
2.5. Usage and Definition of "Shall," "Will," and "Goals" (Chapter 21)

3.0 REQUIREMENTS

3.1. Missions
3.2. Operational Performance Characteristics
- 3.2.1. MOE #1
- …
- 3.2.*n*. MOE #*n*
- 3.2._. Mission Reliability
- 3.2._. Mission Maintainability
- 3.2._. Mission Availability

3.3. Capabilities
- 3.3.1. Capability #1
- 3.3.2. Capability #2
- 3.3.3. Capability #3
- …
- 3.3.*n*. Capability #*n*

3.4. Interfaces
- 3.4.1. External Interfaces
  - 3.4.1.1. External Interface #1
  - …
  - 3.4.1.*n*. External Interface #*n*
- 3.4.2. Internal Interfaces (Not recommended - See discussion)

3.5. Operating Environment Conditions
3.6. Design and Construction Constraints
- 3.6.1. Standards of Manufacture
- 3.6.2. Workmanship
- 3.6.3. Component Parts
- 3.6.4. Human Factors (HF)
- 3.6.5. System Safety
- 3.6.6. Security and Privacy
- 3.6.7. Computer Resources
- 3.6.8. Physical Characteristics
- 3.6.9. Adaption
- 3.6.10. Personnel and Trainin
- 3.6.11. Special Test Equipment
- 3.6.12. Transportability
- 3.6.13. Logistics
- 3.6.14. Sustainment
- 3.6.15. Technical Documentation

3.7. Precedence and Criticality of Requirements

(*continued*)

TABLE 20.1 (*Continued*)

4.0	**QUALIFICATION PROVISIONS** 4.1. Responsibility for Verification 4.2. Verification Methods 4.3. Quality Conformance Inspections 4.4. Qualification Tests
5.0	**PREPARATION FOR DELIVERY (PHS&T)** 5.1. Packaging 5.2. Handling 5.3. Storage 5.4. Transportation
6.0	**NOTES** 6.1. Acronyms and Abbreviations 6.2. Definitions 6.3. Assumptions
7.0	**APPENDICES**

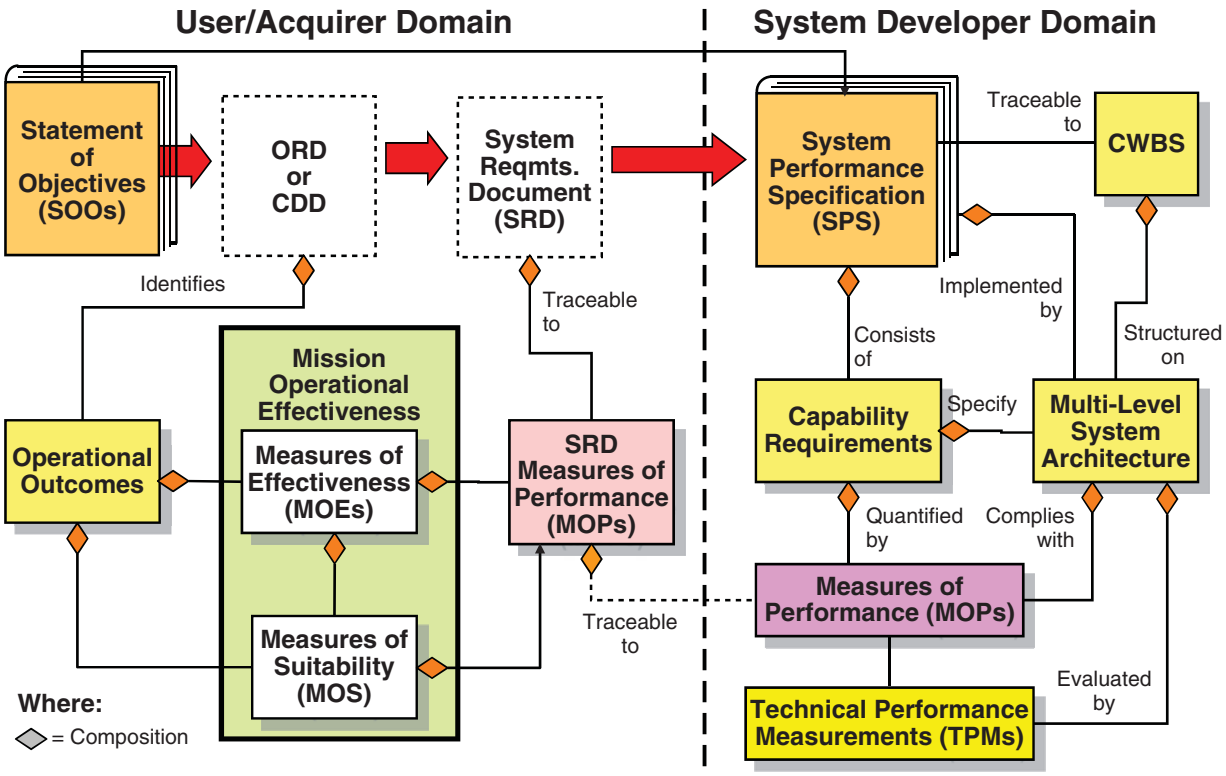


Figure 20.1 MOE, MOS, MOP, and TPM Entity Relationships (ERs) within Specifications

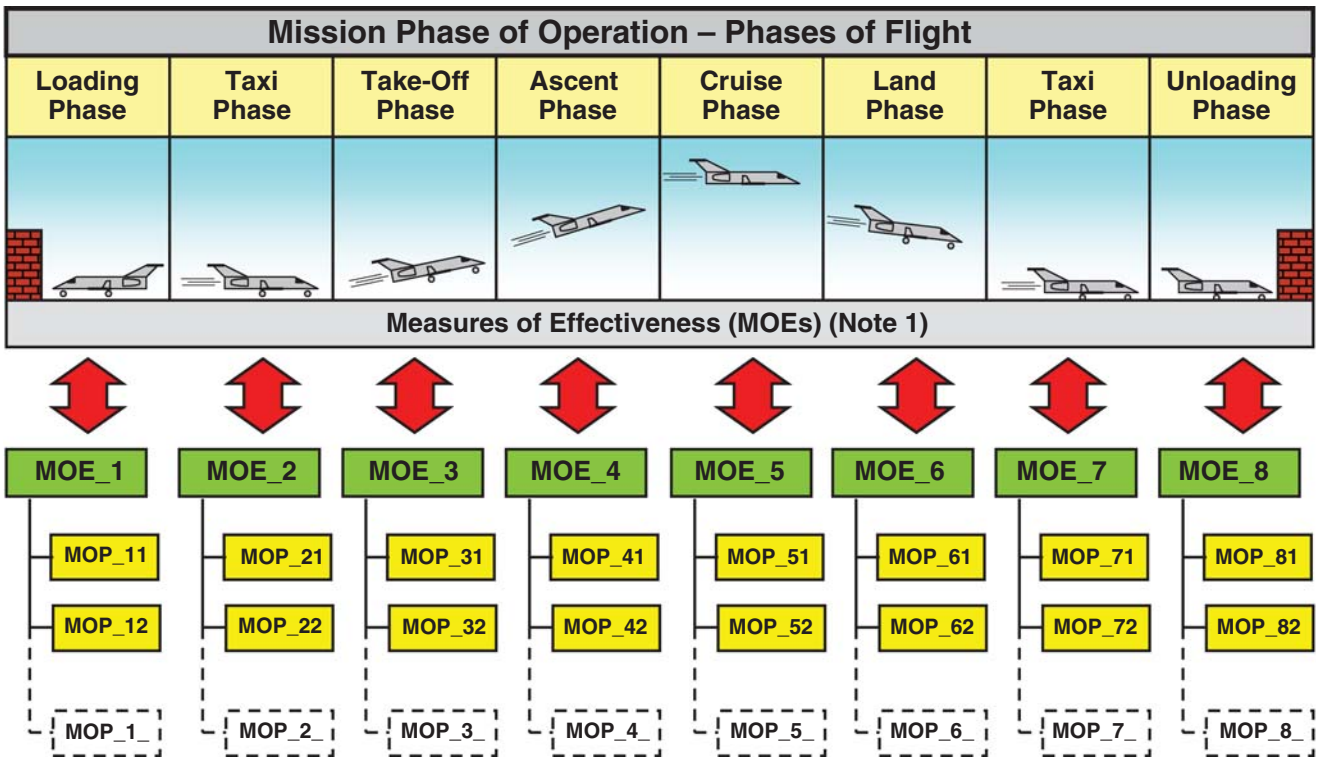
20.3.3.3 Section 3.2 Operational Performance Characteristics

Principle 20.1

Measure of Performance (MOP) Principle

Each specification requirement should specify by one and only one Measure of Performance (MOP) characterized by (1) a Technical Performance Parameter (TPP), (2) a magnitude, and (3) a unit of measure.

When a system, product, or service is employed to perform missions, the System Developer needs to understand what constitutes *mission success* from the perspective of the User.



Notes: 1. To keep the graphic simple, only one MOE is illustrated for each subphase. Subphases may have several objectives and supporting use cases, each of which is assessed by an MOE.

2. Land Phase not shown..

Figure 20.2 Aircraft Mission Cycle MOEs and MOPs

Therefore, this section specifies Measure of Effectiveness (MOE) requirements that express *what* the User expects it to accomplish. MOEs quantify "operational" performance in which the system or product is treated as an object that performs missions. Figure 20.2 illustrates an aircraft mission life cycle in which each Mission phases of Operation and embedded Phases of Flight has an MOE from which contributory MOP requirements are derived.

Based on the operative term *effectiveness*, the metric should be based on a frame of reference such as Miles Per Gallon (MPG), energy/mile/pound (kg) of cargo, For example, 0 - 60 (0 to 97 km/h or 0 to 27 m/s) or 0–100 Miles Per Hour (MPH) are used as the standard for measuring automobile performance in various parts of the world. Therefore, the MOE becomes the number of seconds required to achieve the 60 MPH (97 km/h) or 100 MPH thresholds.

20.3.3.4 Sections 3.2.X Mission Reliability, Maintainability, and Availability (RMA) RMA is often embedded in Section 3.6 DESIGN AND CONSTRUCTION CONSTRAINTS in the back of the document. If you are a System Developer, the RMA requirements can easily become a *showstopper*. Since missions and RMA are mutually dependent (Chapter 34), they need to be stated "up front" as part of the Section 3.2 Operational Performance Characteristics, not in the back of the document. Therefore, RMA is relocated to this section.

20.3.3.5 Section 3.3 Capabilities You may ask: *what is the difference between Section 3.2 Operational Performance Characteristics and Section 3.3 Capabilities*? As noted above Section 3.2 addresses "operational" MOEs. Section 3.3 Capabilities addresses requirements for specific logical capabilities that *are contributory performance effectors* such as propulsion, steering, energy source, and so forth.

Mini-Case Study 20.1

Operational Performance Characteristics Specifications

Let's assume you decide to eliminate specification Section 3.2 Operational Performance Characteristics and derive Section 3.3 Capabilities directly from a User's ORD, CDD, or SRD and specify capabilities for the propulsion, steering, energy source, and so forth. *How do you know that the vehicle delivered by the System Developer will actually go from 0 – 60 MPH in X*

seconds? Specifying discrete capabilities such as engine size and horsepower (hp) does not mean that the automobile, as a system, will achieve that requirement. Where is the *overarching* operational requirement to go from 0 to 60 MPH in X seconds? The requirement only exists in the User's ORD or CDD; the System Developer is only obligated by contract to verify compliance to the SPS based on the System Acquirer's or Users derivation of the SRD from the ORD or CDD.

You may challenge this last statement and observe that deriving SPS requirements from the User's SRD is no different from a System Developer deriving SUBSYSTEM EDS requirements from the SPS. It is still requirements derivation, allocation, and flow down. In principle, that is true. However, here is the difference.

If the System Acquirer/User derived the SRD and issue a contract to deliver to those requirements, *what if that system does not perform*. The System Developer will contend they delivered the system specified by the contract. In contrast, assume that an SPS contains a Section 3.2 Operational Performance Characteristics. SPS requirements are derived, allocated, and flowed down to SUBSYSTEMS. At delivery, the System Developer is still legally obligated by contract to deliver a SYSTEM that is compliant with Section 3.2 Operational Performance Characteristics.

20.3.3.6 Section 3.4 Interfaces Some specification outlines include Section 3.4.1 External Interfaces and Section 3.4.2 Internal Interfaces. As we shall see in the specification development approaches discussion that follows, a specification *specifies* what has to be accomplished and *how well*, not *how to* design the SYSTEM / ENTITY. If you treat any ENTITY from the SYSTEM Level or lower as a "black box," the internal interfaces are *unknown* and *undefined* until the ENTITY's Architecture has been selected—based on the requirements in this specification. Therefore, it is recommended that you consider specifying *only external interfaces* unless there is some compelling reason to do otherwise based on an informed, fact-based decision.

We now shift our focus to understanding various approaches used by Enterprises and Engineers to develop specifications.

20.4 SPECIFICATION DEVELOPMENT APPROACHES

Specification Writing Versus Development Principle

Principle 20.2 Anyone with basic grammar skills can "write" a random set of requirement "shall" statements. However, "development" of a specification requires a higher level of Systems Thinking that requires understanding, organizing, modeling, and translating a User's operational needs and constraints into a coherent set of various types of System/Entity specification requirements.

Enterprises and SEs employ a number of approaches to develop specifications. Typical approaches include:

1. Approach #1—Feature-Based Specification Development.
2. Approach #2—Reuse-Based Specification Development.
3. Approach #3—Performance-Based Specification Development.
4. Approach #4—Architectural Model-Based Specification Development (MBSD).

Let's explore a brief description of each type beginning with the most informal, the Feature-Based Approach.

20.4.1 Feature-Based Specification Development Approach

Specification Requirements Deficiencies Principle

Principle 20.3 Ensure that every specification is free from missing, misplaced, contradictory, and duplicated requirements.

Feature-based specifications are essentially *ad hoc*, brainstormed requirements that capture the Stakeholders' - Users or End Users - imagination and attention. Specifications developed in this manner are often just formalized, loosely coupled *wish lists*.

People who lack formal training in specification development commonly use a feature-based approach. Although Feature-Based specifications may use standard specification outlines, they are often *poorly organized* and prone to missing, misplaced, conflicting, and duplicated requirements. Figure 20.3 provides an illustration.

20.4.1.1 Advantages of the Feature-Based Approach

Irrational Logic Principle

Principle 20.4 It is always easy to rationalize *erroneous logic* that ignores best practices and common sense for the sake of meeting schedule and budget constraints.

The Feature-Based Approach enables developers to quickly *elicit* and *collect* requirements inputs with a *minimal* effort. Specification "writers" spend their time "writing" requirements and very little time analyzing and understanding potential implications and impacts of those requirements. When time is limited or a new system is being developed from *minor* modifications of an existing system, this method may be *marginally acceptable*. Every system is different and should be evaluated on a case-by-case basis.

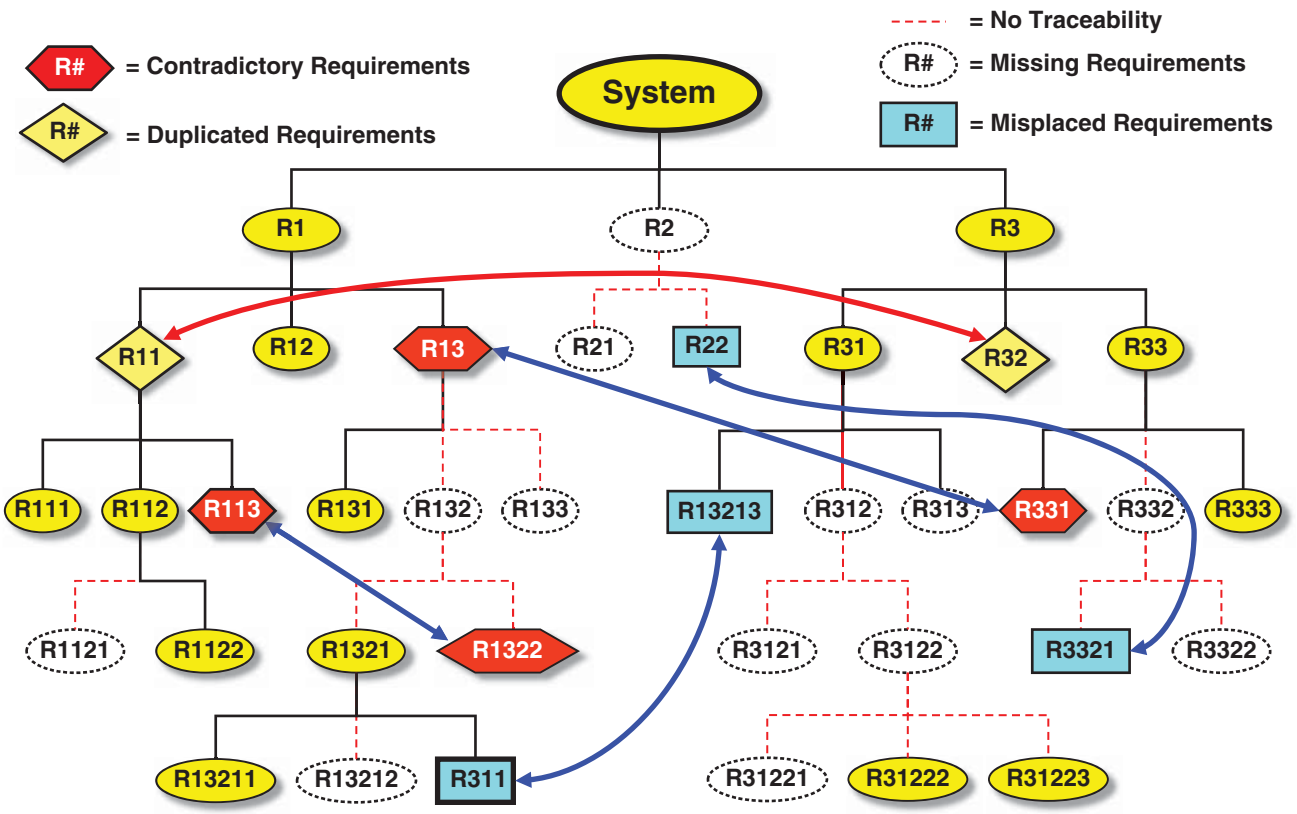


Figure 20.3 Requirements Hierarchy Tree Illustrating Common Specification Requirements Problems

20.4.1.2 Disadvantages of the Feature-Based Approach

Feature-Based specifications, by virtue of the ad hoc approach, exhibit a number of latent defects – errors, omissions, flaws, and so forth. When left uncorrected, the cost-to-correct (Table 13.1) increases over time almost exponentially throughout the System Development Phase. Figure 20.3 provides an illustration of the requirements hierarchy illustrating the following types of latent defects.

1. Missing requirements that were overlooked due to the *ad hoc* approach.
2. Compound requirements written in paragraph style prose.
3. Conflicts with other requirements.
4. Duplication of requirements.
5. Vague and ambiguous requirements statements open to interpretation.

Latent defects such as these often occur in the SDBTF-DPM Engineering Enterprises.

As an illustration of Principle 20.3, consider Requirements R31222 and 31223 in Figure 20.3. These are requirements appear in the specification as random, ad hoc, "wish list" items with no derivation from higher level requirements - R3122 and R312, which are missing. As a result, there is no traceability to the System Level requirement. This point illustrates how a lack of an SE course in Engineering Education (Figure 2.11) allows amateurish methods to proliferate in industry and government.

20.4.2 Reuse-Based Specification Development Approach

The *Reuse-Based Approach* simply exploits or plagiarizes an existing specification or may integrate "requirements" from several specifications. The underlying assumption is that if the existing specifications were "good enough" for those systems, they should apply to this SYSTEM or ENTITY as well. This can potentially be a big mistake! *Think* about it! The source specification you plan to use as a *starting point* may be (1) of poor quality or (2) for an entirely different SYSTEM or application and OPERATING ENVIRONMENT conditions!

The Reuse-Based Approach, which may be the only type of experience an Engineer has, is often used under the *guise of economy* - saving time and money. The specification "writers" *fail to recognize* that corrections during System Integration, Test, and Evaluation (SITE) due to *rework, scrap,* and *redesign* caused by deficiencies in this approach, such as *overlooked* and *incorrect* requirements, often cost more

than employing the Architectural Model-Based Approach discussed later in this chapter.

Specification *reuse* often occurs where there is a standard product line or application similar to a legacy system. You are working from a *known* SYSTEM or ENTITY accepted and refined by the Enterprise. However, the problem with specification *reuse* occurs across product domains, system applications, and product lines that may have not relevance to each other, which can cause *significant* technical and programmatic risk. For example, smartphone specifications are significantly different from desktop compute specification despite commonalities in some capabilities.

The Reuse-Based Specification Approach is generally the standard repertoire for new Engineers, especially if they have not been given formal training in the proper ways to develop specifications. A manager tasks an Engineer to develop a specification. Confronted with meeting schedule commitments, the Engineer decides to contact someone who might have an existing specification and simply adapt it to meet the needs of the task. Constructively speaking, this approach becomes the default survival mechanism for the Engineer – why reinvent the wheel every time - without ever learning the proper way. Unfortunately, most Enterprises and managers contribute to the problem. They lack knowledge themselves and fail to recognize the need to provide the proper training.

The risk of using the Reuse-Based Approach as the source or "model" specification may be intended for a totally different system, system application, or mission. The only commonalities between applications may be the high-level topics of the outline. As a result, the specification could potentially contain two types of flaws:

- References to external specifications and standards that may have been updated to a new version, become obsolete, or cancelled.
- Retention of requirements that are no longer relevant or applicable.

Even if the requirements are topically relevant, they may *miss* or over/under specify the capabilities and levels of performance required for the new system's field application.

Heading 20.1 Our discussion of the Performance-Based and Model-Based Approaches provide better methods of developing a specification. As we will see, the Architectural Model-Based Approach presumes some knowledge of the SYSTEM/ENTITY architecture to specify entities with the architecture.

20.4.3 Performance-Based Specification Development Approach

The Performance-Based Approach specifies SYSTEM/ENTITY capability requirements in terms of performance boundary conditions, value-added transfer function, and interactions with external systems. The SYSTEM/ENTITY is analytically treated as a simple box as illustrated in Figure 20.4. The specification outline presented earlier is then used to bound and specify the System/Entity.

Performance-based specifications represent the *preferred approach* to specification development for many applications, particularly *unprecedented* systems. By *avoiding* design-specific requirements (Principle 20.5), the System Acquirer provides the System Developer with the *flexibility* to innovate and create any number of architectural solutions within contract cost, schedule, and risk constraints. Depending on the System Acquirer's intent, performance-based specifications require extensive System Developer/Subcontractor's structured analysis and derivation of requirements to select a preferred system architecture.

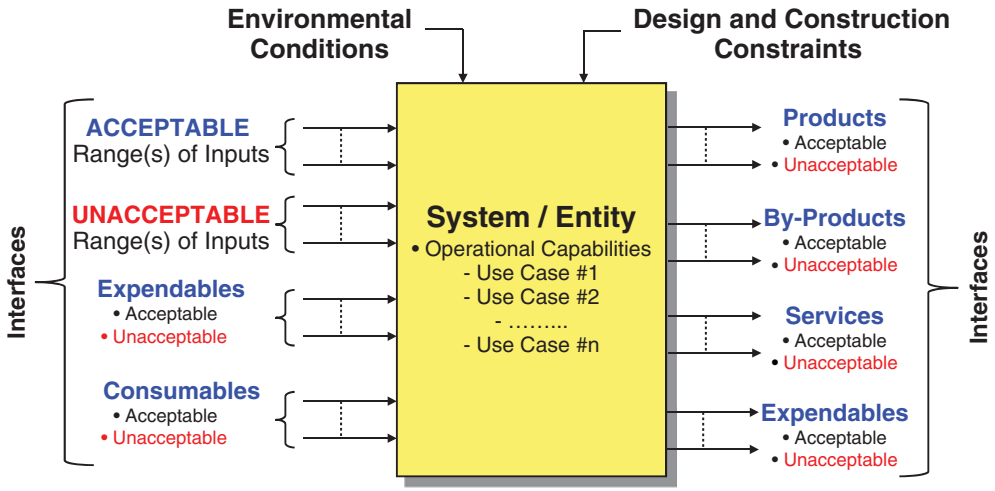


Figure 20.4 Performance-Based Approach to Specification Development

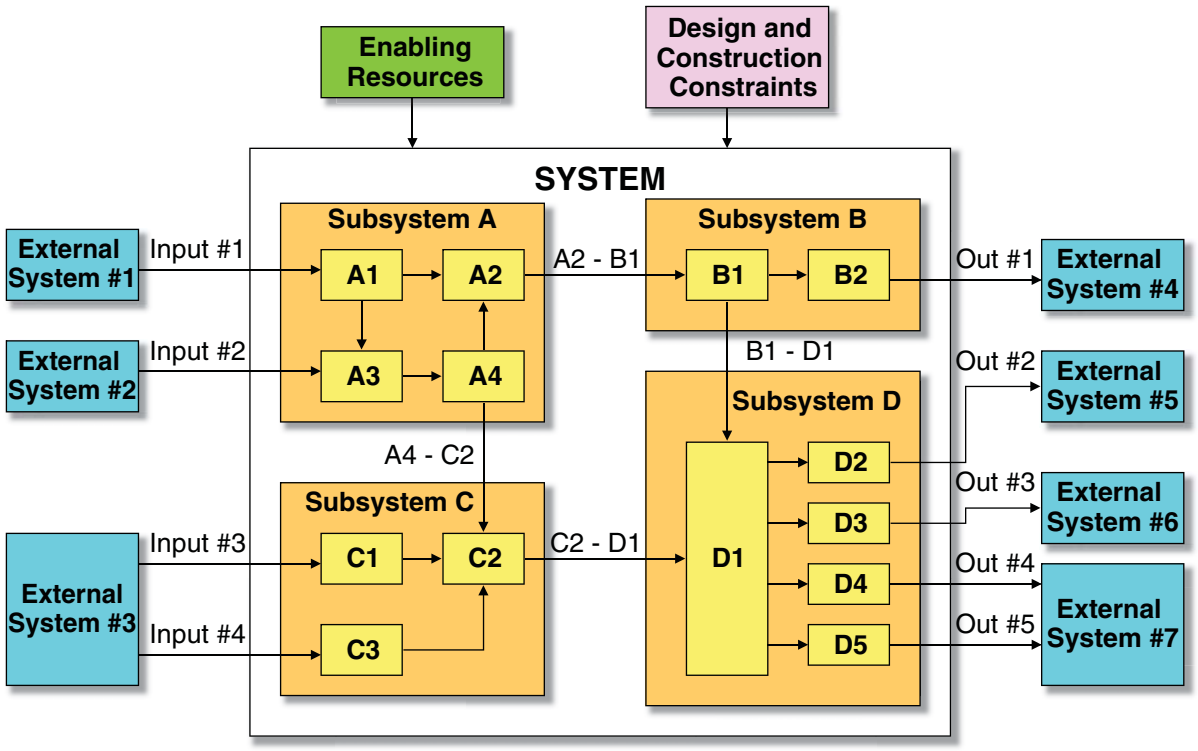


Figure 20.5 Architectural Model-Based Approach to Specification Development

System Acquirers developing an *unprecedented* system tend to favor the Performance-Based Specification Approach as the initial step of a multi-phase acquisition strategy where requirements may be *unknown* or *immature*. The strategy may require a series of Spiral Development (Figure 15.4) contracts to *evolve* and *mature* the system requirements. Consider the following example.

Example 20.1 **Unprecedented System Specification Development**

A System Acquirer plans to develop an *unprecedented* system. After due consideration, the System Acquirer decides to establish a multi-phase acquisition strategy. Phase 1 of the acquisition strategy results in the award of a performance specification-based contract to develop initial prototypes for testing, collecting, and analyzing performance data; selecting a system architecture; and producing a set of requirements as the work product for a Phase 2 follow-on prototype or system. There may even be several other Spiral Development (Figure 15.4) phases, all focused on derisking the final system development.

20.4.4 Architectural Model-Based Specification Approach

The Architectural Model-Based Approach focuses to specification development on bounding and specifying capabilities and performance for a high-level SYSTEM/ENTITY capabilities architecture similar to the one shown in Figure 20.5. This approach is best accomplished via Model-Based Systems Engineering (MBSE) (Chapters 10 and 33).

The Architectural Model-Based Approach begins with treating the SYSTEM as a "black box" similar to the Performance-Based Approach. The system is derived architecturally in successive iterations to create capability behavioral models of System/Entity capabilities based on system UCs and scenarios. To illustrate this approach, consider the following example.

Let's assume we have been tasked to develop the SPS for a land-based vehicle. Specification developers establish Section 3.2 Operational Performance Characteristics. Moving on to Section 3.3 Capabilities, we identify and list the capabilities of the vehicle.

3.3. Capabilities
- 3.3.1. Vehicle Frame Subsystem
- 3.3.2. Body Subsystem
- 3.3.3. Propulsion Subsystem
- 3.3.4. Fuel Subsystem
- 3.3.5. Electrical Subsystem
- 3.3.6. Command and Control (C2) (Chapters 7 and 26)
- 3.3.7. Situational Assessment (Chapter 26)

3.3.8. Cooling Subsystem
3.3.9. Environmental Control Subsystem
3.3.10. Steering Subsystem
3.3.11. Entertainment Subsystem
3.3.12. Storage Subsystem

Let's assume that the vehicle is a precedented system. Therefore, we can assume that the automobile requires the capabilities listed. For example, most automobiles have four passenger doors. You do not have to question the need for four doors unless there is a compelling reason to do so such as shift to a new paradigm.

Given these Capabilities, we can employ tools such as a matrix (Figure 8.10) and N2 Diagram (Figure 8.11) as interim steps to create Entity Relationships (ERs) among System capabilities for an Architectural Block Diagram (ABD) or model similar to Figure 20.5.

Then, for each Section 3.3 Capability, derive the next lower level capability requirements from the parent capability (Sections 3.3.1 – 3.3.12) and interactions with other capabilities based on SYSTEM/ENTITY level UCs and scenarios. Once these are in place, develop SysML™ Use Case Diagrams, Sequence Diagrams, and Activity Diagrams (Chapter 5) to establish behavioral models of each capability.

20.4.4.1 Implementing the Architectural Model-Based Approach To illustrate the application of the Architectural Model-Based Approach, let's assume a SYSTEM or PRODUCT Development Team (SDT/PDT) creates an analytical architecture for a SYSTEM or PRODUCT that includes SUBSYSTEMS A through D as illustrated in Figure 20.5.

SUBSYSTEMS A–D represent *solution spaces* to fill the overall SYSTEM Level *problem space* (Figure 4.7). In general, SUBSYSTEMS A–D represent Capabilities A–D. To simplify our model, let's assume that the System Developer has already decided that these are SUBSYSTEMS and is ready to derive and define each SUBSYSTEM's capabilities.

Continuing our iterative elaboration of each level of abstraction, SUBSYSTEM A becomes a *contextual problem space* (Figure 4.7) and has been determined to be solved by Capability Al through A4 *solution spaces*. The same goes for the SUBSYSTEM B *problem space* consisting of Capability B1 and B2 *solution spaces*. The team constructs a logical capabilities architecture shown in Figure 20.5 that expresses the relationships between:

1. The SYSTEM and External Systems #1 through #7.
2. Capabilities A through D.
3. Capabilities Al–A4, B1–B2, System Monitoring.

Author's Note 20.1

Architectural Models

Observe usage of the term "architectural model." The specification development team creates a model of the SYSTEM / ENTITY that is *informally* or *formally* controlled by the team for their exclusive use.

Based on the analysis, the specification developer(s) elaborates and specifies specification Section 3.3 Capabilities as follows:

3.3. Capabilities (Entity—SYSTEM, PRODUCT, SUBSYSTEM)
- 3.3.1. Capability A
  - 3.3.1.1. Capability A1
  - 3.3.1.2. Capability A2
  - 3.3.1.3. Capability A3
  - 3.3.1.4. Capability A4
- 3.3.2. Capability B
  - 3.3.2.1. Capability B1
  - 3.3.2.2. Capability B2
- 3.3.3. Capability C
  - 3.3.3.1. Capability C1
  - 3.3.3.2. Capability C2
  - 3.3.3.3. Capability C3
- 3.3.4. Capability D
  - 3.3.4.1 Capability D1
  - 3.3.4.2 Capability D2
  - 3.3.4.3 Capability D3
  - 3.3.4.4 Capability D4

The model represents the configuration of capabilities that supports SYSTEM or ENTITY UCs and scenarios. This means that for any UC or scenario, the analysts can trace the "thread" from input through each capability to produce a performance-based outcome (Figures 21.6 and 21.7).

Using this method, the specification developer(s) translate *what* capabilities the SYSTEM / ENTITY is required to provide via text statements positioned in the specification outline. For analysis "working paper" purposes, consider inserting specification paragraph references in the appropriate graphical model element. Each capability is then decomposed into multi-level sub-capabilities that form the basis for outcome-based performance requirement statements.

Principle 20.5

Point Design Avoidance Principle

Specifications do not specify a *point design solution* unless there is a compelling *technical* reason to do so.

For specifications developed for implementation *internal* to the System Developer's Enterprise such as Configuration Items (CIs) (Chapter 16), you may decide to include Figure 20.5 as a convenient reference. However, if you intend to procure the SYSTEM or one or more SUBSYSTEMS from external vendors, you are violating Principle 20.5 by mandating a specific solution. Remember - specifications specify *what* has to be accomplished and *how well*, not *how* to design the SYSTEM. Think about the implications! What if you architecturally:

- Specify the *wrong* solution or one that does not perform?
- Exclude solution options that may prove to have been *optimal*?

Later, if you or the vendor determines that the graphic *overlooked* a key capability, you may be confronted with modifying the contract and paying additional money to incorporate the missed capability. The downside here is that it "opens the door" - provides an opportunity - for the vendor to *recover* costs at your expense for other capabilities they *overlooked* in their original estimate.

Additionally, if the System Developer (1) develops the SYSTEM/ENTITY you mandated graphically in a specification and (2) it fails to satisfy your needs, their response may be "We built the system … you … contracted us to develop."

A Word of Caution 20.1

Risk of Inserting Architectural Graphics into Specifications

Exercise caution when inserting architectural graphics into specifications, especially SYSTEM / ENTITY architecture graphics. Exploit ABDs as analytical working documents for *internal use only*, not in specifications. Under certain conditions, ABDs may be fine for *internal* specifications being developed in-house.

Recognize that when you create an Architectural (Capabilities) Model (Figure 20.5) and model SYSTEM operations (Figures 10.13 – 10.16), each "box" within the model's structure represents a capability that can be *translated* into a specification requirement statement. As a result, the SPS's structure represents the requirements hierarchy of essential requirements. Since the Architectural Model is for *internal analytical use* by the System developer, it is the SPS that is delivered as part of an RFP response, not the architectural model.

By creating an analytical Architectural Model of logical capabilities and relationships such as Figure 20.5 to support specification development, you improve your chances of covering and expressing capabilities without *mandating* a specific solution or specific architectural graphic. If you perform the analysis well and translate the working architecture into capability-based requirements, an Engineer with some intuitive insight should be able to "reverse engineer" the system graphic. The difference is you did not violate Principle 20.5 and mandate *how to* design the system to the System Developer.

20.4.4.2 Implementing the Architectural Model-Based Approach Architectural Model-Based specifications are well suited for both *precedented* and *unprecedented* system applications. For example, HUMAN SYSTEMS often require Command & Control (C2) and situational assessments (Figures 24.12 and 26.6) for steering, propulsion, power, and communications (Chapter 26). However, when the specification developers specify the primary architectural components, they may limit the potential for new and innovative architectures that may be able to exploit new technologies and methods such as combining two traditional components into one.

Additionally, there is always the risk that specifying requirements for a SUBSYSTEM may *unintentionally* constrain the design of an interfacing SUBSYSTEM (Principle 14.3). As a result, cost and schedule impacts may be incurred. Simply state the architectural component capabilities as performance-based entities.

20.4.4.3 Engineering Modeling Paradigms When you model systems, be aware of the influences of Engineering paradigms on your models. Consider the following example.

Example 20.2

Specification of Physical Entities

Traditionally, a commercial aircraft flight crew physically consisted of a Pilot, a Copilot, and a Navigator. However, technology advances, in combination with the need to reduce operating costs, lead to the elimination of the Flight Engineer position. Thus, the paradigm shifted to integrating the Flight Engineer Role into the Pilot and Copilot (1st Officer) Roles and designing navigation systems to support those two roles.

Using the example above, mentally contrast a specification developed using the Architectural Model-Based Approach that would have specified the *traditional* three cockpit crewmember paradigm *versus* the Performance-Based Specification Approach that simply leaves the identification of PERSONNEL Element roles to emerge from the System Developer's Command and Control (C2) analysis of the SPS.

Recognize that the preceding point is not a fallacy of the Architectural Model-Based Approach. The Architectural-Based Approach was simply a method the SEs chose to erroneously instantiate their mental paradigms, which eventually had to shift.

20.4.4.4 Summary Recognize that the Performance-Based Approach and Architectural Model-Based Approaches are not *mutually exclusive*. For example, if a System Acquirer employs the Performance-Based Specification Approach, the System Developer can employ the Architectural Model-Based Approach to derive an SPS that is submitted in response to an RFP. During the System Development Phase, the System Developer may choose to employ the Performance-Based Approach to acquire a SUBSYSTEM from a subcontractor, and so forth.

20.5 SPECIAL TOPICS

Specification development often has nuances that you should learn to recognize. Let's address some these as special topics.

20.5.1 Developing versus Writing Specifications

People often have the *misperception* that you "write" specification requirements as if they were in a Language Writing Composition 101 course; that is what Engineers are educated to do. They provide no supporting analyses or rationale as to *how* they arrived at the capabilities and levels of performance specified in the document (Principle 19.14). This is a paradigm that *exemplifies* why many contracts and System Development efforts "get off to the wrong start" at Contract Award.

Specification development is a multi-level, concurrent, system analysis, conceptual design effort that requires *decision artifacts* and *supporting rationale*. Some Enterprises and SEs say, "We do this. If Person A is writing a specification and has a To Be determined (TBD) that needs to be replaced with a numeric value, they go to Person B who performs a 'back of the envelope' analysis, goes into the lab and runs a simulation, and returns with a performance value. Problem solved!" … That is *not* what we are referring to here.

The point is when you *develop* a specification, you employ a structured analytical approach such as the Capability Architecture-Based approach. Additionally, you need to know the decision criteria and artifacts that *compelled* you to make *informed* decisions about a requirement statement's Measure of Performance (MOP) value. How did you:

1. Identify and derive requirements.
2. *Avoid* missing, misplaced, conflicting, contradictory, or duplicated specification requirements (Principle 20.3)?
3. Derive each requirement's Measure of Performance (MOP)?
4. Rationalize requirement allocations to lower level specifications?

In contrast, people refer to "writing" specification requirements, which is often ad hoc. Typically, the Feature-Based Approach is their in-grained method of choice. Recognize the differences between developing versus writing requirements

20.5.2 Delineating Specifications and Statements of Work (SOW)

Principle 20.6

Specification Scope Principle

Specifications specify *what* SYSTEMS OR ENTITIES are expected to accomplish and *how well*, not work tasks to be performed and accomplished by the project.

Principle 20.7

Project Work Scope Principle

A project contract's work scope—Contract Statement of Work (CSOW) or Project Charter—specifies work tasks to be accomplished, not SYSTEMS OR ENTITY specification requirements.

People often have problems delineating a specification from SOW. As evidence of this confusion, you will typically see SOW language such as activities, tasks, and work products written into specifications. So, *what is the difference between the two types of documents*?

The SOW is an Acquirer's contract document that specifies the *work activities to be performed,* and their work products to be delivered by the System Developer, subcontractor, or vendor to fulfill the Terms and Conditions (Ts&Cs) of the contract. In contrast, the specification specifies and bounds the capabilities, characteristics, and their associated levels of performance required of the deliverable system, product, or service.

How can you avoid these situations or lessen their impact? Develop a rapport and work with the customers long before procurement actions are released to gain their confidence in your enterprise. Professionally and tactfully provide constructive feedback that illustrates how everyone benefits from ensuring the contents of contract documents comply with best practices. There is a reason *best practices* exist—learn to recognize the difference!!

20.6 SPECIFICATION REVIEWS

When a specification reaches a level of maturity, conduct a *specification review* with stakeholders, peers, Subject Matter Experts (SMEs), and others. Specification reviews provide a valuable opportunity to assess how well the specification specifies the essential capabilities, OPERATING ENVIRONMENT conditions, design and construction constraints, consistency, and compatibility with interfacing entities.

20.6.1 Specification Review Challenges

Unfortunately, specification reviews often have limited utility and value. *Why*? Specification reviews often turn into *grammar* correction exercises with little or no substantive review. Reviewers sit around a table or participate in a video or audio conference:

1. Having limited time to review the document prior to the review due to their own schedules.
2. Trying to *avoid critiquing* the specification developer's work —after all they spent a lot of time analyzing the SYSTEM or ENTITY, so they must be an expert. *Avoid* this line of thinking and submit substantive review comments in a professional and tactful manner.
3. Avoiding the wrath of someone else if they do comment - a management issue.

There are several methods of conducting specification reviews. As we explore these methods, think about how you can turn your specification reviews into value-added efforts that produce a specification that will have *minimal* changes or problems.

20.6.2 Specification Review Approaches

Some individuals and Enterprises conduct specification reviews on a line-by-line basis that consumes hours. If you must use this approach, exploit word processor capabilities to append line numbers in the margins of the document. Since documents have *sections* that may restart numbering, use *continuous* line numbering throughout the document; avoid section line numbering restarts.

An alternative approach is to distribute the document electronically and request that comments be inserted as changes in a different color font and returned. Review the comments and incorporate those that are valid and applicable; follow up with the stakeholder for comments that require clarification.

Conduct specification reviews with all Stakeholders. People who casually read specifications in a linear "front to back" or sectional manner for proper grammar and text usage typically *overlook* or *fail* to assimilate and recognize the conditions illustrated in Figure 20.3.

Based on the collection of review comments, identify any *major* issues and conduct a follow-up review with the stakeholders. The purpose of this review is to simply *resolve* major issues. Based on the results of the review, update the document for the next review or approval for baselining and release. This approach avoids consuming hours of Stakeholder time discussing grammar on a line-by-line basis unless it is required to avoid ambiguities.

Whichever approach is best for the review, document the list of stakeholder reviewers, attendees, approved changes, decisions, and action items via *conference minutes* and distribute to participants and others that have been pre-defined and approved by the project.

Principle 20.8 **Specification Baseline Principle**

Every specification should be formally reviewed, approved, baselined and placed under formal Configuration Management Change Management, released, and communicated for technical decision making.

You may ask: *When should specifications be baselined*? Always consult your contract for specific requirements. Specifications, in general, are baselined and released, as listed in Table 20.1. Figure 17.2 provides a graphical view of the example specification release sequences by specification type.

As stated earlier in Principle 14.2, lower level specification and design decisions are dependent on stability of higher level specifications. Therefore, it is important to baseline as soon as the specifications requirements have a level of maturity. Conversely, if you baseline a specification too soon, changes must be approved through formal Configuration Control processes, which can be costly and time consuming. The decision is ultimately determined by the Project Engineer, Lead Systems Engineer (LSE), Configuration Manager, and responsible SDT or PDT.

Author's Note 20.2 **Specification Requirements Updates**

Once a specification is baselined and released, requirements *should not* be added to a specification without formal baseline change management approval, and budgetary resources are negotiated and provided. Remember, each requirement costs money to implement via hardware or software. At the SPS level, requirements changes should be managed as contract modifications. Within the System Developer's Enterprise, any additional requirements should include commensurate cost and schedule modification considerations.

20.6.3 Substantive Specification Reviews

What are *success criteria* for a substantive specification review? In general, here are some examples of criteria for assessing specification review success:

1. Does the specification specify and bound a *solution space* that will satisfy all or a portion of the User's *problem space*? Is this a *solution* or a *symptom* of a *problem*?

2. Are the requirements *essential* in terms of being *necessary* and *sufficient* without imposing unreasonable constraints on the solution set options?
3. Does the specification specify mission-based operational performance characteristics?
4. Are the requirements *complete* and *consistent* with other project specifications?
5. Can each requirement be *verified*? If so, how?
6. Can the SYSTEM/ENTITY be developed within project constraints – cost, schedule, and technology – with a *level of risk* that is *acceptable* to the System Acquirer/User and System Developer?

These criteria require more than a one-time assessment at the first DRAFT specification review. They should serve as a focal point and be assessed at *every* specification review.

20.7 CHAPTER SUMMARY

Our discussion of specification development practices identified the key challenges, issues, and methods related to developing system specifications. As the final part of the concept, we introduced a basic discussion for preparing specifications. The methodology provides an operational approach to specification development based on how the User intends to use the system.

20.8 CHAPTER EXERCISES

20.8.1 Level 1: Chapter Knowledge Exercises

1. What are some common approaches to developing specifications?

2. What is the "feature-based" specification development approach?

3. What is the "performance-based" specification development approach?

4. What is the "reuse" specification development approach?

5. What is the "capability architecture-based" specification development approach?

6. Compare and contrast the four specification development approaches.

7. How are specification reviews performed?

8. How do you know when a specification is ready for release?

20.8.2 Level 2: Chapter Knowledge Application Exercises

Refer to www.wiley.com/go/systemengineeringanalysis2e

20.9 REFERENCE

DI-IPSC-81431A (2000), *Data Item Description (DID): System/Subsystem Specification (SSS)*, Washington, DC: FAA. Retrieved on 2/26/15 from https://sowgen.faa.gov/dids/DI-IPSC-81431A.doc.

21

REQUIREMENTS DERIVATION, ALLOCATION, FLOW DOWN, AND TRACEABILITY

Development of the System Performance Specification (SPS) typically represents only a small portion of a large, complex system's hierarchy of requirements. The challenge for SEs is: *How do we establish the lower level specifications that enable System Developers to create* PART *Level designs*?

Chapter 21 introduces the requirements derivation, allocation, flow down, and traceability practices that enable us to create the specification-based hierarchy of system requirements. Our discussions delineate the terms requirements derivation, allocation, and flow down. We explore how requirements are derived and identify a methodology for deriving requirements and how to apply the methodology.

21.1 DEFINITIONS OF KEY TERMS

- **Child Requirement**—A *contextual* term identifying a requirement that has been derived from and contributes to accomplishment of an abstract *parent* requirement at the next higher level.
- **Leaf Level Requirement**—The lowest-level derived requirement for a specified capability that can be allocated and flowed down to the next lower level entity.
- **Contributory Performance Effector**—A key parameter characterized by a TPP and MOP that impacts overall system, product, or service outcomes and performance.
- **Parent Requirement**—A *contextual* term identifying an abstract requirement that requires been elaboration or refinement into two or more "child" requirements at the next lower level.
- **Requirements Allocation**—The process of assigning accountability for implementation of a requirement to at least one or more lower level contributing System Elements such as EQUIPMENT, PERSONNEL, and FACILITIES or entities within those elements.
- **Requirements Derivation**—The process of *decomposing* or *refining* an abstract *parent* capability requirement into lower level performance-based *child* requirements. *Conditional* accomplishment of the set of derived *children* – sibling - requirements constitutes satisfactory accomplishment of the "parent" capability requirement.
- **Requirements Flow Down**—The process of *delegating accountability* for implementation of a requirement or portion thereof to a lower level of abstraction, such as SYSTEM to PRODUCT or PRODUCT to SUBSYSTEM and so forth.
- **Requirements Testing**—The process of evaluating the content and quality of a requirement statement relative to a predefined set of requirements development criteria. The purpose is to determine if a requirement is *specific, measurable, actionable, realistic, testable, verifiable, and traceable* (Principle 22.3); *complete, consistent, and unambiguous*.
- **Requirements Traceability**—The establishment of bottom-up specification requirements linkages for a SYSTEM/ENTITY from its lowest to the highest levels of

System Engineering Analysis, Design, and Development: Concepts, Principles, and Practices, Second Edition. Charles S. Wasson.

Companion Website: www.wiley.com/go/systemengineeringanalysis2e

abstraction to higher level User *source* or *originating* requirements.

- **Requirements Verification**—An activity performed by a System Developer or System Acquirer using verification methods such as Inspection, Examination, Analysis, Demonstration, or Test to prove *compliance* to a specification requirement based on a review of *objective evidence* such as work products, physical entities, measurement data, and Quality Records (QRs).
- **Requirements Validation**—An activity performed by Stakeholders – System Acquirer, User, End User, and System Developers - to ensure that specification requirements *completely, accurately,* and *precisely* specify and bound a *solution space* that will resolve all or a portion of a User's *intended* operational needs – *problem space* (Figure 4.7).
- **Traceability**—"The ability to identify the relationship between various artifacts of the development process, i.e., the lineage of requirements, the relationship between a design decision and the affected requirements and design features, the assignment of requirements to design features, the relationship of test results to the original source of requirements" (Naval SE Guide, 2004, p. 171).

21.2 APPROACH TO THIS CHAPTER

Chapter 21 begins with an introductory overview of requirements derivation, allocation, flow down, and traceability. Our discussion introduces key terms and concepts required to for understanding the practice.

Based on these concepts, we explore how requirements are derived using UC Thread Analysis. Whereas people often think that requirements derivation is an *ad hoc* brainstorming session of what potential derived requirements might be, we illustrate how UCs and UC Thread Analysis provide an analytical method that enables us to actually derive requirements. Then, advance to a more complex case illustrating how a SYSTEM Level capability requirement representing a UC is used to derived lower level requirements that can be allocated across SUBSYSTEM specification boundaries.

These discussions describe the analytical method and mechanisms for deriving allocating, and flowing down requirements. The reality is: the User Operational Requirements Documents (ORDs) and Capability Development Documents (CDDs) or System Acquirer System Requirements Documents (SRDs) should be based on Measures of Effectiveness (MOEs), Measures of Suitability (MOSs), and their respective Measures of Performance (MOPs) (Chapter 5). Therefore, our next step is to address how SPS MOE, MOS, and MOP requirements are derived, allocated, and flowed down to specifications ate lower levels of abstraction.

We conclude the chapter with a discussion of Special Topics in requirements derivation, allocation, flow down, and traceability.

21.3 INTRODUCTION TO REQUIREMENTS DERIVATION, ALLOCATION FLOWDOWN, & TRACEABILITY

Requirements derivation is commonly used in the vocabularies of Engineers. Yet, few seem to actually understand what is required to derive the requirements. If you ask an Engineer how they derived a requirement, they respond nonchalantly "I just did it" as if there is some obscure, magical formula that only they understand. Objective evidence of this point is illustrated in specifications developed using the multi-level Feature-Based Approach discussed in Chapter 20 and exemplified in the text for Figure 20.3. So, what is *requirements derivation*?

Requirements derivation is the process of refining or elaborating an abstract "parent" capability requirement statement into a set of lower level, "child" capability requirements. The process is similar to an optical prism breaking down the spectral bands of white light as shown in Figure 21.1. Completion of the derivation process results in a multi-level hierarchy of requirements for a SYSTEM/ENTITY as shown in Figure 21.2.

The requirements hierarchy simply illustrates a mesh of requirements. Although not shown in the figure, the mesh is partitioned into the SPS at the highest level and supported by lower level EDSs at lower levels of abstraction. The hierarchy of specifications represents the Specification Tree (Figure 19.2) for the SYSTEM.

On inspection, the requirements hierarchy is more than a derivation of requirements. Figure 21.2 provides an illustration. Observe the *contextual* "parent-child" relationships within the hierarchy. Contextually, a "child" requirement at one level of the hierarchy serves as a "parent" requirement for the next level of decomposition.

The process of deriving an abstract *parent* requirement into two or more *child* requirements means that if you satisfy the *child(ren)* requirements, then you also satisfy the higher level parent requirement. Two key points:

- Bottom-up: Each *child* requirement and its *siblings conditionally* contribute to accomplishment of a *parent* requirement. That is, *child(ren)* requirements express HOW a *parent* requirement is accomplished.
- Top-down: Each *contextual parent* requirement conditionally expresses WHY its *child(ren)* requirements exist.

Observe usage of the term *conditional* in the Bottom-Up point above. *Conditional* accomplishment of "child"

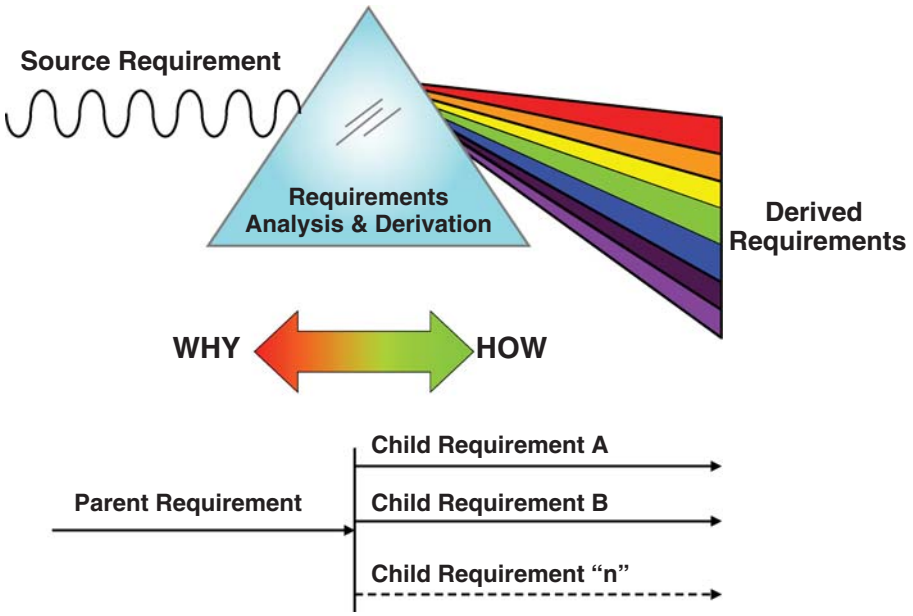


Figure 21.1 Optical Prism Illustration Symbolizing Requirements Derivation

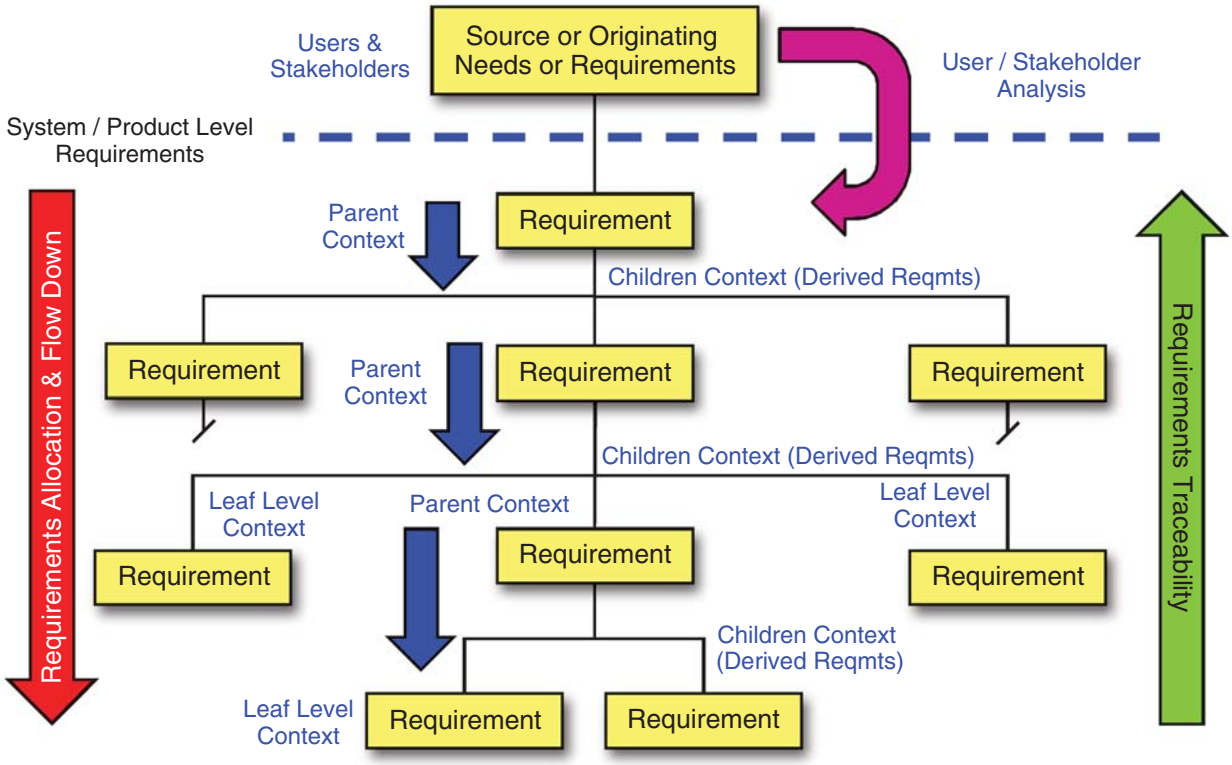


Figure 21.2 Requirements Hierarchy Illustrating Requirements Ancestry and Semantics

requirements may occur in several ways: *separately, in a sequence,* or *simultaneously*.

This discussion leads to a key question: *how do you know when to stop deriving requirements*?

Requirements at higher levels of the hierarchy are fine for expressing what has to be accomplished. However, their *abstractness* is *insufficient* for assigning it to an architectural Entity at the next level of abstraction. For example, "The automobile weight shall not exceed 2,000 pounds" represents abstract requirement. *Of the 2,000 pounds, how much weight should be allocated to the frame, body, engine, and so forth*? We have to partition the 2,000 pound weight requirement – *problem space* – into *solution spaces* that can be allocated – assigned - directly to the frame, body, engine, and so forth. We refer to that lowest level requirement that can be allocated to an architectural ENTITY as a "Leaf" level requirement.

Summarizing Figure 21.2, User *source* or *originating* requirements are decomposed or refined into various levels based on *contextual parent-child* relationships. The refinement of abstract requirements continues top-down until a Leaf Level requirement can be *allocated* – assigned – directly to a Logical Architecture capability or Physical ENTITY. Then, each *child* requirement in the top-down hierarchy is *linked* – traced - to its higher level *parent* requirement until the User's *source* or *originating* requirements are reached.

On inspection, this process may seem to be relatively simple. The reason is it represents the essence of Functional Analysis, which we described as *relevant* but *insufficient* for performing requirements derivation. The deficiency in Functional Analysis is the focus on deriving *children* "functions" – the easy part - instead of capabilities. As a result, the really difficult part – establishing the MOP values for each capability requirement – is postponed until the System Design is underway. *Informed* decision-making requires extensive analysis, trade studies, Modeling & Simulation (M&S), and prototyping. So, *how is this accomplished*?

Observe the left hand side of Figure 21.3. Systems Engineering publicity refers to "flowing SPS requirements to lower levels of specifications." Unfortunately, that is the *misperception* Enterprise executives, managers, and Engineers often have. The *untold* story "behind the scenes" occurs in the SE Process Model and Decision Support Process.

SEs leverage the power - *iterative* and *recursive* characteristics (Chapter 14) – of the Wasson SE Process Model to (1)

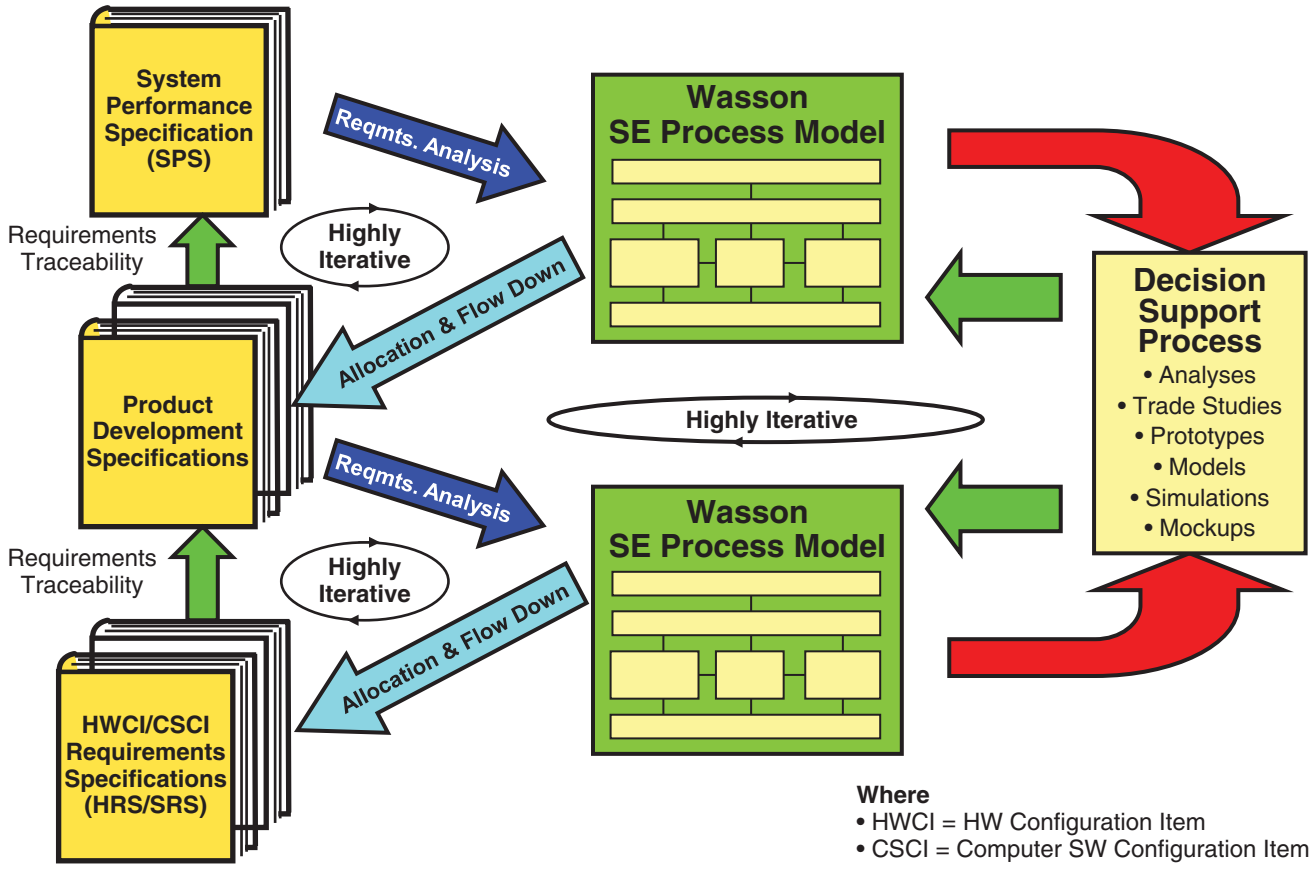


Figure 21.3 Application of the Wasson SE Process to Multi-level Requirements Analysis, Derivation, Allocation, and Flow Down Process

understand the User's *problem* and *solution spaces* and (2) develop the Requirements Domain Solution. Development of the Requirements Domain Solution is based on requirements derivation, allocation, and flow down practices using the System Architecture. Where in-depth analysis, trade studies, Modeling & Simulation (M&S), and prototyping are required, the System/Product Development Team (SDT/PDT) assigns a task to the Decision Support Process.

Decision Support provides *quantitative* analyses, data, and *recommendations* for deriving, allocating, and flowing down requirements from the SPS to PRODUCT Development Specifications. Subsequently, those requirements are allocated and flowed down to Hardware Configuration Item (HWCI) Requirements Specifications (HRSs) and/or Computer Software Configuration Item (CSCI) Requirements Specifications (SRSs). Requirements management tools based on relational databases are used to facilitate the requirements allocation, flow down process, requirements traceability, assignment of verification methods, and support requirements metrics tracking.

21.3.1 Overarching Constraints for Requirements Derivation

Principle 21.1

User Wants, Needs, Can Afford, and Willingness to Pay Principle

When deriving SYSTEM/ENTITY requirements, understand:

- What the User *wants.*
- What the User *needs.*
- What the User can *afford.*
- What the User is *willing to pay.*

You can *academically* derive an infinite number of *solution space* requirements to resolve all or a portion of a *problem space*. However, before you *naively* proceed down that path, there are two realities that constrain the process: (1) the need to identify only *essential* requirements (Principle 19.7) that are *necessary* and *sufficient* balanced with (2) *what* the User wants, needs, can afford, and is willing to pay. Figure 21.4 illustrates this last point. As you read the sections ahead, maintain awareness of these realities.

21.3.1.1 How Many Requirements are "Essential?"

SEs struggle with the rhetorical question: *How many requirements do you need to specify a* SYSTEM/ENTITY*?* There are no specific guidelines or rules—only disciplined and seasoned experience. Specification quality is not measured by the quantity of requirements. Instead, the question should be: What are the *essential* mission or system capabilities that need to be specified without *over-specifying* or *under-specifying* (Principle 19.7)?

Frightening? Yes! But think about it! Every layer of requirements adds restrictions, complexity, costs, and schedule risks that limit the System Developer's *flexibility* and *options to innovate* and achieve lower costs, schedule implementation, and risk.

How many requirements should an ideal specification have? Hypothetically, the answer could be one requirement; however, a one-requirement specification has limited utility. We can state that a properly prepared specification is one that has the *minimal* number of *essential* requirements. This allows the System Developer the flexibility to select the optimal solution that can be *verified* and *validated* to meet the User's operational capability and performance needs.

How do we emerge with this idyllic specification? The answer may be found in performance specifications. Performance specifications enable SEs to treat a system like a

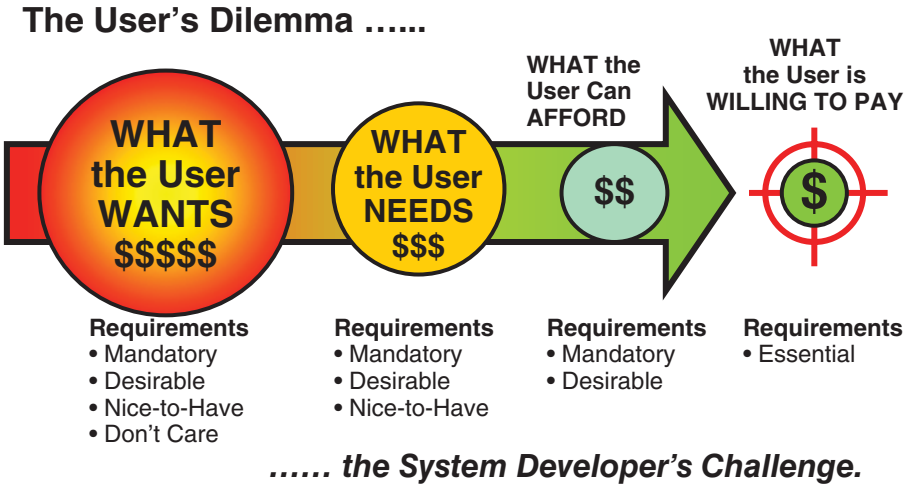


Figure 21.4 Overarching Requirements Derivation Constraint—What the User Needs, Wants, Can Afford, and Willingness to Pay

"black box" with inputs and outputs (Figure 20.4). The intent is to scope and bound the behavior of a system, product, or service relative to scenarios, and conditions in its prescribed operating environment. This is accomplished via *Measures of Effectiveness* (MOEs) and *Measures of Suitability* (MOSs), each with supporting *Measures of Performance* (MOPs) (Figures 20.1 and 20.2) that form the basis for deriving System capabilities that populate specifications. By treating the SYSTEM/ENTITY characteristics as a "performance envelope," we have avoided specifying *how* the system is to be designed.

There is one problem, however, with bounding the SYSTEM/ENTITY at the "performance envelope" level. If you develop the performance specification in a manner that allows too much flexibility, the results may be *unacceptable*, especially to the System Acquirer. This leaves the System Developer with the challenge or interpretation of determining what *essential* capabilities the SYSTEM/ENTITY is to provide and *how well* each capability is to be performed. This has *positive* and *negative* implications, depending on your role as a System Acquirer or System Developer/Subcontractor. Since all capabilities may not be considered to be of *equal value* and *priority* by the User, especially in terms of constrained budgets, the User's priority list may be different from the System Developer's priority list.

Dichotomy of System Acquirer and System Developer Competing Views

Author's Note 21.1 Remember that the User is interested in *optimizing* capabilities and performance while *minimizing* technical, cost, and schedule development costs and risk.

The System Developer, depending on type of contract, is also interested in *minimizing* technical, cost, and schedule risk while *optimizing* profitability. Thus, unless both parties are willing to work toward an *optimal* solution that represents a "win–win" for both parties, conflicts can arise regarding these priority viewpoints and Enterprise objectives may conflict.

How do we solve this dilemma? The System Acquirer may be confronted with having to specify additional requirements that more *explicitly* identify the key capabilities, levels of performance, as well as priorities. The challenge then becomes: *At what level does the System Acquirer stop specifying requirements?* Ultimately, the System Acquirer could potentially *over-specify* or *under-specify* the system, depending on budget factors.

So, *what is the optimal number of requirements?* There are no easy answers. Conceptually, however, we may be able to describe the answer in our next topic, the Optimal System Requirements Concept.

21.3.2 Optimal System Requirements Concept

Anecdotal data suggest that a notional profile can be constructed to express the *optimal* quantity of specification requirements. To illustrate this notional concept, consider the graphic shown in Figure 21.5.

The figure consists of a graphical profile with three curved line segments. For discussion purposes, we will identify the area under each segment as a zone. Beginning at the intersection of the X-axis and Y-axis, every legitimate requirement that is identified at the top hierarchical levels *increases* the *sufficiency* of system definition toward a theoretical Optimal Level. We refer to Zone 1 as the Zone of Increasing System Definition Sufficiency. This theoretical point is the point of inflection in the curve's slope.

At the point of inflection, we should have an *optimal* number of requirements. Hypothetically, the quantity of requirements should be technically *essential—necessary and sufficient*—to specify a System/Entity with the desired capabilities and levels of performance. At this level, the requirements are *minimally sufficient* to specify and bound the User's *intended* operational needs.

Beyond the point of inflection lies the Zone 2—Increasing Requirements Restrictions. As the slope of the curve indicates, requirements can be added but at the expense of *over-specification*. Each additional requirement restricts the SE design options and may increase technical, cost, schedule, technology, and support risk.

As the quantity of requirements continues to increase to the right, you finally reach a *breakpoint* for Zone 3, which represents the region where the requirements become *too restrictive*. Thus, the requirements unduly restrict SE design options and *severely* limit feasibility of the system. Generally, when this occurs, two things can occur:

- Solution #1—The System Acquirer discovers this problem during the draft proposal stage and decides to remove some requirements due to prohibitive technical, cost, schedule, technology, and support costs and risks based on potential Offeror comments.
- Solution #2—If the System Acquirer offers no relief, your option may be to no-bid, which has implications.

The challenge described here is not unique to the System Acquirer. The same problem challenges the System Developer, not just at the SYSTEM Level but for specifications at lower levels. Every requirement at every level has: (1) a cost to implement and (2) a cost to verify in addition to their schedule and risk implications. This impacts the SE Design, the Component Procurement and Development, and the SITE Processes of the System Development Phase (Figure 12.2).

Does this concept answer the question: *What is the optimal number of specification requirements*? No. However, it illustrates some hypothetical conditions—the points

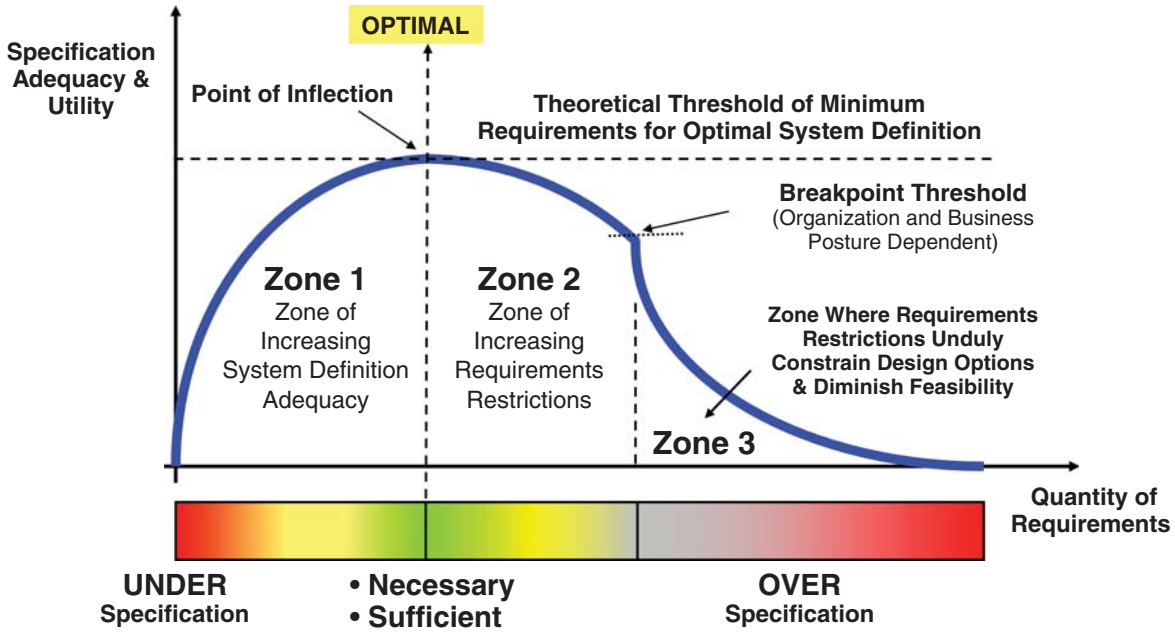


Figure 21.5 Optimal Essential Requirements Concept

of inflection and breakpoints—that should be your focal points. The bottom line is that specification development requires more insight than simply writing ad hoc requirements into a specification outline. You need to think about *what* you require and the potential *ramifications* on the System Developer.

21.3.3 Understanding How Specification Requirements Are Derived

User operational needs created by SYSTEM/ENTITY capability gaps, product obsolescence, external threats, Critical Operational and/or Technical Issues (COIs/CTIs), and so forth represent abstract *problem spaces*, especially for moderate to large complex systems, are too difficult to solve directly. We can, however, solve it by *partitioning* – refining - it into smaller, more manageable pieces with less risk (Principle 4.16 and Figure 4.7). The partitioning of the abstract *problem space* into finer levels of detail enable us to bound and specify *solution space(s)* requirements at various levels of abstraction. The end result is a hierarchy of derived requirements (Figure 21.2) that are traceable back to the User's *source* or *originating* requirements.

21.3.4 Understanding the System Requirements Hierarchy

SPS requirements generally begin with high-level, User mission-oriented requirements statements. These high-level statements are then elaborated into successively lower level requirements that explicitly clarify, specify, and bound the intent of the higher-level requirements. In effect, a hierarchical structure of requirements emerges similar to Figure 20.3. Observe that each of the requirements is labeled using a convention—appended digits for lower levels—that depicts its lineage and traceability to higher-level requirements. Consider the following example:

Example 21.1

Entity Numbering Convention

Parent capability requirement, RI, has three-sibling capability requirements, R11, R12, and R13 (Figure 20.3). The labeling convention simply adds a rightmost digit for each level and begins the numerical sequence with "1." Thus, you can follow the *lineage* and *descendants* for each derived requirement by decoding the numerical sequence of digits. The lineage of requirement, R31223, is based on the following decompositional "chain": R3 → R31 → R312 → R3122 → R31223.

Author's Note 21.2

MOE and MOS Traceability

The hierarchy shown in Figure 20.3 is best described as *ideal,* meaning, all requirements are properly aligned to higher-level requirements. For discussion purposes, we

will restrict the simplicity of this diagram to *generic* requirements. Requirements are derived from and must be traceable to MOEs, MOSs, and Measures of Performance (MOPs) (Chapter 5) at the System and mission objectives levels.

The System Mission represents the highest level requirement in Figures 11.1 and 20.3. The mission is scoped, bounded, and described by three high-level capability requirements, R1, R2, and R3. When a single requirement, such as R12, R32, and R33, effectively ends the chain indicating the requirement is *directly assignable* to a single entity such as a Subsystem or Assembly and no further derivation is required, the term Leaf level requirement is used.

21.4 REQUIREMENTS DERIVATION METHODS

We can illustrate requirements derivation with a simple construct such as the one shown in Figure 21.6. For this illustration, let's assume that the SUBSYSTEM A's Development Specification includes UC-based Capability Requirement A_11. The requirement bounds and specifies a capability to display a sensor measurement in millivolts. Based on a UC analysis, three sequential actions need to be accomplished: (1) collect sensor data, (2) process the data, and (3) display the results. Based on these actions, we translate each UC steps into Capability Requirements A_111, A_112, and A_113.

Observe that the context of this illustration provides insights for deriving capability requirements within a specification. Now, suppose that we have a SYSTEM Level requirement that must be implemented across several SUBSYSTEMS, each with its own EDS.

Within a specification, the requirements derivation simply elaborates each abstraction requirement to lower levels. For example, consider abstract Requirement A_111 Collect Input Data. What are the sources of "input data?" Therefore, since Requirement A_111 represents a capability, its UC Thread might consist of: (1) Collect Sensor #1 Data, (2), Collect Sensor #2 Data, and so forth. Each of these tasks becomes derived Requirements A_1111, A_1112, and so on. Each of these requirements would then be allocated to SUBSYSTEM X that monitors each of these sensors.

Let's expand the complexity of the previous example to the SYSTEM Level SPS.

21.5 REQUIREMENTS DERIVATION AND ALLOCATION ACROSS ENTITY BOUNDARIES

Assume that we have a SYSTEM Level capability requirement, SYS_11, as shown in Figure 21.7. The SDT analyzes the requirement, performs a UC analysis, and derives *child* requirements A_11 and B_11 within the SPS. At this point, the SDT has not allocated Requirements A_11 and B_11 to SUBSYSTEMS A and B. We designate each for now as simply A_11 and B_11 to delineate the uniqueness of each requirement.

UC analysis reveals that Requirements A_11 and B_11 require further Derivation. This brings up an interesting question: *How do you know that requirements within a specification should be derived further*? The answer is: when

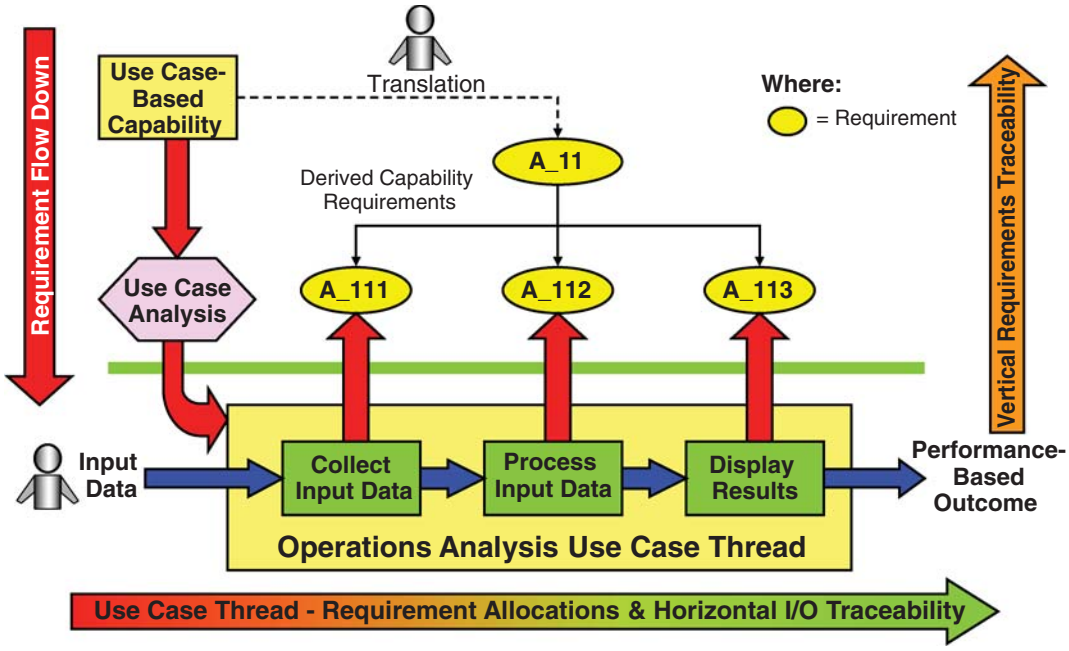


Figure 21.6 Intra-specification Use Case-Based Requirements Derivation and Traceability

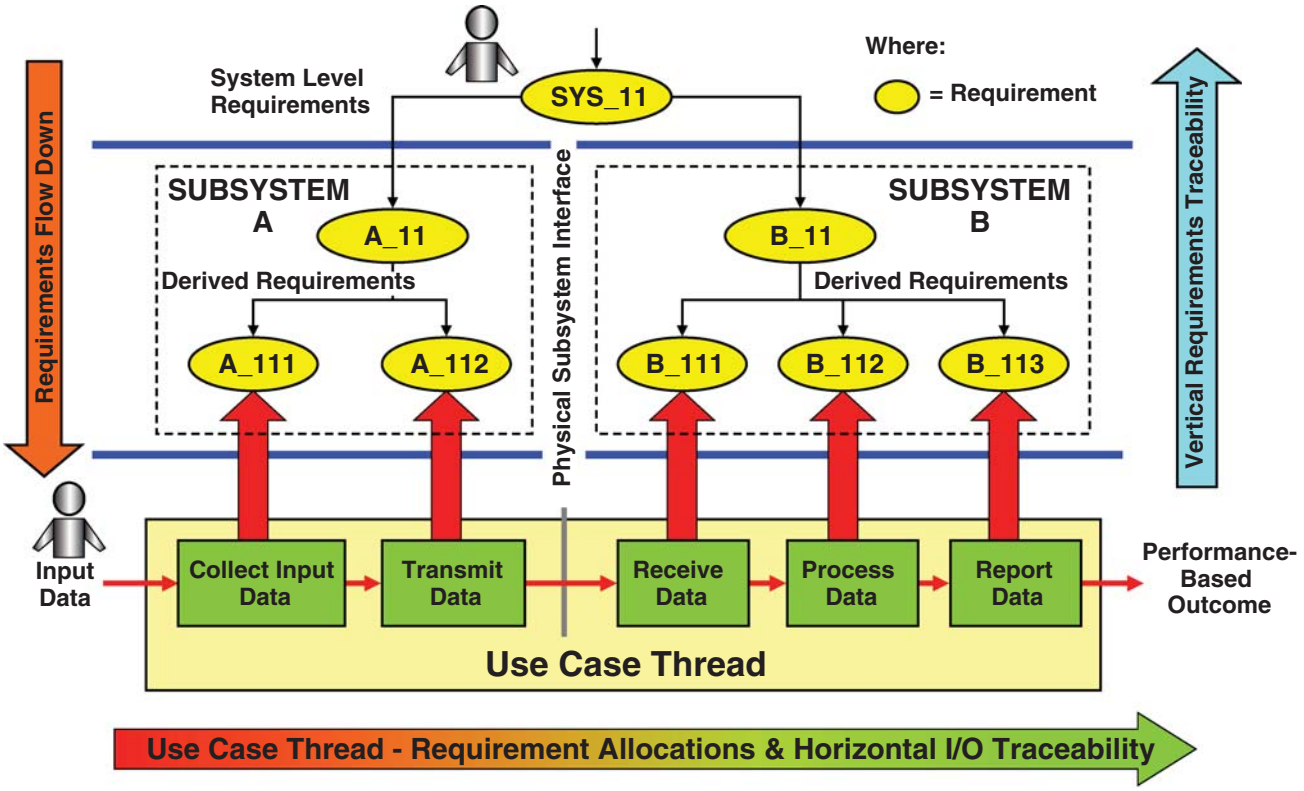


Figure 21.7 Inter-specification UC-Based Requirements Derivation, Allocation, and Traceability

they are too abstract to implement. This brings us to a key principle.

Principle 21.2

Requirements Allocation and Flow Down Principle

SOI specification requirements should be derived to a level that allows direct assignment, allocation, and flow down of each requirement to a specific ENTITY within the SOI's architecture for implementation.

Without becoming immersed into whether Requirement A_11 should be allocated to SUBSYSTEMS A or B, we construct a UC thread at the bottom of the figure. Based on UC thread analysis, SEs derive:

- *Child* Requirements A_111 and A_112 from *Parent* Requirement A_11.
- *Child* Requirements B_111, B_112, and B_113 from *Parent* Requirement B_11.

Since these requirements can be physically implemented, we allocate them to SUBSYSTEMS A and B in the System Architecture and flow them down to their respective SUBSYSTEMS A and B EDSs.

Given these allocations, *what does this mean in terms of what the* SYSTEM *Requirement SYS_11 is expected to accomplish*? It means that if you want to accomplish the performance-based outcome specified by SYSTEM Level Requirement, SYS_11, SUBSYSTEM A must satisfy Requirement A_11 and provide the result (outcome) to SUBSYSTEM B, which must satisfy Requirement B_11 and produce the final result (outcome).

How do SUBSYSTEMS *A and B physically accomplish Requirements A_11 and B_11, respectively*?

- SUBSYSTEM A complies with and satisfies Requirement A_11 by physically implementing Requirements A_111 and A_112.
- SUBSYSTEM B complies with and satisfies Requirement B_11 by physically implementing Requirements B_111, B_112, and B_113.

Consider the following example.

Example 21.2

UC Thread: Capability Requirements Derivation

Suppose that we have a simple SYSTEM that has a remote sensor (SUBSYSTEM A). The

sensor collects input data and transmits it to a central site (SUBSYSTEM B) as illustrated in Figure 21.7.

- The Central Site (SUBSYSTEM B) receives and processes the data to produce a report. Therefore, the Remote Sensor Development Specification must state that it: (1) Collects Data (Requirement A_111) and (2) Transmits Data to the Central Site (Requirement A_112).
- The Central Site Development Specification includes the remainder of the requirements UC Thread to: (1) Receive Data (Requirement B_111), (2) Process Data (Requirement B_112), and (3) Report Data (Requirement B_113).

21.5.1 Measures of Effectiveness (MOEs)

Chapter 20 provided an example specification outline. Brief discussions of the outline noted that Section 3.2 Operating Performance Characteristics (Table 20.1) specified MOES that represented User measures of success. Observe the term *operational* as in mission operations. To illustrate how MOEs are documented consider the example shown in Figure 20.2.

The figure illustrates an airline mission of transporting passengers from one city to another. Observe that we divide the mission into phases with each phase characterized by one or more MOEs. Each MOE should be supported by two or more MOPs. For example, when an aircraft lands, *turnaround time* is critical for "on-time" performance. So, we establish a Turnaround MOE with a t = XX minutes performance value.

To achieve the MOE, we need to establish MOPs that contribute to achievement of the MOE. This requires an MOP for cleaning the aircraft, loading catering supplies, refueling the aircraft, performing maintenance inspections, loading the passengers, and so forth. All of these factors contribute to the overall System Architecture and design. Observe what has occurred here. We have *intermixed* the Aircraft System, which is specified by the SPS, with the Terminal Gate Loading System including the jetway bridge, which are ENABLING SYSTEMS to the Aircraft System – MISSION SYSTEM.

Based on our MOE analysis and derivation of MOPs, we populate the SPS Section 3.2 Operating Performance Characteristics using the Mission Phases of Operation Structure shown in Figure 20.2. Figure 21.8 provides an example illustration of how Section 3.2 might be populated.

21.5.2 Requirements Derivation Components

The preceding discussions approach the concept of derivation of requirements as "objects." Requirements are characterized by two attributes: (1) *what* outcome is to be accomplished and (2) *how well* (Principle 19.6). Chapters 2 and 14 addressed the fallacies of Functional Analysis due to its focus on what, which is the easy part. The difficult part of requirements derivation is deciding how to allocate performance to derived requirements.

Grady (2006, pp. 59–60) identifies three basic methods for allocating performance: (1) *equivalence*, (2) *apportionment*, and (3) *synthesis*. Since these topics are a key focus on performance budgets and safety margins, we will defer our discussion to Chapter 31 (Section 31.2.2).

This concludes our discussion of Requirements Derivation. Now, let's shift our focus to Requirements Allocation Methods that support Requirements Derivation.

21.6 REQUIREMENTS ALLOCATION

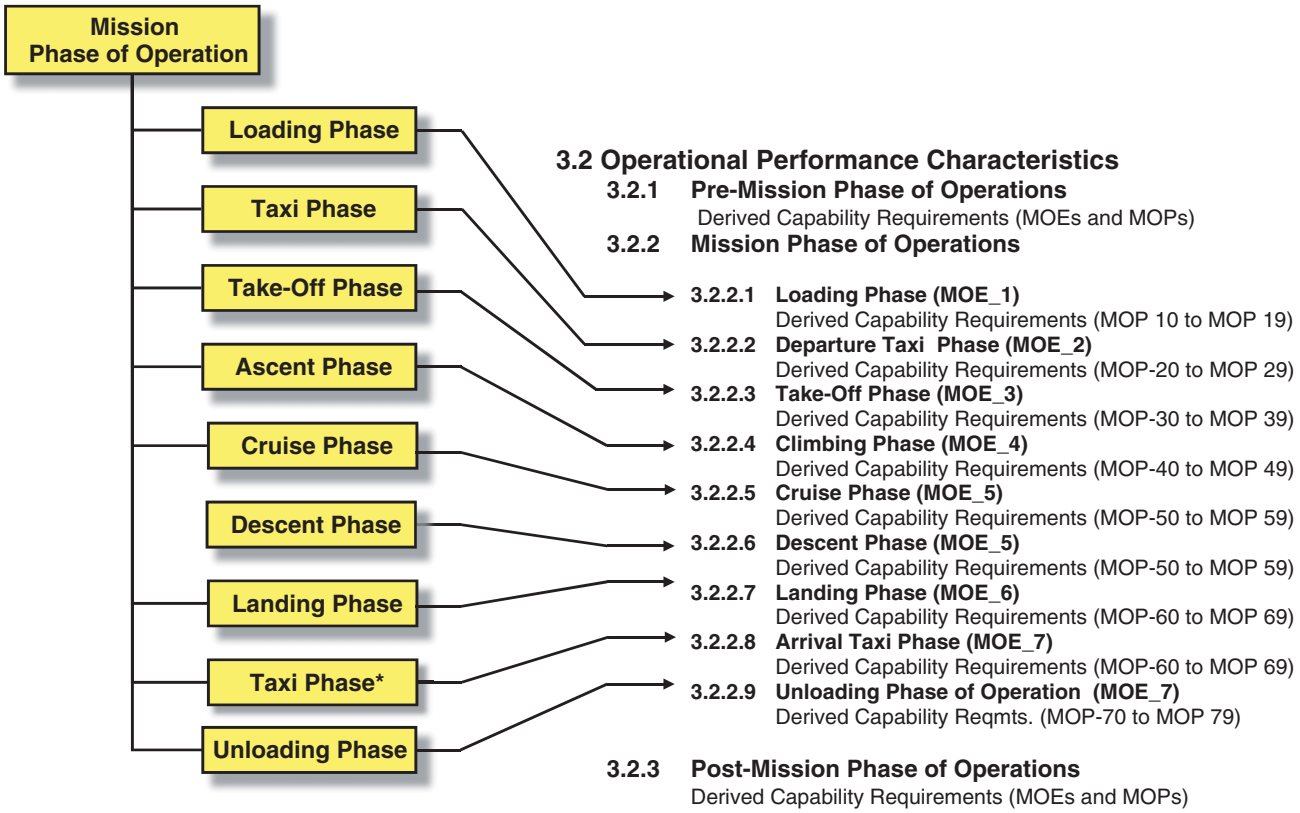
As the Requirements Derivation process progresses, the SDT or PDTs apply the SE Process to begin investigating preliminary architectures for the Operations, Behavioral, and Physical Domain Solutions as shown in Figure 14.1. Unfortunately, many people believe requirements are established. Then, the design is developed. *That is false*!! The reality is the System Design Solution *evolves simultaneously* as the requirements are allocated and flowed down to each level.

21.6.1 Requirements Allocation Concept

Figure 21.9 provides a general overview of the Requirements Allocation process. The SPS specifies capability requirements that we can use to create an Input/Output (I/O) Model similar to Figure 20.4. The Wasson SE Process Model (Figure 14.1) is used to analyze the requirements and formulate and select a logical capabilities architecture. The outcome is the Architectural Representation shown in the upper right consisting of Entities A1–A4. A matrix is created to map – allocate – the SPS capability requirements to the Architectural Elements, A1–A4. At the completion of the exercise, requirements in each A1–A4 column are flowed down to Product A1–A4 Development Specifications.

21.6.2 Multi-Level Requirements Allocation

Now, let's expand the concept to a broader view of the SYSTEM as shown in Figure 21.10. Here each set of requirements are analyzed via the Wasson SE Process Model, logical architectures are formulated and selected, capability requirements are allocated to elements within the architectures. The *iterative* and *recursive* characteristics of repeated applications of the Wasson SE Process Model should be self-evident in the figure. As the Requirements Allocation process continues down the levels of abstraction, if Critical Operational and Technical Issues (COIs/CTIs) are identified, they may require elevation back to higher levels for resolution.



Note: *If capabilities are identical to the Taxi Phase that precedes Take-Off, combine into one paragraph.*

Figure 21.8 Example Illustrating How Figure 20.2 Mission Life Cycle MOEs Provide the Basis for Specification Section 3.2 Operational Performance Characteristics (Table 20.1).

At each level of decomposition and specification development, each requirement has:

1. A value-based priority to the User.
2. A cost to implement.
3. A level of implementation risk.
4. Schedule constraints for implementation.

Therefore, you need to develop *draft* multi-level specifications internal to the program. These specifications enable technical decision-makers to understand the feasibility, ramifications, and risks to lower level entities if we allocate and flow down these capabilities. Whereas higher level specifications tend to be *abstract*, lower level specifications represent where items are physically developed or procured from external vendors. You need this feedback to mature higher-level specifications over time (Principle 14.2). This illustrates *why* specification development is a multi-level top-down, bottom-up, left-right, and right-left process. When you complete specification development, requirements at any level should be *traceable* to the next higher level and subsequently to the User's *source* or *originating* procurement requirements—SOO or SRD.

Given an understanding of Requirements Allocation Methods, let's shift to Requirements Traceability.

21.7 REQUIREMENTS TRACEABILITY

Vertical and Horizontal Requirements Traceability Principle

Principle 21.3 Specification requirements allocations and flow down ensure *vertical* requirements traceability to source or originating requirements. Use Case threads integrate those allocations into *horizontal* capability threads that produce SYSTEM Level performance-based outcomes traceable to specification requirements.

As a result of the requirement derivation, allocation, and flow down, we can say that the *child* requirements are "traceable" to a higher level parent requirement (Principle 19.13). Observe the phrase "traceable" to a higher level parent requirement." We can *only* say that the vertical chain of requirements are traceable back to the User's *source* or *originating* requirements.

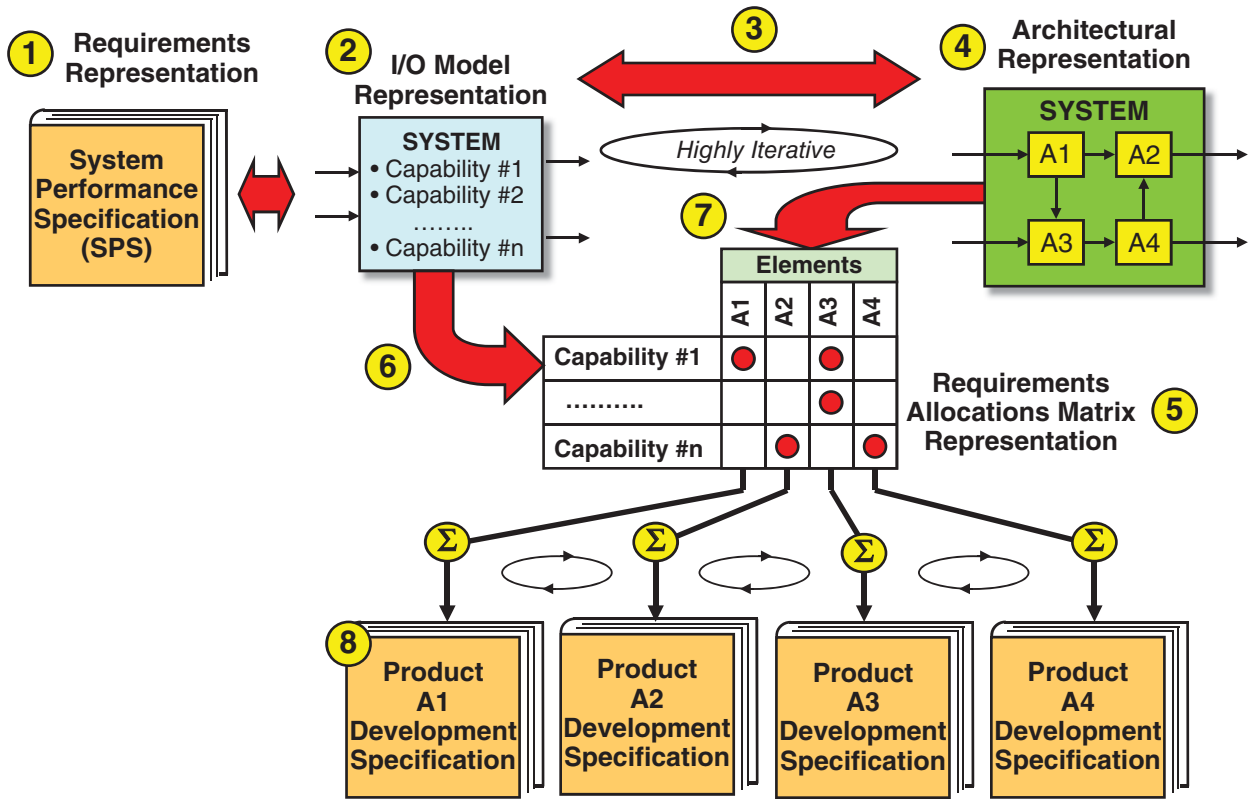


Figure 21.9 Basic Architectural Model-Based Specification Requirements Allocation and Flow Down Method

One of the challenges in System Development is having separate PDTs deliver their ENTITIES for System Integration, Test, & Evaluation (SITE) only to discover that they are *incompatible* or *inoperable* (Principle 10.3). Using Figure 21.7 as an example, let's assume SUBSYSTEM A's PDT and SUBSYSTEM B's PDT *failed* to communicate with each other to ensure that SUBSYSTEM A's EDS stated that Transmit Data (A_112) should be sent to SUBSYSTEM B. If it did, does SUBSYSTEM B's EDS have an interface to Receive Data (B_111). In this context, this brings us to the need for *auditable* vertical and horizontal traceability. Vertical traceability is implicit in the specifications outline. However, horizontal traceability requires manually auditing the respective specifications to ensure continuity across the specification and PDT interface boundaries.

21.7.1 System Level Architecture Perspective: Integrated UC Thread Capabilities

Our discussion to this point has focused on *discrete* SYSTEM or ENTITY Level UC capability threads. However, a System Architecture is not a loosely coupled conglomeration of independent sets of components to accommodate each UC thread. Instead, it is an *integrated framework* of capabilities that enables a User to Command and Control (C2) SYSTEM/ENTITY responses unique to a specific UC. For example, a SYSTEM/ENTITY's Modes and States (Chapter 7) enable the User to C2 sets of capabilities to accomplish specific UC thread performance-based outcomes. Figure 21.11 provides an example.

Observe that the System Architecture consists of SUBSYSTEMS A–E, each configured to provide performance -based capability behaviors and outcomes for a given set of input stimuli, excitations, or cues. For example, a SYSTEM Level UC is implemented via the chain of capabilities provided by SUBSYSTEMS A, B, and D. For this UC, capabilities provided by SUBSYSTEMS C and E are not required. A different System UC thread may require a different routing path through the System Architecture via SUBSYSTEMS A → C → E → D.

21.7.2 Enterprise Requirements Traceability Maturation Levels

Anecdotal evidence suggests that Engineers and Enterprises tend to learn the concept of requirements traceability via a three-stage process. The degree of evolution depends on the size, complexity, and risk of systems being developed and the Enterprise's and individual's desire and willingness to improve performance. Here are descriptions of the evolution of organizations from Stage 1 to Stage 3.

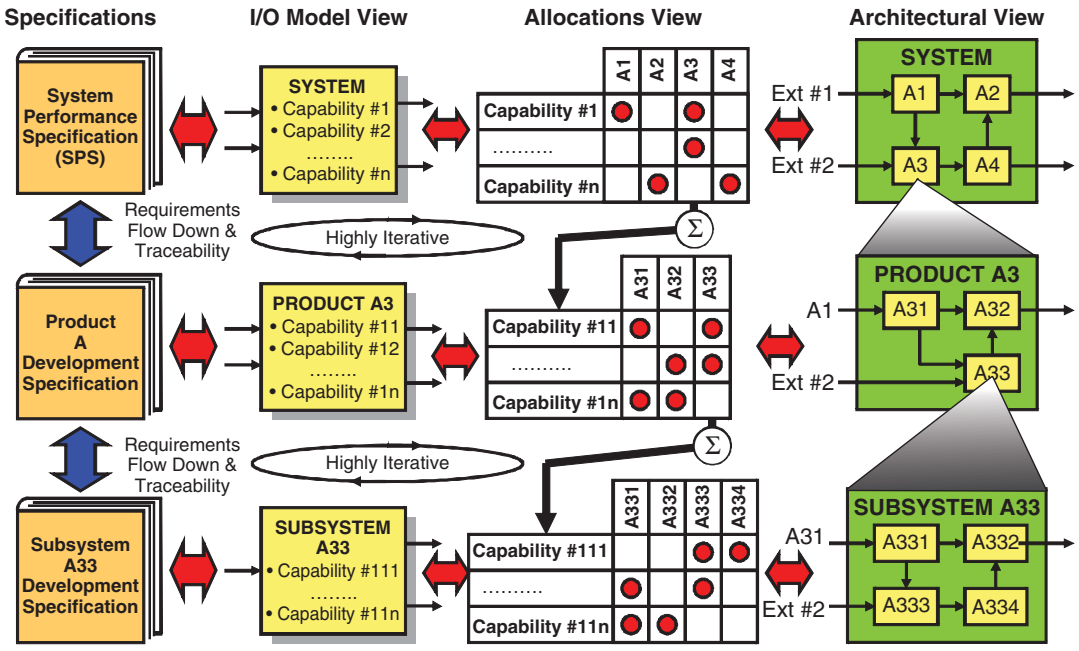


Figure 21.10 Multi-Level Requirements Allocation and Flow Down Process

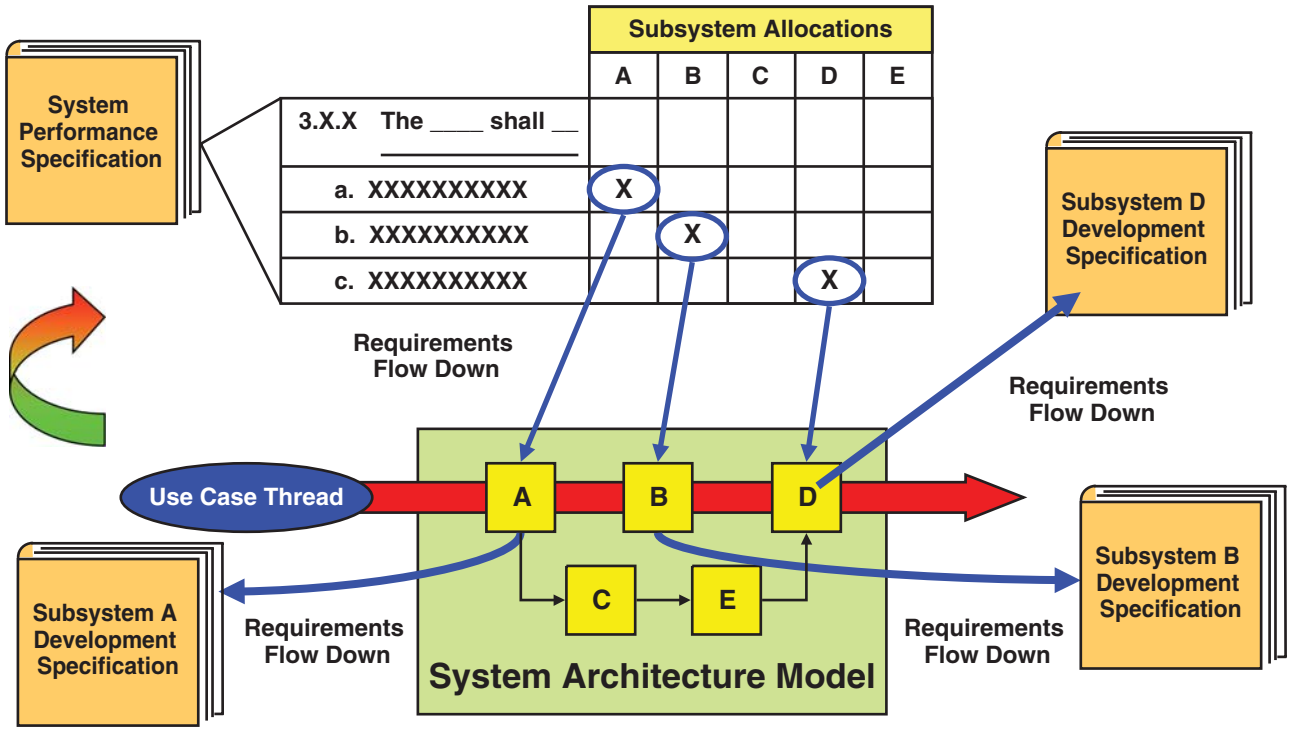


Figure 21.11 Fully Integrated Architectural Framework Illustrating Command and Control (C2) of a Use Case Thread Weaving Through Specific Components.

SE Capability **Stage 1**	General recognition by Enterprise technical, functional, and executive management of the need to organize and structure requirements using a standard specification outline. Over a period of time, the individual and Enterprise lessons learned, such as *missing*, *misplaced*, *conflicting*, *contradictory*, or *duplicated* requirements, indicate that a new level of requirements implementation capability is required. Stage 1 maturity is indicated by continuous process improvement and recognition of the need for advancement to Stage 2.
SE Capability **Stage 2**	At Stage 2, the Enterprise understands and appreciates that multi-level SYSTEM requirements involve vertical requirements traceability via lineage with high-level requirements. Again, over time and many painful experiences in implementing specification requirements, there is a recognition and appreciation that vertical requirements traceability is one-dimensional. Stage 2 maturity is indicated by continuous process improvement and recognition of the need for advancement to Stage 3.
SE Capability **Stage 3**	At Stage 3, the Enterprise understands and appreciates the need for two-dimensional requirements traceability: (1) *vertically* through the requirements hierarchy to the User's *source* or *originating* requirements and (2) *horizontally* via the UC thread traceability.

21.8 TECHNICAL PERFORMANCE MEASURES (TPMs)

Technical Performance Measures (TPMs) Principle

Principle 21.4 Periodically report the status, progress, and maturity of Technical Performance Measures (TPMs) that trace and contribute to achievement of higher level User KPPs and SPS MOEs, or MOSs.

Our discussions of TPPs, MOPs, KPPs, MOEs, and MOSs identify performance effectors. The challenge question is: *how does a project manager, project engineer, or Leads SE (LSE) know that the technical project is progressing toward achieving compliance with its specifications and does not pose any risk to the project delivering on schedule and within cost without major risks*? We do this with

TPMs that represent *graphical* plots that are tracked and updated weekly, biweekly, or monthly and presented at major technical reviews to the System Acquirer.

In general, the TPM plots depict past performance, current status, and predicted performance for selected MOPs at various levels of abstraction that impact overall KPPs and MOEs. TPMs are not restricted to KPPs and MOEs; however, if these are of critical importance to the User and System Acquirer, they demand attention. Initially, TPMs represent analytical estimates. As models and simulations of system performance are validated during the System Development Phase, the *estimates* attain a level of confidence inferring *predicted performance*. As physical components are developed, tested, and verified, the analytical *predictions* become *actual* performance status. For example, you can *estimate* the weight of the SYSTEM, SUBSYSTEM, ASSEMBLY, or SUBASSEMBLY. Once it is produced and verified, *actual* performance replaces *estimates*.

21.8.1 Plotting TPMs

Once the KPPs and MOEs for a system, product, or service are established, the next step is to track and plot TPMs. This is accomplished in a number of ways such as numerical value reporting. The best practice is to plot the values graphically as shown in Figure 21.12.

TPMs should be:

1. Tracked on a weekly basis by those accountable for implementation—such as by development teams or PDTs.
2. Reported at least monthly.
3. Reviewed at each major technical review.

Regarding the last point, note how a PDT at each major technical review reported past performance, current progress, and predicted performance in Figure 21.12:

- Here's the level of performance we *projected* by analysis for TPM XYZ.
- Here's the level of performance we have *achieved* to date.
- Here's the level of performance we *expect to achieve* by the next review.
- Here's the *corrective action plan* for how we expect to align today's level of performance with *projected* performance and *acceptable* control limits – risk mitigation.

Note also how the TPM values consist of analytical projections through CDR with a level of risk associated with achievement. Between CDR and TRR, the physical components become available for system integration and test. As such, actual values are measured and become the basis for final TPM tracking.

Recognize that system component performance varies due to mass properties, manufacturing tolerances, and so forth. This is why the nominal TPM value represents the mean of a Normal Distribution. The challenge for SEs is, for a given TPM value: *What are the allowable upper and lower control limits for a given entity—product or subsystem—that*

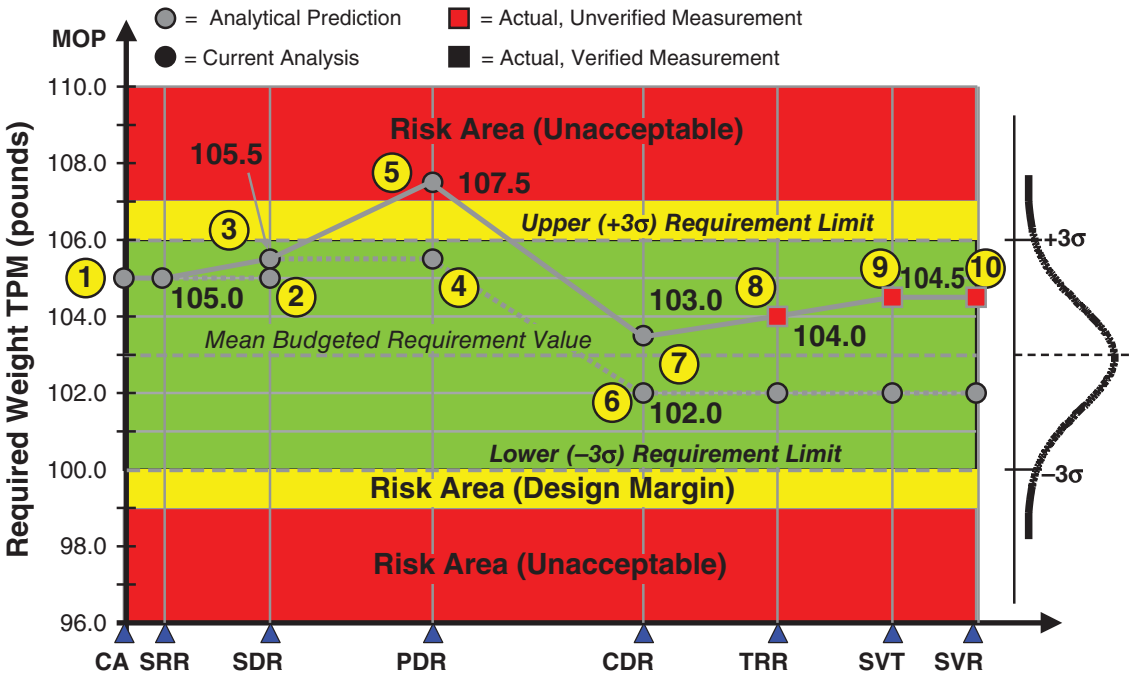


Figure 21.12 Plotting Technical Performance Measurements (TPMs)

do not diminish overall system performance? Based on the results of this decision, $+3\sigma$ and -3σ or other applicable thresholds are established for triggering multi-level risk items and mitigation plans. Within each $+3\sigma$ and -3σ threshold, design safety margins may be established. To facilitate viewing, the design margins zones should be Yellow; anything outside the $+3\sigma$ and -3σ thresholds should be Red.

TPMs along the development path pose potential risks, especially if the SDT or PDT is unable to achieve the analytical projections. When this occurs, each TPM should be required to have *risk thresholds* that *trigger risk items* for mitigation and tracking. If the risk becomes significant, Risk Item Mitigation Plans may be required to provide a *risk profile* for reducing the risk over time and bring it in line with specification requirements. The project's Risk Management Plan should definitize this process and threshold criteria for triggering risk item for tracking and mitigation plans.

21.8.2 Selecting TPMs

TPMs can easily become a very time-consuming activity, especially for reporting purposes. Obviously, every MOP in a specification cannot and does not need to be formally tracked other than through normal requirement ownership and professional accountability. Select those of critical importance such as KPPs and their MOEs and MOSs. How is this accomplished?

When confronted with a System Acquirer requirement to track TPMs, the natural tendency of project technical management is to tell everyone accountable for a specification to go select four (4) to six (6) TPMs, create some charts that plot the history and future predictions for a TPM, and so forth. Often, these TPMs are selected without any form of coordination and integration in terms of meeting overall KPPs or MOEs.

The best starting point begins with SPS KPPs and their MOEs. Once these are selected, follow the requirements allocation threads to lower level specification TPPs and their MOPs that contribute to the achievement of KPPs and MOEs. To illustrate this point, consider the example of a notional vehicle Fuel Efficiency KPP shown in Figure 5.9.

If our SE mission is to design a vehicle that achieves the Fuel Efficiency KPP, we establish an MOE with a performance threshold of XX miles per gallon (MPG) as shown earlier in Figure 5.9. Then, perform an analysis to identify all of the contributory performance effectors that impact the MOE. Examples include: Vehicle Aerodynamics MOP, Vehicle Engine Efficiency MOP, Fuel Quality MOP, Road Conditions MOP, Weather Conditions MOP, etc. Obviously, we cannot control *variable* performance effectors such as Weather Conditions, Road Conditions, Driver Skills, Fuel Quality, etc. even though the vehicle has to be capable of performing in a range of operating conditions. We can, however, identify TPMs for Vehicle Aerodynamics, Vehicle Engine Efficiency, etc. in lower level vehicle body and engine specifications that can be tracked.

As a final point, SE requires "visualizing" the big picture. When TPMs are presented, people in the audience have to *mentally* process the *connectivity* between MOEs and MOPs, which detracts from the presentation. Do yourself, your team, and the audience a favor, create an overview graphic for each TPM such as the one shown in Figure 5.8. Members of the audience need to understand the TPM's performance contribution to a higher level MOE or KPP. for the system, product, or service.

21.8.3 TPM Challenges

TPM tracking involves several challenges. Let's explore a few.

21.8.3.1 TPM Challenge 1: Bureaucratic Metrics Tracking TPMs have two levels of criticality:

- First, TPMs serve as visual indicators to alert an SDT or PDTs to potential technical trouble that requires corrective action. PDT Leads need to clearly understand this. Otherwise, the effort is perceived as nothing but bureaucratic metrics tracking performed to impress the System Acquirer.
- Second, as a Lead SE, Project Engineer, or Technical Director, you need early indicators that provide a level of confidence that the system is going to perform as specified and designed. If not, you need to know in advance to allow time to take corrective action.

21.8.3.2 TPM Challenge 2: Select TPMs Wisely SDTs and PDTs often randomly select a few TPMs to satisfy the metrics tracking task. Select the most difficult, potential showstopper TPMs linkable to higher level KPPs. This ensures that proper attention is focused on addressing the most challenging TPPs to meet. The tendency of most SDTs and PDTs is to select the easiest TPPs to track – looks impressive to management and the customer - only to discover major challenges later trying to meet the critical TPPs linkable to SYSTEM Level KPPs.

21.8.3.3 TPM Challenge 3: Withholding Actual TPM Data If a TPM becomes a risk item, PDT Leads often continue to plot *analytical* predictions to *avoid* the realities of actual data that pose political and technical risks. Don't play games! Your professional obligation as an SE is to report factual, existing data. If you have political problems, deal with the matter in other ways. Remember that if you are not meeting performance levels now and you choose to ignore potentially major problems, wait until you attempt to

explain those problems to technical, program, and executive management when the cost to correct is very expensive.

Conversely, project management should recognize *objective reporting* as constructive to the project. *Avoid* "punishing the messenger." Focus on *constructive* technical solutions that lead to success. Remember that individual and team success leads to project management success.

21.8.3.4 TPM Challenge 4: TPM "Shelf Life" TPMs, especially lower level ones, have a shelf life. During design and early in the SITE phase, low-level TPMs are critical indicators of performance that may affect overall system performance and effectiveness. Once a TPM requirement has been verified, the necessity to track a TPM may be pointless, unless something fails within the system. Remember that lower level MOPs were derived from higher level KPPs and MOEs. Once a lower level MOP has been *verified* and its Configuration Item (CI) or items have been integrated into the next higher level, which has its own set of TPMs, you should not continue to track the MOP. There may be exceptions; *use informed Engineering judgment*.

21.8.3.5 TPM Challenge 5: TPM Reporting As an Engineering best practice, you should track TPMs whether required by contract or not. Tracking TPMs takes work and time, which translates into cost and schedule impacts. Choose TPMs wisely for formal reporting; track others for internal assessment. Some System Acquirer organizations are more mature than others in treating the *openness* with admiration.

A Word of Caution 21.1

Make sure your *candor* and *openness* about TPMs does not become a basis for those with ulterior motives and agendas to transform minor TPM excursions - that are a normal part of everyday System Development - into major technical and project issues. These issues sometimes require System Acquirer and System Developer executive management level intervention and resolution.

21.9 CHAPTER SUMMARY

Our discussion of the requirements derivation, allocation, flow down, and traceability practices described a methodology for deriving multi-level specification requirements. We considered the importance of *vertical* requirements traceability based on requirements allocations and flow down. We also discussed the importance of *horizontal* requirements traceability based on UC continuity thread checks, especially related to Enterprise capabilities.

As an SE, you need to understand how Mission Effectiveness is determined and decomposed into MOEs and MOSs, each bounded by MOPs that are documented in various requirements documents. We addressed how TPMs are used to track *planned* versus *actual* MOP TPM values and the importance of thresholds for triggering risk items and associated Risk Mitigation Plan (RMP).

In closing, people will often state that TPMs are too laborious to perform and track. If this is the case, then how can you and your organization deliver a system, product, or service that meets the User's operational needs and know that you can deliver on schedule in compliance with the specification requirements and within cost without performing the practice?

21.10 CHAPTER EXERCISES

21.10.1 Level 1: Chapter Knowledge Exercises

1. What is requirements derivation?
2. How do you derive requirements?
3. What is requirements allocation?
4. How do you allocate requirements?
5. What is requirements flow down?
6. How do you flow down requirements?
7. What is requirements traceability?
8. How do you trace requirements? To where?
9. What is the relationship between MOEs, MOSs, MOPs, and TPMs?
10. Who is responsible for MOEs, MOSs, MOPs, and TPMs?
11. How do you track and control MOEs, MOSs, MOPs, and TPMs?
12. What is the relationship between TPMs and risk items?
13. When do TPMs trigger risk items for proactive risk mitigation?

21.10.2 Level 2: Chapter Knowledge Application Exercises

Refer to www.wiley.com/go/systemengineeringanalysis2e

21.11 REFERENCES

Grady, Jeffrey O. (2006), *System Requirements Analysis*, Burlington, MA: Academic Press.

Naval SE Guide (2004), *Naval Systems Engineering Guide*, Washington, DC: U.S. Navy. Retrieved on 1/20/14 from http://www.dtic.mil/dtic/tr/fulltext/u2/a527494.pdf.

22

REQUIREMENTS STATEMENT DEVELOPMENT

Once a *solution space* is identified, the challenge for SEs is being able to *accurately* and *precisely* bound and specify a capability via the System Performance Specification (SPS) or ENTITY DEVELOPMENT SPECIFICATION (EDS). Our discussion of specification development practices provided an approach for identifying types of specification requirements.

Chapter 22 focuses on the formulation and development of specification requirements statements. Our discussions introduce and explore various methods for preparing requirements statements. We identify a syntactic structure to facilitate definition of a requirement. The discussions emphasize the need to define the *substantive content* of a requirement before focusing on the grammar. To facilitate preparation of requirements, we include requirements development guidelines and discuss how to analytically "test" a requirement.

We conclude the chapter with a discussion of one of the challenges of specification requirements development: knowing when a set of essential requirements is *necessary* and *sufficient*. We explore the need to minimize the quantity of requirements and avoid *over-specification* and *under-specification*.

22.1 DEFINITION OF KEY TERMS

- **Analysis** – Refer to definition in Chapter 13 Definition of Key Terms.
- **Demonstration** – Refer to definition in Chapter 13 Definition of Key Terms.
- **Essential Requirement**—Refer to the definition in Chapter 19 Definitions of Key Terms.
- **Inspection**—Refer to definition in Chapter 13 Definition of Key Terms.
- **Legacy System**—Refer to Chapter 16 Definition of Key Terms.
- **Primitive Requirement Statement**—"A form of a requirement statement that has no punctuation or formal sentence structure and is not written in a formal specification style" (FAA SE, 2006, Vol. 3,p. B-9).
- **Sibling Requirement**—Two or more requirements at the same level traceable to a common "parent" requirement at the next higher level.
- **Specification Language**—"(1) a language, often a machine-processible combination of natural and formal language, used to express the requirements, design, behavior, or other characteristics of a system or component" (SEVOCAB, 2014, p. 298) (Source: ISO/IEC/IEEE 24765:2010. Copyright © 2012 ISO/IEC. Used by permission).
- **Test**—Refer to definition in Chapter 13 Definition of Key Terms.

22.2 APPROACH TO THIS CHAPTER

Our approach to Chapter 22 can best be described by the Requirement Development Decision Process illustrated in Figure 22.1. In general, the process consists of the following sequence of decisions:

System Engineering Analysis, Design, and Development: Concepts, Principles, and Practices, Second Edition. Charles S. Wasson.

Companion Website: www.wiley.com/go/systemengineeringanalysis2e

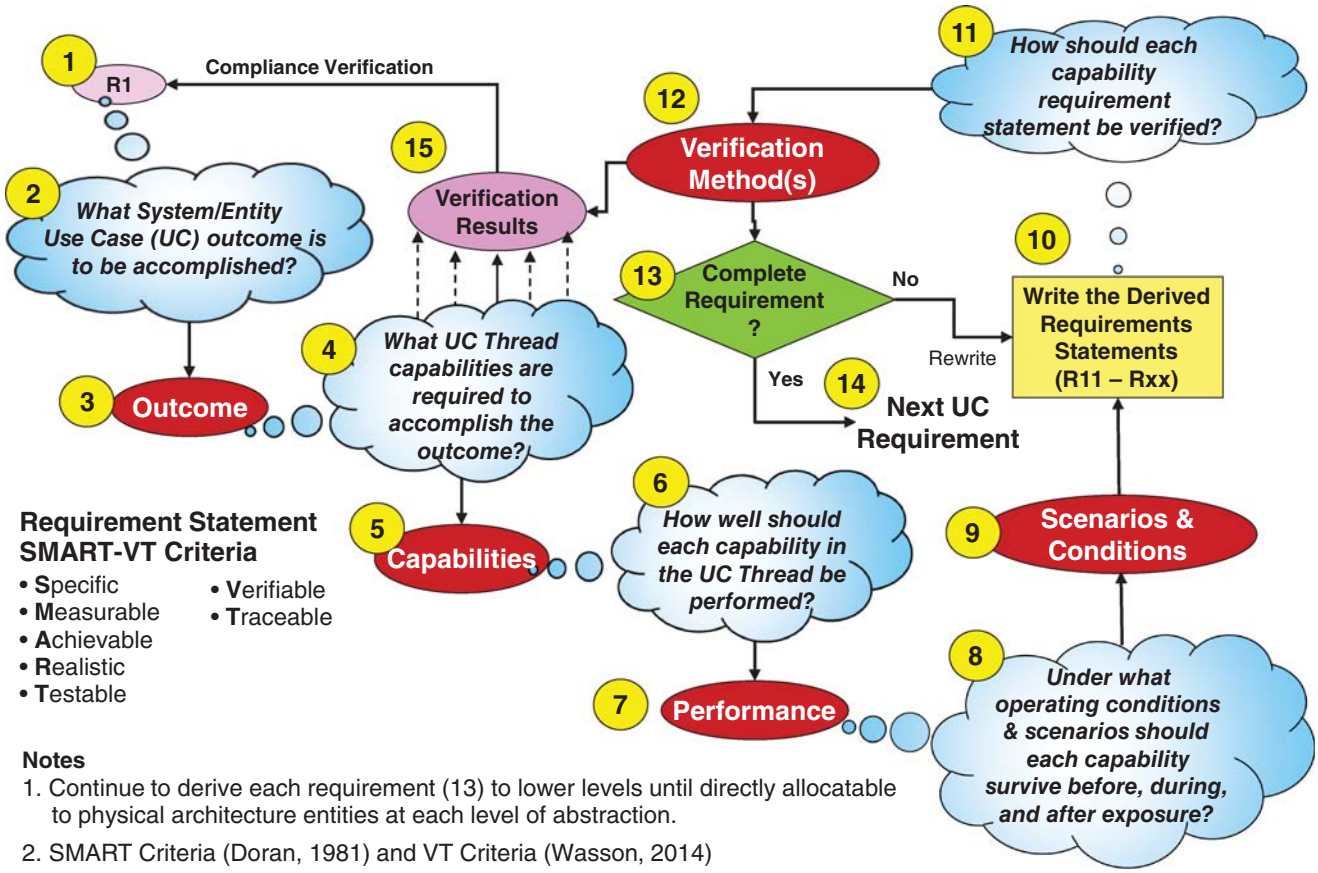


Figure 22.1 Requirement Development Decision Process

1. What System/Entity Use Case (UC) outcome is to be accomplished?
2. What UC Thread (Figure 21.7) capabilities are required to accomplish the outcome?
3. *How well* should each capability in the UC Thread be performed?
4. Under what *operating conditions & scenarios* should each capability survive before, during, and after exposure?
5. How should each capability requirement statement be verified?

The questions provide the infrastructure for our discussions. Let's begin with an Introduction to Requirements Statement Development.

22.3 INTRODUCTION TO REQUIREMENTS STATEMENT DEVELOPMENT

Engineers and Analysts often say that a specification has a "good" set of requirements. The *colloquial* usage of the term "good" makes SEs cringe. *"Good" for whom relative to what "fitness-for-use" standard(s)?* The System Acquirer? The User? The System Developer? If good requirements reside in the minds of each Stakeholder, what makes a *"bad" requirement bad? Does "bad" mean poorly stated, open to interpretation, unachievable, immeasurable, unverifiable, and untestable?* So, *what are well-defined requirements*?

Well-defined requirements statements specify only *essential* requirements that:

1. Are articulated in a brief, concise, and understandable language with low risk of multiple interpretations by two or more independent readers with comparable disciplinary skills.
2. Represent *one and only one* Use Case (UC) and Scenario capability-based (Chapters 5).
3. Are unique within the SPS or EDS without duplication or conflicts within the SYSTEM (Figure 20.3).
4. Specify performance to be achieved *before, during,* and *after* exposure to a *prescribed* set of OPERATING ENVIRONMENT conditions.
5. Adherence to the Doran (1981)-Wasson (2014) SMART-VT Criteria (Principle 22.3).
6. Verification by one or more compliance methods.

To illustrate the importance of the preceding points, let's briefly summarize each one.

22.3.1 Brief, Concisely Worded Statements

Specifications, as part of a legally binding contract, express what the System Acquirer and System Developer understand what they (think) they agreed and committed to perform. Unfortunately, disagreements over what was understood to be developed and delivered occasionally end in conflict. The source of the conflict typically centers around the wording of requirements. So, *how do we avoid these situations*?

The solution begins with writing brief, clear, and concise statements that are easily understood and result in *one and only one* interpretation.

22.3.2 Use Case and Scenario Based Capability Requirements

Requirements statements should originate from higher level UC-based operational capabilities and scenarios. Lower level requirements should (1) elaborate specific action-based outcomes that contribute to achievement of the higher level UC as well as any *most likely* or *probable* scenarios and (2) be traceable to User *source* or *originating* requirements (Principle 19.13).

Author's Note 22.1

Specification writers often have the *misperception* that *well-defined* requirements statements must specify "capabilities." As a result, they write *every* requirement statement repeating the phrase "provide the capability …":

- The System **shall** provide the capability to … (Action A).
- The System **shall** provide the capability to … (Action B).

The irony here is the requirement above is listed in SPS or EDS Section 3.3 "Capabilities." The reader should know they are reading from the SPS or EDS "capabilities" section.

22.3.3 Uniqueness within the System

Principle 22.1

Requirement Uniqueness Principle

Each requirement statement shall be *unique* to one and only one SYSTEM or ENTITY with no other instances of overlaps, duplications, or confliction within the SYSTEM.

Well-defined requirement statements should be *unique* and occur *once and only once* within the entire set of specifications within the SYSTEM or PRODUCT's Specification Tree (Figure 19.2). Unfortunately, Engineers often create similar requirements within the SPS or SDS that are stated slightly differently and conflict with each other (Figure 20.3).

22.3.4 Bounded Operating Environment Conditions

Principle 22.2

OPERATING ENVIRONMENT Conditions Principle

Every specification should contain a *global* requirement specifying that a SYSTEM/PRODUCT comply with the requirements *before, during,* and *after* exposure to the specification's OPERATING ENVIRONMENT conditions and constraints.

Each specification must include at least one requirement statement that establishes, characterizes, and bounds the prescribed OPERATING ENVIRONMENT conditions that constrain the specified System capabilities. For example:

- The System **shall** comply with the requirements specified herein *before, during,* and *after* exposure to the conditions specified in Table 20.1 SPS or EDS Section 3.5, "OPERATING ENVIRONMENT CONDITIONS."

22.3.5 Adherence to the SMART-VT Attributes

Principle 22.3

SMART-VT Criteria Principle

Every specification requirement shall comply with the SMART-VT Criteria- Specific, Measurable, Actionable, Realistic, Testable (Doran, 1981, pp. 35–36) and Verifiable, and Traceable (Wasson, 2014).

Requirements statements must be Specific, Measurable, Actionable – Allocatable, Realistic, Testable, Verifiable and Traceable. If a requirement statement fails any one of these criterion, it should be reworked until it complies.

22.3.6 Requirement Compliance Verification

Principle 22.4

Requirement Verification Method(s) Principle

Compliance to each requirement statement **shall** be proven via one or more of the following verification methods: Inspection, Examination, Analysis, Demonstration, Test, or Similarity (if permissible) that represents the least cost and effort.

Well-defined requirements statements must be supported by one or more verification methods such as Inspection, Examination, Analysis, Demonstration, and Test (Chapter 13).

22.4 PREPARING THE REQUIREMENT STATEMENT

Specification developers often become preoccupied with attempting to state multiple requirements in a single statement. Packing numerous requirements into a single specification requirements statement is viewed as the ultimate achievement. The reality is: that is the *wrong* approach! Specification requirements development is about Systems Engineering; language Composition 101 is simply an ENABLING SYSTEM.

To develop requirements, there is a three-step process that should be followed:

- Step 1: Identify the key elements of the requirement—*who, what, when, where,* and *how well*.
- Step 2: Develop a DRAFT requirement statement.
- Step 3: Refine the requirement statement grammar such as choice and appropriateness of words, sequences, etc.

Let's detail each of these steps.

22.4.1 Step 1: Identify the Key Elements of the Requirement

Humans *unwittingly overcomplicate* the SPS and the EDS requirements statements. The formulation and development of a specification requirements statement involves three aspects: (1) content, (2) syntax, and (3) grammar.

One of the most common problems in writing requirements statements development is that people attempt to perform all three of these aspects *simultaneously*. As a result, the effort skews into *syntax* and *grammar*, neglecting the content. Grammar is simply a means to an end, not initial focal point. *How do we correct this human habit*? One solution to this problem is to develop *primitive* requirements statements, our next topic.

22.4.1.1 Create Primitive Requirements Statements

Primitive requirements statements focus exclusively on the substantive content of a requirement.

The best way to develop *primitives* is to use a tabular approach such as the one illustrated in Table 22.1. Here, we assign a unique Requirement ID SPS-136 for Capability 24 derived from a specific UC. The remainder of the columns focus attention on *substantive content* such as types of relational operators; boundary constraints, tolerances, or conditions; and notes.

Observe how Table 22.1 focuses on the primary content of the requirement and *avoids* becoming mired in grammar. Once the elements of the requirements are established, the next step is to translate the contents into a syntax statement. To illustrate how the primitives method is applied to writing an initial requirements statement, consider the example shown in Table 22.2.

Once the *primitive* contents of a requirement statement is established, the next step is to Develop the Draft Requirement Statement.

22.4.2 Step 2: Develop the Draft Requirements Statement

Specification requirements statements are more than free form "wish lists." As with any type of written language, they have a structure that represents best practices and lessons learned that enable us to avoid mistakes of others.

Principle 22.5

Requirement Structure Principle

Every requirement statement requires (1) a subject of the action, (2) "shall" to express a mandatory action for compliance, (3) an action-based verb, (4) a capability-based performance outcome to be provided, and (5) conditions for accomplishment.

Every specification requirement statement should consist of the following elements:

1. Source of the Action
2. "Shall" (Mandatory compliance)

TABLE 22.1 Table for Identifying Primitive Requirements

Reqmt. ID	Capability to be Provided	Relational Operators	Boundary Constraints, Tolerances, or Conditions	Notes
SPS-136	Capability 24	• Not to exceed • Less than or equal to • Greater than or equal to • In accordance with (IAW) • Optimal	• 50 pounds • 68 ° Fahrenheit • 25 ° Celsius • 6 g's • 6 nautical miles (NM) • 25 hertz (Hz) • 10.000 volts DC +/− 0.010 volts DC • Sea State 3 • 12 megabytes (Mb)	

TABLE 22.2 Examples of primitive requirements using the tabular approach

Reqmt. ID	Subject (Actor)	Action	Outcome to be Accomplished	Relational Operators	Level of Performance	Condition
SYS_1	System	**shall**	operate from an external power source	rated at	• 110 vac +/− 10%	Based on a TBD load.
SYS_123	System	**shall**		not exceed	• 50 pounds	
SYS_341	System	**shall**	operate	over a range of	• −10 °F to +90 °F • +20% to +100% Humidity	
SYS_426	System	**shall**	produce formatted reports	in accordance with	• Regulation 126 Section 3.2.1	On receipt of a command from the operator
SYS_525	System	**shall**	sense overload conditions that	exceed	• 10 ± 0.1 amps DC	
SYS_736	System	**shall**	respond	within	• 100 ± 10 milliseconds	To any operator command

3. Action-based Verb
4. Performance-based Outcome to be Accomplished (What)
5. Level of Performance (How Well)

In some cases, the requirement statement structure may require additional *constraints* such as:

6. Condition-Based Actions (Requirement Dependent) (When)
7. Recipient(s) of the Action (Requirement Dependent)

22.4.2.1 *Source of the Action*

Principle 22.6

Requirement Action Source Principle

Each requirement statement **shall** identify the source and stimulus, excitation, or cue that initiates a system, product, or service capability.

Well-defined requirement statements express the *source of the action*. The source of an action is a *noun-based* object such as a person, place, role, or thing that served as the stimulus, excitation, or cues that triggered or initiated the UC or scenario-based capability (Chapter 5).

UML™ / SysML™[1], for example, apply the term Actor (SysML™) to the source of the action. For example, a requirement statement begins with:

- The System (Actor) …
- The Subsystem (Actor) …

[1]UML™ and SysML™ are trademarks of the Object Management Group (OMG).

Since: (1) a requirement represents a capability derived from a UC or Scenario and (2) a UC defines what outcome the Actor expects from the SYSTEM/PRODUCT (Principle 5.14), therefore each requirement statement *specifies* the capability required to achieve the User's outcome. *How* the designers design the SYSTEM/PRODUCT to accomplish that requirement is not within the scope of specification requirements.

Observe the phrase "what outcome the Actor expects from the SYSTEM/PRODUCT." Remember, the Actor can be a person, place, role, or thing, not just the human User. If SUBSYSTEM A provides a *stimulus* or *excitation* to SUBSYSTEM B and you are developing SUBSYSTEM B's requirements, then SUBSYSTEM A is the Actor that serves as the *source of the action* and its stimulus as the *trigger* for SUBSYSTEM B's UC-based capability.

22.4.2.2 *Shall-Based Requirements Statements*

Principle 22.7

Shall-Based Requirements Principle

Specify every requirement statement with the word "shall" to express *mandatory* actions -outcomes - required for accomplishment to achieve compliance and User acceptance.

Each requirement statement uses "shall" to express the *mandatory action* to be accomplished. When "shall"-based requirements are incorporated into a specification that is an element of an approved contract, the requirement is considered to be *legally binding*.

Requirements, when issued as part of a contract, are considered *legally binding* and *sufficient* for procurement action when expressly stated with the word "shall." Some specification developers have a reputation for expressing requirements with the words "will," "should," and "must." Those terms express an "*intent* to perform," not a mandatory or required action.

A Word of Caution 22.1

Shalls, Wills, and Goals

There is increasing evidence of less discipline in specification wording by using words such as "will." The general viewpoint is that the "context" of the statement is what is important. For example, "… the system will perform the tasks specified below." Based on SE best practices, this requirement statement is *unacceptable!*

Exercise *caution* when writing specification requirements:

- "Shall" requirement statements express *mandatory* actions/outcomes required for accomplishment or to achieve compliance and acceptance.
- "Will" statements express a *non-committal* of *intent to perform* or statement of fact.
- "Goal" statements express *non-mandatory* or *voluntary* desires to strive to achieve and therefore do not contain "shall."

Would you sign a contract with a contractor that is "committed to perform" or one that has a "non-committal intent to perform?" In any case, *always* seek the advice of your Enterprise's legal counsel before committing to a specification that contains *ambiguous* wording that may be subject to legal interpretation.

There are Enterprises that contend it is *acceptable* to use the word *will* when conveying requirements … as long as the definition of *will* is clearly established. *Will* expresses "*a non-committal intent to perform*." When you establish a legally binding contract that commits and obligates the contracting parties, a System Acquirer or User is not interested in signing a contract with a System Developer or Services Provider that has a *non-committal* "best of intentions" to perform. There are two exceptions:

- Exception #1—There may be instances in which both parties agree that there is *uncertainty* that a technical or technology requirement can be *realistically* achieved due to cost, schedule, and other risks. In those cases, the specification might include a *will* statement expressing the intent to work toward achieving a specified outcome and performance value. Where this is the case, explicitly label the statement as a "goal."
- Exception #2—There may be *declarations*, not requirements statements, which state that a System/Product will achieve a hard to reach technical performance level of XXXX (units).

Finally, to facilitate specification readability, consider **boldfacing** each instance of the word "shall" (e.g., **shall).**

To summarize, either the contractor fully *complies* with a contract, its Terms and Conditions (Ts & Cs), and the SPS or EDS, or they do not. Use the word *shall* to express mandatory compliance that is required as a condition for contract compliance and completion acceptance. Therefore, declare these terms in SPS or EDS Section 2.0 Referenced Documents, their meaning, and application.

22.4.2.3 Action-Based Verb

Principle 22.8

Requirement Action Verb Principle

Employ an action-based verb in each requirement statement to express the action to be accomplished.

Requirements statements employ *action-based* verbs to express the *value-added action* to be performed such as *sense* pressure, *convert* sunlight to energy, *process* sensor data, or *store* information, energy, data, or materials. For example:

- "The Vehicle Management System (VMS) **shall** detect faults …
- The computer **shall** store the XYZ data.

22.4.2.4 Performance-Based Outcome to be Accomplished

Principle 22.9

Requirement Outcome Outcome Principle

Each requirement statement shall specify *one and only one* capability with *one and only one* performance-based outcome to be accomplished.

Given the SMART-VT Criteria (Principle 22.3), *how do you develop a requirement statement that complies with the criteria*? From an Engineering perspective the answer resides in *work products* as *objective evidence* – design drawings, physical ENTITIES, and verification results obtained via Inspection, Analysis, Demonstration, or Test.

This leads to another question: *to verify compliance, what do you use as the basis to compare the objective evidence of those work products*? That answer resides in the specification requirements that specify *outcomes* bounded by a *level of performance*. Remember - although we have stated that a requirement statement specified a *capability*, it is the performance-based outcome *result* produced by the capability that is verified for compliance.

To illustrate a performance-based outcome, consider the following example:

- When enabled, the System **shall** broadcast Operational Health & Status (OH&S) data messages at a 1 Hz rate.

Now, consider how specifications are often written. SEs "write" specifications as if they were writing a paper for a Composition 101 class. As Engineers, we typically do not receive formal education in specification requirements development as part of our Engineering education. When confronted with developing specifications in industry or government, we default to the education we did receive – Writing Composition 101. As a result, requirements statements are often written in paragraph style consisting of compound sentences, each containing multiple, embedded requirements.

To illustrate how the lack of formal education in specification development contributes to this problem, consider the following example:

Example 22.1

Poor Requirement Statement Examples

- The vehicle **shall** have the capability to safely transport a driver and three passengers on demand to a destination 300 miles away during all weather conditions on a single tank of fuel while maintaining the cabin temperature at a dashboard setting between 68 °F and 78 °F and storage for four luggage cases measuring 12″ × 24″ × 30″ in the trunk.
- For safety reasons, the vehicle **shall** have tail lights, backup lights, and flashers in the rear of the vehicle as well as carry 200 pounds of luggage in all sizes that are available today but does not restrict placement of a high-quality sound system in the trunk to avoid damage the passengers' eardrums when the doors are closed on a rainy day.

Have you encountered these types of requirements statements? *How many requirements do you suppose are in each of the example statements*? *How would you ever verify the paragraph*? *Always* write simple, brief, and concise statements that express *one and only one* requirement.

Additionally, specification writers often feel compelled to tell the reader *why they know what they know*. Again, write simple, brief, and concise statements that express *one and only one* requirement. System Acquirers pay you to smartly and cost-effectively develop systems, products, or services that meet *their* needs, not express your knowledge via requirements statements.

For a discussion of *acceptable* and *unacceptable* requirements statements, you are encouraged to consult the INCOSE-TP-2010-006-02 (2015) for guidance.

22.4.2.4.1 Level of Performance Accuracy and Precision

Principle 22.10

Requirement Accuracy and Precision Principle

Each specification requirement statement shall quantitatively bound the accuracy and precision of the outcome level of performance and its tolerance limit(s).

Once a capability-based outcome is identified, the next step is to quantitatively bound its level of performance in terms of accuracy and precision. For example,

- Does the nominal outcome level of performance accurately and quantitatively represent the boundary condition(s) for User acceptance of a required system, product, or service outcome.
- What quantitative tolerance levels- tightly controlled variances - in the nominal outcome performance level are required within limits of the technology, test equipment, components and materials, and manufacturing processes without driving up development costs?

22.4.2.4.2 Action Response Time-Based Requirements

Principle 22.11

Requirement Response Time Principle

Where appropriate, each requirement statement **shall** specify the response time that bounds accomplishment of an outcome.

Where applicable, well-defined requirements statements specify time constraints between a triggering event such as a stimulus, excitation, or cue and the required outcome response. For example:

- The System **shall** respond to each command within 250 ± 10 milliseconds of receipt (Action Response Time).
- (Alternate Method) On receipt of each command, the System **shall** respond within 250 ± 10 milliseconds (Action Response Time).

22.4.2.4.3 Action Response Format Requirements

Principle 22.12

Requirement Outcome Format Principle

Where appropriate, each requirement statement **shall** specify the format of an action-based outcome response.

When applicable, well-defined requirements statements specify the format of the action response to cooperative, benign, or hostile interactions based on rules of engagement. For example:

- The System **shall** transmit data messages to the XYZ System formatted in accordance with Table X.

22.4.2.4.4 *Action Completion Time Requirements*

Principle 22.13

Requirement Action Completion Principle

Where appropriate, each requirement statement **shall** bound and specify the time allowed for completion of an action.

Where applicable, well-defined requirements statements may need to specify the *maximum* allowable time for responding to a stimulus, excitation, or cue. For example:

- The System **shall** complete processing of the XYZ information within 100 ± 10 milliseconds (Completion Time) after receipt of the command.
- (Alternate Method) On receipt of the XYZ command, the System **shall** complete processing of the XYZ information within 100 ± 10 milliseconds.

22.4.2.4.5 *Outcome Media Requirements*

Principle 22.14

Requirement Media Principle

Where appropriate, each requirement statement outcome shall specify the media to be used for delivering the outcome.

When applicable, well-defined requirements statements specify the format of the action response to cooperative, benign, or hostile interactions based on rules of engagement. For example:

- The website **shall** provide the option for ordering a Compact Disk (CD) of the artist's body of work.

22.4.2.4.6 *Unacceptable Outcomes*

Principle 22.15

Requirement Unacceptable Outcomes Principle

Where appropriate, each requirement statement **shall** specify one and only one outcome to be avoided when performing an action.

Well-defined requirements statements may need to specify outcomes to be *avoided*. For example:

- System exhaust emissions **shall not** exceed … (Negative connotation)
- (Alternate Method) System exhaust emissions **shall** be less than … (Positive connotation)

22.4.2.5 *Condition-Based Actions Requirements*

Principle 22.16

Requirement Conditional Actions Principle

Where appropriate, each requirement statement **shall** specify the trigger-based conditions that initiate the action.

Sometimes a requirement statement is dependent on *triggering conditions* such as a stimulus, excitation, or cue. As a result, the condition has to be embedded within the statement. Consider the following example:

- The System **shall** … on receipt of the XYZ command.
- (Alternate Method) On receipt of the XYZ command, the System **shall** …

22.4.2.6 *Recipient(s) of the Action*

Principle 22.17

Requirement Action Recipient Principle

Where appropriate, each requirement statement **shall** specify the recipient of an action.

Where applicable, well-defined requirements statements identify the intended recipient(s) of the outcome(s). For example:

- The System **shall** transmit Operational Health and Status (OH&S) data messages to the XYZ System (Recipient of the Action) at a 1Hz rate.

22.4.3 Step 3: Refine the Requirements Statement Grammar

The third step requires transformation of the syntactical statements into clear and concise grammatical statements that are easily read and understood.

22.5 SELECTION OF REQUIREMENT VERIFICATION METHODS

One of the key elements of specification development is establishing agreement between the System Acquirer and the System Developer concerning to *how to prove* that a system, product, or service *complies* with specification Section 3.0 Requirements (Table 20.1). One of the greatest potential risks in System Development concerns System Developer preparation and conduct of formal *Acceptance Tests (ATs)* for the System Acquirer, and both parties *disagree* on the method(s) of verification. Disagreements during AT result in significant impacts to System Developer contract costs,

schedules, and System Acquirer fielding and support costs schedules.

To preclude this scenario, specifications include a Section 4.0 Qualification Provisions (Table 20.1) in the SPS or an EDS. Section 4.0 *explicitly* documents the System Acquirer and System Developer agreement concerning the requirement's verification methods that are recognized as the *acceptable* verification methods for proving compliance. Therefore, a Requirements Verification Matrix (RVM) should be required to be part of the SPS or an EDS Section 4.0. We will address the RVM in a later section.

Author's Note 22.2

System Acquirer and System Developer Contextual Roles

Remember that the terms System Acquirer and System Developer are used in a "role-based" context as shown in the Supply Chain illustrated in Figure 4.1. These terms, as used previously, identify the contract level roles and relationships. They also apply in a similar context between SYSTEM and PRODUCT, PRODUCT and SUBSYSTEM Development teams; System Developer and Subcontractors; and so on. *Why*? As each team flows down requirements to lower level EDS, the implementer of those requirements must demonstrate to their System Acquirer (the higher level team) that the requirements have been accomplished.

Let's shift our focus to how verification methods are selected.

22.5.1 Verification Method Selection Process

In general, four verification methods are available to demonstrate that a system, product, or service complies with a specification requirement. These methods include Inspection or Examination, Analysis, Demonstration, and Test (IADT). A fifth method, Similarity, may be permitted by some Enterprises.

Author's Note 22.3

Please note that the NASA *Systems Engineering Handbook* (NASA, 2007) refers to Analysis, Inspection, Demonstration, and Test as "validation" methods. In this context, NASA defines *validation* as "Proof that the product accomplishes the intended purpose. Validation may be determined by a combination of test, analysis, and demonstration." (NASA, 2007, p. 278)

So *how do SEs select the appropriate verification method(s)*? The answer resides in two key points:

- First, verification methods cover a broad spectrum of costs and schedule impacts. *Inspection* is the least costly on one end of the spectrum, followed by *Analysis*, *Demonstration*, and finally *Test* as the most costly. Please note that Test may be *necessary* to obtain test data but may be *insufficient* for proving compliance. That may require Analysis of the Test data.
- Second, select the most *efficient* and *effective* methods for the least cost and schedule impact (Principle 22.4).

So, *how do we make these decisions*? Figure 22.2 provides the solution via a chain of decisions used to assess each requirement statement. Let's explore each of these decisions in more detail. Please note that Inspection and examination are combined in Figure 22.2 due to space restrictions.

22.5.1.1 Verification by Inspection or Examination Decision Verification by Inspection requires a questioning exercise that has two potential outcomes:

- Is Inspection/Examination *necessary* to prove compliance to specification or design requirement? If the answer is YES, select Inspection/Examination as a verification method.
- Is the Inspection/Examination *sufficient* to prove compliance to a specification or design requirement? If the answer is YES, exit and proceed to identification of verification methods for the next requirement. If the answer is NO, proceed to the Verification by Analysis decision.

Example 22.2

Inspection versus Examination Subtlety

Recognize the subtlety between Inspection and Examination.

- *Inspection*, as defined earlier, can be as simple as performing visual, auditory, or vibratory verification such as a power light illuminates when power is applied.
- *Examination* ranges from simple measurements with a ruler to electron microscopy, radiology, etc.

22.5.1.2 Verification by Analysis Decision Verification by Analysis requires a questioning exercise that has two potential outcomes:

- Is Analysis *necessary* to produce objective evidence—physical data collected from its formal testing—that the ENTITY complies with a specification or design requirement? If the answer is YES, select Analysis as the verification method.
- Is Analysis *sufficient* to prove compliance to a specification or design requirement? If the answer is YES, exit and proceed to Identification of verification methods for the next requirement. If the answer is NO, proceed to the Verification by Demonstration decision.

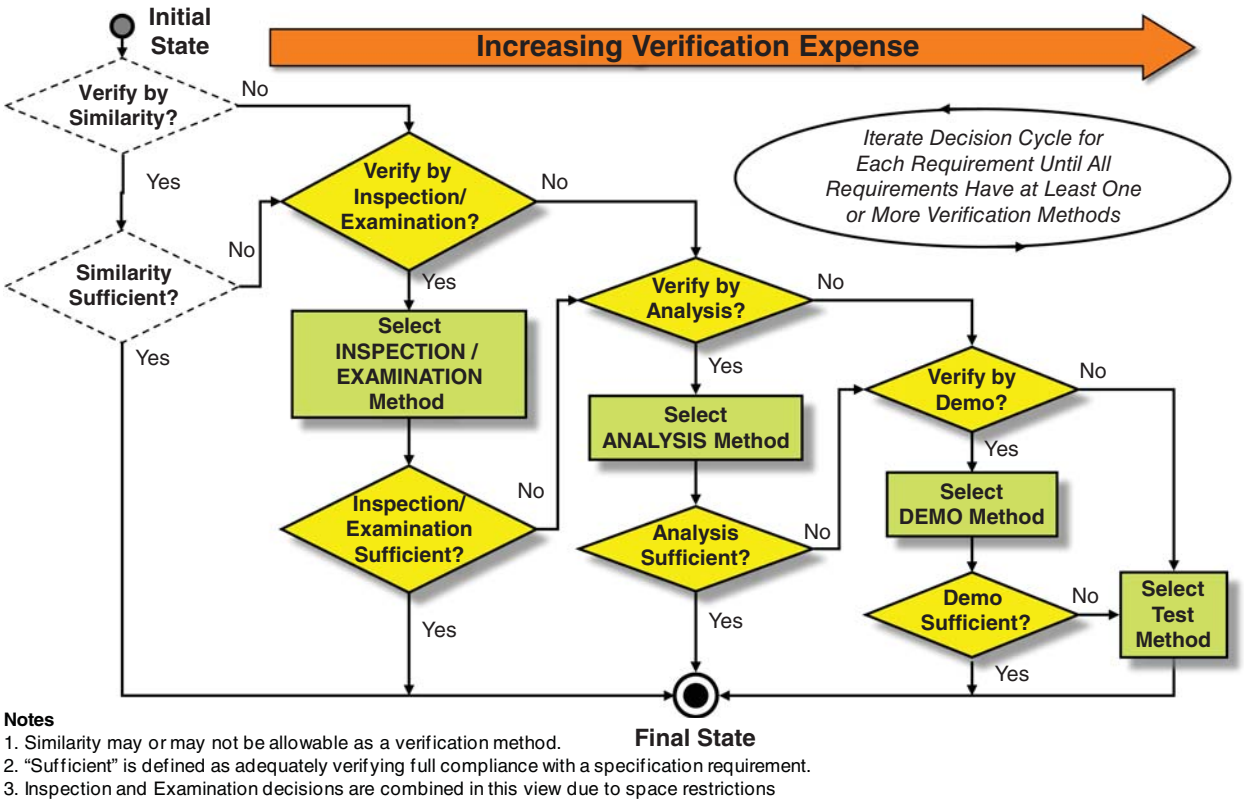


Figure 22.2 Requirements Verification Method(s) Selection Process.

Example 22.3

Analysis, as a verification method, requires presentation of objective evidence based on work products such as Finite Element Analysis (FEA), thermal analysis, review of Modeling and Simulation (M&S) (Chapter 33) data, etc. to verify compliance to a specification requirement.

The FAA *Guide to Reusable Launch Vehicle Safety Validation and Verification Planning* (FAA. 2003, p. 8), for example, states:

> "This method involves technical or mathematical evaluation, mathematical models, simulations, algorithms, and circuit diagrams."

22.5.1.3 Verification by Demonstration Decision Verification by Demonstration requires a questioning exercise that has two potential *outcomes:*

- Is Demonstration *necessary* to prove by formal observation that a physical *entity* produces *repeatable* and *predictable* outcome(s) without having to record formal measurements to prove that it complies with a specification or design requirement? If the answer is Yes, select Demonstration as a verification method.
- Is Demonstration *sufficient* to prove compliance to a specification or design requirement? If the answer is Yes, exit and proceed to identification of verification methods for the next requirement. If the answer is No, proceed to the Verification by Test decision.

Example 22.4

Requirement Demonstration (Verification Method) Example

Demonstration, as a verification method, can be used to verify that a User can log on to a computer system if they have: (1) registered and (2) obtained account log-on authorization.

22.5.1.4 Verification by Test Decision If the answer to *any* of the preceding questions is No, then Test must be selected as a verification method for a specific requirement. The verification method selection process exits at this point and cycles to the next requirement.

Example 22.5

Test (Verification Method) Example

Test, as a verification method, for a jet engine might include installation on a test stand, equipping the engine with sensors

and instruments to assess and collect its performance data on engine performance as well as its operating environment; performing load tests; water, hail, and waterfowl ingestion, etc.

22.5.1.5 Verification by Similarity Decision (Where Permissible) Verification by Similarity serves as an acceptable verification method in some business domains. Development of a new SYSTEM or PRODUCT design can be: (1) very time consuming and expensive and (2) should be pursued only as a last resort after all efforts have been exhausted to locate external Commercial-Off-the-Shelf (COTS) products (Chapter 16) that meet specification requirements. When legacy products already exist in-house based on previous System Development, it makes sense to *reuse* the design, especially if:

- No behavioral or physical design modifications have been made since the ENTITY's formal verification.
- No safety Critical Operational or Technical Issue (COI/CTI) problems or issues have been detected in fielded units.
- Specification requirements of the "As-Verified" (Principle 16.5) reusable design are *equal to* or *exceed* the new SYSTEM's/PRODUCT's capability requirements or OPERATING ENVIRONMENT conditions.

Since the legacy design has been *verified* and *proven* over time in fielded SYSTEMS/PRODUCTS, the only remaining issue is verification of each instance of the deliverable product. Net result: *cost avoidance* instead of new development. As an example, the FAA *Guide to Reusable Launch Vehicle Safety Validation and Verification Planning* (FAA, 2003, p. 8) states:

> "A 'qualification by similarity' analysis is required when using this verification method. … If there are items that are not significantly similar, "delta qualification" tests are performed to bring the item into full compliance with the requirements of the new application."

One of the reasons many Enterprises *discourage* verification by Similarity is that it requires presentation of *objective evidence* such as Inspection or Examination of Records, which may no longer be available.

22.5.1.6 Verification Methods Summary In summary, verification method selection requires insightful forethought. Observe:

- That one or more verification methods (Principle 13.10) may be required to prove compliance to a specification requirement.
- How the verification methods selection process exists when the verification method satisfies the *necessary* and *sufficient* criteria.

Verification method selection decisions range from simple visual Inspections/Examination to complex Tests requiring Analysis of results that can be costly and consume valuable schedule time. *Why would you want to go to the trouble and expense of performing a Test when a simple visual Inspection will provide objective evidence to meet the requirement for being necessary and sufficient*?

Given this understanding of how verification methods are selected, let's proceed with how the methods are documented via the RVM.

22.6 REQUIREMENTS TRACEABILITY AND VERIFICATION TOOLS

Requirements traceability and verification require Requirements Management Tools (RMTs) that enable Engineers to efficiently and effectively enter, mine, and manage specifications, requirements, verification, and verification results data. Highly complex systems, for example, often involve several hundred thousand requirements plus all of the associated data. Given the enormity of the data set, a relational database RMT is the only local way to have a single repository of information. The power of a RMT enables Users to mine, sort, and filter the data, which can be ported to specific types of reports for analysis and review, as well as create special reports and track metrics.

Report examples include: the Requirements Verification Matrix (RVM), the Requirements Traceability Matrix (RTM), and the Requirements Verification Traceability Matrix (RVTM). One of the challenges is people become enamored with creating spreadsheet or database matrix tools and forget that they are trying to manage *requirements knowledge*. Tools are simply *solution spaces* to a broader *problem space* we need to manage concerning requirements. For example, the *problem spaces* are represented by questions such as:

1. Are all requirements in specifications at various levels of abstraction traceable to the User's *source* or *originating* requirements via the SPS? If not, why not? (Principle 19.13).
2. Are first level SPS or EDS requirements statements traceable to User Stories, UCs, and Scenarios?
3. How many requirements are in the SPS, specific EDS, or the entire SYSTEM or PRODUCT requirements hierarchy?
4. How many To Be Determined (TBDs) remain undefined?

5. How many requirements lack assignment of an accountable implementer?
6. How many requirements lack verification methods?
7. How many requirements: (1) have been verified, (2) have verification results discrepancies, and (3) how many requirements remain to be verified?

Rather than jumping into a discussion of RMTs, which are *solution spaces*, let's employ Systems Thinking (Chapter 1) and focus on what knowledge – problem spaces - Engineers need to mine and analyze from the database to answer the questions above.

Let's begin Question 1 above concerning requirements traceability.

22.6.1 Requirements Verification Traceability Matrix (RVTM)[2]

One of the most common ways to address requirements traceability and verification begins with an RVTM. Since the RVM and RTM are customizations of the RVTM, we will focus our discussion on it.

The RVTM is simply a tabular listing of key requirement attributes selected for presentation in a report. Table 22.3 provides an RVTM example.

Observe the construction of the RVTM.

- Column 1 – Assigns a unique Identifier to each requirement statement. The ID is permanently assigned when the requirement statement is created and provides the basis for formal Configuration Management (Chapter 16) of the requirement.
- Column 2 – States the requirement.
- Column 3 – Identifies the SYSTEM or ENTITY architecture element – PRODUCT, SUBSYSTEM, ASSEMBLY, SUBASSEMBLY, or PART Level component – that the requirement is allocated to for implementation and accountability.
- Column 4 – Identifies the Verification Level that represents where the requirement will be verified.
- Columns 5 – 8 – Provide cells (RMT "pick lists") for selecting one or more Verification Methods required to prove compliance.
- Column 9 – Identifies the Next Higher Level requirement for vertical traceability.

Within the RVTM database, other attributes such as Responsibility and Accountability, references to analyses, and so forth can be appended as additional columns in customized reports.

[2]Additional Reading—NASA *Systems Engineering Handbook*, Appendix D, pp. 282–283.

22.6.1.1 Requirements Verification Matrix (RVM)

A Requirements Verification Matrix (RVM) is simply a customized version of the RVTM shown in Table 22.3. The RTM links each SPS (Table 20.1) or EDS Section 3.X Requirement to specific Section 4.X Verification Methods—namely, Inspection, Examination, Analysis, Demonstration, and Test. The row below the RVTM identifies the RVTM columns that appear in most RVMs.

Principle 22.5 The selection of Verification Methods is often considered one of the most mundane tasks Engineers perform. As a result, they spend very little time *seriously contemplating how* they will verify each requirement. Additionally, selection of Verification Methods for each requirement should be a *collaborative* exercise with the Testers (Chapter 28) for two reasons.

1. The Testers are the ones who will be confronted with verifying each requirement. To avoid conflicts, they should be the ones who pass judgment on whether a requirement statement is *well-defined* – complies with the SMART-VT Criteria (Principle 22.3) - or not.
2. Participation by the Testers ensures they are in the loop concerning specific types of test equipment that will be required to test each requirement.

Example 22.6 **Verification Level Application Example**

You are assigned a task to create SUBSYSTEM A EDS. Ideally, you would like to verify SUBSYSTEM A as a physical *work product*. During assignment of verification methods, you discover it is *impractical* to complete verification of Assembly Requirement 3.1.1.2 until it is integrated into the next higher-level SUBSYSTEM. For example, a critical interface – ASSEMBLY A4 – provides inputs to ASSEMBLY 2. ASSEMBLY A4 is lagging behind schedule and has not been verified. *How do we solve this problem*? There are a couple of options. We could:

- Emulate or stimulate (Figure 28.3) ASSEMBLY A4 and complete ASSEMBLY A2 verification.
- Defer completion of ASSEMBLY A2 verification until ASSEMBLY A4 has been verified. However, this may idle laboratory testing and equipment, which is *unacceptable*.

Another solution is to create an attribute for the Verification Level that indicates the level of abstraction in which ASSEMBLY A2 Requirement 3.1.1.2 verification can be completed. Then, when ASSEMBLY A4 verification is complete, ASSEMBLY A2 and A4 are integrated to form SUBSYSTEM A. SUBSYSTEM A Requirement 3.1.1 verification includes verification of ASSEMBLY A2 Requirement 3.1.1.2.

TABLE 22.3 Requirements Verification Traceability Matrix (RVTM) Report

Reqmt. ID	Requirements Statement	Allocated to	Verification Level	Method of Verification				TracesVertically to
				Inspect	Analysis	Demo	Test	
SYS_136	**3.1.1 Capability A** The system shall …	Subsystem 123	Subsystem	X				3.1 Capability XXXX
SYS_137	**3.1.1 Capability A1** The system shall (Capability A1).	Assembly A1	Assembly	X	X			3.1.1 Capability A
SYS_138	**3.1.1.2 Capability A2** The system shall (Capability A2).	Assembly A2	Subsystem		X			3.1.1 Capability A
SYS_139	**3.1.1.3 Capability A3** The system shall (Capability A3).	Assembly A3	Assembly			X		3.1.1 Capability A
SYS_140	**3.1.1.4 Capability A4** The system shall (Capability A4).		Assembly				X	3.1.1 Capability A
RVTM columns Applicability	RTM & RVM	RVTM	RVM	RVM	RVM	RVM	RVM	RTM

22.6.1.2 Requirements Traceability Matrix (RTM)

A Requirements Traceability Matrix (RTM), like the RVM, is a simply customized version of the RVTM shown in Table 22.3. The RTM links each SPS or EDS Section 3.X *child* requirement to the next higher level *parent* requirement. The row below Table 22.3 identifies the RVTM columns that appear in most RTMs.

22.6.2 Requirements Management Tools (RMTs)

Traditionally, hardcopy specifications required the RVM to be presented in the document as part of the SPS or EDS (Table 20.1) Section 4.0 Qualification Provisions. The difficulty with this approach is that the specification developer and readers had to flip back and forth between Section 3.0 Requirements and Section 4.0 (Qualification Provisions). Flipping back and forth is *simply inefficient and time consuming*.

With advancements in RMT technologies based on an object-oriented, relational databases, we can create an RVTM – Table 22.3, for example, as a report directly out of the RMT. RMTs simplify this approach and enable the specification developer to select one or more verification methods, for example, from a set of "pick lists" options in a single cell, not five columns as shown in Table 22.3.

Traditionally, Engineers prefer word processor-based documents. *Why*? They are easier to create with skills and software applications most stakeholders possess. The SE problem is, however, *how do you link requirements in a word processor document to another specification, engineering drawing, graphic, and so on*?

With today's Web-based technologies, you may say this can be done easily with word processor document links. However, these links do not allow you to exploit the power of the database to display or print the "thread" of text statements that are traceable across documents. This is critically important, especially when performing impact assessments as a result of higher level requirements changes.

You can rationalize performing System Development without an RMT; however, the tool saves hours when verifying requirements traceability, collecting requirements metrics, and performing multi-level requirements impact assessments. This is why RT tools provide an advantage and leverage your own ability and time. You can further rationale that RMTs are *unaffordable*. However, an RMT will pay dividends by eliminating manual document linking audits, especially for large, complex system development efforts.

22.7 REQUIREMENTS STATEMENT DEVELOPMENT GUIDELINES

Requirements can be characterized in a range of attributes. These attributes include legal, technical, cost, priority, and schedule considerations. Let's examine each of these briefly.

22.7.1 Title Each Requirement Statement

Principle 22.18

Requirement Identifier and Title Principle

Assign a unique identifier and title to each requirement statement to facilitate searches and readability.

Requirements tend to lose their identities as statements within a larger document. This is particularly troublesome when the need arises to rapidly search for an instance of a requirement. Make it easy for reviewers and users of specifications to easily locate requirements. Tag each requirement with a unique identifier and label each with a "bumper sticker" title. This provides a mechanism for capture in the specification's Table of Contents and makes identification easier.

Example 22.7

Requirement Title Example

SPS_136 **Perform Communications** (Bumper Sticker Title)

- The System **shall** communicate with …

SPS_243 **Convert 28 vdc Power** (Bumper Sticker Title)

- The System **shall** convert 28 vdc power to +5vdc.

22.7.2 References to Other Sections within a Specification

Specification sections often reference other sections *within* the document. Typically, the specification will state, "Refer to Section 3.4.2.6." Instead, reference the section paragraph number and title. *Why*? Specification outline numbering often changes as topics are added or deleted. When this occurs, the sections may be automatically renumbered by the RMT. Thus, a reference such as "Refer to Section 3.4.2.6" may refer to an *unrelated* requirement topic. The same applies to references to external standards and specifications.

22.7.3 References to External Specifications and Standards

Principle 22.19

External References Principle

Specification references to external specifications and standards must include the following: Document ID, Version, and Date. References to internal sections should include Section Number and Title.

When you reference *external* specifications and standards, include the title, document number, version, and date. References to external documents with complete titles, dates, versions, etc. should be listed in the SPS or EDS Section 2.0 Referenced Documents. There are two key points here:

1. List complete information about the document such as "XYZ *System Performance Specification* (SPS) Document No. 123456, Revision A, June 20, XXXX."
2. Identify the version as shown in the preceding point. Specification developers in their haste will often scribble a brief title notation with the intent of coming back later and updating the reference, which is risky. Then, make a broad reference within the specification such as "… in accordance with the latest version of (Standard) …" Standard XXXX may have 400 pages and covers the universe of applications. You do not want to prove compliance to 400 pages of requirements. Simply reference the specific Standard XXXX paragraph number. Then, list the version in (Table 20.1) Section 2.0 References.

22.7.4 Specify Operational and Technical Capability Requirements

Operational and technical capabilities and characteristics requirements document capabilities required for a system, product, or service success. These capabilities drive the System Design Solution and must be an integral part of every SPS or SDS outline (Table 20.1). The DAU (2005, p. B-15) defines these terms as follows:

22.7.4.1 Required Operational Characteristics Required operational characteristics consist of "System parameters that are primary indicators of the system's capability to be employed to perform the required mission functions, and to be supported" (DAU, 2005, p. B-15).

22.7.4.2 Required Technical Characteristics "System parameters selected as primary indicators of achievement of engineering goals. These need not be direct measures of, but should always relate to the system's capability to perform the required mission functions, and to be supported" (DAU, 2005, p. B-15). Remember that *operational* and *technical* characteristics should follow the SMART-VT Principle (Principle 22.3)—specific, measurable, achievable, realistic, verifiable and traceable.

22.7.5 Avoid Paragraph-Based Text Requirements

Principle 22.20

Compound Requirements Principle

Avoid usage of compound requirements statements written in paragraph style.

Paragraph-based *declarative* statements of fact are acceptable in specifications for Specification Section 1.0 Introduction. The problem is System Developers must spend valuable time separating or *parsing* compound requirements into *singular* requirements when they appear in SPS or EDS Specification Section 3.0 Requirements. Do yourself and the System Developer (contractor, subcontractor, etc.) a favor and develop the SPS and EDS with *singular* requirements statements that cover *one and only one* capability. This saves valuable time that can be better spent on higher priority tasks and creates a consistent format that promotes *readability, understandability*, and *verifiability*.

Example 22.8

Compound Requirement Statements

The vehicle **shall** be capable of carrying a family of four on ski trips to (location) in all types of conditions up and down mountains, over gravel roads, in the vicinity of the lake near the river that is close to (city).

Remember that if you have a contract and are obligated to *demonstrate* and *verify* compliance to technical requirements, every *unnecessary* word in a requirement statement increases the risk of *misinterpretation* and *conflict*. Requirements statements should be brief, practical, clearly stated, and concise. To illustrate this point, consider the following example based on the popular children's *Dick and Jane Series* (Gray and Sharp, 1946) elementary reader. The intent is not to be absurd but to illustrate the difference in the two requirements writing styles – Example 22.9 versus Example 22.10.

Example 22.9

Simple Writing Style

1. Watch spot run.
2. Dick ran up the hill.
3. Jane fell down.

Although these are extreme examples, it illustrates that brief statements with few words explicitly communicate. *How do we put this into practice*? Consider the following examples:

Example 22.10

Requirements Statements

1. The System **shall** operate from an external +28vdc power source.
2. The System **shall** respond to commands from the XYZ within 250 ± 10 milliseconds (MS).

22.7.6 Eliminate Compound Requirements Statements

When requirements statements specify compound requirements, allocation of the requirement becomes difficult. *Why*? Assume multiple requirements are specified in a single requirement statement. Each is allocated to different ENTITIES. Here is the question - *How do you link which portion of a requirement statement to a specific Subsystem?*

When the SYSTEM must be verified, *how can you check off a requirement as complete when portions remain to be verified due to missing equipment and the like*? The only alternative is to uniquely number each requirement within the statement, which is impractical. Verification requires that all elements embedded within the statement are required to satisfy verification of the statement as a whole. Consider the following example.

Example 22.11

SPS_178 Transmit Weather Data

The System **shall** transmit weather data messages containing the following information:

- Date
- Time of Day (TOD)
- Ambient Temperature
- Relative humidity
- Barometric pressure

SPS_179 Formatted Weather Messages

The System **shall** transmit weather data messages formatted in accordance with Table XX.X (Refer to Figure 27.5):

22.7.7 Eliminate Endless Requirements

Principle 22.21

Endless Requirements Principle

Eliminate usage of endless requirements terms such as all, etc., et al., and and/or, which are impractical to specify, bound, and verify.

A note to specification writers: Eliminate every use of the word "all" in every specification! Appreciate the legal significance of this statement "… **all** instances of an XYZ will be verified …." Contemplate how large in scope the "all" universe is! "All", a limitless word for practical purposes, is *boundless* and subject to *interpretation*. As a specification developer, your mission is to specify and bound the *solution space* as simple and practical; not for verifying every conceivable scenario or instance of an entity. Do yourself a favor and *avoid* usage of "all."

22.7.8 Eliminate the Term and Abbreviation "Etc."

Et cetera (etc.) is another word Specification writers use but should not. Engineers discover early in their careers that there will always be someone who questions miniscule instances of physics, science, or a natural occurrence – rightfully or wrongly - and chastise the Engineer for their oversight. To avoid this situation, Engineers default to "etc." to "cover all bases". As an SE, your mission is to specify and bound the *solution space*. When you specify a requirement with "etc." such as "The system shall consist of a, b, c, etc.," the System Acquirer could say later, "Well, we also want a d, e, and f." You may reply we did not bid the cost of performing "d, e, and f," which may then bring a response from the System Acquirer, "You agreed to the requirements and 'etc.' means we can request anything we want to." Do yourself a favor and eliminate all instances of "etc." in specifications.

22.7.9 Eliminate the Term "And/Or"

Specification writers often specify requirements via enumerated lists that include the term "and/or."

Example 22.12

Enumerated Lists of Requirements

For example, "The system shall **consist** of capabilities A, B, C, and/or D." The statement, as written, sounds like a salesperson selling an automobile …" You can purchase the car with options A, B, C, and/or D … whatever you choose.

Explicitly specify what is required. Either the SYSTEM consists of A, B, C, or D or it does not. If not, so state and bound exactly what the system is to contain. Remember that specifications must state *exactly* what capabilities are required to be verified at system delivery and acceptance, not wish lists made by the System Acquirer and the User.

22.7.10 Analytical "Testing" Derived Requirements Statements

Many people are often surprised to learn that you can analytically "test" requirements. Requirement testing takes the form of technical compliance audits with Enterprise standards and conventions, such as *software* coding standards and graphical conventions. Modeling & Simulation (M&S) provides another method for testing requirements. Execution of those models and simulations provides insights into the *reasonableness* of a requirement, performance allocation, potential conflict, and difficulty in verification.

Requirement testing also includes *inspection* and *evaluation* of each requirements statement in accordance with predefined criteria. *What are the criteria*? This brings us to our next topic, Requirement Validation Criteria.

22.7.11 Requirement Validation Criteria

When testing the validity of a requirement by Inspection or Examination, there are a number of criteria that can be applied to determine the necessity and sufficiency of the requirement. Table 22.4 provides an example listing of key criteria.

The criteria stated in Table 22.4 are comprehensive. You may be thinking: *How could one conceivably evaluate a specification with potentially hundreds of requirements using these criteria?* Seasoned SE professionals subconsciously imprint most of these criteria into their mental processes. With experience, you will learn to test specification requirements rapidly by inspection. This further reinforces the importance of training all SEs in the proper methods of requirements writing and review to ensure a level of confidence and continuity in specification results.

22.8 WHEN DOES A REQUIREMENT BECOME "OFFICIAL"?

Official Requirement Principle

A requirement is not considered *official* until the following criteria have been met:

Principle 22.22

- Is accepted by a consensus of its Stakeholders.
- Is traceable to *source* or *originating* requirements via an RVM.
- Satisfies requirements validation criteria listed in Table 22.4.
- Is assigned one or more *verification* methods.
- Is approved and released for implementation.

Our previous discussions focused on the content of well-defined requirements. A key question commonly asked is: *when is a requirement considered to be official*? There are two stages of answers to the question: (1) unofficial and (2) official.

Simply identifying and specifying a requirement statement is only a prerequisite to official Stakeholder recognition and acceptance. This first stage merely establishes the statement as *relevant* and *worthy* of consideration as a requirement. That takes us to the next stage, official approval and subsequent release.

Many SEs erroneously believe that simply preparing a requirement statement makes the requirement *complete* and *acceptable*. Preparing a requirement statement *does not* mean that the statement will pass the SMART-VT (Principle 22.3) and Requirement Validation Criteria in Table 22.4. *Why is it necessary to test completeness*? There are two key factors to consider.

- First, identifying the verification method(s) forces the requirement developer to *think* about *how* the requirement will be verified. For example, write a one- or two-sentence verification test plan that takes two measurements, compares the two, and evaluates the differences relative to the requirement statement.
- Conversely, if you have difficulty identifying a test plan, maybe you should consider rewriting and re-scoping the requirement statement.

Author's Note 22.4

Engineers tend to defer identification of the verification method(s) to the start of the System Integration, Test, and Evaluation (SITE) phase. This is *unacceptable*! You need to recognize the need to identify the verification methods *early* in the requirements development process, not only for cost estimates, but to validate the *realism* and *reasonableness* of the requirement.

Regarding the second point, an approach may be to identify a preliminary set of verification methods when the requirements are derived. This will provide an initial *completeness check* for the verification requirement. Then, continue the analysis to lower levels. Prior to the need to baseline the higher-level requirements, review and reconcile the preliminary verification methods.

While the potential ramifications for *premature* baselining are known (Chapter 16) such as increased costs due to stability of decisions, *human procrastination* can pose an even greater challenge and risk. Whether intentional or not, project schedules often become a convenient excuse for SEs to shift identification of verification methods to the back of their priority list as discussed earlier in our discussion of Verification Methods selection. As a result, Engineers sometimes produce them just before they are needed in SITE, which is the wrong. Establish verification methods prior to specification approval.

22.9 CHAPTER SUMMARY

Our discussion of the requirements statement development practices:

- Provided principles for developing well-defined requirements.
- Discussed when a requirement should be officially recognized as a requirement.
- Described the key attributes that characterize a well-defined requirement.
- Introduced a basic methodology for preparing requirements statements via a three-step approach that avoids common problems of developing requirement content and grammar simultaneously.

TABLE 22.4 Requirement Statement Validation Criteria

ID	Criteria	Criteria Question	Reference
1.	Unique Identifier	Does the requirement have its own unique identifier such as specification nemonic and indexed numeric within the specification?	Principle 22.18
2.	Unique Title	Does the requirement have an outcome-based title that: (1) is unique within the SYSTEM and (2) represents the outcome to be achieved?	Principle 22.18
3.	Uniqueness	Is the requirement statement *unique* within the SYSTEM's specification tree (Figure 19.2) without duplication?	Principle 19.5
4.	Essentialness	Is the requirement *essential* to the development of a system, product, or service?	Principle 19.7
5.	Validity	Is this a *valid* requirement that reflects a required capability traceable to the User's intended operational needs or simply a symptom of a problem or need?	Principle 4.13
6.	Traceability	Is the requirement *traceable* to the User's source or originating requirements?	Principle 19.13
7.	SOW language	Does this requirement include language that logically belongs within the contract or project charter Statement of Work (SOW)?	Principles 20.6 and 20.7
8.	Specification Scope	Does this requirement fall within the *scope of this specification* or does it belong in another specification?	Principle 19.5
9.	Hierarchical level	If this requirement is within the *scope of this specification*, is the requirement positioned at the right level within the requirements hierarchy?	Figure 20.3
10.	User priority	What is the User's *priority level* for this requirement such as can live without it, nice to have, desirable, mandatory?	Principle 19.15
11.	Realism and Achievability	Is the requirement realistic and actionable?	Principle 22.3
12.	Feasibility	Can the requirement be implemented within reasonable and justifiable need priorities and budgetary cost without limiting the minimum required set of requirements?	Principle 19.4
13.	Completeness	Is the requirement complete in terms of satisfying the structural syntax criteria?	Principle 22.5
14.	Semantics and Terminology	Does the requirement contain any terminology or semantics that may have multiple interpretations and require scoping definitions for clarity?	Principle 19.12
15.	Conciseness	Is the requirement stated in a clear, concise, and unambiguous language that results in one and *only one interpretation* by its stakeholders?	Principle 19.9
16.	Understandability	Is the requirement simply stated in language that is easily understood by the document's *stakeholders*?	Principle 19.12
17.	Consistency	Is the requirement *consistent* with SYSTEM terminology and nemonics used: • Throughout the specification? • Among specifications within the Specification Tree (Figure 19.2)? • By the System Acquirer, User, and System Developer(s)?	Principle 19.11
18.	Assumptions	Does the requirement make assumptions that should be communicated in the *Notes and Assumptions* section of the specification?	Principle 24.20
19.	Overlap	Does the scope of this requirement overlap the scope of another requirement?	Principle 22.1
20.	Accuracy	Is the requirement stated in language that *accurately* and *precisely* bounds the required outcome and level of required performance?	Principle 22.10
21.	Precision	Is the precision of the required level of performance and tolerance levels adequate or too restrictive?	Principle 22.10
22.	Testability	If required, can a test or series of tests be devised and instrumented economically with available resources—for example, knowledge, skills, equipment, etc.?	Principle 22.3
23.	Measurability	If testable, can the outcome and level of performance be measured *directly* or derived *indirectly* by analysis of test results?	Principle 22.3
24.	Verifiability	Can the outcome and level of performance be verified by *inspection, analysis, demonstration*, or *test* to prove full compliance with the requirement?	Principle 22.4
25.	Level of Risk	Does this requirement pose any significant technical, technology, cost, schedule, or support risks?	Principle 19.16

- Provided a list of suggestions for developing well-defined requirements.
- Highlighted the need to focus on singular requirements statements that are unique within the hierarchy of requirements.
- Described the process for selecting requirement verification methods.
- Described and illustrated differences between an RVTM, RTM and an RVM.

22.10 CHAPTER EXERCISES

22.10.1 Level 1: Chapter Knowledge Exercises

Answer each of the What You Should Learn from This Chapter questions identified in the Introduction.

1. What are the attributes of a well-defined requirement?
2. When is a "requirement" officially recognized as a requirement?
3. What are some principles for preparing well-defined requirements?
4. What are some of the *common pitfall* in preparing requirements statements?
5. What is the syntactical structure of a requirement statement? List its sequences.
6. What *criteria* do you use to validate – test – a requirement?
7. Describe the decision process used to identify a requirement's verification method(s)?
8. What is an RVM?
9. How do you develop an RVM?
10. What is meant by requirements minimization? Why would you want to reduce the quantity of requirements?
11. What is the optimal number of requirements in a specification?

22.10.2 Level 2: Chapter Knowledge Application Exercises

Refer to www.wiley.com/go/systemengineeringanalysis2e

22.11 REFERENCES

DAU (2005), *Test and Evaluation Management Guide*, 5th ed., Ft. Belvoir, VA: Defense Acquisition University (DAU) Press. Retrieved on 1/16/14 from http://www.dtic.mil/cgi-bin/GetTRDoc?AD=ADA436591.

Doran, George T. (1981), "There's a S.M.A.R.T. way to write management's goals and objectives," *Management Review* (AMA FORUM), Vol. 70 No. 11: pp. 35–36. Cambridge, MA: MIT Sloan Management Review.

FAA (2003), Guide to Reusable Launch Vehicle Safety Validation and Verification Planning, Office of the Associate Administrator for Commercial Space Transportation, Version 1.0, Washington, DC: Federal Aviation Administration (FAA), Sept. 2003. Retrieved on 1/20/14 from http://www.faa.gov/about/office_org/headquarters_offices/ast/licenses_permits/media/VV_Guide_9-30-03.pdf.

INCOSE-TP-2010-006-02 (2015), *Guide for Writing Requirements*, Requirements Working Group (RWG), San Diego, CA: International Council on System Engineering (INCOSE).

ISO/IEC/IEEE 24765:2010 (2010), *Systems and software engineering—Vocabulary*, Geneva: International Organization for Standardization (ISO).

NASA SP-2007-6105 (2007), *System Engineering Handbook*, Rev. 1. Washington, DC: National Aeronautics and Space Administration (NASA). Retrieved on 6/2/15 from http://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/20080008301.pdf.

Gray, William S, and Sharp, Zerna (1946), *Dick and Jane Series*, Glenview, IL: Scott, Foresman, and Company.

SEVOCAB (2014), Software and Systems Engineering Vocabulary, New York, NY: IEEE Computer Society. Accessed on 5/19/14 from www.computer.org/sevocab.

23

SPECIFICATION ANALYSIS

When systems, products, and services are acquired, the System Acquirer typically provides a System Requirements Document (SRD) or Statement of Objectives (S00) with the formal Request for Proposal (RFP) solicitation. The SRD/S00 expresses the required set of capabilities and performance Offerors use as the basis to submit solution-based proposals. The challenge for System Acquirers and System Developers is to formulate, derive, and negotiate an SPS that:

1. Clearly, concisely, and completely bounds the *solution space.*
2. Is well understood by all Stakeholders—Users and End Users.
3. Establishes the basis for deliverable system, product, or service technical acceptance.

Our discussion in this section describes various Specification Analysis Practices. We explore various methods and techniques used to analyze specification requirements for completeness from several perspectives. We introduce common specification practice deficiencies and investigate methods for identifying, tracking, and resolving these deficiencies. We also consider semantic ambiguities of terms such as "comply" versus "conform" versus "meet" that System Acquirers and System Developers employ that do have literal significance in interpretation.

As the concluding chapter in the SYSTEM SPECIFICATION PRACTICES series, this chapter has two application contexts:

- Context #1—Provides guidance for reviewing specifications developed *externally* by the SRD or procurement specification.
- Context #2—Provides guidance for evaluating *internally* developed specifications—SPS or Entity Development Specification (EDS)—for proposal or procurement purposes.

23.1 DEFINITION OF KEY TERMS

- **Availability**—"A measure of the degree to which an item is in an operable state and can be committed at the start of a mission when the mission is called for at an unknown (random) point in time. See Inherent Availability (AI), Achieved Availability (AA), and Operational Availability (AO)." (DAU, 2012, p. B-18).
- **Efficiency**—"(1) The degree to which a system or component performs its designated functions with minimum consumption of resources" (SEVOCAB, 2014, p. 104) (Source: ISO/IEC/IEEE 24765:2010 (2010) – Copyright by ISO/IEC. Used by permission).
- **Maintainability**—"The ability of an item to be retained in, or restored to, a specified condition when maintenance is performed by personnel having specified skill levels, using prescribed procedures and resources, at each prescribed level of maintenance and repair." (DAU, 2012, p. B-131).
- **Portability**—"(1) ease with which a system or component can be transferred from one hardware or software environment to another." (SEVOCAB, 2014, p. 225) (Source: ISO/IEC/IEEE 24765:2010 (2010) – Copyright by ISO/IEC. Used by permission).

System Engineering Analysis, Design, and Development: Concepts, Principles, and Practices, Second Edition. Charles S. Wasson.

Companion Website: www.wiley.com/go/systemengineeringanalysis2e

- **Producibility**—"The relative ease of manufacturing an item or system. This relative ease is governed by the characteristics and features of a design that enables economical fabrication, assembly, inspection, and testing using available manufacturing techniques" (DAU, 2012, p. B-175).
- **Reconfigurability**—The ability of a system, product, or service configuration to be modified manually or automatically to support mission objectives.
- **Reliability**—"The ability of a system and its parts to perform their mission without failure, degradation, or demand on the support system under a prescribed set of conditions. See Mean Time Between Failure (MTBF) and Mean Time Between Maintenance (MTBM)" (DAU, 2012, p. B-189).
- **Serviceability**—"A measure of the degree to which servicing of an item will be accomplished within a given time under specified conditions" (DAU, 2012, p. B-202).
- **Supportability**—"A key component of availability. It includes design, technical support data, and maintenance procedures to facilitate detection, isolation, and timely repair and/or replacement of system anomalies. This includes factors such as diagnostics, prognostics, real time maintenance data collection, and Human System Integration (HSI) considerations (JCIDS Manual)" (DAU, 2012, p. B-216).
- **Survivability**—The capability of a system and its User to avoid or withstand a man-made hostile environment without suffering an abortive impairment of its ability to accomplish its designated mission (DAU, 2012, p. B-217).
- **Susceptibility**—"The degree to which a device, equipment, or weapon system is open to effective attack as a result of one or more inherent weaknesses. Susceptibility is a function of operational tactics, countermeasures, probability of an enemy threat, etc. Susceptibility is considered a subset of survivability" (DAU, 2012, p. B-217).
- **Sustainability**—"The ability to maintain the necessary level and duration of operational activity to achieve" a mission and its objectives. "Sustainability is a function of providing for and maintaining those levels of" readiness, personnel, expendables, and consumables necessary to support a System of Interest (SOI) (Adapted from DAU, 2011, p. B-258–259).
- **System Safety**—"The application of engineering and management principles, criteria, and techniques to optimize safety within the constraints of operational effectiveness, time, and cost throughout all phases of the system life cycle" (DAU, 2012, p. B-221).
- **System Security**—The level of protection that characterizes a system, product, or service's ability to *reject* or *deter* intrusion and access by external threats or unauthorized systems.
- **Testability**—"(3) degree to which a system or component facilitates the establishment of test criteria and the performance of tests to determine whether those criteria have been met. (SEVOCAB, 2014, p. 325) (Source: ISO/IEC/IEEE 24765:2010 (2010) – Copyright by ISO/IEC. Used by permission).
- **Transportability**—"The capability of material to be moved by towing, self-propulsion, or carrier through any means, such as railways, highways, waterways, pipelines, oceans, and airways. Full consideration of available and projected transportation assets, mobility plans and schedules, and the impact of system equipment and support items on the strategic mobility of operating military forces is required to achieve this capability" (DAU, 2012, p. B-232).
- **Usability**—"**(2)** ease with which a user can learn to operate, prepare inputs for, and interpret outputs of a system or component (SEVOCAB, 2014, p. 339) (Source: ISO/IEC/IEEE 24765:2010 (2010) – Copyright by ISO/IEC. Used by permission).
- **Vulnerability**—"The characteristics of a system that cause it to suffer a definite degradation (loss or reduction of capability to perform the designated mission) as a result of having been subjected to a certain (defined) level of effects in an unnatural (man-made) hostile environment. Vulnerability is considered a subset of survivability" (DAU, 2012, p. B-239).

23.2 ANALYZING EXISTING SPECIFICATIONS

Chapters 19–22 focus on how a System Acquirer might develop an SRD, SPS, or EDS for procurement of a system, product, or service or the System Developer might develop lower-level EDSs. However, *what if a specification already exists*? How does the System Acquirer or a System Developer candidate analyze the specification for *completeness*, *reasonableness*, and *feasibility*?

There are two key contexts regarding analysis of specifications:

1. Acquirer verification *prior to* formal solicitation.
2. System Developer analysis during the proposal process and following Contract Award (CA)

Let's investigate these contexts further.

23.2.1 System Acquirer Role Perspective

System Acquirers start by reducing the project and technical risk of the procurement action. *How can this be*

accomplished? By releasing a high-quality draft specification that *accurately, precisely*, and *completely* specifies and bounds the *solution space* system, product, or service. Key review questions to ask might include:

1. Have all User System Deployment Phase; System Operations, Maintenance, and Sustainment (OM&S) Phase; and System Retirement/Disposal Phase (Figure 12.2) Stakeholder requirements been adequately identified, prioritized, scoped, and specified?
2. Have we bounded the correct *solution space* within the *problem space*?
3. Have we identified the *right* system to fill the prescribed *solution space* and cope with its Operating Environment?
4. Does this specification *accurately*, *precisely, and completely* specify the *selected solution* space?
5. If we procure a system based on these requirements, will the deliverable work product satisfy the User's intended operational needs documented in the Operational Requirements Document (ORD) or Capability Development Document (CDD)?
6. Can the system specified be developed within the Total Cost of Ownership (TCO)—such as acquisition, deployment, Operations, Maintenance, and Sustainment (OM&S), and retirement/disposal costs—budgets that are available?

What happens if you inadequately address these and other questions? Later, if it is determined that the requirements have *latent defects* such as errors, deficiencies, or omissions, the cost to modify the contract can be very expensive assuming the other party agrees to the modifications. To *minimize* specification risk, System Acquirers often release a pre-solicitation draft specification to qualified candidate RFP Offerors for comments.

23.2.2 System Developer Role Perspective

In contrast, the System Developer must reduce contract cost, schedule, and technical and technology risk. To do this, specification analysis must answer several key review questions that include:

1. Do we fully understand the scope of the work effort we are agreeing to perform?
2. Do the SRD requirements, as stated, specify a system that satisfies the User's operational needs? If not, what approach must we use to inform them?
3. Have we thoroughly investigated and talked with a representative cross-section of the Stakeholder community to validate their requirements and needs?
4. Do we understand the *problem space* the User is attempting to solve by procuring this system? Does the specification specify and bound the *problem* or a *symptom* of the problem?
5. Can the requirements, as stated, be verified within reasonable expectations, cost, schedule, and risk?
6. Do these requirements mandate technologies that pose *unacceptable* risks?

23.3 SPECIFICATION ASSESSMENT CHECKLIST

When we engineer systems, the general mind-set is to *propose* and *develop* solutions that solve User *solution spaces* within *problem spaces.* The problem with this mind-set is that it lacks a central focal point that captures what the User needs or seeks. For example: *Is the User concerned about Growth*, *Reliability*, *Maneuverability*, and so forth? If you do not understand (1) what the User needs and (2) what priorities they place on those needs, you are just going through a "check-the-box" exercise. So, *how do SEs avoid this mind-set*?

This section introduces SPS topics based on User objectives that are often key drivers in System Development. These objectives form the basis for proposal and System Development activities responding to System Acquirer formal solicitations and contracts. The broad, far-reaching ramifications of technical decisions made in support of these objectives clearly focus on the need to integrate Subject Matter Experts (SMEs), as key members of System/Product Teams (SPTs/PDTs), to ENABLING SYSTEM development. The discussions that follow scope the context of each objective.

Author's Note 23.1

Essential Requirements

- First, specifications presumably document all of the System Acquirer's *essential* requirements (Principle 19.7). You can reach a point where you "can't see the forest for the trees," as such immersion in the details tends to obscure what is important to the User. Once you read a specification, talk with the User via System Acquirer contracting protocol to determine the key objectives that matter most and impact System Developer decision-making.
- Second, collaborate with the Users to identify and prioritize the *essential* capability requirements.

If you are part of the *ad hoc, endless loop* Specify-Design-Build-Test-Fix (SDBTF)-Design Process Model (DPM) Engineering Paradigm (Chapter 2), *avoid* the notion that you can use the types of requirements that follow and translate them into specification requirements.

Remember that most of the discussions that follow address System Attributes (Chapter 3), not *capabilities* per se.

23.3.1 Single Use, Reusable, and Multi-Purpose System Requirements

Principle 23.1

System Applications Principle

Specification requirements should be written to support any one of three types of system applications: *single use, reusable*, and *multi-purpose*.

User mission applications of a system ultimately drive its specification requirements. Most systems are designed for three primary types of mission applications: *single use, reusable*, and *multi-purpose*. Let's clarify these terms:

- **Single Use Systems** are designed for a one-time use. Examples include: satellites, rockets, missiles, munitions, medical syringes, and batteries.
- **Reusable Systems** are systems *configured* to perform numerous missions with periodic maintenance. Examples include: automobiles, computers, aircraft, flashlight, rechargeable batteries, and homes.
- **Multi-Purpose Systems** represent Reusable Systems that are *reconfigurable* to accommodate several types of mission applications. For example: aircraft can be *reconfigured* to carry passengers or cargo, computers can be reconfigured to execute various types of software applications, and external hardware devices such as printers or modems.

Users acquire most systems and products for reusable applications. Reusable SYSTEMS and PRODUCTS must be designed for usability; Reliability, Maintainability, and Availability (RMA); and sustainability over a specified number of operational usage cycles. Therefore, reusable applications requirements require particular attention in areas such as *modularity* and *interchangeability*. In contrast, *single-use* applications may require a focus on cost per unit, reliability, and other factors results.

23.3.2 Efficiency Requirements

Some SYSTEMS and PRODUCTS require focused consideration on *efficient* utilization of resources. In these cases, *efficiency* requirements must be established.

23.3.3 Effectiveness Requirements

One of the key contributors to User satisfaction is a system, product, or service's degree of *effectiveness*. Did the missile locate, impact, and destroy the target? Does the flight simulator improve pilot *effectiveness*? Establish requirements where system effectiveness is a key determinate for mission success.

23.3.4 Usability Requirements

Durable Systems and Products require a high usability. *Usability*, as viewed by the User and End User, is often *vague* and *ambiguous*. Classic requirements include: *user-friendly interfaces … easy to use … easy to understand … easy to control … easy to drive over rough terrain … easy to lift … easy to carry …* and so forth. From an SE perspective, it is critical that the *usability* requirements be *accurately* and *precisely* bounded in SPS and EDS requirements.

Rapid prototypes may need to be developed to present to Users for *usability* evaluation and constructive feedback. User decisions must then be captured as graphical displays or text requirements. Application examples include:

- Special considerations for the physically challenged.
- Special equipment for *ingress* and *egress* to/from a vehicle, building, etc.

23.3.5 Comfort Requirements

Vehicular systems developed for Users such as operators and passengers for travel purposes over long periods of time require a *level of comfort* related to fatigue, boredom, hygiene, and so on. As in the Usability objective, requirements must be explicitly identified, scoped, and bounded. *Comfort* requirements are key drivers in the development of systems such as homes, vehicles, spacecraft, and offices.

23.3.6 Compatibility and Interoperability Requirements

Systems and products that require mechanical interfaces with external systems require *compatibility* requirements. Where those or wireless interfaces require unilateral or bi-directional communications that must be understood and encoded/decoded by both systems, *interoperability* requirements must be established. If the Interoperability objective is selected, strict interface design standards and protocols must be established and controlled for consistency.

23.3.7 Growth and Expansion Requirements

Systems, products, and services often require the capability to be expanded to accommodate future *upgrades*. This may require additional processing or propulsion power, flexibility to increase storage capacity of expendables or consumables, increased data communications ports or bandwidth, or organizational or geographical expansion.

A Word of Caution 23.1

Reliability, Maintainability, and Availability (RMA) Requirements

Please note that the purpose of the RMA requirements discussion that follows is to highlight and recognize the potential need for requirements in these areas. Employ a qualified, competent expert in RMA to assist you in making these decisions. Remember that RMA requirements have major cost and safety ramifications!

23.3.8 Reliability Requirements

Every system, product, and service has an intrinsic value to its Stakeholders in being able to support Enterprise missions and objectives. Depending on the operating condition of a system or product, mission, and OPERATING ENVIRONMENT, System Reliability (Chapter 34) relative to achieving mission success may be a critical issue.

23.3.9 Maintainability Requirements

Single-use, *multi-use*, and *multi-purpose* systems require some form of Maintainability objective (Chapter 34). This may include *preventive maintenance, corrective maintenance,* calibration, upgrades, and refurbishment. Key *maintainability* requirements considerations include maintenance operator access and clearances for hands, arms, tools, and EQUIPMENT.

Additional considerations include the availability of electrical power, need for batteries, or electrical generators, external air sources for aircraft while on the ground, and so on. These considerations emphasize the need for a Concept of Operations (ConOps) Maintenance Concept (Chapter 6) to provide a framework for deriving Maintainability requirements.

23.3.10 Availability Requirements

The first criterion for systems, products, and services success is *system readiness* to perform a mission when called upon.

Establish *availability* requirements commensurate with the available budget and mission. This includes implementation of Daily Operational Readiness Tests (DORTs) or User-Initiated Test (UIT), Built-In Tests (BITs), Built-In Test Equipment (BITE), indicators, and display meters. These activities and design considerations can provide early indications of potential problem areas and thus enable corrective action to be taken in advance of mission needs.

23.3.11 Producibility Requirements

Systems and products planned for production must be PRODUCIBLE in a cost-effective manner that:

- Has acceptable risk.
- Has *repeatable* and *predictable* processes and methods.
- Can be produced within budgets and schedules at a reasonable profit.

Engineering development of *first articles* are often capability demonstrators that may include lesser quality materials, technologies, improvised components, and "add-on" instrumentation to support performance verification. Production items may not require these additional items or weight that limit performance and increase cost. A *producibility* objective often drives a search for alternative materials and processes to reduce cost and risk and improve or maintain system performance.

23.3.12 Storage Requirements

Most systems require some form of storage albeit for energy, operator's manuals, mission data storage, tools and equipment, etc. Additionally, access to the storage areas or compartments by the System Operator is critical. For example, the glove compartment of an automobile is inside the front dash, not in the trunk. Establish storage requirements commensurate with the mission, system, and PERSONNEL Element.

23.3.13 Integration, Test, and Evaluation Requirements

Modular systems and products that undergo multiple levels of integration and test require a Design for Integration, Test, and Evaluation objective. Ideally, you *isolate* each Configuration Item (CI) and test it with actual, simulated, stimulated, or emulated interfaces. If the system or product is to be integrated at various facilities, special interface considerations should be given to physical constraints and EQUIPMENT, and the tools available should be investigated.

Finally, systems and products must be designed to facilitate multi-level *System Verification* and *Validation* (V & V). This requires the incorporation of temporary or permanent test points and access ports or doors to support calibration and alignments.

23.3.14 Verification Requirements

Once the system or product is integrated, tested, and ready to be verified, designers must consider *how* the item will be *verified*. Some requirements can be physically verified; others may not. Establish a Verification objective to ensure all data required for verification are *accessible* and can be easily *measured*.

23.3.15 Enabling System Requirements

Multi-use applications require continuing support throughout their planned life cycle. Enabling System requirements ensure that appropriate design considerations are given for replenishment of expendables and consumables and for corrective and preventive maintenance.

Thus, Enabling System requirements are critical for establishing Mission Support requirements.

23.3.16 Deployment Requirements

Many systems, products, and services require Deployment Requirements to support deployment or shipment to duty station mission areas. Design considerations include tie-downs, safety chains, anchors, accelerometer sensor instrumentation packages, controlled environmental atmospheres, and shock and vibration proofing. Additionally, en route deployment constraints such as bridge heights and maximum weights, road grades, and hazardous waste routes constraints must be factored into SPS (Table 20.1) Section 3.6 Design & Construction Constraints decisions.

23.3.17 Transportability Requirements

Vehicular systems, military troops, and others require design considerations based on *transportability* objectives. This requires understanding who/what is to be transported, what volumetric space and carrying capacity is required, what *tie-downs*, *lift points, or safety lights, and markers* are required, how the cargo is to be secured, who requires what access to the cargo and when, and what protection mechanisms are required from environmental and Human System threats.

23.3.18 Mobility Requirements

Vehicular systems and military troops are Users that require design considerations based on a mobility objective. The requirements considerations include *how* the system will be physically moved, how often, and how the system is to be secured when it reaches its destination.

23.3.19 Maneuverability Requirements

Systems often require the capability to maneuver through their Operating Environment. This requires piloting/steering mechanisms that enable the operators to physically or remotely move the vehicle from one location, navigational systems, change orientation relative to a frame of reference, or make directional vector-headed changes.

23.3.20 Portability Requirements

Some systems and products must be developed with a Portability objective. *Portability* requirements have two key contexts: (1) *physical properties* and (2) software:

- Portability physical properties of a system or product such as the acceptable size and weight that a human can lift, move, or transport.
- Software portability refers to the ease of integrating and executing CSCIs on various types of computer systems with minimal adaption.

23.3.21 Training Requirements

Most systems and products require some form of Training objective. Where this is the case, training is a Critical Operational Issue (COI), not only for the students but also for the instructors, general public, and environment. Additionally, scoring and debriefs of training sessions are important. Therefore, the ConOps Sustainment Concept (Table 6.1) becomes a key input into system and product requirements. In some cases, systems and products may require re-modification to include instructor controls and scoring results.

23.3.22 Reconfigurability Requirements

Some systems and products are designed to accommodate a variety of missions as well as a quick turnaround between missions. Therefore, some may have to be reconfigurable within a specified timeframe.

23.3.23 Security-and-Protection Requirements

Various systems and products require a *security-and-protection* objective that limits system or product access to only *authorized* individuals or organizations with a "need to know" justification. Requirements considerations could include: layers of armor, Internet firewalls, authorized accounts, passwords, and encryption.

23.3.24 Vulnerability Requirements

Systems that operate in hostile threat environments or that could be *misused* or *abused* by the operator(s) should have a Vulnerability objective. This applies to buildings, safes, vehicles, computers, and electrical circuits.

When a system, product, or service is anticipated to be *vulnerable* to Operating Environment threats, mission analysis and UC analysis should identify the threats and threat scenarios and prioritize design capabilities to *resist or minimize* the effects of those threats. The generalized solution acknowledges the interaction; the specialized solution incorporates key capabilities or features to protect the system and its operators from external threats.

23.3.25 Lethality Requirements

Some systems such as munitions and missiles are intended to penetrate vulnerable areas and destroy mechanisms that enable the targeted system to survive. The *lethality* objective focuses on the system design and material characteristics that enable a system such as a missile to achieve this objective against an external system.

23.3.26 Survivability Requirements

Some systems and products are required to operate in harsh, hostile OPERATING ENVIRONMENTS. They must be capable of surviving to complete the mission and, as applicable, return safely. Examples include thermal insulation, layers of armor, elimination of Single Points of Failure (SPF), (Chapters 26 and 34). Systems such as these require a Survivability objective.

23.3.27 Safety Requirements

The application of SE requires strict adherence to laws, regulations, and engineering principles and practices that promote the safety of system and product stakeholders—the operators, maintainers, general public, personal property, and environment. *Safety* requirements focus on ensuring that systems, products, and services are: safe to deploy; operate, maintain, and sustain; and dispose. This includes not only the physical product but also establishing training and instructional procedures, hazards, cautions, and warnings, and potential consequences for failure to perform.

23.3.28 Disposal Requirements

Systems and products that employ Nuclear, Biological, or Chemical (NBC) materials ultimately wear out, become exhausted, damaged, or destroyed intentionally or by accident. In any case, the system or product requires a Disposal objective. This includes mechanisms for monitoring and removal of hazardous materials such as NBC substances or traces. For items that can be reclaimed and recycled, special tools and equipment—categorized as Peculiar Support Equipment (PSE)—may be required.

23.4 SPECIFICATION ANALYSIS METHODS

Given the System Acquirer and System Developer perspectives, *how do they approach analysis of the specification*? The answer encompasses the methods and techniques identified in earlier chapters. Due to the broad scope of this answer, we will briefly address some high-level approaches you can apply to specification analysis.

Examine the outline structure for missing sections and topics that are crucial to developing a System to fill the *solution space* and solve all or a portion of the User's/End User's *problem, issue, or concern.*

23.4.1 System Requirements Analysis (SRA)

Perform an SRA to understand what the system is expected to do. Ask key questions such as:

1. Does the list of requirements appear to be generated as a feature-based "wish list" or reflect structured analysis such as Model-Based Systems Engineering (MBSE)?
2. Do the requirements follow the Requirements Statement Validation Criteria (Table 22.4)?
3. Do the requirements appear to have been written by a seasoned SME or by a semi-knowledgeable person needing a task?
4. Do the requirements adequately capture User operational needs? Are they *necessary* and *sufficient*?
5. Do the requirements *unnecessarily* constrain the range of viable solutions? (Figure 21.5)
6. Are all system interface requirements identified?
7. Are there any undefined annotations (TBD or TBS) remaining in the specification?
8. Are there any Critical Operational/Technical Issues (COIs/CTIs) that require resolution or clarification with the System Acquirer, User, or End User?

23.4.2 Perform Engineering Graphical Analysis

1. Based on the SPS or EDS requirements, as stated, can we draw a simple graphic such as a Context Diagram (Figure 8.1) or Architecture/System Block Diagram (ABD/SBD) (Figure 20.5) of the SYSTEM and its interactions with its OPERATING ENVIRONMENT?
2. Are there any obvious inconsistencies in the graphic that are not specified as requirements in the specification such as missing requirements (Figure 20.3)?

23.4.3 Hierarchical Analysis

1. Are there any *misplaced, missing, duplicated, or conflicting* requirements (Figure 20.3)?
2. Are the requirements positioned and scoped at the proper levels of abstraction? (Figure 8.4)

23.4.4 Technology Analysis

Do the specification requirements indicate a *willingness* or *unwillingness* by the System Acquirer to consider and accept new technologies or solutions?

23.4.5 Competitive Analysis

Do the specification requirements inappropriately favor or target a competitor's products, services, or organizational capabilities?

23.4.6 Modeling and Simulation (M & S) Analysis

If appropriate, is it worthwhile to develop models and simulations (Chapters 10 and 33) as decision aids to analyze system performance issues (COIs/CTIs)?

23.4.7 Requirements Verification Analysis

1. Are there any requirements that are *unreasonable*, *unverifiable*, *cost prohibitive*, or too *risky* using the verification methods specified?
2. Does verification require any special test facilities, tools, equipment, or training?

23.4.8 Requirements Validation Analysis

When SEs analyze specifications, especially those prepared by external Enterprises, most engineers *presume* the specification has been prepared by someone who:

1. Understands the User's *problem space* and *solution space(s)*.
2. Accurately analyzes, translates, and articulates the *solution space* into requirements that can be implemented economically with *acceptable* risk, and so forth.

Exercise *caution* with this mind-set! *Avoid* assuming anything until you have validated the specification requirements.

23.5 SPECIFICATION DEFICIENCIES CHECKLIST

If you analyze specification requirements practices in many Enterprises, there are a number of common deficiencies that occur frequently. These include:

- Deficiency 1: Failure to budget reasonable time for specification analysis.
- Deficiency 2: Requirements traceability.
- Deficiency 3: Failure to follow a standard specification outline.
- Deficiency 4: Inadequate specification development approach.
- Deficiency 5: Lack of specification and requirements ownership.
- Deficiency 6: Referenced versus Applicable Documents.
- Deficiency 7: Specifying broad references.
- Deficiency 8: References to unapproved specifications.
- Deficiency 9: Use of ambiguous words and phrases.
- Deficiency 10: Missing fault detection, isolation, containment, & recovery requirements
- Deficiency 11: Over-/Under-specification of Requirements
- Deficiency 12: Specification change management.
- Deficiency 13: Requirements applicability—configuration effectivity.
- Deficiency 14: Failure to identify verification methods
- Deficiency 15: Failure to define technical terms
- Deficiency 16: Failure to identify assumptions

Deficiency 1: Failure to Budget Reasonable Time for Specification Analysis

One of the ironies of system development is failure to allocate the proper amount of time to reasonably analyze or develop specifications.

Wasson's Task Significance Principle

Principle 23.2 The time allocated by management for most program and technical decision-making tasks is inversely proportional to the significance of the tasks or decision to the deliverable product, User, or Enterprise.

Three of the most crucial specification analysis tasks during a proposal effort are understanding:

1. What problem the User is trying to solve.
2. What system, product, or service does the System Acquirer's formal solicitation's SRD/S00 specify.
3. What you have committed your Enterprise to via the draft SPS submitted as the proposal response.

Despite their significance, these three tasks often fall back in the priority list behind multi-level managerial briefings, which are important, and other "administrative" tasks. Seriously spend the appropriate amount of time getting specification requirements right!

Deficiency 2: Requirements Traceability

Enterprises and Engineers often review specifications without ever asking the most fundamental question. *Where did these requirements originate and what was the methodology for allocating and flowing them down to this specification* (Principle 19.13)? Before you spend time reviewing a specification, you should authenticate the answer to this question.

Deficiency 3: Failure to Follow a Standard Specification Outline

Many specification issues are traceable to a lack of commitment to establish and employ standard specification development outlines and guidelines (Principle 19.3). Standard outlines represent industry *best practices* and organized *lessons learned* that reflect problem areas or issues that someone else has encountered and must be corrected in future efforts. Over time, they incorporate a broad spectrum of topics that may or may not be applicable to all programs. The natural tendency of SEs is to delete non-applicable sections of a standard specification outline. Additionally, management often *dictates* that specific topics be deleted because "We don't want to bring it to someone's attention that we are not going to perform (topic)."

The reality is standard outlines include topics that are intended to *keep you out of trouble*! A cardinal rule of system specification practices requires you to provide rationale as to why a standard outline topic is not applicable to your Project. The rationale communicates to the reader that you:

1. Considered the subject matter.
2. Determined the topic is not relevant to your system development effort for the stated rationale.

Therefore, if someone determines later that "Yes, it is relevant," you can correct the deficiency and applicability statement. This applies to plans, specifications, and other types of technical decision-making documents.

Problems arise when SEs *purposefully delete* sections from a standard outline. Once deleted, the section is "out of sight, out of mind." Since contract success is dependent on delivering a properly designed and developed system on schedule and within budget, you are better off identifying a topical section as "Not Applicable." Then, if others determine that it is applicable, at least you have some lead time to take corrective action *before it is too late*.

If you follow this guideline and go into a System Requirements Review (SRR), (Chapter 18) any "Not Applicable" (N/A) issues can be addressed at that time. All parties emerge with a record of agreement via the conference minutes concerning the applicability issue.

Cost-to-Correct Specification Requirements Errors and Omissions

Author's Note 23.2 Remember that the cost-to-correct specification requirements *latent defects* increases almost exponentially (Table 13.1) throughout the System Development Process (Figure 12.2).

Deficiency 4: Inadequate Specification Development Approach

Some Engineers pride themselves in being able to quickly "assemble a specification" by duplicating legacy system specifications (Specification Reuse Approach—Chapter 20) without appropriate mission, operational, and system analysis. *Guess what?* Management takes great pride in the practice as well. Ahead of schedule, life is rosy! Later, SEs discover that key requirements were *overlooked* or *ignored*, (Figure 20.3) were not estimated, and have significant cost to implement. *Guess what?* Management is very unhappy!

Where *legacy precedented* systems exist and are used as the basis to create new specifications with only minor modifications, the *practice of plagiarizing existing specifications* may be acceptable. However, be cautious of the practice! Learn when and how to apply specification reuse effectively.

Deficiency 5: Lack of Specification and Requirements Ownership

Specification requirements are often *ignored* due to a *lack of ownership*. Two SEs argue that each thought the other was responsible for implementing an SPS or EDS requirement. Every requirement in every specification should have an assigned owner accountable for its analysis, implementation, verification, traceability, and proposed updates (Principle 19.8). The challenge is: *if you review a specification and need to clarify a requirement, no one seems to know who is accountable for its origination*.

Deficiency 6: Referenced Documents versus Applicable Documents

Specification Referenced Documents Principle

Principle 23.3 Specification Section 2.0 "Referenced Documents" lists only documents explicitly referenced by Section 3.0 "Requirements" and 4.0 "Qualification Provisions"; remove all other non-referenced documents.

Referenced documents refer to those documents—namely, specifications and standards—explicitly specified in the SPS and EDS Section 3.0 "Requirements" and listed in Section 2.0 "Referenced Documents" or "Applicable Documents." In contrast, Applicable Documents are those containing information germane to the topic but are not referenced by the specification. *Do not list* these documents in a specification's Section 2.0 "Referenced Documents."

Deficiency 7: Specifying Broad References

Principle 23.4

Broad References Principle

Avoid broad, general references to external documents. Always, specify document title, version, release date, section number(s) and section title(s).

Specification writers spend the bulk of their time word-smithing, tweaking, and correcting documents and very limited time, if any, performing systems analysis—even less on specification references. References are often inserted at the last minute. *Why*?
Typically, those same SEs do not have the time to thoroughly research the references. They make broad, non-specific references such as "in accordance with MIL-STD-1472" intended as a "placeholder" because:

1. Management is demanding completion.
2. We'll "clean it up" later.
3. We don't understand what the reference means, but it sounds good; "saw it in another spec one time so let's use it."

Since this is a specification and the System Developer is required by contract to implement the provisions of the SPS, are you prepared to *pay the bill* to incorporate all provisions of MIL-STD-1472? Absolutely not! With luck, the problem may work itself out during the proposal phase via the Offeror formal Question and Answer (Q&A) process.
This problem is a challenge for the System Acquirer and System Developers.

1. The referenced document may be outdated or cancelled. Professionally speaking, this can be embarrassing for the Enterprise and specification developer.
2. Unskilled specification writers—despite 30 years of experience in other non-topic areas—may inadvertently make technical decisions that may specify requirements that have legal, safety, and risk ramifications.
3. System Developers often *fail* to properly research RFP references, thereby costing significant amounts of money to implement the reference as stated in the SPS.

Heuristic 23.1 Undocumented Verbal Agreements

Undocumented verbal agreements and requirements vaporize when either party gets into trouble or a dispute emerges.

A Word of Caution 23.2

Thoroughly Analyze and Understand RFP Reference Requirements

Thoroughly research RFP references, formally request System Acquirer (role) clarifications or confirmations, and then document in brief form what the reference explicitly requires. Remember that *undocumented comments are magically forgotten by the source when things go wrong*!

Deficiency 8: References to Unapproved Specifications
When the System Developer's project plans to develop several multi-level EDSs, the efforts must be synchronized. One of the ironies of specification development is a perception that multi-level specifications can be written simultaneously to cut development time. This is an *erroneous perception* that ultimately leads to technical chaos, conflicts, and inconsistencies! The sequencing and approval of specification development at lower levels is dependent on maturation and approval of higher-level specifications. There are ways of accomplishing this, assuming teams at multiple levels communicate well and mature decisions quickly at higher levels.

Deficiency 9: Use of Ambiguous Words and Phrases
Specification writers are notorious for "writing" requirements statements that employ ambiguous words that are open to interpretation (Principle 19.9). In accordance with specification practices that promote explicitness, ambiguous words and terms should be avoided or defined. Consider the following example:

Example 23.1

Ambiguous Customer Specification Phrases

Simulation community specifications often include *ambiguous* requirements statement wording such as "realistic representation of the real world" and "effective training."

The challenge for the System Acquirer and System Developer is: *How do you know when a "realistic representation" or "effective training" has been successfully achieved*? The answer is to explicitly define the terms. The risk is: What subjective criteria will the User, subconsciously in their minds and perceptions, use to determine successful achievement of the requirements? (NAVAIR TSD, 2014) provides examples of ambiguous terms and phrases.

Deficiency 10: Missing Fault Detection, Isolation, Containment, & Recovery Requirements
Most SEs write specifications for the "ideal" world. The reality is that systems and interfaces fail, sometimes with catastrophic consequences. When specifications are written, make sure the requirements

are specified to address Normal, Abnormal, Emergency, and, if appropriate, Catastrophic Operations (Figure 19.5). This requires robust architectural solutions (Chapter 26) that enable the SYSTEM to tolerate and/or recover from the following types of faults:

1. External interface faults.
2. Internal faults resulting from component faults and internal interface faults.

For mission critical systems that involve human life, public safety, property loss, and the environment, specifications often fail to specify requirements for detecting, isolating, and containing faults (Figure 26.8). Where this is the case, fault detection, isolation, and containment requirements need to be specified as well as requirements for recovering from external and internal faults (Figure 10.17).

Deficiency 11: Over-/Under-specification of Requirements

Specification developers are often confronted with *uncertainty* regarding the adequacy of the requirements. Specification requirements can be *over-specified* or *underspecified* (Figure 21.5) The way to *avoid* over-/under-specification involves four criteria:

1. Identify only essential requirements (Principle 19.7).
2. Focus on specifying the SYSTEM or ENTITY's performance envelope such as boundaries for performance-based capabilities.
3. *Avoid* specifying *how* the performance-based capability envelope is to be implemented (Principle 19.6).
4. *Avoid* specifying capabilities more than one level below the SYSTEM or ENTITY's level of abstraction.

Deficiency 12: Specification Change Management

Principle 23.5

Decision-Making Reference Principle

Under standard operating practices, project personnel perform to current, baselined work products that have been: (1) *approved* for release and decision-making and (2) communicated to all stakeholders.

Once a specification is approved, baselined, and released, changes must be formally approved and communicated to Stakeholders. Because of a lack of communications, two problems can occur:

- First, program and technical management often lack an appreciation of the need to communicate specification changes to project personnel. Ironically, the *technical integrity* of the project is at *risk*, which is the very topic management seems to be averse to.
- Second, when management does communicate changes, Project personnel often *ignore* announcements about specification changes such as Specification Change Notices (SCNs) and continue to work with a previous version.

System *integrity* demands *formal* change management process discipline to track approved specification updates, incorporate changes immediately, and notify all stakeholders of the latest changes. This includes proper versioning to enable stakeholders to determine currency.

Deficiency 13: Requirements Applicability—Configuration Effectivity

Principle 23.6

Specification Effectivity Principle

When specifications apply to different model numbers, versions, or blocks serial numbers, explicitly designate each requirement with the appropriate effectivity.

Some specification requirements may only be applicable to specific configurations and units—that is, configuration effectivity. Where this is the case, label the requirement as only applicable to the Configurations A, B, C, etc., or Serial Numbers XXXX–YYYY. If this occurs, include a statement to the reader on the cover and in the Section 1.0 Introduction that this specification applies to configurations A, B, and C and Serial Number effectivity XXXX–YYYY.

A Word of Caution 23.3

Multi-Purpose Product Model Specifications

Realistically, this can become a very complicated challenge over time as models change. Enterprises often have "core" requirements for a product line with customized elaborations for variations for a specific product model. Apply *Systems Thinking* (Chapter 1) concerning the ramifications of these decisions and avoid their unintended consequences (Principle 24.20).

Deficiency 14: Failure to Identify Verification Methods

Every specification requirement must be verified. Yet, the identification of verification methods is one of the last activities to be performed. As a result, most specifications are deficient in specifying how the system or product will be verified, especially early in the specification's development. A showstopper rule should be that no specification is baselined until every requirement is assigned at least one or more verification methods (Principles 13.11 and 22.4).

Deficiency 15: Failure to Define Technical Terms
Specifications should be written for their Users using language and terminology familiar to them. Specifications are often deficient in terms of technical terms that may be ambiguous and have multiple meanings. Where this is the case, assumptions should be documented as illustrated in Table 20.1 Specification Section 6.2.

Deficiency 16: Failure to Identify Assumptions
Underlying most specifications is a set of assumptions that are never documented. Where this is the case, assumptions should be documented as illustrated in Table 20.1 Specification Section 6.3

23.6 RESOLUTION OF SPECIFICATION COI/CTI ISSUES

Specifications often contain abstract or ambiguous requirements that require *clarification* to ensure the proper *interpretation and understanding of the requirement.* Some requirements, however, raise COIs/CTIs that present major challenges to implementation. The issues may reflect technical, technology, cost, schedule, or support risks as well as TBDs, and so on.

Requirements issues and the need for clarifications occur *before* and *after* Contract Award (ACA). Let's briefly explore the handling of issues during these two timeframes.

23.6.1 Requirement Issues Resolution Prior to CA

System Acquirers often release the draft version of an SRD as part of formal solicitation such as a RFP for review and comment. The draft SRD may contain various specification requirements issues and the need for clarifications.

The first step of any specification analysis should be to *identify* and *tag* all requirements requiring clarification—technical, cost, schedule, technology, and support risks, and so on. Remember that publicly surfacing issues and clarifications during the RFP process may potentially tip competitors regarding your proposal strategy and enable them to gain competitive insights into your solution. Therefore, the proposal team must make a decision regarding which issues to submit for clarification and which issues to give additional internal analysis and/or follow-up.

The key point is that Offerors—System Developers—must thoroughly analyze, scrutinize, and resolve all requirements issues and clarifications *before* they submit their specification as part of the proposal and sign a contract.

If the SRR has not been conducted (Chapter 18), there may be an opportunity to address the need for clarifications or to correct deficiencies. Even then, some System Acquirers may be reluctant to agree to specification changes via contract modifications.

From the System Acquirer's perspective, the need to change always goes back to the proposal response prior to CA "… why wasn't the matter addressed at that time or during contract negotiations?" The problem is exacerbated if the Offeror (System Developer) voluntarily proposed specification requirements language without adequate analysis or consideration. The situation potentially has degrees of Enterprise, technical, and professional embarrassment.

23.6.2 Requirement Issues Resolution ACA

Requirements issue resolution ACA varies by contract and Acquirer. The *willingness* to modify specification requirements language may depend on how recently the CA has occurred.

23.6.3 Requirement Issues Resolution Prior to the SRR

HUMAN SYSTEMS, Enterprise and Engineered, even with the best of intentions, are not perfect. Inevitably, every contract SPS has blemishes, degrees of goodness, and strong and weak points. Although the degree of goodness has an academic connotation, "goodness" resides in the minds and perceptions of the System Acquirer and System Developer. Remember the cliché, *one person's work of art may be viewed by another person as an unorganized set of thoughts*.

Discussions by both parties reach a point whereby *willingness* to entertain contract modification to eliminate specification blemishes or latent defects are rejected. The System Acquirer may agree in principle to changes, but is reluctant to request changes due to the System Developer taking advantage of the situation via revised cost changes.

Conversely, the System Developer may want changes, but the System Acquirer is *unwilling* to allow any changes for fear of the unknown that may result from the changes. Even when both parties agree, there may be SPS *latent defects* (Principles 13.2 and 13.7) that lie *dormant* and go *undiscovered* until late in the System Development Phase of the contract. The best that can occur is for both parties to accommodate each other's wishes at no cost, assuming that is the appropriate and reasonable solution.

Regardless of the scenario, you may have a situation where the SPS contains defects, deficiencies, or errors and the System Acquirer refuses to modify the contract. *What do you do?*

One solution is to create an electronic System Design Notebook (SDN); some people refer to this as a Design Rationale Document (DRD). *Why do you need an SDN or DRD?* You need a mechanism to record design assumptions and rationale for requirements allocations and design criteria.

Under the Terms and Conditions (Ts&Cs) of the contract, the System Developer must perform and comply with the

SPS requirements that are subsequently flowed down to lower-level EDSs. Therefore, lacking a definitive set of SPS requirements, you may want to consider an *at risk* solution that expresses your Enterprise's interpretation of the ambiguous SPS requirements. Based on contracting protocol, an Engineering Change Proposal (ECP) should be provided to the System Acquirer Contracting Officer (ACO) for review and follow-up action.

Importance of Clarifying Requirements before Signing the Contract

Author's Note 23.3 The preceding discussion highlights the need to *think smartly up front* about requirements and *avoid* this situation. You will find *impatient* people who insist that you move on and not worry about interpretations. "Besides, it's perfectly clear to me!" Beware! Any time investment and *effort* spent up front clarifying SPS requirements before they become contract obligations will be significantly *less risk* and *costlier* than ACA. Remember that Figure 13.2 includes all types of latent defects—design errors, flaws, and specification requirements deficiencies (Figure 20.3).

When you conduct the SRR (Chapter 18), any residual SPS latent *defects*—deficiencies, errors, deficiencies, and so forth—should be addressed with the System Acquirer. The requirements defects discussion and decisions should be recorded in the SRR meeting minutes. If the System Acquirer refuses to allow corrections via contract modification, they may acknowledge via the System Acquirer Contracting Officer (ACO) your chosen approach. Therefore, consider documenting your design assumptions in a *Design Rational Document* (DRD) and include open distribution or accessibility to the System Acquirer.

Resolution of Specification Requirements Issues "Up Front"

Author's Note 23.4 Every contract and situation is different and requires decision-making on its own merits. Professionally and technically speaking, the System Acquirer and System Developer should emerge from the SRR with no outstanding specification issues. The reality is:

- The System Acquirer may have to settle for a negotiated acceptance of the System *as is* with known deficiencies at System delivery.
- The System Developer may be unable to perform to the Ts&Cs of the contract.

All Stakeholders need to emerge from the contract as winners! Therefore, *avoid* this problem and resolve it up front during the proposal phase or not later than the SRR and throughout the System Development Phase, as appropriate.

23.6.4 COI/CTI Post-SRR Requirements Issues

There are occasions when latent defects—flaws, errors, deficiencies, etc.—in specification requirements go *undiscovered* until after SRR, generally due to poor analysis. System Acquirers may reluctantly consider and may or may not accept ECPs for requirements changes. Depending on the situation, the only *workable solution* may be to submit a Request for Deviation to the specification requirements.

23.6.5 Tracking Requirements Issues and Clarifications

Requirements Issues Tracking Principle

Principle 23.7 Track specification requirements issues to closure.

When specification requirements issues and clarifications are identified, it is critical to bring all of them to closure quickly. The preferred approach for tracking closure status is to establish a metric that represents the number of actionable COIs, CTIs, TBDs, TBSs, and clarifications remain "open."

23.6.6 Closing Thoughts

When specification requirements issues are discovered, surfacing the issue to the surprise of the System Acquirer program manager in a major technical review typically has consequences. Stakeholders—System Acquirers, Users, End Users, Executives, and Project Managers —do not like surprises, especially in public forums such as project reviews and conferences. Although the handling of an issue depends on the personnel and Enterprises involved, the best advice may be for the System Developer's program or technical director to informally introduce the issue "off-line" in private conversation with the System Acquirer program manager prior to a review. Depending on the outcome and response, the System Developer may choose to address the issue formally through normal procurement channels.

Finally, the *reluctance* and *unwillingness* of the System Acquirer to entertain the idea of specification requirements changes may be driven by having to implement a contract modification that requires numerous approvals and justifications, a laborious and career risk process. It also challenges the initiator(s) to explain to your management *why you failed* to recognize this situation and rectify it during contract negotiations.

23.7 REQUIREMENTS COMPLIANCE

As a System Developer, one of the most common questions posed by management to SEs is: *Can we meet the specification's requirements*? The response typically involves words

such as *comply*, *conform*, and *meet*. Since SEs often lack training on the proper usage of the terms, you will find these terms are typically used interchangeably. So, *what do the terms mean*?

If you delve into the definitions of *comply*, *conform*, or *meet* in most dictionaries, you may emerge in a state of confusion. The reason is that most dictionary definitions of these terms employ either of the other two terms as part of the definition, resulting in circular references. So, to bring some clarity to this confusion, consider the following explanations.

23.7.1 Compliance

The term *compliance* is often used in reference to specification or design requirements compliance, process compliance, regulatory compliance, and so forth. In general, compliance infers mandatory *strict adherence* or obedience to the "letter" of a requirement with no exceptions. Despite the term's intent, you will often find *degrees of compliance*. Within the Contract System Development domain contracting protocol requires the performing Entity – System Developer – to: (1) formally notify the Acquirer's Contracting Officer (ACO) immediately anytime they will be unable to achieve full compliance with contract specification requirements and (2) propose risk mitigation corrective actions.

Example 23.2

Compliance Example

People are expected to fully *comply* with the letter of contract law established by international, federal, state, and local governmental authority.

23.7.2 Conformance

We often hear the phrase "We will *conform* to your requirements." *What does the seemingly evasive term "conform" really mean*? In general, it says the Enterprise has Organizational Standard practices (OSP) such as processes, methods, and behavioral patterns that we use on a regular basis. However, to promote harmony among team members and a positive relationship, we will *adapt* or *adopt*—that is, to *conform* to—a set of processes, methods, and behavioral patterns to be mutually acceptable to the other party. Here's an example.

Example 23.3

Conformance Example

You visit a country and discover their eating habits are different from yours. You have a choice: (1) *conform* to their culinary practices, (2) go without food, or (3) bring your own chef and food.

23.7.3 "Meet" Requirements

You often hear someone say they or their Enterprise can "meet" the requirements. *What does this mean*? In general, the term "meet" is a very weak, non-committal response that carries a *minimum* threshold connotation. In effect, they are stating, "We will meet your *minimum* requirements."

To summarize our Requirements Compliance section discussion, a subcontractor might tell a System Developer, "For this subcontract, we will *conform* to your documentation system's review and approval process. When we submit our documents for review and approval, we will *comply* with the subcontract's instructions for document format and submit via the Contracting Officer."

23.8 CHAPTER SUMMARY

In summary, our discussions in Chapter 23 addressed key topics you can employ to analyze specifications. This applies to specifications written both external and internal to your project. One of the greatest challenges in Specification Analysis is the failure to fully review a specification in terms of its requirements, references, and verification methods. Your analysis should reveal if the document was created by an Enterprise that employs the SDBTF-DPM Engineering Paradigm or one that employs MBSE methods.

We began our discussion of specification analysis by addressing System Acquirer and System Developer perspectives for a specification. We introduced various methods that System Developers employ to provide a high-level assessment of the "goodness" of the specification.

Next, we introduced common deficiencies such as missing, misplaced, overlapping, or duplicated requirements that plague many specifications. We described how specification issues and concerns should be resolved between the System Developer and Acquirer prior to and ACA. Last, we delineated three terms—*comply*, *conform*, and *meet*—that contractors express interchangeably but have different connotations.

23.9 CHAPTER EXERCISES

23.9.1 Level 1: Chapter Knowledge Exercises

1. How do you methodically analyze a specification?
2. What are some common types of specification requirements *defects*?
3. When requirements *defects* are identified, how should you resolve them internally and externally with the System Acquirer?
4. What does it mean to *comply* with a requirement?
5. What does it mean to *conform* to a requirement?
6. What does it mean to *meet* a requirement?

23.9.2 Level 2: Chapter Knowledge Application Exercises

Refer to www.wiley.com/go/systemengineeringanalysis2e.

23.10 REFERENCES

DAU (2011), *Glossary: Defense Acquisition Acronyms and Terms*, 14th ed. Ft. Belvoir, VA: Defense Acquisition University (DAU) Press. Retrieved on 3/27/13 from: http://www.dau.mil/pubscats/PubsCats/Glossary%2014th%20edition%20July%202011.pdf.

DAU (2012), *Glossary: Defense Acquisition Acronyms and Terms*, 15th ed. Ft. Belvoir, VA: Defense Acquisition University (DAU) Press. Retrieved on 6/1/15 from http://www.dau.mil/publications/publicationsDocs/Glossary_15th_ed.pdf

ISO/IEC/IEEE 24765:2010 (2010), *Systems and software engineering—Vocabulary*, Geneva: International Organization for Standardization (ISO).

NAVAIR TSD (2014), *Acquisition Guide: "Guide to Specification Writing," Naval Air Warfare (NAVAIR) Center Training Systems Division (TSD)*, Orlando, FL: NAVAIR. Retrieved on 3/6/15 from http://www.navair.navy.mil/nawctsd/Resources/Library/Acqguide/spec.htm.

SEVOCAB (2014), *Software and Systems Engineering Vocabulary*, New York, NY: IEEE Computer Society. Accessed on 5/19/14 from www.computer.org/sevocab

24

USER-CENTERED SYSTEM DESIGN (UCSD)

Enterprise and Engineered systems, products, or services require some form of human leadership and direction via interactions to initiate, operate (Monitor, Command, and Control (MC2), maintain, sustain, and terminate their operation. As technologies advance and deployment, Operations, Maintenance, and Sustainment (OM&S) costs increase, we continually strive to automate systems to *minimize* the number of human interactions and decisions to improve productivity, efficiency, and effectiveness and reduce costs and risk.

In general, most people tend to view systems, products, or services as stand-alone entities. The reality is that systems, products, or services are simply *tools* as *enablers* that provide humans the capability to accomplish missions and outcome-based performance objectives. For example, early man exploiting the capability of an integrated fulcrum and lever system to move heavy objects that exceeded their own physical capabilities and limitations.

User's Mental Model Principle

Principle 24.1 To reduce or eliminate human error, thoroughly understand the User's conceptual mental models for task performance before designing EQUIPMENT-HARDWARE and SOFTWARE.

One of the common aspects of all Engineering disciplines is designing and selecting components to ensure that progressive *stages* and *levels* of interfaces are (1) *compatible* physically and *interoperable* with each other, (2) durable and reliable, and (3) able to survive in their prescribed OPERATING ENVIRONMENTS for the duration of a mission. Engineering spends countless hours performing this task with an exclusionary focus on designing the EQUIPMENT Element; that's what we have been educated to perform. Then, after the EQUIPMENT is developed, proceed with the *undesirable* task of writing the operator, maintenance, and training manuals that are later deemed to be useless by the User. Hypothetically, the operator is told to "place their right foot against their left ear and take their left hand and reach under their right leg to activate a switch on a floor level panel on the right side of the operator." Enterprises award project teams and publicize accolades for "outstanding technical performance and innovation" in designing systems, products, or services that may have limited degrees of usability and User satisfaction.

Humans, as "elements" of a system, are not contortionists! Like physical components and materials, humans have bounded physical capabilities that limit our abilities. Our inherent performance is disrupted by distractions—*inefficient* and *ineffective* interfaces—that induce stress, which has *physiological* and *psychological* implications on human decision-making and performance. The irony here is that Engineers are educated and trained to research the physical characteristics and compatibilities of electrical, mechanical, and software components; materials; and chemicals in manufacturer's data and spec sheets. Yet, they are not educated or trained to view the human operators and maintainers as "components" *worthy* of similar *compatibility* and *interoperability* considerations. "Just tell the operator to reach down to the floor and flip the switch on. It should be *intuitively obvious*. They (Users) are smart enough to figure it out for themselves and don't need us telling them what to do!" is a common Engineering mind-set.

System Engineering Analysis, Design, and Development: Concepts, Principles, and Practices, Second Edition. Charles S. Wasson.

Companion Website: www.wiley.com/go/systemengineeringanalysis2e

Chapter 24 answers this challenge with a focus on User-Centric System Design (USCD). This topic is often referred to under a variety of terms such as Human Systems Integration (HSI), Human Factors (HF), Human Factors Engineering (HFE), Human Engineering (HE), Human-Centered Design (HCD), and User-Centered Design. As new Engineering graduates enter industry, government, and academia, they learn to *adapt* to the local Enterprise usage of these terms over time. Our discussions in this chapter are about the need to:

1. *Design* and *engineer* multi-level EQUIPMENT, MISSION RESOURCES, PROCEDURAL DATA, SYSTEM RESPONSES, and FACILITIES System Elements to be *compatible* and *interoperable* with human operator, maintainer, and sustainer characteristics.
2. *Educate* and *train* PERSONNEL System Element components in *how to* safely, effectively, and efficiently MC2 systems, products, or services to conduct Enterprise missions and perform tasks to achieve performance-based outcomes and objectives.

On completion of this chapter, one of the most important points is recognition and appreciation that humans, as PERSONNEL Element "components" of a system, have bounded capabilities and limitations. Those capabilities and limitations become constraint requirements levied on the design and implementation of the EQUIPMENT, PROCEDURAL DATA, MISSION RESOURCES, SYSTEM RESPONSES, and FACILITIES Elements, not vice versa! The bottom line is Enterprises need to recognize and shift their *ad hoc* Plug and Chug … Specify–Design–Build–Test–Fix (SDBTF)-Design Process Model (DPM) EQUIPMENT-based Engineering paradigm to a USCD paradigm.

24.1 DEFINITIONS OF KEY TERMS

- **Anthropometry**—"The scientific measurement and collection of data about human physical characteristics and the application (engineering anthropometry) of these data to the design and evaluation of systems, equipment, and facilities" (MIL-HDBK-1908B, 1999, p. 6).
- **Biomechanics**—"A subspecialty of human factors, primarily concerned with human movements, muscle strength, and muscle power" (Chapanis, 1996, p. 12).
  "The study of the structure and function of biological systems by means of the methods of mechanics" (Hatze, 1974, 189–190).
- **Concurrency**—The degree to which the PROCEDURAL DATA Element information—operator and maintenance manuals, training instruction, materials, devices, and aids—*accurately* and *consistently* match the MISSION SYSTEM and ENABLING SYSTEM EQUIPMENT Element design and its implementation.
- **Duty**—"A set of operationally-related tasks within a given job, e.g., driving, system or product servicing, communicating, target detection, self-protection, operator maintenance" (Adapted from MIL-HDBK-1908B, 1999, p. 32).
- **Egress**—"the action of going out of or leaving a place" (Oxford Online Dictionary, 2013a).
- **Ergonomics** (or HF)—"The scientific discipline concerned with the understanding of interactions between humans and other elements of a system, and the profession that applies theory, principles, and data and methods to design in order to optimize human well-being and overall system performance" (IEA, 2013).
- **Ergonomists**—"Those who perform Ergonomics … (and) … contribute to the design and evaluation of tasks, jobs, products, environments, and systems" (HFES, 2013).
- **Fail-Safe Design**—A design " … provided in those areas where failure can cause catastrophe through damage to equipment, injury to personnel, or inadvertent operation of critical equipment" (MIL-STD-1472G, 2012, p. 12).
- **Failure**—"The event, or inoperable state, in which any item or part of an item does not, or would not, perform as previously specified" (MIL-HDBK-470A, p. G-5).
- **Fault**—"Immediate cause of failure (e.g., maladjustment, misalignment, defect)" (MIL-HDBK-470B, G-5).
- **Haptic**—"Refers to all the physical sensors that provide a sense of touch at the skin level and force feedback information from muscles and joints" (DoD 5000.59-M, 1998, p. 117).
- **Haptics**—"The design of clothing or exoskeletons that not only sense motions of body parts (e.g., fingers) but also provide tactile and force feedback for haptic perception of a virtual world" (DoD 5000.59-M, 1998, p. 117).
- **Harm**—"physical injury or damage to the health of people, or damage to property or the environment" (ISO/IEC Guide 51:1999, 2009, 3.3).
- **Hazard**—"Any real or potential condition that can cause injury, illness, or death to people; damage to, or loss of, a system (hardware or software), equipment, or property; and/or damage to the environment" (FAA, 2006, p. B-6).
  Hazard—"A real or potential condition that could lead to an unplanned event or series of events (i.e. mishap) resulting in death, injury, occupational illness, damage

to or loss of equipment or property, or damage to the environment" (MIL-STD-882E, 2012, p. 5).

- **Human System Integration (HSI)** The interdisciplinary technical and management processes for integrating human considerations within and across all system elements; an essential enabler to SE practice. (INCOSE SEHv4, 2015, Appendix C).
- **Human-Centered (HCD)** (User-centered design)—A "design approach that is characterized by the active involvement of users, a clear understanding of user and task requirements, an appropriate allocation of function between users and technology, iterations of design solutions, and multi-disciplinary design." (Except from ISO 1503:2008, Paragraph 3.7 on page 3, with the permission of ANSI on behalf of ISO. (c) ISO 2014—All rights reserved.)
- **User-Centered System Design (UCSD)**—A multidisciplinary activity that places human capabilities and limitations at the forefront of System Design Solution development to achieve (1) *optimal* System performance across all System Elements (Personnel, Equipment, Procedural Data, Mission Resources, System Responses, and Facilities), (2) acceptable risk, and (3) acceptable System/Product Life Cycle OM&S Phase costs.
- **Human Engineering (HE)**—"*gen*. The application of knowledge about human capabilities and limitations to system or equipment design and development to achieve efficient, effective, and safe system performance at minimum cost and manpower, skill, and training demands. Human engineering assures that the system or equipment design, required human tasks, and work environment are compatible with the sensory, perceptual, mental, and physical attributes of the personnel who will operate, maintain, control and support it" (MIL-STD-1908B, 1999, p. 17).
- **HE Requirements**—"… requirements … established to develop effective human interfaces and preclude system characteristics that require extensive cognitive, physical, or sensory skills; complex manpower or training-intensive tasks; or result in frequent or critical errors" (MIL-STD-1472G, 2012, p. 11).
- **Human Factors (HF)**—"A body of scientific facts about human characteristics. The term covers all biomedical and psychosocial considerations; it includes, but is not limited to, principles and applications in the areas of human engineering, personnel selection, training, life support, job performance aids, and human performance evaluation" (MIL-HDBK-1908B, 1999, p. 17).
  "The discipline of Human Factors (HF) is devoted to the study, analysis, design, and evaluation of human-system interfaces and human organizations, with an emphasis on human capabilities and limitations as they impact system operation" (NASA SP 2007-6105, 2007, p. 246).
- **Human Factors Engineering (HFE)** —"A multi-disciplinary effort to generate and compile information about human capabilities and limitations, and apply that information to (the design and acquisition of complex systems) produce safe, comfortable, and effective human performance" (FAA SEM, Vol. 3, 2006, p. B-6).
  "Involves understanding and comprehensive integration of human capabilities (cognitive, physical, sensory, and team dynamic) into system design, beginning with conceptualization and continuing through system disposal" (USAF AFD-090121-055, 2009, p. 58).
  "… an interdisciplinary approach to evaluating and improving the safety, efficiency, and robustness of work systems, such as healthcare delivery" (National Center for Human Factors Engineering in Healthcare, 2013).
- **Human Factors Test and Evaluation (HFTE)**—"*acq*. Part of the system testing effort conducted in accordance with approved test plans. HFTE includes all testing directed toward validation and evaluation of HF analyses, studies, criteria, decisions, and operational and maintenance design characteristics, and features. These may include engineering design tests, model tests, mockup evaluations, demonstrations, and subsystem tests conducted to verify system level requirements. HF tests are a part of system developmental test and evaluation and operational test and evaluation" (MIL-HDBK-1908B, 1999, p. 18).
- **Human Performance**—"A measure of human functions and action in a specified environment, reflecting the ability of actual users and maintainers to meet the system's performance standards, including reliability and maintainability, under the conditions in which the system will be employed" (MIL-HDBK-1908B, 1999, p. 18).
  "The ability of actual users and maintainers to meet the system's performance standards, including Reliability and Maintainability (R&M), under the conditions in which the system will be employed" (DAU, 2012, p. B-98).
- **Human-System Integration (HSI)**—"The integrated and comprehensive analysis, design, assessment of requirements, concepts, and resources for system manpower, personnel, training, safety and occupational health, habitability, personnel survivability, and human factors engineering" (MIL-STD-882E, 2012, p. 6).
  "the systems engineering process and program management effort that provides integrated and comprehensive analysis, design, and assessment of requirements, concepts, and resources for human

engineering, manpower, personnel, training, system safety, health hazards, personnel survivability, and habitability" (MIL-STD-46855, 2011, p. ii).

- **Ingress**—The act of entering a physical system through an access portal designed to accommodate and facilitate human entry and needs.
- **Job**—"The combination of all human performance required for operation and maintenance of one personnel position in a system, e.g., driver" (MIL-HDBK-1908B, 1999, p. 32).
- **Job Hazard Analysis**—"identifies problem jobs and risk factors associated with them" (OSHA 3125, 2013, p. 7).
- **Man–Man Interface (MMI)**—"The actions, reactions, and interactions between humans and other system components. This also applies to a multi-station, multi-person configuration or system. Term also defines the properties of the hardware, software or equipment which constitute conditions for interactions" (MIL-HDBK-1908B, 1999, p. 21).
- **Person–Person Interface**—"*gen*. The actions, reactions, and interactions (i.e., transactions) among persons as they perform jobs, duties, and tasks to operate and maintain a manned system, including peer-peer and subordinate-supervisor interactions" (Adapted from MIL-HDBK-1908B, 1999, p. 21).
- **Safety Critical**—"A term applied to a condition, event, operation, process, or item whose mishap severity consequence is either Catastrophic or Critical (e.g., safety-critical function, safety-critical path, and safety-critical component)" (MIL-STD-882E, 2012, p. 7).
- **Safety Design**—The application of "… system and personnel safety factors, including minimizing potential human error in the operation and maintenance of the system, particularly under the conditions of alert, battle stress, or other emergency or non-routine conditions" (MIL-STD-1472G, 2012, p. 12).
- **Simplicity of Design**—"… the simplest design consistent with functional requirements and expected service conditions … and … capable of being operated, maintained, and repaired in its operational environment by personnel with a minimum of training" (MIL-STD-1472G, 2012, p. 12).
- **Subtask**—"An activity (perceptions, decisions and responses) which fulfills a portion of the immediate purpose within the task, e.g., remove lug nuts" (MIL-HDBK-1908B, 1999, p. 32).
- **Task**—"A composite of related activities (perceptions, decisions, and responses) performed for an immediate purpose, written in operator/maintainer language, e.g., change a tire" (MIL-HDBK-1908B, 1999, p. 32).
- **Task Analysis**—"A systematic method used to develop a time-oriented description of personnel-equipment/software interactions brought about by an operator, controller or maintainer in accomplishing a unit of work with a system or item of equipment. It shows the sequential and simultaneous manual and intellectual activities of personnel operating, maintaining or controlling equipment, in addition to sequential operation of the equipment. It is a part of system engineering analysis where system engineering is required" (MIL HDBK-1908B, 1999, pp. 31–32).
- **Task Element**—"The smallest logically and reasonably definable unit of behavior required in completing a task or subtask, e.g., apply counterclockwise torque to the lug nuts with a lug wrench" (MIL-HDBK-1908B, 1999, p. 32).
- **Use Error**—"a repetitive pattern of failure that indicates that a failure mode is likely to occur with use and thus has a reasonable possibility of predictability of occurrence" (NAP, 2007, p. 256).

24.2 APPROACH TO THIS CHAPTER

UCSD is a multi-disciplined SE activity requiring insightful collaboration with the User community and specialty engineering through a strategy of HSI that applies HFE, HE, Safety, and discipline design principles. Our objective is to:

1. Promote the importance of SEs integrating User HF into the design of systems, product, and service at the beginning of a project.
2. Delineate and clarify the context of various HF skills and disciplines.
3. Develop *insights* and *awareness* to enable SEs to plan for, recruit/employ, challenge, and review the decisions and work products of HSI, HF, HFE, HE, and Ergonomics professionals who design and engineer Human–System interactions.

A Word of Caution 24.1

Despite the fact that every Engineering discipline is a specialty discipline, HSI, HF, HFE, HE, and Ergonomics are commonly referred to as Specialty Engineering skills. Each requires specialized knowledge, experience, and *domain* expertise not only in Engineering but also behavioral science disciplines. ALWAYS employ the services of a competent, qualified professional of your Enterprise's own choosing. Remember you and your Enterprise are *solely* accountable for your own decisions and actions or the lack thereof concerning the performance of system, products, and services you deliver to your customers and their Users and End Users.

Most textbooks focus on HSI as an abstract, ambiguous term with discussions that encompass topics such as HF, HFE, HE, and Ergonomics. Unfortunately, the *state of the practice* is that Engineers and Enterprises amateurishly employ these terms *interchangeably*. Thrown into the quagmire are additional terms that include Anthropometrics and Biomechanics. Instead of helping SEs understand what is to be accomplished, detailed discussions focus on selecting display font sizes, colors, workstation chair heights and viewing angles while ignoring the higher-level view of *what* is required to achieve mission success. Chapter 24 is intended to bring clarity to these disciplines and their applications in the Systems Engineering and Development (SE&D) of systems, products, or services.

Given that most engineers are not educated or trained to competently perform HF, Chapter 24 will begin with a holistic view of how SYSTEMS or PRODUCTS can fail due to human interactions. The key construct for our discussion will be the System Element Architecture (SEA) introduced earlier in Figure 8.13. The construct depicts the integrated set of System Elements—PERSONNEL, EQUIPMENT, MISSION RESOURCES, PROCEDURAL DATA, and SYSTEM RESPONSES—comprising a MISSION SYSTEM and its ENABLING SYSTEMS plus an additional FACILITIES Element unique to ENABLING SYSTEMS.

Principle 24.2

Personnel Dual Accountability Principle

The Personnel Element in each MISSION SYSTEM or ENABLING SYSTEM has two levels of accountability: (1) mission accountability and (2) system accountability.

Since the PERSONNEL Element is an *operational* entity within the SEA, our discussions up to this juncture have treated it *architecturally* as a peer-level component to the other System Elements. *Observe the "architectural" context here*. The reality is that HIGHER-ORDER SYSTEMS— Enterprises—employ each SOI, MISSION SYSTEM, and ENABLING SYSTEMS to accomplish missions that have outcomes and performance-based objectives. Who does a Higher-Order System hold accountable for mission performance - inanimate objects such as the Equipment Element - Hardware and Software, Procedural Data, or Mission Resources? No, the Mission System or Enabling System operators, maintainer, and trainers that comprise the Personnel Element. As a result, the Personnel Element has two levels of accountability:

- System accountability as a performing System Element within the SEA (Figure 9.2) for safely and properly controlling Mission System performance.
- Mission accountability to Higher-Order Systems for accomplishment of mission task order outcomes and performance.

Given these accountabilities, Chapter 24 focuses on answering the following line of thought:

1. *Optimal* mission performance is determined by *optimal* PERSONNEL Element decision-making and actions coupled with *optimal* EQUIPMENT Element performance within a prescribed set of OPERATING ENVIRONMENT conditions.
2. *Optimal* PERSONNEL Element performance is a function of various HF that address human capabilities and limitations in terms of EQUIPMENT, PROCEDURAL DATA, MISSION RESOURCES, SYSTEM RESPONSES, and FACILITIES Element. Specifically Critical Operational and Technical Issues (COIs/CTIs) concerning ease of use—*compatibility* and *interoperability*—stress, and environmental safety.
3. This line of thought leads us to a key question: if User performance is a primary contributor to overall Mission and System performance and success, how do we engineer the EQUIPMENT, PROCEDURAL DATA, MISSION RESOURCES, SYSTEM RESPONSES, and FACILITIES to resolve these COIs/CTIs?

To address these points, we need to first understand the Specialty Engineering disciplines that enable us to "Engineer" systems, product, and services. Let's begin from a high-level (100,000 foot level) view and zoom in through a series of levels that ultimately take us down to the "switchology" of User–System interfaces.

24.3 INTRODUCTION TO UCSD

The success of Enterprise and Engineered Systems and their missions is ultimately determined by human performance—leadership, strategy, resources, education, training, experience, and luck. Humans, however, have finite mental and physical capabilities. As a result, we are dependent on the Engineered systems, product, or services as tools that enable us to leverage our own finite capabilities to achieve higher-level Mission and System outcomes with minimal Human error. Examples include space exploration, transportation, energy, medical, and technology advancements.

Chapter 2 highlighted the shortcomings of the traditional, *ad hoc* SDBTF Design Process Model (DPM) Engineering Paradigm:

- **SDBTF-DPM Shortcoming #1**—Quantum leaps (Figure 2.3) from specification requirements to a physical solution—hardware and software—without due diligence in understanding: (1) the operational need or Problem Space the User needs to solve, (2) how the User wants to use the system, and (3) behavioral responses of the system.

- **SDBTF-DPM Shortcoming #2**—Erroneous perceptions and ingrained beliefs *uncorrected* by Engineering education that application of Archer's Design Process (Chapters 2, 11, and 14) is Systems Engineering.

There is a third problem that is the result of Shortcoming #1:

- **SDBTF-DPM Shortcoming #3**—Limited Consideration of the User—operator, maintainer, and trainer—in the design of the EQUIPMENT Element.

Shortcoming #3, in general, is referred to as *HF*. History is filled with case studies in which the focus on EQUIPMENT Design-"Engineering the Box"-while ignoring human capabilities and limitations-"Engineering the System"-resulted in Mission and SYSTEM failure as well as injury, damage, and loss of life. Some of these results occurred due to poor EQUIPMENT Element design. Other failures occurred because the MISSION SYSTEM or ENABLING SYSTEM:

- Failed the Operational Utility, Suitability, Availability, Usability, Effectiveness, and Efficiency criteria (Principle 3.11).
- The PERSONNEL Element was not consulted or considered in the making of EQUIPMENT Element design decisions.
- The owner failed to train the PERSONNEL Element in the proper handling and safety procedures—use, misuse, abuse, and misapplication—of the EQUIPMENT Element.

You may ask: *if HF is obviously this important, why is it neglected*? In general, these include reasons such as:

- As Engineers, we naturally apply HF when we design systems.
- Automating a system is erroneously perceived to eliminate the need for HF.
- Humans will adapt to whatever system we give them.
- The design HF wants to review is already committed and fixed.
- This project is understaffed. We need Engineers, not HF looking over our shoulders.
- When you think about it, HF is just common sense.
- HF is not required by our contract.

To better understand and appreciate the impact of HF on system, product, or service performance, let's employ *Systems Thinking* (Chapter 1). This brings us to a key concept in SE and Risk Management: Reason's "Swiss Cheese" Accident Trajectory Model.

24.3.1 Reason's "Swiss Cheese" Accident Trajectory Model

In analyzing root causes of various types of industrial and medical *incidents* and *accidents*, Reason (1990b, p. 208) developed what has become known as the "Swiss Cheese" Accident Trajectory Model. The concept is based on a chain of "slices" that represent *defenses*, *barriers*, or *safeguards* intended to prevent potential *hazards* from slipping through "holes" in each slice. Figure 24.1 provides an illustration of Reason's concept. Think of this model as a symbolic representation of the SEA shown in Figure 8.13. As the dynamically grow/shrink and align themselves, a hazard penetrates the various layers via a trajectory resulting in an *incident* or *accident* event.

The underlying premise of the model is based on HIGHER-ORDER Enterprise Systems instituting various forms of *defenses*, *barriers*, or *safeguards* such as policies, processes, procedures, training, and EQUIPMENT design methods to prevent or minimize the occurrence of safety-related incidents or accidents. For example, system design *safeguards* might include:

- Training PERSONNEL to *properly* operate EQUIPMENT to help reduce the opportunity, frequency, and severity of incidents or accidents.
- Establishing and training Organizational Standard Processes (OSPs)—policies, processes, procedures, and methods—to restrict what the PERSONNEL Elements (operators, maintainers, trainers) are permitted to do in terms of authority, accountability, and safety.
- Designing EQUIPMENT with a safety-driven culture and focus to help reduce accidents.

The series of *defenses*, *barriers*, or *safeguards* are intended to *detect* and *prevent* a potential hazard trajectory becoming an *incident* or *accident*.

This discussion has two levels of application: (1) MISSION SYSTEM or ENABLING SYSTEM design and development and (2) the User's HIGHER-ORDER Level 0 System (Figure 8.4) that integrates the MISSION SYSTEM and ENABLING SYSTEMS as assets as a System of Interest (SOI) to perform Enterprise missions.

Conceptually, each System Element inherently has *weaknesses* or *latent defects* such as design flaws, errors, deficiencies, and omissions that represent potential hazards that may go *undetected* through Enterprise design review processes. From an Engineering perspective, Reason's *holes* are considered *faults*. Symbolically, the level of significance of the faults inherent in each of the System Elements—PERSONNEL, EQUIPMENT, and *fault*—is represented as time-dependent or time-independent holes of varying sizes.

When *faults* are: (1) discovered by subsequent slices in the chain, and (2) corrective actions—*defenses*, *barriers*,

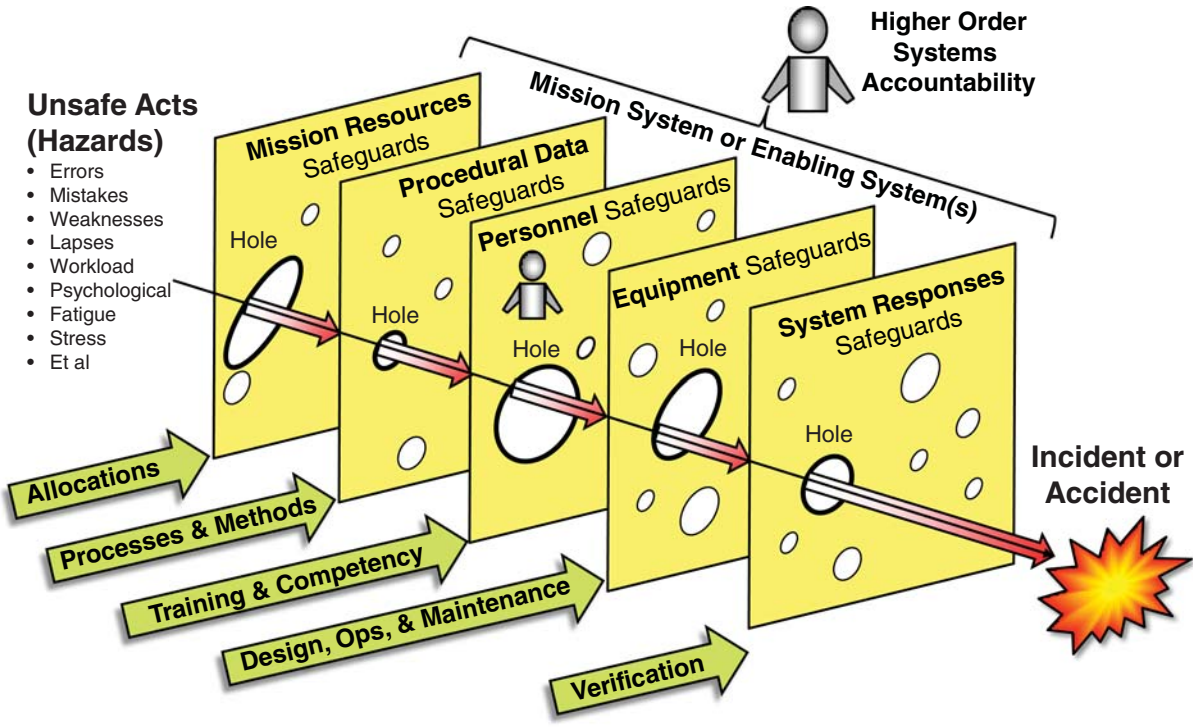


Figure 24.1 Application of Reason's (1990) Accident Trajectory Model to a MISSION SYSTEM or ENABLING SYSTEM's Elements. Derivative Work—Used with Permission.

or *safeguards*—*are* performed through proactive, continuous process improvement, the risk of system, product, or service failures should be reduced. However, the challenge occurs when faults in the chain of slices align in a way that allows the hazard to slip through and manifest itself as an incident or accident. The result may have *negative* consequences such as injury to the User(s) and/or public, damage to the SYSTEM or environment, or *catastrophic* consequences such as loss of life.

From an SE or System Analyst perspective, *what are some examples of the "faults" that should be considered in the design of each System Element*?

- **MISSION RESOURCES Element Faults**—Examples include inadequacies, contradictions, disruptions, ambiguities, tasking, resources, and quality that may be *static* or *dynamic* and close over time.
- **PROCEDURAL DATA Element Faults**—Examples include document errors, inaccuracies, inconsistencies, and omissions that remain *static* until corrected.
- **PERSONNEL Element Faults**—Examples include human *dynamics* such as memory lapses, sleep deprivation, fatigue, and stress; taking shortcuts; ignoring cautions and warnings; *unsafe acts*—abuse, misuse, or misapplication—of the EQUIPMENT; or mistakes such as issuing the wrong command and flipping the wrong switch.
- **EQUIPMENT Element Faults**—Examples include failures, intermittent cables or shorts, computational errors, corrupted data, mechanical degradation, cracked seals, poor lubrication, overheating, out-of-spec tolerances, and lack of maintenance.
- **SYSTEM RESPONSES Element Faults**—Examples include *inappropriate* or *unacceptable* outputs or by-products and latent defects—design errors, flaws, and deficiencies.
- **FACILITIES Element Faults**—Examples include lack of services, lack of the appropriate tools and parts, inadequate space, or environmental conditions.

Faults may be *static* such as design defects or *dynamic* due to wear and tear or subject to human decision-making. Reason (2000, p. 769) observes that the dynamic "holes"—*faults*—are in a continual state of change and symbolized graphically by the size of the hole as a function of time.

Engineering Accountability Principle

Principle 24.3 Engineering is a professional discipline that demands performance and accountability for all aspects of our work, including "Engineering the System", not just the "Engineering of Boxes"!

When each System Element's *defenses*, *barriers*, or *safeguards* are *weak* or *fail* across the series of slices allowing *holes* to form and align, a hazard follows an open trajectory through Reason's Accident Trajectory Model as represented by the arrow in Figure 24.1. As a result a potential hazard materializes into a physical incident or accident.

Reason (1995, p. 82) classifies two types of failures in the model: *active failures* and *latent conditions or failures*:

- **Active failures** represent "immediate adverse outcomes" that penetrate the layers of *safeguards* and *barriers* (Figure 24.1). For example, if you (PERSONNEL Element) are driving down the road at 60 MPH and shift the transmission (EQUIPMENT Element) from FORWARD to REVERSE, there will be an "immediate adverse outcome!."
- **Latent conditions or failures**, which Reason (2000, p. 769) refers to as "resident pathogens," represent outcomes resulting from a "delayed action." *Latent conditions* (Reason, 2000, p. 769) or *failures* (Reason, 1995, p. 82) originate from decisions made by others—system designers, reviewers, and management—that lie dormant until a specific set of factors triggers an undesirable outcome. For example, if you (PERSONNEL Element) fail to maintain the level and integrity of the coolant in an automobile to acceptable performance conditions, the engine (EQUIPMENT Element) will eventually fail mechanically that could have potential catastrophic consequences for the passengers.

These descriptions lead to an unanswered question: *what constitutes a failure*? MIL-HDBK-470A, p. G-5 defines a failure as follows:

- **Failure**—"The event, or inoperable state, in which any item or part of an item does not, or would not, perform as previously specified" (MIL-HDBK-470A, p. G-5).

Observe that *active* and *latent failures* are not mutually exclusive; *latent failures* could be triggered by external factors or conditions to become *active failures*. Consider the following example:

Example 24.1 **Latent Failure Conditions Leading to an Active Failure**

Suppose you have neglected to properly maintain the coolant level in your automobile (a potential *latent failure* condition) and then drive the automobile at a high rate of speed, a triggering factor that leads to an engine overheating and subsequent mechanical failure (an *active failure* with an immediate adverse outcome).

Given Reason's (1995, 2000) types of failures—*active* and *latent*—consider the following points:

- THE PERSONNEL and MISSION RESOURCES Elements being human dependent are *dynamic* and in a *continual state of change* as a function of the physiological, emotional, and psychological states of the operators and maintainers.
- Now, consider the symbolic nature of EQUIPMENT Element *holes*. These holes represent *latent defects*—design errors, flaws, deficiencies, and poor workmanship or materials—that lie dormant until they are exposed as a result of system usage, degradation of components over time, lack of maintenance, abuse, misuse, and so forth. In general, (1) a *design* latent defect either exists or it doesn't, and (2) a specific instance—Serial Number (S/N)—of a *product* may be free of latent defects at delivery but could acquire one such as a nicked wire during routine maintenance.
- PROCEDURAL DATA Element latent defects are analogous to the EQUIPMENT Element. A document or manual either (1) contains latent defects, or (2) it is not represented symbolically by a fixed size hole. As these latent defects are identified and corrected over time, you could say that the hole is *dynamic* and closes up.
- SYSTEM RESPONSES Element outcomes are a function of the PERSONNEL–EQUIPMENT Element interactions—"active failures"—due to the timing of Human decision-making and latencies represented by the HF Interactions Model discussed later in Figure 24.9.

Author's Note 24.1 **Defect Reporting and Corrective Action**

Engineers and Analysts view documenting and reporting the quantity of defects found during an In-Process Review (IPR) as a bureaucratic exercise in futility. If errors are found in a specification or Operator's Manual, it gets corrected after the meeting—no need for non-value-added statistics is the response. Unfortunately, it is bureaucratic paperwork, especially if managers (1) never analyze and isolate the *who*, *what*, *when*, *where* and *how* the *latent defects* entered in the work product development process stream or (2) take corrective actions.

Hopefully, you recognize and appreciate *why* early removal of *latent defects* is so critically important due to their cost-to-correct (Table 13.1) downstream and potential impact on mission performance. If you fail to remove latent defects, the consequences can lead to injury; damage to EQUIPMENT, property, or the environment; or catastrophic loss of life. Engineering is a professional discipline that demands *performance* and *accountability* for all aspects of our work, not just designing widgets!

So, *what is the purpose of highlighting these points*? Ultimately, mission success is determined by SOI—Mission System(s) and Enabling System(s)—success. You should recall from our Chapter 1 definition that every system has a probability of success. System probability of success translates into Mission probability of success. The question is: *how do SEs ensure that proper safeguards and barriers are designed into the System Elements to ensure that probability of success is achieved*? The answer depends on the competence of the System Developer, User resources for System Development, and the level of *acceptable risk*—cost versus performance trade-offs—the User can tolerate.

24.3.2 Contributory or Root Causes: Human Slips, Lapses, Errors, Mistakes, and Violations

Principle 24.4

Root Causes Principle

Accidents are typically the result of a chain of weaknesses, not just one, in a system's *barriers* or *safeguards* that are intended to prevent a *hazard* from escalating into an incident or event with *negative* consequences.

Reason (1990a, b, c,) has published extensive research concerning contributory causes related to his Accident Trajectory Model illustrated in Figure 24.1 as a derived work. Based on his research, he identified four types of human error classifications: (1) slips, (2) lapses, (3) mistakes, and (4) violations (Reason, 1990b, p 9). *Why are these seemingly esoteric terms of critical importance to SE*? The answer is mission success. *How do you lead the development of a multi-discipline System Design Solution that eliminates or minimizes the impact of slips, lapses, errors, mistakes, and violations?* Specifically, Personnel (operator/maintainer actions), Equipment (hardware and software), Procedural Data, Mission Resources, and System Responses Element requirements and designs. Let's define each of these types of Human errors:

- **Errors** are "occasions in which a planned sequence of mental or physical activities fails to achieve its intended outcome" (Reason, 1990, p. 9), for example, a miscalculation.
- **Slips** represent "A simple frequently performed physical action goes wrong" (HSE 2013a, p. 3).
- **Lapses** represent "A lapse of attention or memory" (HSE 2013a, p. 3), for example, sleeping on the job or performing an action simultaneous with a distraction.
- **Mistakes** are (1) *misinterpretations* of the facts or conditions or (2) deviations from established practices or procedures leading to actions that produce *unexpected* outcomes with potentially *negative consequences* such as failure or catastrophe, for example, "Not understanding properly how something works or an error of diagnosis or planning" (HSE 2013a, p. 3).
- **Violations** represent "A deliberate breach of rules and procedures … " (HSE 2013a, p. 3).

Reason (1990b, p. 1–18), HSE (2013a, p. 2–3), and Nelson, W. (2013a) provide descriptions of these topics in greater detail.

The NAP (2007) highlights an important distinction concerning *human error*, *user error*, and *use error*: " … in the area of medical products, regulator and standards bodies make a clear distinction between the common terms 'human error' and 'user error' in comparison to 'use error.' The term 'use error' attempts to remove the blame from the user and open up the analyst to consider other causes, including the following:

- Poor user interface design (e.g., poor usability).
- Enterprise elements (e.g., inadequate training or support structure).
- Use environment not properly anticipated in the design.
- Not understanding the user's tasks and task flow.
- Not understanding the user profile in terms of individual differences in training, experience, task performance, incentives, and motivation" (NAP, 2007, pp. 256–257).

Once a System or Product is fielded, Reason (2000, p. 768) stresses the importance of detailed analysis to understand how various *incidents* and *events* such as "near misses" may provide indications of discovering potential *active* or *latent failures*—sometimes referred to by SEs as *unknown-unknowns*—or understanding the boundary conditions that trigger them. A classic example is the Test Pilot Chuck Yeager Mini-Case Study 24.1 below.

Mini-Case Study 24.1

F-86 Sabre Jet Accidents

Test Pilot Chuck Yeager had been flying an F-86 Sabre jet chase plane in support of other aircraft testing. A number of other pilots had died while flying the F-86 due to mysterious mechanical problems. The only physical evidence available was the remains of the aircraft scattered across the landscape.

During a solo mission unrelated to the accidents, Yeager attempted a slow roll when suddenly an aileron became locked when the aircraft was 150 feet above the ground. In reaction to the plane's behavior, Yeager backed off on the g's causing the nose to tilt upward, which then caused it to fly toward the ground. To his amazement, the aileron unlocked; he recovered from the roll and took the plane to an altitude of 15,000 feet. Being a typical test pilot driven to

understand cause-and-effect relationships, he decided to try the maneuver again and again recovering each time.

On a return to his base, he informed his superiors that he believed the problem was due to the wings bending during stress, thereby causing the aileron to lock. An investigation was initiated to disassemble the wings part by part. Their investigation revealed that a bolt had been installed upside down. On further investigation, they learned that a worker had purposely ignored assembly instructions—a *violation*—on the premise that everyone knows that bolt heads are installed with the head at the upper end, not at the lower end.

Mini-Case Study 24.1 exemplifies several key points related to Reason's Model:

Manufacturing (Enabling System) Personnel Element

1. Violation—Worker ignored work instructions, Reason's safeguards and barriers
2. Error—Worker installed bolt incorrectly
3. Lapse—Lack of apparent verification of proper bolt installation

Aircraft (Mission System)

1. Equipment Element—Latent failure initiated by an active failure, triggering conditions
2. Personnel Element—Instinctive situational assessment as a test pilot to recognize catastrophic conditions and perform compensating actions (Chapter 34), Reason's safeguards and barriers, to recover.

Figure 24.2 illustrates how Reason's Error Classifications—human slips, lapses, mistakes, and violations—can enter a SYSTEM or PRODUCT and impact the overall mission performance. The challenge question is: *how do multi-disciplined SEs—hardware and software—incorporate HF to (1) design the* EQUIPMENT *and* PROCEDURAL DATA *Elements to reduce the likelihood and risk for User errors and (2) train the Users to properly and safely operate the Equipment*?

As a System Developer or User SE, you have three key objectives:

- **Objective #1**—Employ or acquire the services of a qualified, competent HF Engineer to perform HFE.
- **Objective #2**—Collaborate with the User, HFE, safety, hardware, and software, to mitigate both technical and Enterprise level risks. *Prevent* or *minimize* the potential for User slips, lapses, errors, mistakes, and violations from occurring subject to technology, cost, and schedule limitations.

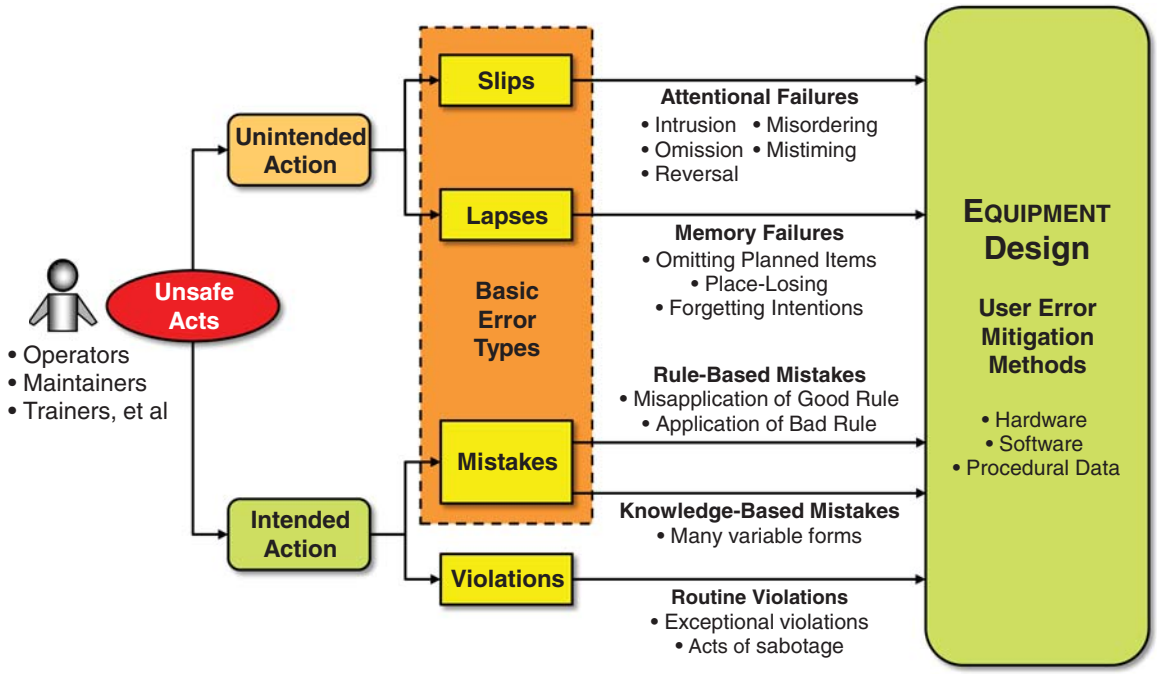


Figure 24.2 Reason's Error Classifications Applied to EQUIPMENT Element Design. Derivative Work—Adapted from Reason, J.T. (1990). Human error. Cambridge, England: Copyright © 1990 Cambridge University Press. Used with permission.

- **Objective #3**—Validate that optimal HSI has been achieved through User assessment of system, product, or service operation, maintenance, and sustainment.

During an incident or accident investigation, executives, project managers, or project engineers often publicize the need to identify the *single root cause*. The reality is most incidents or accidents are the result of a combination of contributory causes, not just one. The UK Health and Safety Executive (HSE, 2013a, p. 5) observes "People may debate whether particular factors should be classed as root causes, contributing causes, or neither. However, major accidents generally involve more than one root cause." HSE (2013a, p. 5) continues with a quote "Virtually none of the accidents investigated involved only a single cause. More commonly, half a dozen root and contributing causes were identified." Belke (1998, p. 1) reinforces this point and notes that "Virtually none of the accidents that EPA (US Environmental Protection Agency) and OSHA (US Occupational safety and Health Administration) investigated involved only a single cause. More commonly, half a dozen root and contributing causes were identified."

This discussion leads to another question: *what motivates humans to commit slips, lapses, errors, mistakes, or violations*? This brings us to the identifications of human Performance-Influencing Factors (PIFs).

How well suited is Reason's Accident Trajectory Model to systems, products, or services? The EEC (2006) provides a discussion of the model's suitability and limitations that includes participation by Dr. Reason.

So, *what are the contributory performance effectors that allow leakage of a potential hazard through the defenses, barriers, and safeguards that result in an incident or accident*? This brings us to our next topic, PIFs.

24.3.3 Human Performance-Influencing Factors (PIFs)

The UK HSE recommends several PIFs in three types of factors—job, person, and Enterprise—that influence human performance. The HSE suggests that optimization of these factors can "reduce the likelihood of all types of human failure" (HSE, 2013b) (Table 24.1).

24.3.4 Shifting to a User-Centric Design Paradigm

Principle 24.5

Do No Harm Principle

Design Enterprise and Engineered Systems, products, or services to DO NO HARM to their Users, End Users, the public, or the environment.

To further emphasize the criticality of HF in System Design, consider the implications of flight safety in today's world. Figure 24.3 provides a graphical illustration from FAA-H-8083-30 (2008) concerning root causes of aviation accidents.

In the early days of aviation beginning in 1903, the primary driver for Accidents in Aviation was *Technical Causes*. As a result of a robust Engineering focus on SE and Specialty Engineering integration—Reliability, Maintainability, and Availability (RMA); HF; and Flight Safety—over several decades, Technical Causes diminished significantly. Today, *Human Causes* are often the primary driver.

Although this is an FAA graphic, the concept applies similarly to other business domains such as military, medical, transportation, financial, and educational. Simply stated, if these systems fail, they may inflict fear, loss of confidence, mistrust, injury, damage, or loss of life to the public, EQUIPMENT, and/or the environment. The bottom line is System Design Solutions should incorporate HF through robust HSI activities to ensure that a system, a product, or service will DO NO HARM.

Based on the need to shift to a new User HF and Ergonomics-Based System Design Paradigm, *how do SEs accomplish this*? The answer resides in a concept referred to as HSI.

24.3.5 Human System Integration (HSI)

Principle 24.6

HSI Application Principle

HSI is a continuous, multi-disciplined activity that spans the entire System/Product Life Cycle, not just System Integration, Test, and Evaluation (SITE).

The concept of HSI has existed for several decades. However, one of the difficulties in understanding HSI begins with its name. The fallacy of the HSI title, which can be taken literally and erroneously, is its *ambiguity*—ambiguity that infers that a System Developer is going to develop a system, product, or service; obtain User acceptance; and deliver it to the User to integrate their operators, maintainers, and trainers for System Operations, Maintenance, and Sustainment. In this context, the application of the term appears to be an *event*.

HSI, however, is not an event. It represents the integration of HE, HFE, and Ergonomics knowledge with Engineering Design to produce systems, products, or services to that are simply compatible and interoperable with and usable by the User(s). To illustrate this point, consider a couple of perspectives from the USAF and INCOSE.

- The USAF HSI Handbook (2009, p. 8) states "Human Systems Integration (HSI) is a robust process by which to design and develop systems that effectively and affordably integrate human capabilities and limitations."
- From an SE perspective, the INCOSE SE Handbook (2012, p. 328) notes "Human Systems Integration brings human-centered disciplines and concerns into

TABLE 24.1 Performance-Influencing Factors (PIFs) That Influence Human Performance (HSE, 2013b)

Factor	PIFs
Job factors	1. Clarity of signs, signals, instructions, and other information 2. System/equipment interface (labeling, alarms, error avoidance/ tolerance) 3. Difficulty/complexity of task 4. Routine or unusual 5. Divided attention 6. Procedures inadequate or inappropriate 7. Preparation for task (e.g., permits, risk assessments, checking) 8. Time available/required 9. Tools appropriate for task ○ Communication, with colleagues, supervision, and contractor, and other working environment conditions (noise, heat, space, lighting, ventilation)
Person factors	Physical capability and condition 1. Fatigue (acute from temporary situation or chronic) 2. Stress/morale 3. Work overload/underload 4. Competence to deal with circumstances 5. Motivation versus other priorities
Organization factors	Work pressures—for example, production versus safety 1. Level and nature of supervision / leadership 2. Communication 3. Manning levels 4. Peer pressure 5. Clarity of roles and responsibilities 6. Consequences of failure to follow rules/procedures 7. Effectiveness of organizational learning (learning from experiences) 8. Organizational or safety culture, for example, everyone breaks the rules

the SE process to improve the overall system design and performance."

The scope of HSI begins with the initial contact with the System Acquirer, User, or End User and continues throughout the System/Product Life Cycle. Figure 24.4 provides a better perspective of the scope of HSI as an activity that simply text descriptions. The graphic, which uses a V-Model (Figure 15.2) example, illustrates how HSI as an SE activity is accomplished via HFE or how Ergonomics is performed throughout the System Acquisition, System Development, and System OM&S Phases of the System/Product Life Cycle. Although not illustrated in this graphic, HSI is equally important during the System Retirement/Disposal Phase.

As an SE activity, HSI is performed across numerous phases of the System/Product Life Cycle as shown in Figure 24.4:

- System Acquisition Phase HSI requirements considerations—who the Users and End Users are—and acquisition specifications.
- System Development Phase HSI design and development considerations.
- System Deployment and OM&S Phase HSI considerations - ease of use, maintenance, and sustainment labeling - identifiers, cautions and warnings; access ports; test points, and LRU removal.
- System Retirement/Disposal Phase HSI considerations - easy identification and removal of sensitive, toxic, and hazardous materials.

Chapter 8 stressed the importance of making a determination of whether the User is *internal* or *external* to the system, product, or service you are acquiring or required to deliver. Context diagrams such as Figure 8.1 provide valuable insights in bounding and specifying "the System." If you are

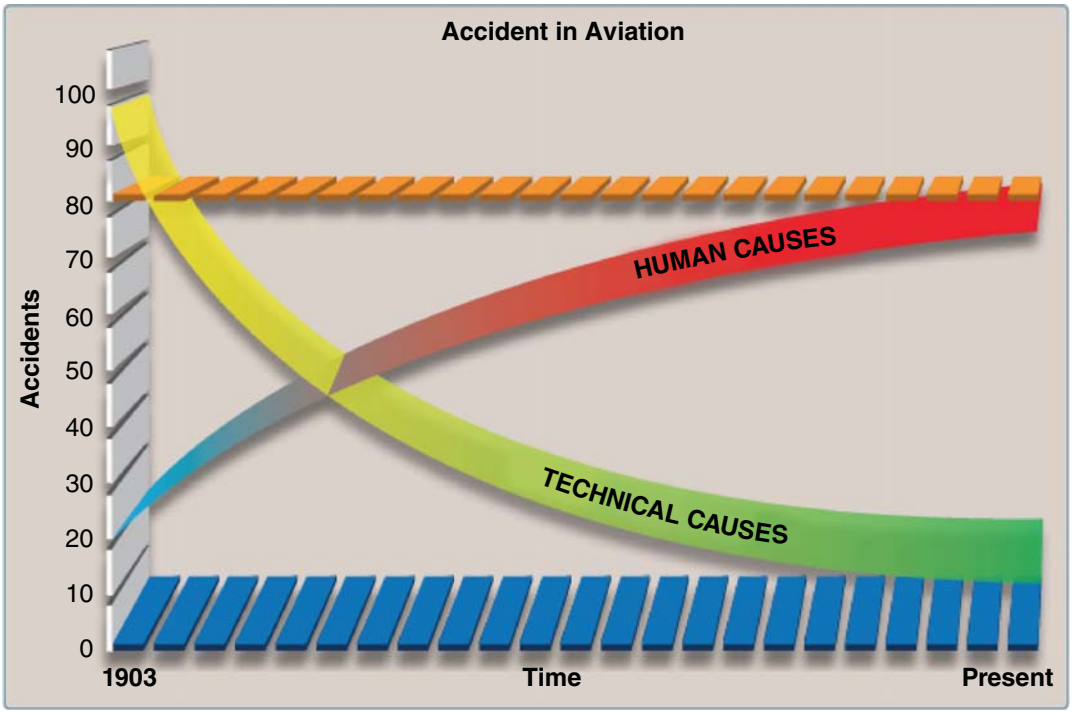


Figure 24.3 Accidents in Aviation: Statistical Graph Showing That 80 Percent of All Aviation Accidents Are Caused by HF (Source: FAA-H-8083-30, 2008, Figure 14–34, p. 14–28).

a System Developer that is only accountable for delivering the EQUIPMENT and the PROCEDURAL DATA Elements, you should create a Level 0 Customer System that architecturally integrates the User's operators, maintainers, and trainers with your EQUIPMENT and PROCEDURAL DATA. Although the *human condition* of the User is beyond the System Developer's control in terms of their decisions and actions, when missions are performed, a measure of success is for Users to emerge highly pleased in terms of customer satisfaction with mission performance.

A Word of Caution 24.2

System Architecting and Human Factor Considerations

Avoid the notion that System Architecting is simply developing an architecture based on the EQUIPMENT Element. Many times, System Architects focus on developing the SYSTEM or PRODUCT hardware and software and treat HF as an afterthought after the design is complete when User's manuals are being finalized. UCSD represents a major *paradigm shift* from the typical System-Level Architecting that focuses exclusively on hardware and software design.

To illustrate this point, NASA (SP 2007-6105, 2007, p. 247) notes "Most methods involve judgment and so are highly dependent on the skill and expertise of the analyst. In addition, both experienced and inexperienced operators provide valuable information about the strengths and weaknesses of old systems and how the new system might be used." Madni (2011, p. 5) refers to this condition as a "Human Role–Architecture Mismatch."

If you are a System Architect, UCSD should be an integral part of your education, training, and daily work. If you doubt this, purchase a product and evaluate *how well* the System Architect considered User Stories (Chapter 15), Use Cases (UCs) (Chapter 5), and scenarios.

24.3.6 HSI Domains

HSI is a very broad topic encompassing a lot of considerations that impact human and other performance. MIL-STD-46855A (2011) partitions HSI into nine domains that include[1]:

1. HE
2. Manpower
3. Personnel

[1]Refer to USAF *HSI Handbook* (2009, pp. 11 and 16), Clark and Goulder (2002, p. 90), and INCOSE (2011, pp. 332–336) for detailed discussions of these domains.

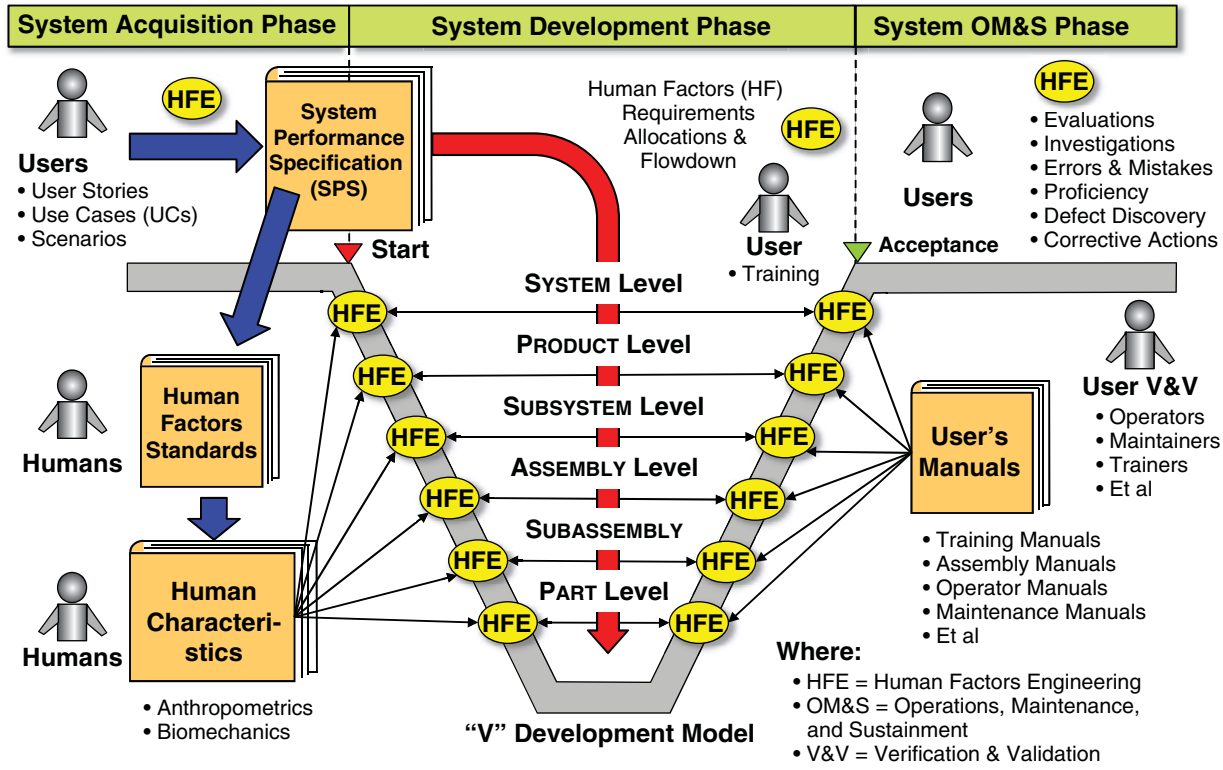


Figure 24.4 Human System Integration (HSI) Application to V-Model System Development.

4. Training
5. System safety
6. Health hazards
7. Personnel
8. Survivability
9. Habitability

Is HSI a consideration for SE&D? Hopefully, our discussion of Reason's Accident Trajectory Model is self-explanatory in answering this question. INCOSE (2011, p. 328) notes that HSI "… is an essential enabler to SE practice…." Domains such as aviation and ground transportation, marine, medical, and energy have statutory, regulatory, and other types of requirements for HF concerning aircraft, vehicles, trains, ships, and medical devices. As an illustrative example:

- DoDI 5000.02 (2008), for example, states, "The PM (Project Manager) shall take steps to ensure ergonomics, human factors engineering, and cognitive engineering is employed during systems engineering over the life of the program to provide for effective human-machine interfaces and to meet HSI requirements. Where practicable and cost effective, system designs shall minimize or eliminate system characteristics that require excessive cognitive, physical, or sensory skills; entail extensive training or workload-intensive tasks; result in mission-critical errors; or produce safety or health hazards" (DODI 5000.02 (2008, p. 60)).

In summary, given this Introduction to UCSD, let's shift our focus to understanding HF.

24.4 UNDERSTANDING HUMAN FACTORS (HF) AND ERGONOMICS

News coverage and other discussions often attribute accident causes to the "human element." Nelson (2013b, p. 2) observes that people "having little or no true knowledge of 'human factors' or 'human factors engineering' engage in conversations that use the terms "human element" versus "human factors." He cautions usage of the term "human element" as an abstract reference to individuals when things go wrong. The appropriate term is "human factors."

24.4.1 The Emergence of HF and Ergonomics

To address the needs and challenges addressing human considerations into System Design, two fields of study, HF and

Ergonomics, emerged in the same general timeframe. HF, for example, has origins in the United States. In contrast, Ergonomics has origins traceable to European roots. Chapanis (1996, p. 13) describes HF as common throughout North America and Ergonomics has a widespread use throughout Europe and Asia. He adds "Those working in both domains view HF and Ergonomics as equivalent" (Chapanis 1996, p. 13).

Both terms have traceability back to ancient Greek civilizations to the works of Hippocrates:

"Prior to World War I the focus of aviation psychology was on the aviator himself, but the war shifted the focus onto the aircraft, in particular, the design of controls and displays, the effects of altitude and environmental factors on the pilot" (Wikipedia, 2013a). Later, with automobiles gaining in popularity, studies expanded into areas such as driver behavior. Subsequently, World War II brought new emphasis as systems became more complex and expensive and mission success became more critical. As a result, several professional organizations emerged:

- In 1946, the Institute of Ergonomics and Human Factors, formerly The Ergonomics Society, was formed for HF specialists and ergonomists (Wikipedia, 2013a).
- In 1957, the Human Factors and Ergonomics Society (HFES) was founded (HFES, 2013b).
- Finally, the International Ergonomics Association (IEA) evolved representing a federation of ergonomics and HF societies around the world (IEA, 2010a).[2]

Based on a review of HF and Ergonomics definitions, one might ask: *why two different terms if they both accomplish the same objectives*? That question drives the need to understand the subtle differences.

Ergonomics is derived from two Greek words *ergon*, meaning work, and *nomos*, meaning laws—"to denote the science of work, ergonomics is a systems-oriented discipline, which now applies to all aspects of human activity" (IEA, 2010b). Observe the operative term *work* in this description inferring a focus on *what people do* and *how well they perform*. This is a critical point in understanding the difference between two perspectives as illustrated in Figure 24.5:

- **Ergonomics**—Understanding how stressors such as workload tasking, EQUIPMENT design, and OPERATING ENVIRONMENT impact human productivity and performance to "fit" the job to the worker
- **HF**—Applying human capabilities and limitations as constraints to EQUIPMENT design, tasking, and OPERATING ENVIRONMENT conditions.

[2]For additional information about the history of HF and Ergonomics, refer to FAA-H-8083-30 (2008 pp. 14-6 to 14-8) and Meister (1999, p. 146).

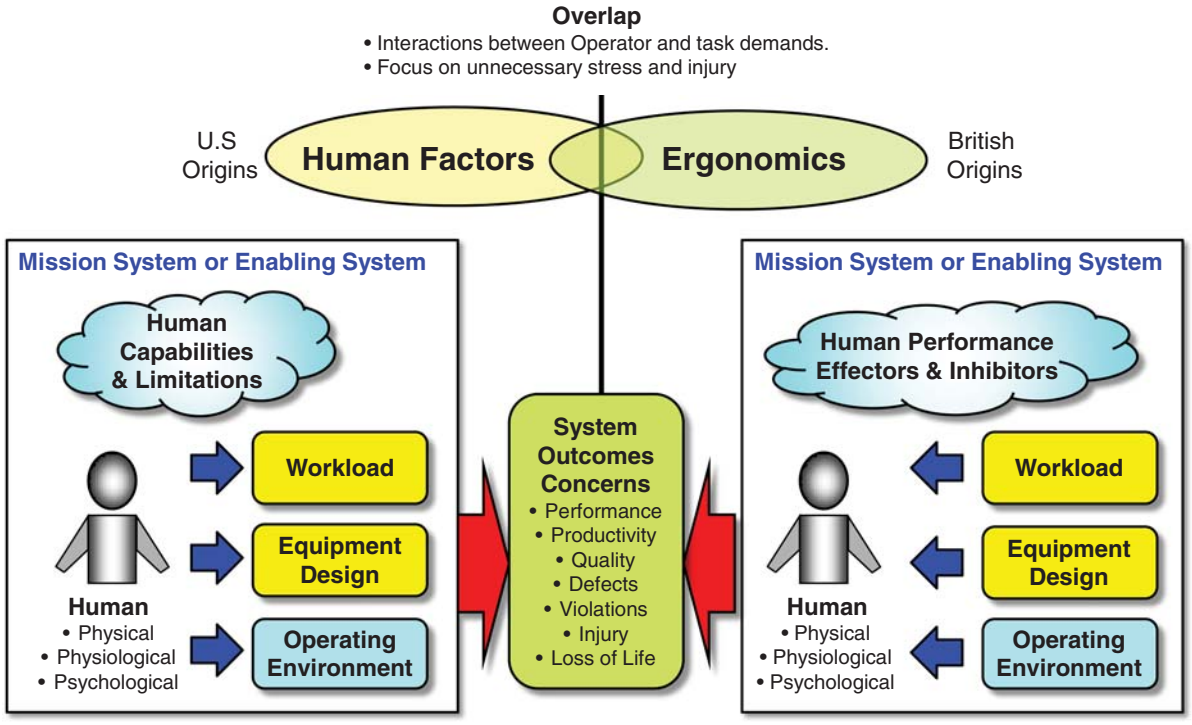


Figure 24.5 Subtle Perspective Differences between Human Factors (HF) and Ergonomics

Nelson (2010b, p. 2) observes "Ergonomics has traditionally focused on how work *affects* people, while the emphasis in human factors (engineering) is on the design of systems that reduce the potential for system errors and prevent injury." Nelson (2013b, p. 2) notes that:

- "Ergonomics may involve studies of physiological response to physically demanding work; environmental stressors such as heat, noise, and illumination; the performance of complex psychomotor tasks; and activity involving visual monitoring.
- The emphasis in ergonomics has been on ways to reduce fatigue by designing tasks within people work capabilities.
- The goal of an ergonomics work program is to achieve the optimal match between persons doing work and the overall work environment" (Nelson 2013b, p. 2).

In a discussion about Ergonomics, Wikipedia (2013) states, "The expression '*human factors*' is a North American term which has been adopted to emphasize the application of the same methods to non-work-related situations." Observe the phrase *non-work-related* situations. The inference here is on the development of systems such as aircraft and missiles. However, aircraft, for example, still has work-related environments via pilot operations in the cockpits, flight attendants in the cabin, and maintenance crews refueling the plane, during various phases of aircraft ground and flight operations.

As a result of a stigma created by commercial product advertising campaigns, people are often led to believe that Ergonomics focuses on the ergonomic design of office chairs and cubicles to improve worker comfort and productivity. Chapanis (1996, pp. 12–13) notes that the popular press refers to "the more physiological, anthropometric, and biomechanical aspects" of human factors but professionals regard ergonomics as *synonymous* with human factors."

The Psychology Wiki (2013) identifies, for example, Five Aspects of Ergonomics:

1. Safety
2. Comfort
3. Ease of Use
4. Productivity/Performance
5. Aesthetics such as signage

Whether you work in HF or Ergonomics, both apply to SE&D. In the remaining discussions, we will use HF; either term is equally applicable.

A key question is: from a system, product, or service perspective, what are the objectives *HF considerations intended to accomplish*?

TABLE 24.2 Chapanis' (1996) Objectives of HF

Primary Objectives	Supporting Objectives
Basic operational objectives	• Reduce errors • Increase safety • Improve system performance
Objective bearing on: • RMA • Integrated Logistics Support (ILS)	• Increase reliability • Improve maintainability • Reduce personnel requirements • Reduce Training requirements
Objectives affecting: • Users and operators	• Improve the working environment • Reduce fatigue and physical stress • Increase human comfort • Reduce boredom and monotony • Increase ease of use • Increase user acceptance • Increase aesthetic appearance
Other objectives	• Reduce losses of time and equipment • Increase economy of production

24.4.1.1 HF Objectives Chapanis (1996, p. 16) suggests the HF objectives provided in Table 24.2.

24.4.1.2 What Are Human Factors (HFs)? HF, as the name infers, are comprised of several elements that represent various categories of human performance. The DoD Critical Process Assessment Tool (CPAT) (1998, Table 1), for example, identifies five key factors or elements concerning human characteristics that have an impact on PERSONNEL–EQUIPMENT Element interactions that drive to system design considerations:

1. Anthropometric factors
2. Sensory factors
3. Cognitive factors
4. Psychological Factors
5. Physiological Factors

To better understand the scope each of these factors, Table 24.3 lists general human characteristics related to each of these factors.

Since the DoD is the source, does this list apply to any business domain? In general, *yes*; specifically, *no*. One of the things you discover about HF is that the subject is typically domain dependent. For example, Table 24.3 reflects a US military perspective at the time. In contrast, NASA (SP 2007-6105, 2007, p. 67), for example, identifies four similar but different categories of HF listed in Table 24.4 that represent key considerations in aerospace research and space exploration domains.

TABLE 24.3 Common Human Characteristics Associated with HF

HF	Human Characteristics
Anthropometric factors*	• Human physical dimensions • Body posture • Repetitive motion • Physical interface
Sensory factors	• Hearing • Vision • Touch • Balance
Cognitive factors	• Mental ability • Skills • Decision-making • Training requirements
Psychological factors	• Human needs • Attitudes • Expectations • Motivations
Physiological factors	• Human reactions to environments • Strength (lifts, grip, carrying) • Endurance

*Introduced here and discussed later in this chapter.
Source: DoD CPAT, 1998, Table 1, p. 7.

Other domains such as medical or transportation have their own sets of HF. Employ those factors appropriate for your business domain.

24.4.1.3 HF Disciplines A key question is: *what types of disciplines are required to implement HF as an HSI strategy*? The answer depends on the type, complexity, technology, and risk of the product. The FAA-H-8083-30 (2008, pp. 14.2–14.6), for example, identifies ten disciplines that comprise its HF considerations. These include[3]:

1. Clinical Psychology
2. Experimental Psychology
3. Organizational Psychology
4. Educational Psychology
5. Anthropometrics
6. Computer Science
7. Cognitive Science
8. Safety Engineering
9. Medical Science
10. Industrial Engineering

[3]Refer to the FAA-H-8083-30 (2008) for detailed descriptions of each of these disciplines and their relevance to HF.

TABLE 24.4 NASA HF (NASA SP 2007-6105, 2007, p. 67)

Human Factor Category	Attributes
Anthropometry and biomechanics	• Physical size, shape, and strength of the humans
Sensation and perception	• Primarily vision and hearing • Senses such as touch are also important
Environment	• Ambient noise and lighting • Vibration • Temperature and humidity • Atmospheric composition • Contaminants
Psychological factors	• Comprise memory • Information processing components such as pattern recognition, decision-making, and signal detection • Affective factors, for example, emotions, cultural patterns, and habits

In another example, the National Center for Human Factors Engineering in Healthcare (NCHFEH) (2013) states "Human Factors scientists and engineers study the intersection of people, technology, policy, and work across multiple domains, using an interdisciplinary approach that draws from:

1. Cognitive Psychology
2. Organizational Psychology
3. Human Performance
4. Industrial Engineering
5. Systems Engineering
6. Economic Theory"

For general HF applications that are not domain specific, Chapanis (1996, p. 14) identifies eight "technical disciplines contributing to human factors/ergonomics:

1. Anthropometry
2. Applied Physiology
3. Environmental Medicine
4. Engineering
5. Statistics
6. Operations Research
7. Industrial Design
8. Psychology"

Chapanis (1996, pp. 13–15) describes how these disciplines contribute to HF development of systems, products, and services.

Given HF composition of the specialty skills listed above, *what are they intended to accomplish?* This brings us to User–System Design Factors.

24.4.2 User–System Design Factors

MIL-STD-1472G (2012, p. 11) specifies that the following Design Factors "reflect human engineering, life support, and biomedical factors that affect human performance, including when applicable:

1. "Satisfactory atmospheric conditions including composition, pressure, temperature, and humidity.
2. Range of acoustic noise, vibration, acceleration, shock, blast, and impact forces and safeguards against uncontrolled variability beyond safe limits.
3. Protection from thermal, biological, toxicological/chemical, radiological/nuclear, mechanical, electrical, electromagnetic, pyrotechnic, and other hazards.
4. Adequate space for personnel, their equipment, and free volume for the movements and activities they are required to perform during operation and maintenance tasks under normal, adverse, and emergency conditions.
5. Adequate physical, visual, auditory, and other communication links between personnel, and between personnel and their equipment, under normal, adverse, and emergency conditions.
6. Efficient arrangement of operation and maintenance workplaces, equipment, controls, and displays.
7. Provisions for ensuring safe, efficient task performance under reduced and elevated gravitational forces with safeguards against injury, equipment damage, and disorientation.
8. Adequate natural or artificial illumination for the performance of operation, control, training, and maintenance.
9. Safe and adequate passageways, hatches, ladders, stairways, platforms, inclines, and other provisions for ingress, egress, and passage under normal, adverse, and emergency conditions.
10. Provision for acceptable personnel accommodations including body support and restraint, seating, rest, and sustenance, that is, oxygen, food, water, and waste management.
11. Provision for non-restrictive personal life support and protective equipment.
12. Provisions for minimizing psycho-physiological stress effects of mission duration and fatigue under normal, adverse, and emergency conditions.
13. Design features to assure rapidity, safety, ease and economy of operation, and maintenance in normal, adverse, and emergency maintenance environments.
14. Satisfactory remote handling provisions and tools.
15. Adequate emergency systems for contingency management, escape, survival, and rescue.
16. Compatibility of the design, location, and layout of controls, displays, workspaces, maintenance accesses, stowage provisions, passenger compartments, allocated tasks, and control movements with the clothing and personal equipment to be worn by personnel operating, riding in, or maintaining military systems or equipment.
17. Design of work stations shall be considered in all human-system interactions for mobile operations."

Now, consider how this list has commonalities and differences for various types of systems such as:

- Medical devices such as infusion pumps, X-ray devices, magnetic resonance imaging devices, and da Vinci surgical device (Figure 25.9).
- Portable consumer products such as smartphones and tablet computers.
- Office buildings.

To illustrate how User–System Design Factors might be applied, consider the two contextual examples below (Examples 24.2 and 24.3) in which humans are integrated into a MISSION SYSTEM (contextual role) and serve as an ENABLING SYSTEM to other MISSION SYSTEMS.

Example 24.2 **Fully Integrated, Self-Contained Aircraft System UCSD**

As an integrated, self-contained system, a MISSION SYSTEM such as a commercial aircraft system consisting of PERSONNEL (pilots, stewards/stewardesses, passengers), EQUIPMENT, PROCEDURAL DATA, MISSION RESOURCES, and SYSTEM RESPONSES is designed to incorporate HF into the development of each of these entities.

Example 24.3 **Architecturally Distributed Unmanned Aerial System (UAS)**

A UAS consists of a pilotless aircraft controlled from the ground perhaps thousands of miles away by an operator—surrogate pilot—in a control center. Since the pilot is not onboard the aircraft, there is no need to provide an environment with physical cockpit space,

flight controls, and max gravity ("g") limitations other than visual onboard cameras and sensors and instrumentation data to downlink to the ground control workstation. There is, however, a need to create some form of cockpit that is realistically similar to being onboard piloting the aircraft. In that case, the ground station EQUIPMENT Element must incorporate human capabilities and limitations.

In these two examples, observe how HF is an integral to the EQUIPMENT Design but in different ways. In the case of the UAS, Flight Control System (FCS) downlink data and uplink C2 transmission lag time latencies present new challenges to the pilot operator that are not present in the commercial airline example.

As another example, pilots are used to feeling the bumps in flight due to turbulence; hearing engine noise, deployment, and retraction of landing gear; and feeling the effects of the aircraft touch down on the runway. A UAS pilot, which is an external, remote User of the SYSTEM, lacks these sensations. This is particularly problematic in landing in which the pilot may inadvertently drive the aircraft onto the runway with a hard landing expecting to feel the "bump" only to have the aircraft bounce back into the air.

This last point emphasizes the need to determine whether the User is or is not part of the System being developed as discussed in Chapter 8 (Figure 8.1). Examples 24.2 and 24.3 illustrate how HFs change for the User based on whether they are onboard an aircraft or remotely piloting it.

Engineered Systems are not *sold* with a User as part of the SYSTEM/PRODUCT. The Users are considered "external systems" that interface to the system or product. Exceptions include Enterprises that:

- Develop Systems internally for use in performing Enterprise missions.
- Train their personnel to operate and maintain those systems.
- Deploy them to perform analytical services for clients such as energy oil and gas well exploration.

In general, most systems, products, and services are developed under contract or for the marketplace for use by Enterprises and consumers (Figure 5.1).

Since Chapter 24 focuses on the User, you need to (1) understand *who* the System Users are; (2) determine if they are Internal Users or External Users; (3) determine their HF capabilities, limitations, and skillsets; and (4) develop the System Elements with a User-Centric mind-set to meet their needs. From an HF perspective, developing commercial products for a global, diverse marketplace of External User skill sets may be *very different* from developing a system, product, or service for a known set of Internal Users.

In adapting to and targeting specific market segments with different Users and skillsets, technical decisions and assumptions must be made to determine what level of capability and automation we incorporate into system, product, or service designs. In this context, we need to understand (1) what the Users want to do in terms of User Stories and UCs, (2) what their capabilities, skills, and limitations are, and (3) what they are willing to pay (Figure 21.4) for a specific level of performance. This brings us to a key question: *how do we determine the proper mix of User–System tasks to be performed by each*? Let's begin with a discussion of User–System Tasking.

24.4.2.1 Defining Task Attributes If you analyze most Enterprise System operator, maintainer, or trainer tasks, you will discover that they share a common set of attributes. These attributes enable HF Engineers to understand the mission and scenario conditions that bound the User–System Interactions. For example, MIL-HDBK-1908B (1999, p. 32) identifies the attributes of a task shown in Table 24.5.

24.4.2.2 PERSONNEL Element Tasking and Accountability As stated in earlier Principle 24.2, the Personnel Element has two levels of accountability: (1) system accountability and (2) mission accountability. Analytically, the Personnel Element is accountable as a peer-level architectural System Element contributor to ensure the safe and proper operation of the Mission System - system accountability. However, unlike its System Element peers - Equipment- Hardware and Software, Procedural Data, Mission Resources, or Facilities Elements, which are inanimate objects - the Personnel Element is accountable to Higher Order Systems for exercising decision-making C2 authority over its System Element peers to successfully complete each mission - mission accountability.

Our discussion of the SEA shown in Figure 9.2 the PERSONNEL Element as "peer level" to the EQUIPMENT, PROCEDURAL DATA, MISSION RESOURCES, SYSTEM RESPONSES, and FACILITIES Elements. From a Modeling and Simulation (M&S) and System Design perspective, this is true. However, the PERSONNEL Element has *dual roles* to not only perform mission tasks but also to exercise authoritative C2 over the respective MISSION SYSTEM or ENABLING SYSTEM and its System Elements. *Why so*?

Principle 24.7

Stress Points (SPs) Principle

Optimal SYSTEM performance is achieved when the User–System Stress Points (SPs) have been avoided, eliminated, or reduced.

MISSION SYSTEMS and ENABLING SYSTEMS have mission outcome and performance accountability (Principle 24.2).

TABLE 24.5 MIL-STD-1908B Definitions of Task Attributes

Item	Task Attribute	Definition
1.	Mission	What the system is supposed to accomplish, for example, mission.
2.	Scenario/conditions	Categories of factors or constraints under which the system will be expected to operate and be maintained, for example, day/night, all weather, all terrain operation.
3.	Function	A broad category of activity performed by a system, for example, transportation.
4.	Job	The combination of all human performance required for operation and maintenance of one personnel position in a system, for example, driver.
5.	Duty	A set of operationally related tasks within a given job, for example, driving, system servicing, communicating, target detection, self-protection, and operator maintenance.
6.	Task	A composite of related activities (perceptions, decisions, and responses) performed for an immediate purpose, written in operator/maintainer language, for example, change a tire.
7.	Subtask	An activity (perceptions, decisions and responses), which fulfills a portion of the immediate purpose within the task, for example, remove lug nuts.
8.	Task element	The smallest logically and reasonably definable unit of behavior required in completing a task or subtask, for example, apply counterclockwise torque to the lug nuts with a lug wrench.

Source: MIL-HDBK-1908B, Definitions, para. 3.0, p. 32.

Ultimate mission success accountability resides with the PERSONNEL Element. So, the physical, mental, and emotional states—human capabilities and limitations—of the PERSONNEL Element (operators and maintainers) throughout the mission ultimately determine mission success. These states are reflected in PERSONNEL Element performance via factors such as:

- Training and experience to *efficiently* and *effectively* operate and maintain the EQUIPMENT Element safely and properly.
- Workload and related stress.
- Human safety.
- MISSION RESOURCES Element—Clear communications and timely and accurate flow of mission information and data.
- EQUIPMENT Element—*Ease of use*, integrity and trust, cautions, and warnings.
- PROCEDURAL DATA Element—Training manuals and operator/maintenance manuals.
- Operating Environment—Noise levels, lighting, temperature, and humidity.

Observe how each of the interfaces (Figure 24.6) between the PERSONNEL Element and the EQUIPMENT, PROCEDURAL DATA, MISSION RESOURCES, SYSTEM RESPONSES, and FACILITIES (not shown) Elements represents potential SPs that can impede, distract, or impact PERSONNEL Element performance.

24.4.3 MISSION SYSTEM and ENABLING SYSTEM Task Environments

If you perform a domain analysis of various Human Task Environments, the three most common types are illustrated in Figure 24.7.

- **Desk-based Task Environments** such as offices, space stations, and manufacturing plants.
- **Standing Task Environments** such as machinery, copy machines and faxes, space stations, and manufacturing plants.
- **Vehicle-based Seated Task Environments** such as automobiles, aircraft, spacecraft, construction, and agricultural equipment.

The question is: *how do you develop Systems and Products for a diverse set of Users—operators and maintainers*? The answer resides in Anthropometrics and Biomechanics.

24.4.4 Anthropometry and Biomechanics Relationships to HF

Principle 24.8

User-Centric Design Principle

Design the EQUIPMENT Element and adjust the task workload to fit the User's HF capabilities and limitations, not vice versa.

Humans have diverse ranges of physical characteristics that impact how they work and perform. In the case of HF

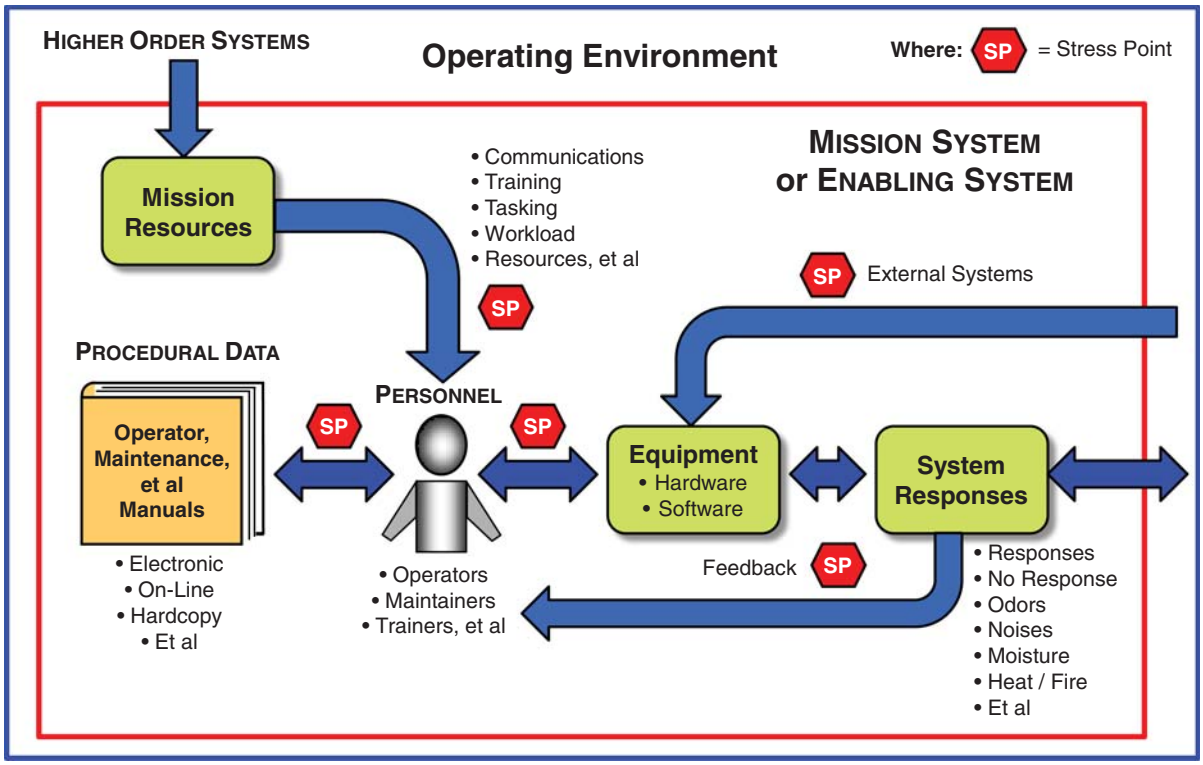


Figure 24.6 MISSION SYSTEM or ENABLING SYSTEM Tasking Model

and Ergonomics, it is important to *match* the person to the job and task. This requires insightful study into how humans are structured, their capabilities, sensitivities, and limitations. As a result, two fields of study, *Anthropometry* and *Biomechanics*, emerged to collect, analyze, and summarize published databases to support HF and Ergonomic decision-making. Figure 24.8 provides simple graphical examples to aid our discussion.

In general, Anthropometry focuses on scientific measurements of human physical characteristics—male and female, size, weight, and structure—in various positions such as standing, sitting, and lifting. In contrast, Biomechanics

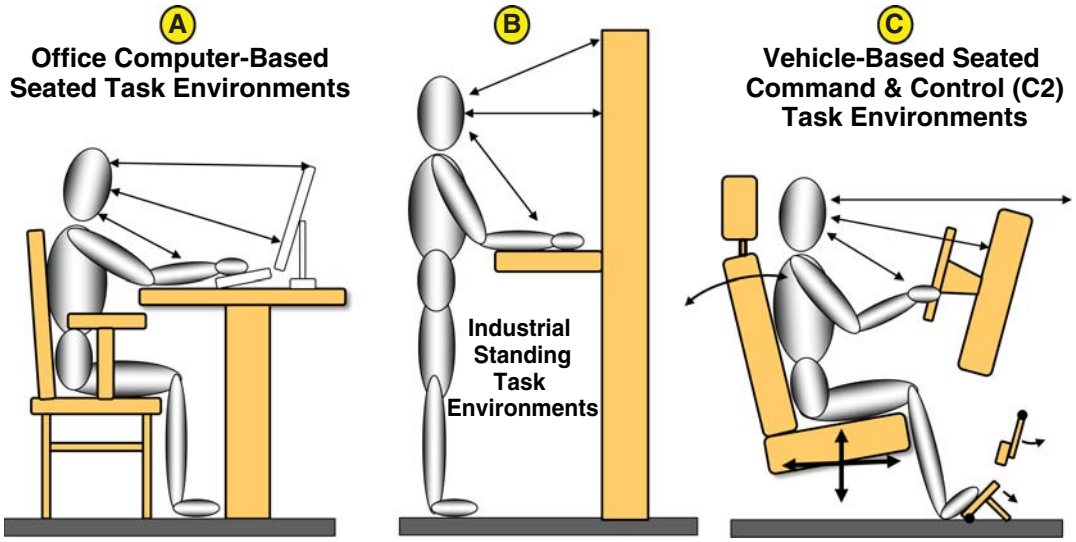


Figure 24.7 Common PERSONNEL–EQUIPMENT Task Environment Interactions

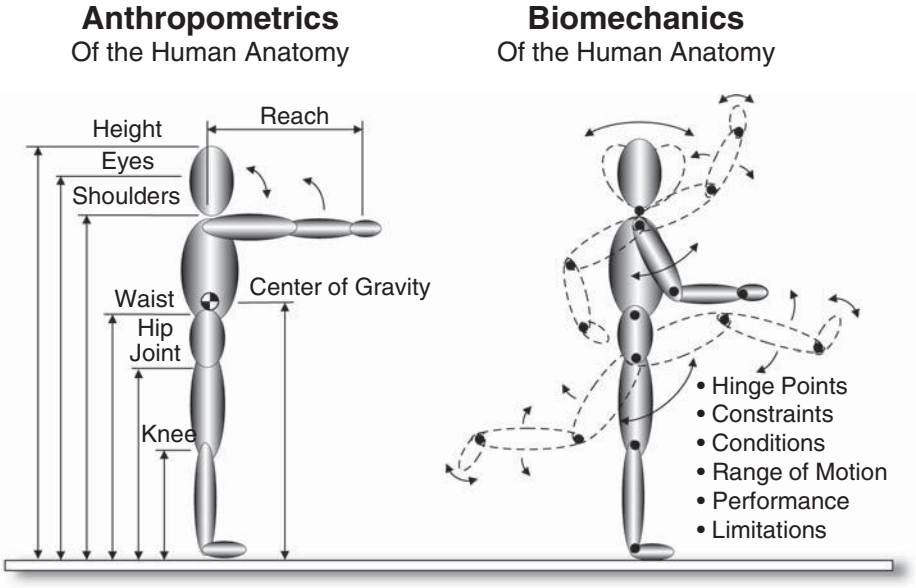


Figure 24.8 Example Graphics of Anthropometrics and Biomechanics

focuses on human capabilities and limitations related to body movements such as standing, sitting, and range of motion. Example human characteristics sources include:

- DoD-HDBK-743A (1991), Military Handbook: Anthropometry of U.S. Military Personnel.
- NASA SP-2010-3407 (2010), *Human Integration Design Handbook* (HIDH).
- NASA-STD-3000 (1995), Manned Systems Integration Standards.
- NASA RP 1024 (1978), Anthropometric Source Book, Vols. 1–3.

Roebuck et al. (1975, p. 7) note that Biomechanics "concerns primarily the dimensions, composition, and mass properties of body segments; the joints linking the body segments together; the mobility in the joints; the mechanical reactions of the body to force fields, vibrations, and impacts; the voluntary actions of the body in bringing about controlled movements, in applying forces, torques, energy, and power to external objects like controls, tools, and other equipment." They add that it is "very difficult (and useless) to draw demarcation lines between anthropometry, biomechanics, and engineering anthropometry."

Although Biomechanics may appear to be a relatively new field of study, its origins are traceable back to Aristotle in his book *De Motu Animalium*—translation *On the Movement of Animals*. Later, Leonardo da Vinci may be the first person to study the mechanics of anatomy (Wikipedia, 2013b). Leonardo da Vinci (1452–1519) through his studies of human anatomy and Giovanni Alfonso Borelli (1608–1679) through his studies of biomechanics collaborated to integrate existing knowledge in physics, anatomy, and physiology. Borelli published his texts *De Motu Animalium I and De Motu Animalium II* based on Aristotle's original title. Borelli also introduced the concept of a "stick person" as a human model with articulated linkages and interconnecting muscles (Kroemer, 2010, p. 97).

24.4.5 Understanding Personnel–Equipment Interactions

User–System interactions manifest themselves via physical Personnel–Equipment interfaces (Figures 10.15 and 10.16) and communications. The question is: *Who is the communicator and what is being communicated*?

Conceptually, we can say that the Equipment Element communicates only essential information to the User on a *need-to-know* basis (Principle 24.9). Although the Equipment Element serves as the *physical* communicator, the real communicators are the virtual hardware and software designers that collaborated with the User in the development of the Equipment. So, the real challenge becomes one of the virtual designers communicating with operators, maintainers, or trainers. Then, translating and transforming those communications - requirements - into an Equipment Element design that enables the User to conduct specific types of missions and be well-pleased with the interaction and results. The question is: *what needs to be communicated*?

In general, a User needs to

- View current status—Situational Assessment
- Query the System or Product for information
- Store mission information for future reference

- Command and Control (C2) C2 System outcomes and performance.
- Perform Guidance and Navigation (G&N).
- Communicate Information
- Report System Operational Health and Status (OH&S)

We can summarize this information into three key System capabilities: (1) obtain or have access to the Situational Assessment of the EQUIPMENT OH&S, (2) be able to C2 a mission, and (3) engage and interact with other external systems.

From an SE perspective, for clear, error-free communications to occur, two aspects of the PERSONNEL–EQUIPMENT interface must be considered:

1. The EQUIPMENT Element must be capable of sensing, detecting, and communicating information via a selection of dashboard display colors, lights, dials, or bar scales, based on an understanding of human physical, physiological, and psychological characteristics. The information must be valid, understandable, accurate, and precise for the task.
2. The PERSONNEL Element must be capable of reading display information, hearing audible alerts and alarms, or feeling vibrations, as applicable, being communicated. Operator perceptions, interpretations, and understanding of this information are critical and must match what the virtual designers intended to communicate. This is why Operator training, PROCEDURAL DATA, and experience are paramount to fill any gaps or correct any perceptions when performed under stress.

If we model these interactions, *what is actually occurring from a scientific perspective*? This brings us to Meister's HF Interactions Model.

24.4.6 Meister's HF Interactions Model

Most PERSONNEL–EQUIPMENT Element operational interactions involve three HFs Cognitive Factors, Physical Factors, and Physiological Factors. One of the best models for depicting these interactions is based on the work of Meister (1971) and illustrated as a derivative work in Figure 24.9. Both the FAA (2014) and NASA (SP 2007-6105, 2007, p. 247) employ this model.

Three key points about this figure are

- The PERSONNEL Element, as a performing Entity and the controlling operator accountable for System performance, is shown in reverse to the FAA and NASA renditions. The intent is to depict a general left-to-right PERSONNEL C2 stimulus flow to the EQUIPMENT with a return behavioral response back to the PERSONNEL Element.
- The graphic uses an older term "Machine" in lieu of EQUIPMENT, a more preferable term.

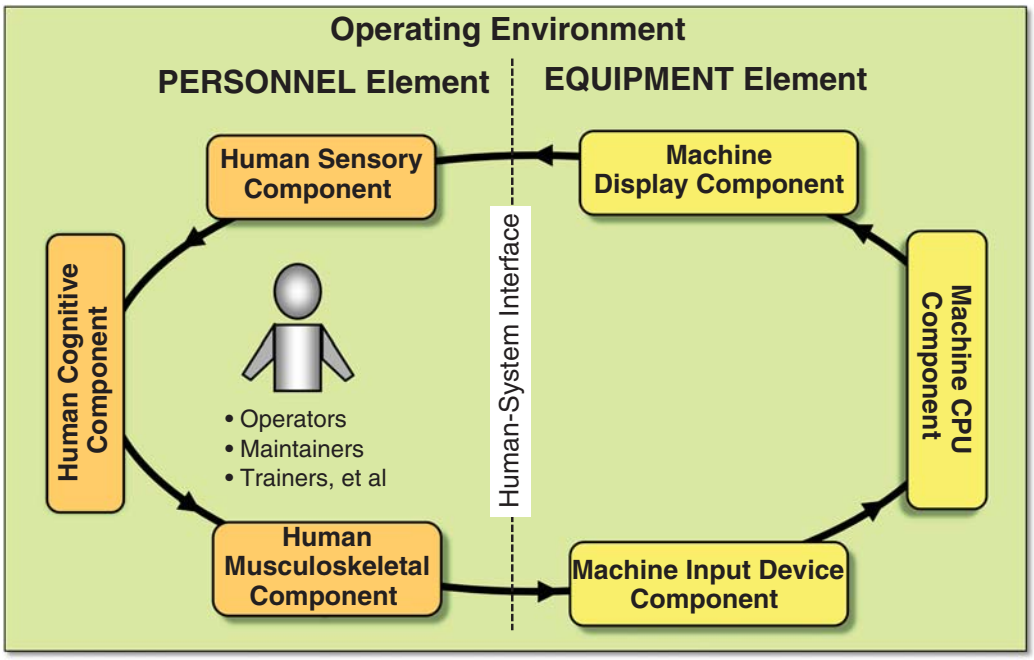


Figure 24.9 "Engineering the System" Considerations - Adaptation of the FAA's and NASA's Version of Meister's (1971) Human Factors (HF) Interactions Model. Derivative work — used with permission.

- The term *Machine CPU Component* is a physical implementation, not a behavioral capability. *Processing* would more appropriately represent key capabilities such as Sensing, Situational Assessment, C2, and storage.
- Some refer to the boundary between the PERSONNEL and the EQUIPMENT Elements as Man–Machine Interfaces (MMIs) or Human–Computer Interfaces (HCIs).

Principle 24.9

Display Information Principle

Display only *essential* information based on the User's need-to-know and priorities; avoid data overload.

Principle 24.10

Superfluous Information Principle

Eliminate superfluous, non-value-added information unless the User has consciously requested it and the information would be available.

The FAA (2013) describes each of the components in Figure 24.9 as follows:

- **Human Sensory Component**—"Vision, hearing, taste, smell, and touch."
- **Human Cognitive Component**—"Attention, memory (short and long term), information processing, decision-making, and action initiation."
- **Human Musculoskeletal Component**—"Motor coordination, action performance, and object manipulation."
- **Machine Input Device**—"Receives data via sensors; controls, switches, and levers; keyboard, mouse, and trackball; touchscreen, and voice."
- **Machine CPU Component**—"Performs programmed procedures, stores data, retrieves data, transmits response."
- **Machine Display Component**—"Displays response – visual, authority, and tactile – and initiates queries."
- **Environment**—"Illumination, noise level, air quality, vibration, and climate."

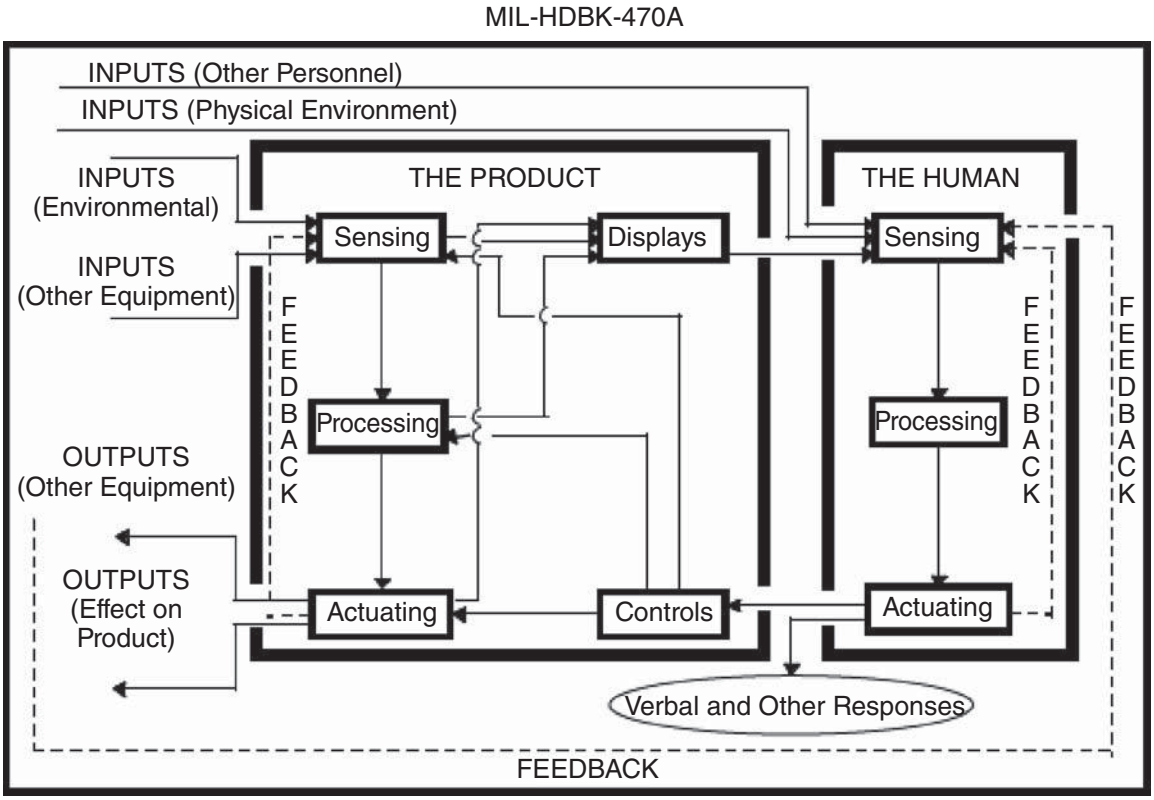


Figure 24.10 MIL-HDBK-470A Depiction of Meister's Human Interactions Model (Source: MIL-HDBK-470A, Figure 8, p. 4–11.)

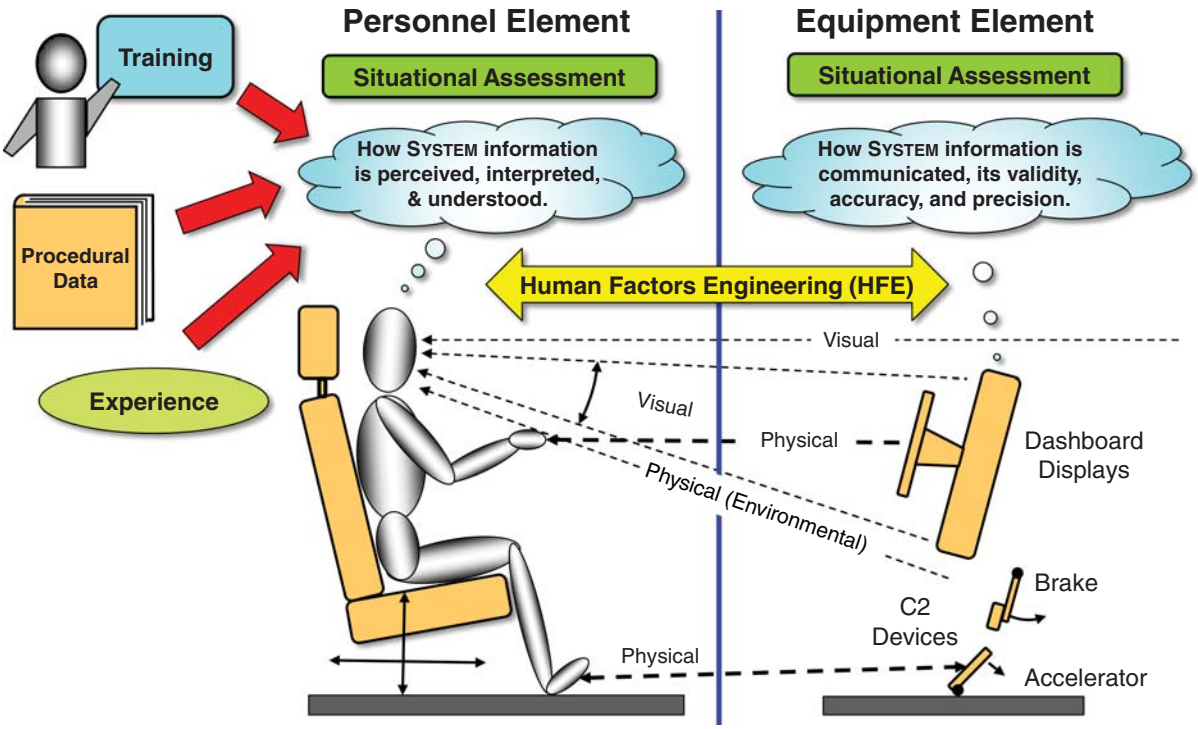


Figure 24.11 Generalized Personnel–Equipment Situational Assessment Interactions Model

MIL-HDBK-470A (Figure 8, p. 4-11) provides a more detailed version of the model in Figure 24.10.

Given this basic construct of Personnel–Equipment Interactions, let's explore how these interactions occur using the simple closed loop Car–Driver C2 system shown in Figure 24.11. Observe that this graphic isolates each side of a Vehicle-based Task Environment shown in Figure 24.7 Panel C.

Figure 24.11 illustrates that in order to drive the vehicle, the Driver employs their training, Procedural Data (vehicle's owner manual), and personal experience to interact with the car via its (1) dashboard displays for OH&S information and (2) control devices such as the steering wheel, accelerator, and brake. These interactions occur via Situational Assessments and decision-making by the driver based on the vehicle's Situational Assessment information and physical dynamics.

Although our discussion here focuses on Situational Assessment interactions, observe that all of the HF listed in Table 24.3—Anthropometrics and Biomechanics, Sensory, Cognitive, Physiological, and Psychological Factors—apply to the design of this interface.

We can model these interactions using a SysML™ Activity Diagram shown in Figure 24.12. Observe that both the Personnel and the Equipment Elements perform Situational Assessment and C2 tasks. Consider the following Car–Driver System example.

Example 24.4

Car–Driver System: Situational Assessment and C2

- The driver of a car performs a Situational Assessment of the vehicle's OH&S and then performs C2 to the vehicle to crank the engine.
- The car's C2 responds to the driver's command, cranks the engine, and provides Situational Assessment information to the driver via the dashboard displays.
- The driver continuously performs a Situational Assessment - scans vehicle and roadway status, interprets the results, and responds accordingly based on the vehicle, road, and driving conditions.

This discussion represents more than simply an introduction of Meister's HF Interactions Model (Figure 24.9). More importantly is the fact that both HF and Ergonomics recognize that Mission and System performance is ultimately dependent primarily on human performance. Human performance, in turn, is *bounded* by the capabilities—strengths and limitations—of the set of Users.

If the Equipment, Procedural Data, Mission Resources, System Responses, and Facilities interfaces, designs, and implementations are *mismatched* and *incompatible* with User capabilities and limitations, overall System performance will suffer. Such is the case with the User–System HF Interactions Model interfaces shown in

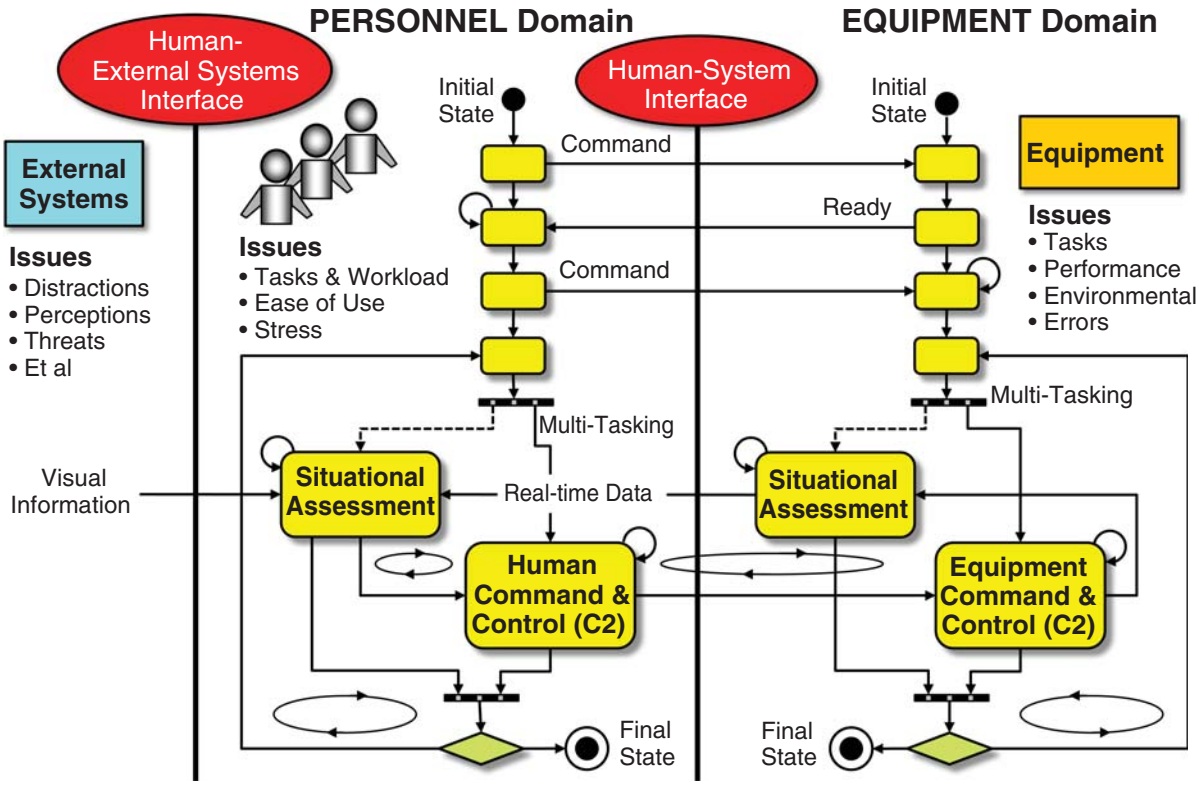


Figure 24.12 Generalized SysML™ Activity Diagram Model of Personnel–Equipment Task Interactions

Figures 24.9 and 24.10 and listed below. Failure to properly address these *incompatibility* issues up front could lead to injury or even worse catastrophic failure resulting in loss of life. Therefore, the following Human System Integration (HSI) between the following interfaces in Figure 24.9 become critical for reducing human errors and improving proficiency.

- Machine Display Component-to-Human Sensory Component Interface
- Human Musculoskeletal Component-to-Machine Input Device Component Interface

This illustration highlights the fallacies of the traditional System Design approach that focuses almost exclusively on the EQUIPMENT Element—HARDWARE and SOFTWARE. As a result, the PERSONNEL Element (Figure 24.9) is *ignored* and *assumptions* are made that the Users will *conform* and *make up* for shortcomings in the EQUIPMENT design. *Humans simply cannot perform beyond their own HF capabilities and limitations or do so for extended periods of time*. As a result, there is a need to shift the Enterprise paradigm from traditional EQUIPMENT Design to User-Centric Design. This requires recognition that the boundaries of human capabilities become performance requirement *constraints* levied on EQUIPMENT and PROCEDURAL DATA, *not vice versa*. Figure 24.13 illustrates how User-Centered Design levies and allocates those PERSONNEL HF constraints as requirements to the remaining System Elements.

Given these insights into HF, let's address how we integrate HF into HSI and SE. This brings us to our next topic, HFE.

24.4.7 Human Factors Engineering (HFE)

Principle 24.11

Human Factors (HF) Application Principle

"Anywhere you find technology and people interacting, there is a need for human factors engineering" (Nelson, G. (2013b, p. 1), paraphrasing Kantowitz (1983)).

Principle 24.12

HF Qualified Professional Principle

Employ the services of competent, qualified professionals to perform HFE.

When human interactions are a key element of "Engineering the System", challenging technical decisions have to be made regarding the PERSONNEL and EQUIPMENT Elements.

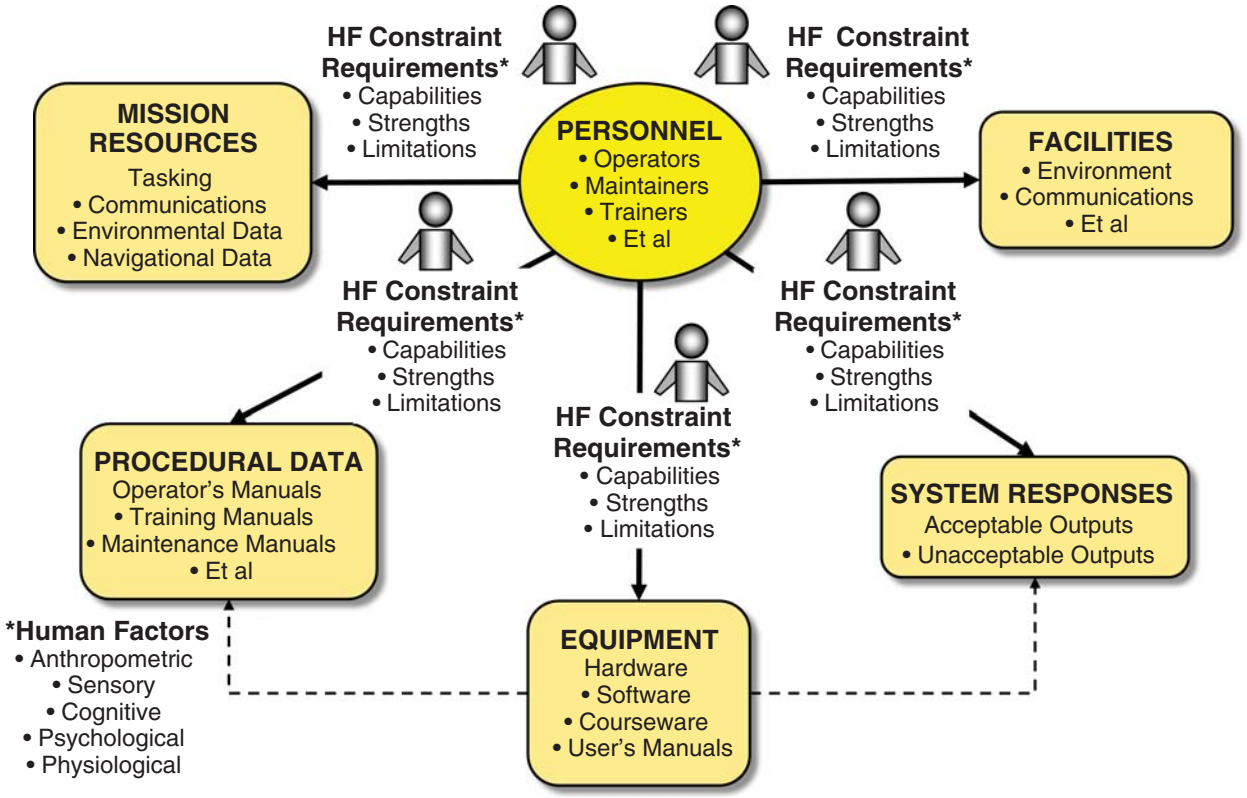


Figure 24.13 Paradigm Shift from Traditional EQUIPMENT Design to User-Centered Design

For a given set of operating conditions and constraints, key questions to be answered include:

1. Compared to the EQUIPMENT Element, what tasks can the PERSONNEL Element perform best?
2. Compared to the PERSONNEL Element, what tasks can the EQUIPMENT Element perform best?
3. What controls—capabilities and overrides—should be allocated for PERSONNEL to perform versus EQUIPMENT? For example, *automated* versus *manual* controls.
4. What work task ergonomic factors should be considered in HUMAN–SYSTEM interface designs?
5. What are the performance effects of HUMAN–SYSTEM interactions and outputs such as products, by-products, and services on the environment, safety, and health of Support Systems and the general public?

Answers to these questions require Specialty Engineering skills that include HFE and System Safety.

How does HFE enable us to develop a system, product, or service for compatibility and interoperability with its human operators, maintainers, and sustainers? Simply stated, HSE encompasses more than an Engineer "looking up" a few human physical characteristics and convincing the User during technical reviews that they actually spent time *considering* HF as part of the System Design Solution.

Observe the phrase "looking up a few human physical characteristics." Hopefully, you recognize and appreciate at this point in our discussion that "human physical characteristics" are one of five HFs listed earlier in Table 24.3 to be considered. This mind-set may impress the System Acquirer or User but fails in terms of satisfying the HF objectives stated earlier.

As a final point, people sometimes think that HF Engineers EQUIPMENT Element HARDWARE or SOFTWARE. In some Enterprises, if they are competent, they may perform both roles. However, what most people do not realize is that the role of most Specialty Engineering Disciplines such as HFE, Safety, Reliability, and Maintainability are to (1) support HF requirements development, (2) *assess compliance* to specification requirements via design reviews and prototype evaluations based on discipline-based design principles and best practices, and (3) report issues and recommendations for achieving compliance. Their discipline-based mission is not to design the EQUIPMENT - HARDWARE or SOFTWARE. If this is the case, they may have a potential conflict of interest.

Another term that is often used interchangeably with HFE is HE.

24.4.7.1 Human Engineering (HE) HE evolved as a discipline beginning with military organizations. MIL-STD-1472 Human Engineering, for example, has been a DoD Design Criteria Standard for many decades. For those who have worked with this standard, its context is well understood and has served the Aerospace and Defense SE&D community well. However, over time, advancements in knowledge and technology such as medicine bring new dimensions to the meaning of the term. Taken literally, the connotation is that we are going to "engineer humans", which gets into professional, ethical, and moral issues. Although that is not the context or intent of the term, HE is confusing for some people. Nelson (2013b, p. 2) cautions that "The words and order of words used in this term can imply to the uninformed that it is the human that is to be 'engineered,' or 'changed,' rather than the system."

On inspection, one might assume that HF and HE are the same. However, Licht (undated) references the following delineation provided in AFSC (1977, p. 2-1):

- **HF and HE**—"… human engineering is not synonymous with human factors. The term 'human factors' is more comprehensive, covering all biomedical and psychosocial considerations applying to man in the system. It includes not only human engineering, but also life support, personnel selection and training, training equipment, job performance aids, and performance measurement and evaluation."

24.4.7.2 Evaluations of Prototypes and Demonstrations One of the best approaches to support design decision-making is to prototype design areas that may have a moderate to high level of risk. Spiral Development discussed in Chapter 15 provides a good strategy for refining human–system interfaces with Users via rapid prototyping to drive out risk-based requirements. Rapid prototyping includes cardboard model mock-ups, sample displays, and so forth.

What do HFE prototype evaluations accomplish? The DoD *HFE CPAT* (1998, Section 1.1.3) suggests "Operator/maintainer interfaces should be prototyped to

1. Develop or improve display/software and hardware interfaces.
2. Achieve a design that results in the required effectiveness of human performance during system operation and maintenance.
3. Develop a design that makes economical demands upon personnel resources, skills, training and costs."

This leads to the question: *how do prototypes enable HFE to evaluate design solutions*? Prototypes are mechanisms that enable the HF Engineer to (1) evaluate basic concepts such as workflow, haptics of controls such as touch and feel, PERSONNEL–EQUIPMENT interaction sequences, and elimination of non-value-added steps and (2) assess performance based on instrumented test results.

As the HF Engineer performs his/her work, *what types of analyses do they perform that enable them to evaluate the design solutions*? The answer resides in understanding our next topic, HFE Analyses.

24.4.8 HFE Analysis

HFE performs various analyses such as manpower, personnel, training, and safety/health hazards to ensure that System Performance Specification (SPS) and lower-level Entity Development Specification (EDS) requirements are met. HF engineers employ various tools and methods to perform operational sequence evaluations, timeline and task analyses, and error analyses.

Since these decisions have an impact on the SPS and EDS requirements, HFE should be an integral part of SPS and EDS development activities beginning during the proposal phase. *Failure* to do so may have a *major* impact on contract technical, technology, cost, and schedule delivery performance as well as severe consequences if catastrophic failures that may have been avoidable occur after deployment of the system, product, or service.

The DoD *HFE CPAT* (1998, Section 1.1.3, p. 6) identifies four analytical HFE techniques for application to HSI design decision-making:

- **Operational Sequence Evaluations**—"… Describe the flow of information and processes from mission initiation through mission completion. The results of these evaluations are then used to determine how decision–action sequences should be supported by the human–system interfaces … "
- **Task Analysis**—"… The study of task and activity flows and human characteristics that may be anticipated in a particular task. Task analysis is used to detect design risks associated with human capabilities, such as skill levels and skill types. Task analysis also provides data for man–machine trade-off studies. The results of a task analysis allow the system designer to make informed decisions about the optimal mix of automation and manual tasking … "
- **Error Analysis**—"… is used to identify possible system failure modes. Error analysis is often conducted as part of human–machine trade-off studies to reveal and reduce (or eliminate) human error during operation and maintenance of the system. The error analysis results eventually are integrated into reliability failure

analyses to determine the system level effects of any failures … ”

- **Tests and Demonstrations**—“ … are often necessary to identify mission critical operations and maintenance tasks, validate the results of the HF related analyses, and verify that HF design requirements have been met. These tests and demonstrations are used to identify mission critical operations and maintenance tasks. Therefore, they should be completed at the earliest time possible in the design development process.”

24.4.9 PERSONNEL–Equipment Elements Trade-Offs

Optimal UCSD Principle

Principle 24.13 Optimal System performance is achieved when the PERSONNEL and EQUIPMENT Elements are *compatible*, *interoperable*, and *balanced* based on what each does best.

The SPS and each EDS should specify and bound SYSTEM capabilities, interfaces, design, and construction constraints related to PERSONNEL capabilities—skill levels and limitations. You should recall that:

- One of the fallacies of the *ad hoc* SDBTF-DPM Enterprise environments is making a *quantum leap* from SPS or ENTITY specification requirements to a Physical Domain Solution (Figure 2.3).
- When specifications are written, they should specify *what* has to be accomplished and *how well*, not dictate *how* to design the system, product, or service—Physical Domain Solution.

If a system, product, or service must perform missions that have specified results and performance-based outcomes, *how are the System's requirements allocated architecturally to the System Elements—Personnel and Equipment*? In some extreme environments characterized by conditions that are too harsh for humans such as space, the EQUIPMENT Element becomes the survival buffer between the OPERATING ENVIRONMENT and its PERSONNEL Element. Examples include space travel, aircraft in the atmosphere, nuclear plants, and undersea exploration.

In contrast, Earth-based environments have habitable conditions for humans such as driving an automobile or office environments. In either case, the PERSONNEL Element is *accountable* for the C2 of the MISSION SYSTEM or ENABLING SYSTEM. As a result, SEs must find an *optimal* balance across the System Elements based on an acceptable mix of technology, budget, schedule, and risk, more specifically SYSTEM requirements allocations to PERSONNEL–EQUIPMENT interactions. Specifically, making *informed* decisions concerning what the Human—PERSONNEL—does best *versus* what the EQUIPMENT does best.

We know that both the MISSION SYSTEM and ENABLING SYSTEM operationally perform tasks to accomplish an overall mission. Each SYSTEM-Level operational task is accomplished by a sequence of interactions between the PERSONNEL and the EQUIPMENT Elements (Figure 10.16). The capability to interact requires that the PERSONNEL and EQUIPMENT Elements perform their own sets and sequences of tasks. So, *how do we get from System-Level operational tasks to* PERSONNEL AND EQUIPMENT ELEMENT *tasks*? The answer resides in performing Task Analysis via an Analysis of Alternatives (AoA)—trade study— (Chapter 32) to determine the *optimal* balance between PERSONNEL capabilities and limitations versus EQUIPMENT capabilities and limitations or a mixture of the two. Figure 24.14 provides an illustration. Observe the top of the figure. Some types of tasks are specifically suited without debate to either the EQUIPMENT Element or the PERSONNEL Element as noted by the rectangular regions at either end. For example:

- Tasks requiring query search criteria or parameters for mission data retrieval should be allocated to the PERSONNEL Element.
- Tasks requiring execution of the high-speed data searches of vast amounts of mission data should be allocated to the EQUIPMENT Element.

Between these two regions lie areas that may require a mixture of tasks requiring both PERSONNEL and Equipment actions as noted by the complementary diagonal lines, for example, tasks requiring PERSONNEL and EQUIPMENT Element monitoring of normal operating conditions, cautions, and warnings, and making timely evasive or defensive decisions or actions.

In general, *avoid* specifying operational tasks to be performed explicitly by the PERSONNEL Element unless there are compelling reasons supported by an AoA. As the SE Process Model (Figure 14.1) is applied through multiple levels of SYSTEM design (Figure 21.3), trade-offs are supported by Chapters 30–34, “DECISION SUPPORT PRACTICES.” Organizationally, HFE should be tasked to determine the optimal mix of PERSONNEL versus EQUIPMENT tasks based on their respective strengths and limitations. This may require analysis supported by development of prototypes, models, and simulations to ensure that overall SYSTEM-Level performance is *optimal*.

So, *how do HFEs evaluate which tasks should be allocated to the* PERSONNEL *and* THE EQUIPMENT *Elements*? The answer resides in establishing criteria that characterize the capabilities–strengths and limitations—of PERSONNEL and EQUIPMENT Elements. Let's explore the key strengths and limitations of HUMAN and EQUIPMENT performance.

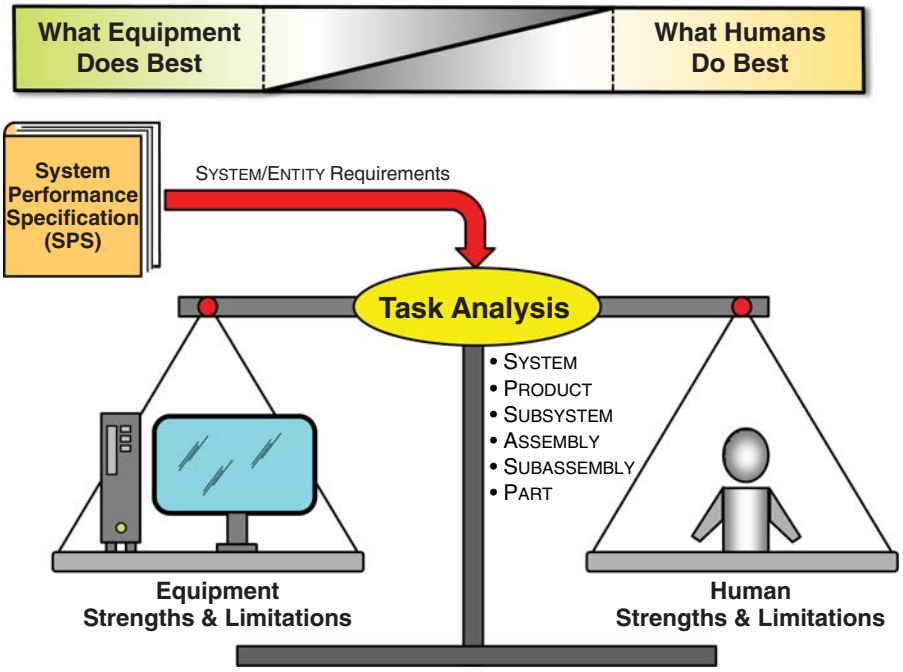


Figure 24.14 PERSONNEL–EQUIPMENT Task Analysis of Alternatives (AoA) Trade-Offs

Humans, in general, excel in mental strength and a number of skills when contrasted with the EQUIPMENT Element. In general, human performance *exceeds* EQUIPMENT Element performance in the following areas:

1. Value-based judgments and decisions
2. Priority selections
3. Resource allocations—over time
4. Impromptu tasks
5. Creative, non-repetitive tasks
6. Sensitivity to painful conditions
7. Human communications
8. Smell and touch
9. Adaptive behavior

Author's Note 24.2

Numbered items in the list above and below are for reference purposes only and *should not* be interpreted as a rank ordering of abilities.

For a detailed listing of PERSONNEL versus EQUIPMENT *capabilities* and *limitations*, refer to McCormick and Sanders (1964, 1982)

- Human strengths (McCormick and Sanders, 1982, pp. 489–490)
- Human limitations (McCormick and Sanders, 1964, pp. 573)
- Machine strengths McCormick and Sanders (1982, p. 490)
- Machine limitations (McCormick, 1964, p. 574)

Author's Note 24.3

Currency and Validity of McCormick's Criteria

Observe that McCormick's criteria were established in 1964. However, they are equally valid today with only minor exceptions due to new technologies and current terminology.

Now that we have an understanding of Human (Personnel)–Equipment strengths and limitations, let's address how they are applied to SE&D.

24.5 SITUATIONAL ASSESSMENT: AREAS OF CONCERN

Mission success of any system, product, or service requires knowledge and insights concerning Situational Assessment. Key Situational Assessment *areas of concern* include:

1. Mission Objectives Status
2. Mission Objectives Status
3. Task Status
4. Crew Status, as applicable
5. Vehicle Attitude

6. Resources Status
7. Phases, Modes, and States of Operation
8. Operational health & Status (OH&S)
9. Guidance & Navigation (G&N)
10. C2
11. Communications
12. Environmental Conditions

Each of these elements requires PERSONNEL–EQUIPMENT Interactions as illustrated by Figures 24.9–24.12. The question is: *how do HFEs communicate* PERSONNEL–EQUIPMENT *information in a clear and concise manner that is well understood by the operator, maintainer, and trainer, without the need for explanation other than basic training*? There are several ways.

24.5.1 Situational Assessment Approaches

Principle 24.14

Situational Assessment Principle

Provide real-time Situational Assessment information to the User to evaluate Mission performance and System Operational Health & Status (OH&S).

Once the initial PERSONNEL-EQUIPMENT Task allocations are made, the next question is: *how will the* PERSONNEL *and* EQUIPMENT *Elements interact with each other to accomplish interface objectives*. We refer to these PERSONNEL interactions as Input/Output (I/O) operations that include:

- Audio-visual stimuli and cues.
- Tactile feedback—such as touch and counter-reaction cues.
- Physical products and services—such as hard copies and data files.

As this information is collected and processed, the EQUIPMENT Element produces various preprogrammed cues as well as cautionary or warning alerts such as those that, at a *minimum*, include:

- Prompting the User to react to a query or make a decision.
- General health status.
- Problem reporting.
- Progress reporting in performing a task.

Since visual, auditory, and vibratory cues are an integral part of Human–System interfaces, let's scope the context of each term.

24.5.1.1 Visual Cues

Visual cues consist of optical warnings, cautions, or normal indications or messages to inform operators and maintainers of the current system conditions, status, or health. These include hand signals, console indicators, video display messages, and flashing lights.

24.5.1.1.1 Visual Display Messages

Principle 24.15

Visual Information Principle

Present visual information to the User in accordance with established Human Engineering (HE) and SYSTEM Safety design principles.

Visual display messages include various types of formatted dialogue boxes that request an operator response. Examples include color-coded boxes, selection options, and flashing lights. Depending on the level of emphasis required, visual display messages may be paired with various types of audio alerts or alarms.

24.5.1.1.2 Visual Notification, Alert, and Alarm Indicators

Principle 24.16

Visual Indicators Principle

Provide visual notifications, alerts, and alarms to inform the User about conditions that may impact System performance, cause damage, or threaten the safety of its User(s).

Three types of cues are commonly used as indicators for completions, cautions, and warnings for conditions to be discussed later in Figure 30.1. Examples include Power On, Completion, Interrupt, Caution, Master Caution, and Warning notifications, alerts, and alarms. For example, an aircraft has an Integrated Cautions and Warnings (ICAW) cockpit instrumentation alarm. Most of these are implemented as individual control panel lamps or in combination with visual display status screen information, display dialogue boxes with completion percentages, or bar graphs:

- **Power Indicators**—Provide visual feedback that power has been applied, is active, or is available; that overload conditions have occurred, or back-up power has been applied.
- **Status Indicators**—Provide status of a process that has been activated or in a state of completion.
- **Completion Indicators**—Provide visual indications that an activity has completed processing.
- **Caution Signals (Cues)**—"alert the operator to an impending dangerous condition requiring attention, but not necessarily immediate action" (MIL-HDBK-1908B, 1999, p. 8).

- **Warning Signal**—"*gen.* A signal which alerts the operator to a dangerous condition requiring immediate action" (MIL-HDBK-1908B, 1999, p. 34).
- **Master Caution (Warning) Signal (Cues)**—"*gen.* A signal which indicates that one or more caution (warning) lights have been actuated" (MIL-HDBK-1908B, 1999, p. 21).

Other types of visual notifications include signage that alerts the User to various levels of dangers and perils. Nelson, G. (1990) provides additional information in his paper *Essential Elements of Warnings and Instructions*.

Mini-Case Study 24.2

Signal Technology Advancements

Traditionally, large, complex systems consisted of control rooms, cockpits, hospital nurse stations, and security systems that monitored numerous performance parameters via indicator lights. State of the practice at the time consisted of hardwiring signals to a central monitoring location. When Caution or Warning conditions occurred such as illustrated in Figure 30.1, electronic circuits set to detect threshold conditions activated visual and auditory Caution or Warning Signals. The cost of implementation, reliability, and continual maintenance were continual issues.

Some of these still exist today requiring a single operator to monitor each individual light. As a result, operators and pilots were trained in visual scanning techniques to ensure that all indicators were viewed periodically. In terms of Reason's Error Classifications—slips, lapses, and mistakes—Caution Signals were sometimes electronically Boolean "ORed" into a single Master Caution Signal to indicate when any one Caution Signal is activated.

Fortunately with today's technology and automation, these same parameters can be tracked via distributed or remotely located systems networked via Ethernet or wirelessly into a central control station and then globally via the Internet to smartphones and other devices. Despite the technology advancements, there are still needs for indicator lamps, especially for locating problem devices such as transformers on electrical power poles; massively large facilities with limited lighting; and indicator panel lights on modems.

24.5.1.1.3 Auditory Cues

Principle 24.17

Auditory Cues Principle

Provide User auditory cues for notifications, alerts, and alarms commensurate with the level of significance and their spectral audio range.

Audio cues consist of warnings, cautions, and alerts that notify the operator or maintainer of specific equipment health and status conditions that require attention with varying levels of urgency for action. Various tonal frequencies as well as sequences and patterns of tones are employed to symbolize system-operating conditions. For example, a smartphone can be programmed to initiate specific types of ring tones with various callers, alerts, and emails. Auditory cues are often employed to alert system operators, especially when they are not attentive or observing the visual cues that require immediate attention such as conditions illustrated later in Figure 30.1:

- **Audio output devices** consist of electromechanical mechanisms such as speakers and headphones that communicate tones or messages to the SYSTEM operator(s).

24.5.1.1.4 Vibratory Cues

Where auditory or visual cues are undesirable, inappropriate, and impermissible, *vibratory* cues may be employed to alert system operators. Vibratory cues consist of devices that employ electronic mechanisms that vibrate on command for a pre-programmed period of time. Examples include cell phones and pagers.

Heading 24.1

Given an understanding of types of human–system interaction cues, *how are these cues communicated*? This brings us to our next topic, SYSTEM C2 devices.

24.5.2 Operator C2 Devices

Principle 24.18

C2 Devices Principle

Select C2 devices that are *appropriate*, *compatible*, and *interoperable* with User anthropometrics and biomechanics.

Principle 24.19

Escape Mechanisms Principle

Provide operational escape mechanisms—safety latches, stop switches or alarms, abort processing, restarts, and fire suppression systems—that allow the User to *safely* override or exit conditions that require immediate corrective actions without causing panic, injury, damage, or catastrophic results.

PERSONNEL require I/O mechanisms to MC2 SYSTEM or PRODUCT operations and performance. Let's identify some of the various types of I/O devices that serve as candidate solutions for the C2 portion of MC2:

- **Data entry devices** consist of electromechanical–optical mechanisms such as keyboards or touch panels that enable system operators and maintainers to enter alphanumeric information and data.

- **Pointing control devices**, such as a trackball, eyeball trackers, touchpad, or mouse, enable system operators and maintainers to point, select, manipulate, or maneuver data such as "drag and drop."
- **Steering devices** consist of steering wheels and joysticks on ground vehicles; yokes, sticks, and rudder pedals on aircraft; and thrusters on spacecraft.
- **Mechanical control I/O devices** include mechanical tools that enable operators to calibrate, align, control, or adjust the system configuration, operation, and performance.
- **Electronic control I/O devices** consist of electronic or electromechanical mechanisms, such as remote controls, toggle or rotary switches, dials, and touch screen displays, configured to communicate operator-controlled pointing position or displacement to specific data items.
- **Translational displacement control devices** such as joysticks and track balls employ electronics that transform or translate mechanical movements by angular displacement, stress, or compression into electronic signals that are used to control systems.
- **Sensory I/O devices** consist of mechanisms that sense the presence, degree, proximity, and strength of human interaction.
- **Audio input devices** consist of microphones that translate sound waves into inputs that are compatible with and recognized by the system's voice recognition software commands to perform an action or perform speech-to-text actions—dictation—for notes and records.

Heading 24.2

At this juncture, we have explored HSI beginning with a high-level perspective using Reason's Accident Trajectory Model and driven down to the physical implementation of the User–System interface. Using this information as a backdrop, let's address how it applies to Complex System Development.

24.6 COMPLEX SYSTEM DEVELOPMENT

Principle 24.20

Unwarranted Assumptions Principle

"*Unwarranted* assumptions in design can produce *unintended* consequences" (Madni, 2011, p. 5).

On inspection, HF, HE, HFE, and Ergonomics may appear to be simple. However, the reality is that humans develop a level of *trust* and *confidence* in a system, product, or service, especially when their lives depend on it, to prevent injury or death. Madni (2011, p. 4 - 5) identifies several challenges that increase the complexity of these decisions. The challenges include

1. Human Performance
2. Human Error
3. Human Adaptivity
4. Multitasking
5. Decision-making under Stress
6. User Acceptance
7. Risk Perception and Behavior
8. HSI

In general, a potential problem may be an increasing PERSONNEL dependence on technology and automation to supplement human shortcomings such as Reason's (1990b) slips, lapses, mistakes, and violations. Madni (2011, p. 5) observes that "Poor Automation Design Can Degrade Human Performance." Examples include:

- Cognitive Load in Supervising Automation
- Automation-Induced Complacency
- Partially Automated System with Incomplete Knowledge
- Mistrust of Automation
- Erosion of Operator's Expertise and Engagement

To illustrate these points, Pasztor (2013) reported in a *Wall Street Journal* article that a study commissioned by the US FAA indicates that pilots may rely too heavily on automation—*automation dependence*. As such, there may be a reluctance to intervene with the automation when confronted with unusual circumstances. As a result of the automation dependence, piloting skills required to fly the plane *manually*, *if required*, may also be diminishing.

24.7 SE HF AND ERGONOMICS ACTIONS

In summary, *what actions should be taken to ensure successful HSI*? The answer resides in five key objectives for SEs.

24.7.1 Objective #1: Employ Competent, Qualified HF Engineers

A key theme of Chapter 24 is that Engineers, in general, do not have the education, qualifications, or experience to perform HFE. What is important is to recognize the need to employ the services of HFEs who are *competent* and *qualified*. Then, based on information provided here, Engineers should (1) be able to converse and understand the discipline and (2) oversee implementation of these actions,

(3) be a User's Advocate, and (4) collaborate with the HFEs to make technical, technology, cost, schedule, and risk AoA decisions.

24.7.2 Objective #2: User Advocacy and System Usability

Principle 24.21

System Usability Principle

System, product, or service *usability* is key to winning User hearts, minds, confidence, and acceptance—key elements of customer satisfaction.

SEs should always collaborate with the Users and factor their reviews comments into decision-making processes. When the User may not be available, be a User's Advocate. A central focus on User Advocacy is ensuring the *usability* of a system, product, or services.

ISO 9241-11 (1998) defines *usability* in terms of its *effectiveness*, *efficiency*, and *satisfaction* for a specific context of use. The NAP (2007, p. 192) defines each of these attributes as follows:

- **Effectiveness**—"a measure of how well users can perform the job accurately and completely."
- **Efficiency**—"a measure of how quickly a user can perform work and is generally measured as task time, which is critical for productivity."
- **Satisfaction**—"the degree to which users like the product - a subjective response that includes the perceived ease of use and usefulness. Satisfaction is a success factor for any products with discretionary use, and essential to maintain workforce motivation."

In other words,

$$\text{Usability} = f(\text{effectiveness, efficiency, and satisfaction}) \tag{24.1}$$

Nielsen (2013) defines *usability* in terms of Five Quality Components, which include two of the items listed above by ISO 92411-11 (1998):

1. "**Learnability**: How easy is it for users to accomplish basic tasks the first time they encounter the design?
2. **Efficiency**: Once users have learned the design, how quickly can they perform tasks?
3. **Memorability**: When users return to the design after a period of not using it, how easily can they reestablish proficiency?
4. **Errors**: How many errors do users make, how severe are these errors, and how easily can they recover from the errors?
5. **Satisfaction**: How pleasant is it to use the design?"

For additional information about HF and Usability, refer to the following references:

- Nielsen (1995), 10 Usability Heuristics for User Interface Design
- Zhang, (2003), Using Usability Heuristics to Evaluate Patient Safety of Medical Devices
- Fennigkoh (2011), The Complexities of the Human-Medical Device Interface

24.7.3 Objective #3: Do No Harm

Principle 24.22

Fail Safe Design Principle

Where necessary, every system, product, or service should incorporate *fail safe* design features that prevent or minimize human error that could result in injury or loss of life or damage to the environment.

When humans interact with EQUIPMENT, the potential for performing *unsafe acts*—slips, lapses, errors, mistakes, and violations—increases. Design and implement the EQUIPMENT Element to DO NO HARM—*prevent* or *minimize* the potential for human errors that may injure the User or the public, damage or pollute the environment, or result in loss of life. Consider *fail-safe* designs where any of the *unsafe acts* are likely to occur.[4]

24.7.4 Objective #4: Reduce PERSONNEL Element Stress Points (SPs)

Optimal PERSONNEL Element performance occurs where focused knowledge, education, and training are applied with minimal or no distractions. Human performance under *stressful* conditions can transform potential hazards into *unsafe acts* that can lead to *incidents* and *accidents* as illustrated by Reason's (1990b) Accident Trajectory Model. *How can you reduce* PERSONNEL *Element stress*? There are several ways using Figure 24.6 as noted by the SP icons. Examples include collaboration with the Users to design, prototype, and develop the:

1. EQUIPMENT Element to be (1) *compatible* and *interoperable* with the capabilities and limitations of its operators, maintainers and trainers and (2) free of *latent defects*.

[4]Refer to Nelson (1990, 1993, and 2007) for additional information concerning the basic elements and core principles of product safety engineering.

2. PROCEDURAL DATA Element information—processes, methods, and procedures—that (1) is *concurrent*, *accurate*, and *consistent* with the EQUIPMENT Element design; (2) eliminates *redundant* or *unnecessary* non-value-added steps; and (3) is easy to understand, comprehend, and implement.
3. MISSION RESOURCES Element information to *clearly* and *concisely* (1) communicate MISSION SYSTEM and ENABLING SYSTEM tasking, objectives, outcomes, and performance; (2) avoid, eliminate, or prevent overlaps, misunderstandings, misinterpretation, or Person-to-Person conflicts; and (3) reduce workload to an acceptable level based on the skills of the Users.
4. SYSTEM RESPONSES Element to produce acceptable outputs and avoid unacceptable outputs that are incompatible with the User. For example, a human pilot may not be able to survive the *unacceptable* g-forces that new, high-performance aircraft technology might provide. In turn, this may lead to the need for development of a new pilotless aircraft that can achieve a level mission performance that is unattainable with existing piloted aircraft.
5. FACILITIES Element, if applicable, to accommodate the (1) training of the PERSONNEL Element to *properly* and *safely* operate, maintain, and sustain the EQUIPMENT Element and (2) maintenance and sustainment or the EQUIPMENT Element.

24.7.5 Objective #5: Continuously Assess System Usability and Performance

Despite best-made plans and HFE, User acceptance of a system, product, or service ultimately comes down to whether they emotionally "like it or not"—its *Usability*. Recall from our earlier discussions in Chapter 3, the attributes that impact a User's acceptance of a system, product, or service—Operational Utility, Suitability, Availability, Usability, Efficiency, and Effectiveness. Implicitly, these represent Psychological HF identified in Table 24.3.

Therefore:

- System Developer SEs and HFEs should collaborate with the User to assess these attributes throughout the System Development Phase of the System/Product Life Cycle.
- User SEs and HFEs should continuously assess these attributes throughout the System OM&S Phase (Chapter 29) to (1) institute *corrective actions* to improve overall performance and eliminate *latent defects*, (2) assess User performance to determine the need for *proficiency* or *remedial* training, and (3) formulate and develop next-generation HF requirements as new technologies evolve.

24.8 CHAPTER SUMMARY

In summary, we have addressed the importance for User HF and Ergonomics System Design. Key points include:

- Understanding how HF and Ergonomics impact system, product, or service performance via Reason's (1990b) "Swiss Cheese" Accident Trajectory Model.
- How the System Elements—PERSONNEL, EQUIPMENT, MISSION RESOURCES, PROCEDURAL DATA, SYSTEM RESPONSES, and FACILITIES—should be designed to serve as defenses, barriers or safeguards addressed by Reason (1990) to minimize hazards from evolving into accidents.
- Why Reason's (1990b) Human Error Classifications—slips, lapses, mistakes, and violations—are important and how PERSONNEL Element training, EQUIPMENT Element design, and Procedural Data should be designed and implemented as safeguards to prevent hazards from becoming accidents.
- Accidents are typically not the result of a single root cause but several weaknesses in the *defenses*, *barriers* and *safeguards* of Reason's (1990b) Accident Trajectory Model.
- The objective of HSI is to ensure that HF and Ergonomics requirements, design, and implementation considerations occur throughout the System Acquisition, Development, Deployment OM&S, and Retirement/Disposal Phases of the System/Product Life Cycle.
- Understand the contexts and differences between HSI, HFE, HE, and Ergonomics.
- HF and Ergonomics are viewed as *synonymous* but have different approaches that focus on the same outcomes—human workloads, stress, safety, injury reduction, productivity, and performance:
- HF focus on levying human capabilities and limitations as constraints on task workloads, EQUIPMENT design, PROCEDURAL DATA Element instruction, and OPERATING ENVIRONMENT conditions.
- Ergonomics focuses on how workload tasks, EQUIPMENT design, and Operating Environment conditions create stressful situations that impact human performance and productivity.
- Anthropometrics focus on the physical characteristics of humans.
- Biometrics is a subspecialty of Anthropometry that focuses on the mechanical movements of humans, their capabilities, and levels of performance.

- Human–System interactions can be modeled using an HF Interactions Model (Figures 24.9 and 24.10) based on the work of Meiser (1971).
- The applications of HF and Ergonomics are vital for reducing stress in various types of Human–System interactions—desk, standing, vehicle, and environments—that require Situational Assessment and C2 (Figures 24.11, 24.12).

HFE performs a vital role in:

- Identifying HF and Ergonomics requirements.
- Allocating HF requirements to the Personnel and Equipment Elements using the strengths and limitations of each (Figure 24.14) to determine the optimal mix to achieve overall System performance.
- Evaluating Equipment, Procedural Data (operator and training manuals), hardware and software designs and prototypes in collaboration with the Users to assess usability and compliance to HF and Ergonomics design principles.
- Supporting SITE Verification and Validation (V&V) activities.
- Analyzing User field experiences, problems, and issues to identify any latent defects and the need for corrective action.
- Evaluate User performance concerning the need for basic, remedial, or proficiency training (Chapter 33).
- Participating in *incident* or *accident* investigations to determine root causes and recommend corrective actions.
- *Usability* is a function of the EQUIPMENT Element's *efficiency*, *effectiveness*, and level of *satisfaction*. Nielsen (2013) adds *learnability*, *errors*, and *memorability* as other quality components.
- For additional reading, refer to Norman (2013, Figure 1.11, p.32) concerning the importance of understanding the User's conceptual mental models and their relationship to good system design practices.

24.9 CHAPTER EXERCISES

24.9.1 Level 1: Chapter Knowledge Exercises

1. What are HF?
2. What are the five types of HF?
3. What is Anthropometry?
4. What is Biomechanics?
5. What are haptics?
6. What is Ergonomics?
7. What is HSI and what are the HSI domains?
8. What types of I/O devices are available for Human–System interfaces?
9. What are the key attributes of human tasks?
10. What is HFE and how is it applied to the System/Product Life Cycle?
11. What is Reason's (1990) "Swiss Cheese" Accident Trajectory Model, and how does it apply to HF and Ergonomics?
12. What is the Human Factors Interactions Model, and how does it apply to PERSONNEL–EQUIPMENT Element design?
13. What are the criteria HF Engineers use to allocate specification requirements to the PERSONNEL and EQUIPMENT Elements?
14. Human Factors and Ergonomics share at least 7 System outcome concerns. What are they?
15. Compare and contrast the DoD CPAT and NASA HF perspectives listed in Tables 24.3 and 24.4.

24.9.2 Level 2: Chapter Knowledge Application Exercises

Refer to www.wiley.com/go/systemengineeringanalysis2e

24.10 REFERENCES

AFSC (1977), *Air Force Systems Command Design Handbook*, 3rd Edition, Wright-Patterson Air Force Base (WPAFB): Air Force Systems Command.

Belke, James C. (1998), Recurring Causes of Recent Chemical Accidents, U.S. Environmental Protection Agency (EPA), Chemical Emergency Preparedness and Prevention Office, San Antonio, Texas: AIChE/CCPS Workshop on Reliability and Risk Management.

Chapanis, Alphonse (1996), *Human Factors in Systems Engineering*, New York: John Wiley & Sons, Inc.

CPATS HFE (1998), CPATS (Critical Process Assessment Tool) – Human Factors Engineering (HFE), Military Specifications and Standards Reform Program (MSSRP), 14 August, 1998, Los Angeles, CA: USAF SMC/AXMP. Retrieved on 3/7/15 from http://www2.mitre.org/work/sepo/toolkits/risk/taxonomies/files/CPATS_HSI_14Aug98.DOC.

Clark, James J. and Goulder, Robert K (2002), *Human Systems Integration (HSI) - Ensuring Design & Development Meet Human Performance Capability Early in Acquisition Process*, PM Magazine: July – August 2002, Ft. Belvoir, VA: Defense Acquisition University (DAU).

DAU (2012), *Glossary: Defense Acquisition Acronyms and Terms*, 15th ed. Ft. Belvoir, VA: Defense Acquisition University (DAU) Press. Retrieved on 3/27/13 from http://www.dau.mil/pubscats /PubsCats/Glossary%2014th%20edition%20July%202011.pdf.

DoD 5000.59-M (1998), *DoD Modeling and Simulation (M&S) Glossary*, Washington, DC: Department of Defense (DoD). Retrieved on 8/1/13 from http://www.dtic.mil/whs/directives /corres/pdf/500059m.pdf.

DOD-HDBK-743A (1991), *Anthropometry of U.S. Military* Personnel *(Metric),* Washington, DC: Department of Defense (DoD).

DoDI 5000.02 (2008), *Operation of the Defense Acquisition System*, Washington, DC: Department of Defense.

EEC (2006), *Revisiting the Swiss Cheese Model of Accidents*, EEC Note No. 13/06 Brétigny-sur-Orge, France: Eurocontrol Experimental Centre (EEC). Retrieved on 10/22/13 from http://www.eurocontrol.int/eec/gallery/content/public/document /eec/report/2006/017_Swiss_Cheese_Model.pdf.

FAA SEM (2006), *System Engineering Manual*, Version 3.1, Vol. 3, National Airspace System (NAS), Washington, DC: Federal Aviation Administration (FAA).

FAA-H-8083-30 (2008), *Aviation Maintenance Technician Handbook - General*, Washington, DC: U.S. Department of Transportation (DOT) Federal Aviation Administration (FAA). Retrieved on 10/16/13 from http://www.faa.gov/regulations_ policies/handbooks_manuals/aircraft/media/AMT_Handbook_ Addendum_Human_Factors.pdf.

FAA (2014), *Human Factors Model webpage*, Washington, DC: U.S. Department of Transportation (DOT) Federal Aviation Administration (FAA). Retrieved on 5/22/14 from http://www .hf.faa.gov/Webtraining/HFModel/HFInterModel/overview.htm.

Fennigkoh, Larry (2011), *The Complexities of the Human-Medical Device Interface*, January/February, 2011, *Biomedical Instrumentation & Technology*, Arlington, VA: Association for the Advancement of Medical Instrumentation (AAMI).

Hatze, Herbert (1974). "*The meaning of the term biomechanics*", Vol. 7, New York: Elsevier Science *Journal of Biomechanics.* Retrieved on 10/24/13 from http://biomechanics.psu.edu.

HFES (2013a), *Definitions of Human Factors and Ergonomics webpage*, Santa Monica, CA: Human Factors and Ergonomics Society (HFES). Retrieved on 10/19/13 from http://www.hfes.org /Web/EducationalResources/HFEdefinitionsmain.html.

HFES (2013b), *About HFES webpage*, Santa Monica, CA: Human Factors and Ergonomics Society (HFES). Retrieved on 10/19/13 from https://www.hfes.org//Web/AboutHFES/about.html.

HSE (2013a), *Humans and Risk*, HSE Human Factors Briefing Note No. 3, Liverpool, Merseyside, England: UK Health and Safety Executive (HSE). Retrieved on 10/16/13 from http://www.hse .gov.uk/humanfactors/topics/pifs.pdf.

HSE (2013b), *Performance Influencing Factors (PIFs)*, Liverpool, Merseyside, England: UK Health and Safety Executive (HSE). Retrieved on 10/16/13 from http://www.hse.gov.uk /humanfactors/topics/pifs.pdf.

IEA (2010a), *Definitions of Ergonomics webpage*, Zurich, Switzerland: International Ergonomics Association (IEA). Retrieved on 10/19/13 from http://www.iea.cc/01_what/What%20is%20 Ergonomics.html.

IEA (2010b), *About IEA webpage*, Zurich, Switzerland: International Ergonomics Association (IEA). Retrieved on 10/19/13 from http://www.iea.cc/02_about/About%20IEA.html.

INCOSE TP-2003-002-03.2.2 (2011), *System Engineering Handbook*, Version 3.2.2, San Diego, CA: International Council on System Engineering (INCOSE).

INCOSE (2015). *Systems Engineering Handbook: A Guide for System Life Cycle Process and Activities*, 4th ed. D. D. Walden, G. J. Roedler, K. J. Forsberg, R. D. Hamelin, and, T. M. Shortell (Eds.). San Diego, CA: International Council on Systems Engineering.

ISO 1503:2008 (2012) *Spatial orientation and direction of movement – ergonomic requirements*. Geneva: International Organization for Standards (ISO).

ISO 9241-210:2010 (2010) *Ergonomics of human-system interaction – Part 210: Human-centred design for interactive systems*. Geneva: International Organization for Standards (ISO).

ISO/IEC Guide 51:1999 (2009) *Safety aspects -- Guidelines for their inclusion in standards*. Geneva: International Organization for Standards (ISO).

Kantowitz, Barry H. (1983), *Human Factors: Understanding People-System Relationships*, Hoboken, NJ: John Wiley & Sons, Inc.

Kroemer, Karl H. E. (2010), *Engineering Physiology: Bases for Human Factors Engineering/Ergonomics*, 4th Edition, Berlin: Springer-Verlag.

Licht, Deborah M., Polzella, Donald J., Boff, Kenneth R (undated), Human Factors, Ergonomics, and Human Factors Engineering: An Analysis of Definitions, Wright-Patterson Air Force Base (WPAFB): Crew System Ergonomics Information Analysis Center (CSERIAC). Retrieved on 10/22/13 from http://www.hfes.org/Web/EducationalResources/HFDefinitions .pdf.

McCormick, Ernest J. (1964), *Human Factors Engineering*, 2nd Edition, New York: McGraw-Hill Company.

McCormick, Ernest J. and Sanders, Mark S. (1982), *Human Factors in Engineering Design*, 5th Edition, New York: McGraw-Hill Company.

Madni, Azad M. (2011), "*Integrating Humans With and Within Complex Systems*," University of Southern California, *Crosstalk* - May/June 2011, Hill AFB, UT: *CrossTalk*. Retrieved on 9/27/13 from http://www.crosstalkonline.org/storage /issue-archives/2011/201105/201105-Madni.pdf.

Meister, David (1971), *Human factors: theory and practice*, Hoboken, NJ: Wiley-Interscience.

Meister, David (1999), *The History of Human Factors and Ergonomics*, Mahwah, NJ: Lawrence Erlbaum Associates.

MIL-HDBK-1908B (1999), *DoD Definitions of Human Factors Terms,* Washington, DC: Department of Defense (DOD).

MIL-STD-470B (1989), *Maintainability Program for Systems and Equipment*. Washington, DC: Department of Defense (DoD).

MIL-STD-882E (2012), *System Safety,* Washington, DC: Department of Defense (DoD).

MIL-STD-1472G (2012), *DoD Design Criteria Standard: Human Engineering*, Washington, DC: Department of Defense (DoD).

MIL-STD-46855A (2011), *Human Engineering Requirements for Military Systems, Equipment, and Facilities*, Washington, DC: Department of Defense (DoD).

NAP (2007), Human-System Integration in the System Development Process: A New Look, Committee on Human-System Design Support for Changing Technology, Richard W. Pew and Anne S. Mavor, Editors, Committee on Human Factors, National Research Council. Washington, DC: The National Academies Press. Retrieved on 10/22/13 from http://www.nap.edu/catalog.php?record_id=11893.

NASA RP-1024 (1978) *Anthropometric Source Book*, Vols. 1–3. Washington, DC: NASA Scientific & Technical Office.

NASA SP-2007-6105 (2007), *System Engineering Handbook*, Rev. 1. Washington, DC: National Aeronautics and Space Administration (NASA). Retrieved on 5/1/13 from https://acc.dau.mil/adl/en-US/196055/file/33180/NASA%20SP-2007-6105%20Rev%201%20Final%2031Dec2007.pdf.

NASA SP-2010-3407 (2010), *Human Integration Design Handbook (HIDH)*, Washington, DC: NASA. Retrieved on 10/21/13 from http://ston.jsc.nasa.gov/collections/TRS/_techrep/SP-2010-3407.pdf.

NASA-STD-3000 (1995), *Man-Systems Integration Standards*, Vols. 1–5, Washington, DC: NASA.

National Center for Human Factors Engineering in Healthcare (2013), *What is Human Factors Engineering webpage*, Washington, DC: National Center for Human Factors Engineering in Healthcare. Retrieved on 10/19/13 from http://medicalhumanfactors.net/what-is-hfe.

Nielsen, Jakob, (2013), *Usability 101: Introduction to Usability*, Fremont, CA: Nielsen Norman Group. Retrieved on 10/22/13 from http://www.nngroup.com/articles/usability-101-introduction-to-usability/.

Nielsen, Jakob, (1995), *10 Usability Heuristics for User Interface Design*, Fremont, CA: Nielsen Norman Group. Retrieved on 10/22/13 from http://www.nngroup.com/articles/ten-usability-heuristics/.

Nelson, Gary (1990), *Essential Elements of Warnings and Instructions*, TX: Nelson and Associates. Retrieved on 10/22/13 from http://www.hazardcontrol.com/factsheets/pdfs/essential-elements-of-warnings-and-instructions.pdf.

Nelson, Gary (1993), *Basic Elements of Product Safety Engineering*, TX: Nelson and Associates. Retrieved on 10/22/13 from http://www.hazardcontrol.com/factsheets/pdfs/basic-elements-of-product-safety-engineering.pdf.

Nelson, Gary (2007), *Core Principles of Safety Engineering and the Cardinal Rules of Hazard Control*, TX: Nelson and Associates. Retrieved on 10/22/13 from http://www.hazardcontrol.com/factsheets/pdfs/core-principles.pdf.

Nelson, Gary (2010a), *"Human Error" vs. "Human Nature"*, Bryan, TX: Nelson and Associates. Retrieved on 10/22/13 from http://www.hazardcontrol.com/factsheets/pdfs/human-error-vs-nature.pdf.

Nelson, Gary (2010b), *Human Factors and Ergonomics*, Bryan, TX: Nelson and Associates. Retrieved on 10/22/13 from http://www.hazardcontrol.com/factsheets/humanfactors/human-factors-and-ergonomics.

Norman, Don (2013), *The Design of Everyday Things*, New York: Basic Books.

OSHA 3125 (2000 Revised), *Ergonomics: the Study of Work*, Washington, DC: Department of Labor Occupational Safety and Health Administration (OSHA). Retrieved on 10/25/13 from https://www.osha.gov/Publications/osha3125.pdf.

Pasztor, Andy (2013), *Pilots Rely Too Much on Automation, Panel Says*, Wall Street Journal, Nov. 17, 2013, New York, NY: Wall Street Journal. Retrieved on 5/22/14 from http://online.wsj.com/news/articles/SB10001424052702304439804579204202526288042.

Psychology Wiki (2013), *Human Factors Engineering webpage*, San Francisco, CA: Wikia, Inc. Retrieved on 10/22/13 from http://psychology.wikia.com/wiki/Human_factors_engineering.

Reason, James (1990a), *The contribution of latent human failures to the breakdown of complex systems, Series B*, Vol. 327, pp. 475–484, London: Philosophical Transactions of the Royal Society.

Reason, James (1990b), *Human Error*, Cambridge, UK: Cambridge University Press.

Reason, James (2000), *Human error: models and management*, Vol 320, London: *British Medical Journal*.

Roebuck, J. A. Jr.,, Kroemer, K. H. E., and Thomson, W. G. (1975), *Engineering Anthropometry Methods*, New York: John Wiley & Sons, Inc.

USAF AFD-090121-055 (2009), *Human Systems Integration Requirements Pocket Guide*, Falls Church, VA: USAF Air Force Human Systems Integration Office. Retrieved on 10/25/13 from http://www.wpafb.af.mil/shared/media/document/AFD-090121-055.pdf.

USAF AFD-090121-054 (2009), *Human Systems Integration Handbook*, Brooks City-Base, TX: USAF Human Performance Optimization Division. Retrieved on 10/27/13 from http://www.wpafb.af.mil/shared/media/document/AFD-090121-054.pdf.

Wikipedia (2013a), *Human Factors and Ergonomics webpage*, San Francisco, CA: Wikimedia Foundation. Retrieved on 10/19/13 from http://en.wikipedia.org/wiki/Ergonomics.

Wikipedia (2013b), *Biomechanics webpage*, San Francisco, CA: Wikimedia Foundation. Retrieved on 10/16/13 from http://en.wikipedia.org/wiki/Biomechanics.

Zhang, Jiajie; Johnson, Todd R.; Patel, Vimla L.; Paige, Danielle L.; and Kubose, Tate (2003), "*Using Usability Heuristics to Evaluate Patient Safety of Medical Devices*," *Journal of Biomedical Informatics* 36 (2003) 23–30, Miamisburg, OH: Reed Elsevier Inc. - ScienceDirect Division). Retrieved on 9/27/13 from http://www.sciencedirect.com/science/article/pii/S1532046403000601.

25

ENGINEERING STANDARDS OF UNITS, COORDINATE SYSTEMS, AND CONVENTIONS

As the System Architecture evolves, interactions between the SYSTEM/PRODUCT and its OPERATING ENVIRONMENT and internal elements such as SUBSYSTEMS must be *compatible* and *interoperable*. This requires that both sides of the interface comply with interface requirements. For some systems, the interface may require:

- Collaboration between interfacing parties to establish interface requirements
- Agreement to comply with industry, professional, or scientific interface standards
- Use of previously established legacy interfaces

Whichever is the case, multi-disciplined SEs on both sides of the interface must synchronize in thought, process, and methods, and share a common system design perspective for accomplishing the interface. One of the first steps for ensuring a common perspective resides in the establishment of Engineering standards of units, coordinate systems, and conventions that form the basis of interface requirements.

Engineers often treat Engineering standards of units, coordinate systems, and conventions as secondary topics … "something every engineer knows from Engineering statics, dynamics, and physics." However, there is a major difference between an Engineer's *knowledge* at graduation versus Systems Engineering *leadership*, o*rchestration, and application* of a shared vision and mindset among groups of multi-disciplined engineers developing systems, products, or services. Many systems have been developed that failed System Integration, Test, and Evaluation (SITE) or their missions due to simple Engineering errors in interpreting, translating, and communicating a SYSTEM's: (1) standard of units, (2) weights and measures, (3) coordinate reference system, and (4) its location and dynamics as a free body in geophysical space.

Chapter 25 emphasizes one of the critical leadership roles of SEs, to establish Engineering standards, coordinate systems, and conventions "up front" as one of the cornerstones of system design Decision making. Our discussion explores each of these topical areas and provides examples to illustrate the importance of synchronizing design mindsets. It is critically important that everyone have a common, shared viewpoint of a project's Standards of Units, Weights and Measures, and Coordinate Systems at the beginning of a project. Simply stated in Principle 26.12, if you want to Monitor, Command, and Control (MC2) a system, product, or service, you have to measure its performance.

25.1 DEFINITIONS OF KEY TERMS

- **Altitude**—The vertical distance measured outward into free space from a point of interest located on the surface of a standard body reference such as Mean Sea Level (MSL).
- **Azimuth**—The clockwise angle formed by an arc between a reference plane such as True North (0°), and a vertical plane passing through a point of interest.

System Engineering Analysis, Design, and Development: Concepts, Principles, and Practices, Second Edition. Charles S. Wasson.

Companion Website: www.wiley.com/go/systemengineeringanalysis2e

- **Compliance**—The process of fully adhering to a requirement without exception.
- **Conformance**—The process of adapting or customizing Enterprise work products, processes, and methods to meet the *spirit and intent* of a required action or objective.
- **Convention**—A method established by project, industry, national, or international standards organizations for conveying how engineers are to interpret and apply System or Entity configurations, orientations, directions, or actions.
- **Coordinate System**—A two- or three-dimensional axis frame of reference used to establish a system configuration and orientation conventions, support analysis, and facilitate mathematical computations.
- **Dimension**—A physical property inherent to an object that is independent of the system of measure used to quantify its magnitude.
- **Elevation**—An angle formed by a plane tangential to the Earth's surface at an observer's location and a point of interest above or below that plane.
- **Euler Angles**—"A set of three angles used to describe the orientation of an entity as a set of three successive rotations about three different orthogonal axes (x, y, and z). The order of rotation is first about z by angle (psi), then about the new y by angle (theta), then about the newest x by angle (phi). Angles psi and phi range between +/− pi, while angle theta ranges only between +/− pi/2 radians. These angles specify the successive rotations needed to transform from the world coordinate system to the entity coordinate system. The positive direction of rotation about an axis is defined by the right-hand rule" (DoD 5000.59-M (1998), p. 113).
- **Open Standards**—"Widely accepted and supported standards set by recognized standards organizations or the market place. These standards support interoperability, portability, and scalability and are equally available to the general public at no cost or with a moderate license fee" (DAU, 2012, p. B-153).
- **Performance Standard**—A requirement that establishes an outcome-based level of proficiency or performance to be achieved for an activity or competency.
- **Rotational Movements** – Positive or negative angular rotations and rates of change of a free body relative to the principal axes of its inertial frame of reference.
- **Standard**—"In work measurement, any established or accepted rule, model, or criterion against which comparisons are made" (DAU, 2012, p. B-211).
- **Technical Standard**—"A common and repeated use of rules, conditions, guidelines or characteristics for products or related processes and production methods. It includes the definition of terms, classification of components, delineation of procedures, specification of dimensions, materials, performance, designs, or operations. It includes measurement of quality and quantity as well as a description of fit and measurements" (NASA SOW NPG 5600.2BE (1997), Appendix B, Definitions).
- **Translational Movements** – Directional travel of a free body as a function of time - velocity –relative to the principal axes of its inertial frame of reference.

25.2 APPROACH TO THIS CHAPTER

Engineering Standards Principle

Principle 25.1 Engineering standards of units, coordinate systems, and conventions should be specified in *one and only one* official document that has been reviewed, approved, baselined and placed under formal Configuration Management (CM) control, released, and communicated for decision making.

Chapter 25 is not intended to be a refresher from Engineering and Physics concerning standards of units, coordinate system frames of reference, or conventions. Our purpose is to simply provide a general checklist that you, as a Systems Engineer and technical leader, need to consider when planning and implementing a technical program that manifests itself in specifications, architectures, designs, and interfaces.

Engineers cavalierly say, "We already know this!!"

- First, the collective "we" do not have a shared, common knowledge unless … you, as the Lead Systems Engineer (LSE) … establish, document, and communicate a project standard based on collaboration and consensus of the project's Stakeholders.
- Secondly, history is filled with cases even with these standards in place where by *simple* errors—*Metric* versus *English* units—on opposite sides of an interface resulted in failure. For example, National Aeronautics and Space Administration's (NASA) Mars Climate Orbiter (MCO, 1999) and (MCO, 2000).

This topic should be one of the first tasks to consider. Once this information is established, document it as Project Engineering Standard. The tendency is a scatter of this critical information across a series of project memoranda. Avoid this and incorporate it into a *single* document that has:

1. A document number, title, data, and version.
2. Been reviewed, approved, under formal CM control, and released for use on the project.

As a final point, one of the best case studies that illustrate coordinate systems NASA's Space Transportation System (STS) or Space Shuttle Program that ended in 2012. We use some of the Shuttle graphics to illustrate our discussions. As you read the chapter, these illustrations and coordinate systems apply similarly.

25.3 ENGINEERING STANDARDS

Engineering Standards, Coordinate Systems, and Conventions Principle

Principle 25.2 Engineering standards, coordinate systems, and conventions are the foundational Achilles Heal of Engineered Systems and Enterprise System development. Neglect them and the systems will fail.

Engineering standards provide a mechanism for Enterprises and industries to:

- Establish a consensus of performance requirements for development of systems, products, and services.
- Audit compliance of those deliverable work products.
- Provide a framework for targeting improvements related to performance and safety.

Standards evolve from lessons learned, best practices, and methods within an Enterprise and across industry domains. They:

- Ensure product compatibility and interoperability.
- Ensure consistency, uniformity, precision, and accuracy in materials, processes, weights, and measures.
- Promote modularity and interchangeability.
- Ensure the safety to the public and environment.
- Avoid the consequences of lessons learned.
- Promote ethical business relationships.

When a standard is employed as the basis for evaluating work-related performance, the term *performance standard* is employed.

25.3.1 Standard Normative and Informative Clauses

In general, standards express performance requirements via clauses. Standards clauses generally fall into two types: *Normative* clauses and *informative* clauses.

- *Normative clauses* or requirements express mandatory criteria for compliance with the standard and include the word "shall" to indicate required performance.
- *Informative clauses* express information for voluntary compliance or to provide guidance/clarification for implementing the normative requirements.

Local Enterprise Engineering standards should clearly delineate and designate *normative* and *informative* requirements.

25.3.2 Engineering Standards Authorities

Engineering standards are established by corporate, academic, professional, and international Enterprises. To posture themselves for this position, they must be recognized and respected within a given industry or business domain as the authoritative proponent or issuer of standard practices. Consult your contract, industry, and Engineering discipline for specific standards that may be applicable to your business and contract.

25.3.3 Dimensional Properties and Systems of Units

Standards express two types of information that serve as the frame of reference for measurements: *dimensional properties* and *systems of units*.

- *Dimensional properties* represent inherent physical properties of an object such as mass, length, width, and weight.
- *Systems of units* are standards of measure that form the basis for measuring the magnitudes of an object's dimensional properties.

25.4 STANDARDS FOR UNITS, WEIGHTS, AND MEASURES

Engineering standards represent a consensus of industry stakeholders that establish "Fitness for Use" Criteria (Figure 4.1) for a variety of applications. This includes: documentation, processes, methods, materials, interfaces, frames of reference, weights and measures, domain transformations, demonstrations, and conventions. Of these items, Engineering weights and measures, conventions and frames of reference require specific emphasis, especially in creating consistency across a contract program.

Perhaps the most fundamental concept of Engineering is establishing a system of *weights* and *measures*. Technical expressions that describe the form, fit, and function of a System, Product, or Service are totally dependent on usage of standard units for weights and measures. There are two primary standard systems of units in use today:

- International System of Units (SI)
- British Engineering System (BES)

Let's contrast each of these systems.

25.4.1 International System of Units (SI)

The SI was approved by the11th General Conference on Weights and Measures (CGPM) in 1960 (NIST, 2013). The CGPM adopted the SI designation from the French Le Système International d'Unitès. The SI, which is the Brochure published by the International Bureau of Weights and Measures (BIPM), is "… revised from time to time in accordance with the decisions of the CGPM and the International Committee for Weights and Measures (CIPM)" (Taylor and Thompson, 2008, p. 70).

The SI, which is sometimes referred to as the Metricor MKS for Meter–Kilogram–Second System, is based on seven base units that are determined to be independent. Table 25.1 provides a listing of the seven *base units*, which are considered mutually independent.

Principle 25.3

SI Fonts Principle

Document *quantity symbols* using an italic font and symbols for dimensions in sans serif roman capitals" (Taylor and Thompson, 2008, p. 4).

Taylor and Thompson (2008a, p. 33) note "Quantity symbols, which are always printed in italic … are, with few exceptions, single letters of the Latin or Greek alphabets that may have subscripts or superscripts or other identifying signs." They also provide tables of conversion factors for converting various types of units to the SI (Taylor and Thompson (2008a, pp. 45–69).

25.4.2 British Engineering System (BES)

The BES consists of five base units listed in Table 25.2.

25.4.3 Scientific Notation

In addition to defining the system of units for measurement, we need to express the magnitudes associated with those units in a manner that is easy to read. We do this with scientific notation as illustrated in Table 25.3.

TABLE 25.2 Base Units of the BES

Base Quantity	Name	Symbol
Length	Foot	ft
Force	Pound	lb
Time	Second	s
Temperature	Fahrenheit	°F
Luminous Intensity	Candle	

TABLE 25.3 Scientific Notation Symbology (Taylor and Thompson, 2008b, p. 29)

Power	Prefix	Notation
10^{-9}	nano	n
10^{-6}	micro	μ
10^{-3}	milli	m
10^{-2}	centi	c
10^{-1}	deci	d
10^{1}	deka	da
10^{2}	hecto	h
10^{3}	kilo	k
10^{6}	mega	M
10^{9}	giga	G
10^{12}	tera	T
10^{15}	peta	P

25.4.4 Standard Atmosphere

Systems and products interact with other external systems in their operating environment under a variety of NATURAL and INDUCED ENVIRONMENT conditions. As part of the *problem space* and *solution space* definitions, SEs and others have to make assumptions (Principle 24.20) that characterize and bound operating environment conditions.

Adding to the complexity of these assumptions is the fact that NATURAL ENVIRONMENT conditions vary throughout the day, month, year, and world. So, *how do we standardize to support informed operating environment decision-making*? The answer resides in creating the *US Standard Atmosphere*

TABLE 25.1 Seven Base Units of the International System of Units (SI) (NIST, 2008a, p. 4; 2008b, 11)

Base Quantity	SI Base Unit*	SI Base Symbol	Symbol for Quantity**	Symbol for Dimension**
Length	meter	m	*l, x, r,*	L
Mass	kilogram	kg	*m*	M
Time, duration	second	s	*t*	T
Electric current	ampere	A	*I, i*	I
Thermodynamic Temperature	kelvin	K	*T*	Θ
Amount of Substance	mole	mol	*n*	N
Luminous Intensity	candela	cd	I_v	J

*Taylor and Thompson (2008a), Para. 4.1 SI Base Units Table 1, p. 4.
**Taylor and Thompson (2008b), Para. 1.3 Dimensions of Quantities, p. 11.

developed by the US National Oceanic and Atmospheric Agency (NOAA), NASA, and the USAF (NASA, 1976). "Parameters listed include temperature, pressure, density, acceleration caused by gravity, pressure scale height, number density, mean particle speed, mean collision frequency, mean free path, mean molecular weight, sound speed, dynamic viscosity, kinematic viscosity, thermal conductivity, and geo-potential altitude" (NASA, 1976).

Standard Atmosphere models and tables for geographical locations are available from government organizations such as the US NOAA and NASA.

25.4.5 Data Accuracy and Precision

When data are measured or computed, it is critical for a program to establish a policy for data *accuracy* and *precision*. In addition to establishing mathematical models, transformations, and conversions, the integrity of the chain of computations is totally dependent on the accuracy of the data that feed each one.

To illustrate the importance of data precision, consider a simple definition of the mathematical symbol, pi (π). Are two digits of precision—namely 3.14 for pi—*necessary* and *sufficient* criteria for downstream computations? Four digits? Eight digits? You have to decide and lead agreement for the program standard.

Finally, a reminder that multiplication of two numbers, each with two-digits of decimal accuracy, does not yield a product that has four-digits of decimal accuracy (Principle 30.1).

25.4.6 Weights and Measures Summary

As the technical lead for a program, SEs should establish a program consensus regarding a standard system of units for expressing physical quantities. This requires three actions:

1. Establish a project standard for weights, measures, and unit conversion tables via project memorandums or references to an established professional standard.
2. Thoroughly *scrutinize* standards of units for interface compatibilities and interoperability on *both* sides of an interface throughout the System Development Phase, especially at reviews.
3. Promote professional discipline and compliance from project teams.

25.5 COORDINATE REFERENCE SYSTEMS

In addition to the establishment of standards for weights and measures, one of the key roles and initial tasks for SE during System Development is the establishment of coordinate reference systems. System Engineering design decisions require more than discipline specific topics such as designing power supplies, electronic circuits, or coding software. Integration of the set of components requires insightful decision making that goes beyond traditional discipline "form, fit, and function" considerations. Specifically, physical placement of components and their interactions within the next higher-level System, Subsystem, Assembly, or Subassembly (Figure 8.4).

General placement of components within higher levels requires consideration of Operating Environment conditions (Figure 9.6) such as: Electromagnetic Interference (EMI), radiation, thermal heat transfer, and mechanical shock and vibration. Physical placement of those components within the framework of an aircraft, spacecraft (Figures 25.6 and 25.7), satellite, ship, or missile; medical device (Figure 25.5) or surgical device (Figure 25.9); or machining a part (Figure 25.8) requires establishing an observer's frame of reference, designation of principal axes, and assignment of axis rotational conventions.

If you listen to Engineers at the beginning of a project, you will hear conversations such as, "We need to establish a coordinate system for the project. Let's use a Right-Handed Cartesian Coordinate System as the standard." As you will see in the sections that follow, establishing a coordinate reference system requires more insight than conversations like this. SE in its technical leadership role is accountable for ensuring that all ambiguities are eliminated and the selected coordinate system is clearly communicated to everyone.

Author's Note 25.1

Discussions of Coordinate Reference Systems are often mixed by ambiguities related to key terms such as the Left-Hand (LH) Rule and the Right-Hand (RH) Rule, each with two different application configuration contexts. Rather than perpetuate these ambiguities, let's briefly address their origins as a way of differentiating the application contexts.

25.5.1 Coordinate Reference System Origins

Coordinate Reference Systems originate from mathematician René Descartes in the seventeenth century, electronic pioneer Sir John Ambrose Fleming in the late nineteenth century, and Hans Christian Ørsted in the late eighteenth and early nineteenth centuries.

- Descartes devised the two-dimensional (2-D) and three-dimensional (3-D) Cartesian coordinate systems provided the first "systematic link between Euclidian Geometry and algebra" (Wikipedia, 2013a).

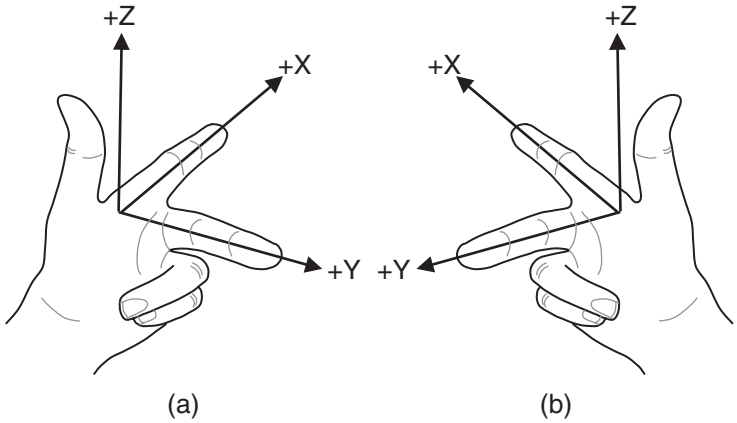


Figure 25.1 (a) Fleming's Left-Hand (LH) Rule for Motors and (b) Right-Hand (RH) Rule for Generators Applied to SE Applications

- Fleming devised the following rules for explaining the force, field, and current directions in motor and generator applications as shown in Figure 25.1:
  - LH Rule for explaining motion in electric *motor*. The configuration consists of three orthogonal axes formed by the fingers of the left hand in which the thumb represents the motor's thrust or force of motion, index finger represents the field direction, and middle finger represents the current flow direction. (Fleming, 1902, p. 149–150)
  - The RH Rule originally defined as *The Hand Rule for the Direction of the Electromotive Force* for explaining electrical current flow in a *generator*. The configuration consists of three orthogonal axes formed by the fingers of the right hand in which the *thumb* represents the conductor's direction, the *index finger* represents the magnetic flux (field) direction, and the *middle finger* represents the direction of the induced Electromotive Force (EMF) (Fleming, 1902, p. 173–174).
- The RH Grip or Maxwell's Corkscrew Rule (Wikipedia, 2013c) is based on Ørsted's discovery that a circular magnetic field is created by a current flowing through a conductor (Wikipedia, 2013d). The RH Grip Rule states that if the *thumb* of the right hand represents the direction of current flow in a conductor, fingers coiled around the conductor represent the counterclockwise magnetic flux or *positive* (+) rotation. (Fleming, 1902, p. 149). Figure 25.2 provides an illustration.

Since Coordinate Reference Systems enable us to establish geophysical and spatial relationships, we will simply assign +X, +Y, and +Z to the respective LH or RH convention axes as shown in Figure 25.1.

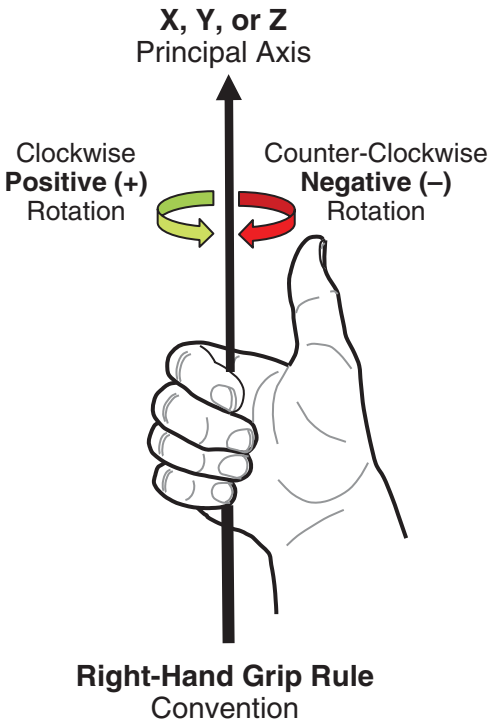


Figure 25.2 Right-Hand (RH) Grip rule Application to an Axis to Illustrate the Clockwise (+) and Counter-Clockwise (−) sign Conventions

As a level of abstraction, coordinate reference systems are comprised of: (1) an Observer's Frame of Reference, (2) conventions, and (3) units of measure that enable engineers and analysts to communicate spatial information about a current location, its shape, or distance from other objects. The question is: *from what observer's perspective*? This brings us to our first topic, the Observer's Frame of Reference.

25.5.2 Observer's Frame of Reference

The Engineering of Systems and Products often require that an Observer's *frame of reference* be established for characterizing the Engineering mechanics and dynamics of the System as a *free body* in motion relative to other systems in its Operating Environment. Basic concepts for Observer frames of reference originate from physics. To get started, let's begin with a few refresher principles.

Principle 25.4

Observer's Frame of Reference Eyepoint Principle

Human coordinate systems consist of an Observer's Frame of Reference structured with three principal axes labeled +X, +Y, and +Z that are orthogonal to each other and intersect at an origin located at the Observer's *eyepoint*.

Figure 25.3 provides an example illustration.

Principle 25.5

Coordinate System Principle

The Observer's *eyepoint* serves as an origin for an Observer's of Frame Reference; a coordinate system establishes the spatial location of a point or object in space relative to that origin using *x*-, *y*-, and *z*-axis coordinates.

Once the Observer's Frame of Reference is selected, we need to establish a convention to recognize travel along the axes. This leads to a key question: what is a convention?

A *convention* establishes an orientation for describing actions observed relative to the Observer's Frame of Reference with the *eyepoint* located at the origin. Coordinate systems serve as a static Observer's Frame of Reference for locating an object's position at a specific instant in time relative to the frame's origin and axes.

People often equate an Observer's Frame of Reference as a Coordinate Reference System. In fact, the Observer's Frame of Reference provides the multi-axis framework for assigning direction of travel and units of measure that form the Coordinate Reference System. The *right* side of Figure 25.3 provides an illustration.

25.5.2.1 LH and RH Conventions

Principle 25.6

Observer's Frame of Reference Convention Principle

An Observer's Frame of Reference is characterized by a Left-Hand (LH) or Right-Hand (RH) orientation.

One of the first steps in establishing an Observer's Frame of Reference is deciding to use a LH or a RH Rule convention. Figure 25.1 provides illustrative examples. In both conventions, the thumb of the designated hand points upward, the index finger points forward, and the middle finger points *orthogonally* toward the other hand.

Johanningmeier (2013) notes that most Computer Numerical Control (CNC) machinery employs an RH coordinate system—+X Vertical. However, he cautions that some CNC machines may employ LH coordinate systems with X Vertical. Consider the implications of LH and RH Coordinate Systems on the following example:

Example 25.1

A manufacturer employs a Computer-Aided Design (CAD) to create a 3-D model of a mechanical housing to be machined from a block of aircraft grade aluminum. Once the design is reviewed and approved, model data will be sent to a Numeric Control (NC) manufacturing cell for machining. When the 3-D model dimensional data are transferred, the manufacturing cell's coordinate system becomes the frame of reference for positioning the block of aluminum and machining it to the dimensions specified by the 3-D model. Your mission as a Mechanical SE lead is to collaborate with the machine shop team and establish a seamless strategy for transferring the data to ensure a successful part fabrication.

25.5.2.2 Frame of Reference Multi-Axis Configurations

Principle 25.7

Observer's Multi-Axis Configuration Principle

Pick one and only one multi-axis configuration for a given Observer's Frame of Reference *eyepoint*, preferably to serve the whole project.

Declaring a LH or RH Rule convention is a *necessary* but *insufficient* criterion for characterizing the Observer's Frame of Reference. The reality is the +X, +Y, and +Z axes can be rotated resulting in new structural reference configurations. For example, a RH Cartesian Frame of Reference has three possible orientations relative to an Observer's eyepoint. Figure 25.4 illustrates the fallacy of simply declaring the RH Rule as *the* frame of reference without regard to the orientation of the principal axes. Observe that the key is establishing the Local vertical axis orientation such as +X Axis Vertical, +Y Axis Vertical, or +Z Axis Vertical.

A Word of Caution 25.1

Exercise caution in applying the term "Vertical" as if it is fixed in place. Vertical in this context refers to an outward projection from an Observer's Frame of Reference origin into free space opposite of gravitational forces. As we shall see in our discussion of the Human Anatomical Coordinate Reference System, "Vertical" is a relative term depending on whether the human is

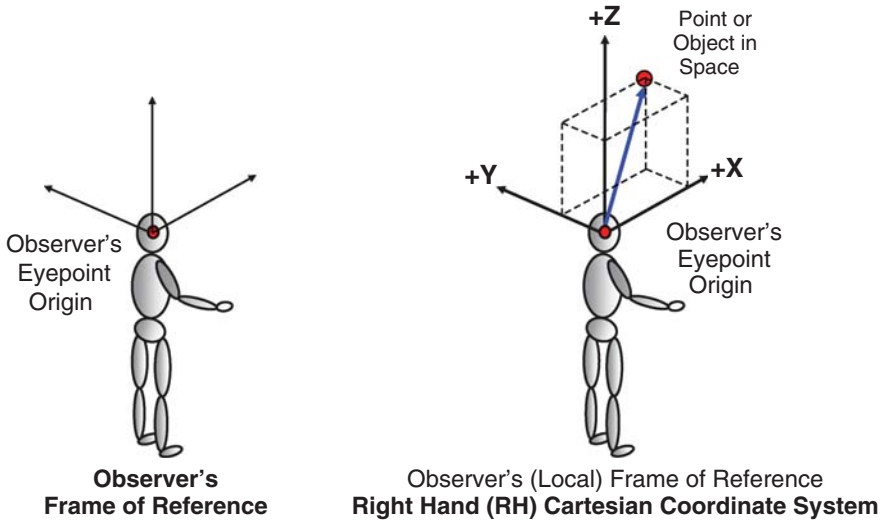


Figure 25.3 Illustrative Example of an Observer's Frame of Reference and Coordinate System

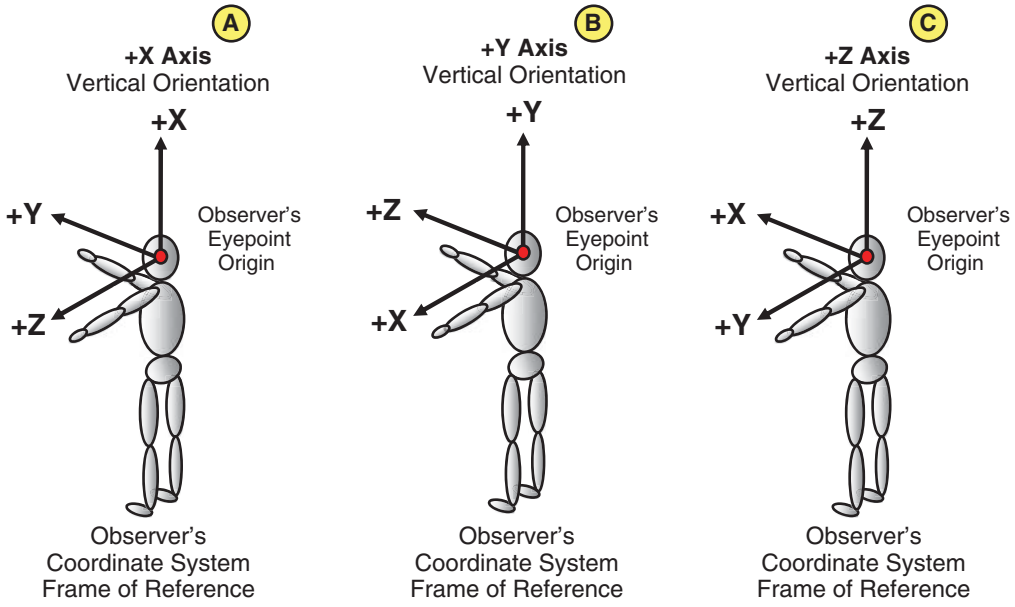


Figure 25.4 Three Types of Principal Axes Orientations of an Observer's Right-Hand (RH) Coordinate System

standing or *sitting*, *prone* on their stomach, or lying on their left or right side or back. The Observer's Frame of Reference, which is fixed relative to a body, rotates with body position, which makes "Vertical" relative to the situation. Figure 25.5 illustrates the point.

To illustrate application of these three configurations to the real world, consider the following example:

Example 25.2

Application Examples of the Three RH Rule Configurations

Using Figure 25.4 as a reference, ground, sea, air, and space-based vehicle applications commonly employ the +X-Axis (Panel C) to represent the *direction of travel*. However, free bodies in space such as aircraft and satellites may employ either of two

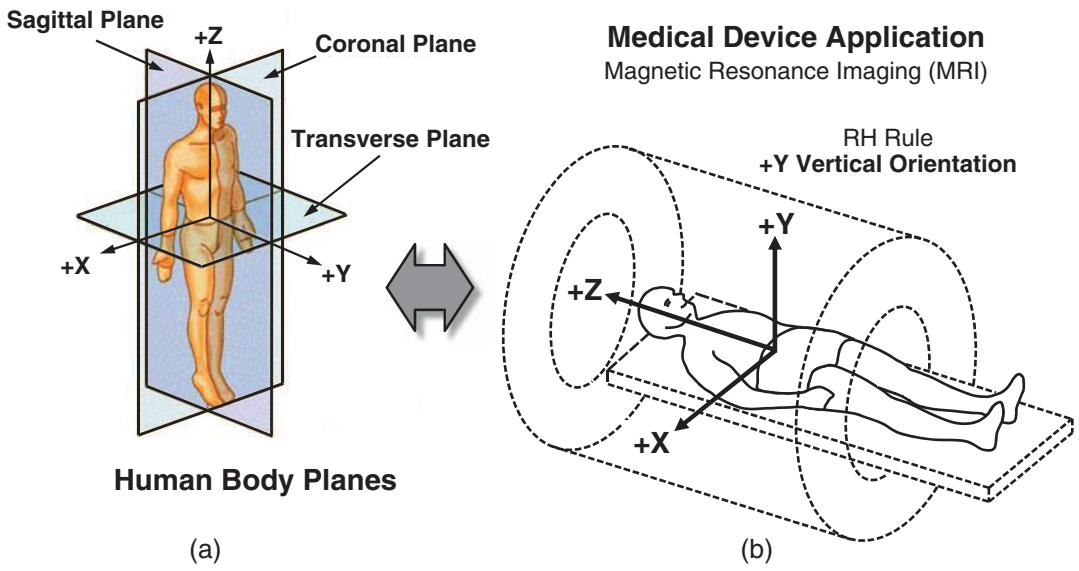


Figure 25.5 (a) Anatomical Coordinate Reference System Depicting the Intersecting Coronal, Sagittal, and Transverse Planes (Source: Wikimedia Commons, 2013) and (b) Magnetic Resonance Imaging (MRI) Application

Body Frame orientations: (1) +Z-Axis Up referred to as the East-North-Up (ENU) or (2) +Z-Axis Down orientation referred to as North-East-Down (NED) discussed later in this chapter (Figure 25.11).

25.5.3 Cartesian Coordinate Systems and Applications

One of the most basic coordinate reference systems used in Engineering and Physics is expressed as a RH or LH 2-D (X, Y) or 3-D (X, Y, Z) Cartesian Coordinate System. Both consist of an origin located at an Observer's *eyepoint* or a designated point internal or external to a SYSTEM.

To illustrate how Cartesian Coordinate Systems are applied to the real world, let's use some real world examples.

25.5.3.1 Medical Device Example and Applications of Coordinate Systems We often think of Cartesian Coordinate Systems as applying to mechanical applications. However, humans are no exception, especially in medical device technologies.

The medical industry, for example, employs an anatomical coordinate reference system using Human Body Planes as shown on the *left* side of Figure 25.5. Here, we see three intersecting, anatomical planes an origin positioned in the midpoint of a body.

Now, consider how the Human Anatomical Coordinate Reference System might be applied to a real-world situation. The *right* side of Figure 25.5 provides an example illustration of a human undergoing a Magnetic Resonance Imaging (MRI) procedure.

As a follow-up to the earlier cautionary note in the preceding section concerning the three possible multi-axis configurations, a word of caution!

A Word of Caution 25.2

As noted previously, when we refer to any one of the +X, +Y, +Z Axes as being vertical, exercise *caution*. "Vertical" is situational dependent and relates to SYSTEMS / PRODUCTS that are *stationary* and *fixed* in place. Free body systems may be different. Although the coordinate frame of reference is *identical*, what may be "vertical" in one situation *may not* be vertical in other situations. Such is the case with human body planes (Figure 25.5).

Consider the following example:

The Situational Dependence of "Vertical" in Fixed and Free Body Systems

Example 25.3 Note the *left* side of Figure 25.5 depicting Human Body Planes—Coronal, Sagittal, and Transverse. In this illustration, a human in a *standing or sitting position*, the +Z-Axis is "vertical."

Now, observe the *right* side of Figure 25.5 of a human in a *reclining position* during a MRI procedure. Note that in this view, the free body +Y-Axis has been rotated in the Sagittal Plane toward the +Z-Axis. In this view, +Y is now "vertical." As a result, MRI devices, in general, view the human subject as having the +Y-Axis vertical.

Metaphorically, establishment of a coordinate system is equivalent to laying the first cornerstone for a building. That cornerstone serves as the frame of reference for developing your SYSTEM/PRODUCT. This requires a leadership role by the LSE to establish not only the coordinate system but also the conventions to the employed concerning the axes. To illustrate this point, consider following Mini-Case Study 25.1 for an MRI device.

Mini-Case Study 25.1

MRI Medical Device Coordinate Systems and Conventions

MRI devices such as the one illustrated in Figure 25.5 are referred to as cylindrical superconducting systems. The origin of the MRI coordinate system is defined at *imaging isocenter*, the point at which all three gradient fields—+X, +Y, and +Z—are zero. The underlying assumption regardless of manufacturer is that B0 is coincidental with the human body's +Z-axis shown on the left side of the figure. Any exceptions (e.g., reverse ramped systems) will be made clear. Beyond this point, commonality across MR System Developers ends.

+Z-Axis Convention

Some System Developers choose a +Z-axis (B0) convention to point:

- *Into* the "bore" from the patient entry end such as shown in Figure 25.5.
- *Out of* the bore—toward the patient's feet.

+Y-Axis Convention

Some System Developers choose a +Y-axis to point:

- *Upward* as in Figure 25.5. Following the RH Rule, this means that the +X-axis points to the *left* as one looks into the bore from the patient entry end.
- *Downward* resulting in the +X-axis pointing to the *right* as one looks into the bore from the patient entry end.

B0 Field Polarity

Note that alignment of the Z-axis with B0 does not mean the B0 field always *points toward* +Z. Some MRI magnets can be ramped either *forward* or *reverse*, which changes the polarity of B0, relative to +Z. However, the direction of +Z remains unchanged; *it is always into the bore from the patient end*.

Source: Eash (2013).

25.5.3.2 Mechanical Engineering Example: Space Shuttle Coordinate System

One of the challenges in the Engineering of systems is spatially establishing the relative position and orientation of physically integrated components within the SYSTEM/PRODUCT. The objective is to ensure they:

- Are compatible and interoperate in terms of *form, fit*, and *function.*
- Do not interfere with each other unintentionally.
- Do not have an *adverse*, *negative* impact on the performance of the system, its operators, and its facilities or mission objectives—*Law of Unintended System Consequences* (Principle 3.5).

This challenge requires positioning integrated components in a virtual frame of reference or structure and attaching them at Integration Points (IPs) (Chapter 28) or nodes to ensure *interoperability*. Then, identifying additional attachment points such as *lift points* that enable external systems to lift or move the integrated system. This requires establishing a dimensional coordinate system.

The last NASA Space Shuttle mission was completed in 2011 after a very remarkable history despite two major accidents. The architectural configuration of the Space Shuttle required bringing four major systems together into a "stacked" configuration. To ensure that the integration of the "stack" would proceed according to plan, multi-discipline SEs System Analysts established the dimensional coordinate reference system[1] shown in Figure 25.6.

The integrated "stack" consists of the Orbiter Vehicle (OV), External Tank (ET), and two Solid Rocket Boosters (SRBs), each with their own respective frame of reference coordinate systems. These systems are then referenced relative to an integrated vehicle coordinate system with the origin designated as X_, Y_, Z_ in which the "_" represents the subscripts – T, B, O, and S - listed on the left side of Figure 25.6. This figure illustrates several key decision areas for SEs regarding Dimensional Coordinate Reference Systems.

- Aircraft designers establish *virtual* origins along the longitudinal X-Axis in front of the nose of aircraft as shown in Figure 25.6. One rationale, for example, includes accommodating free space *forward* of the nose for additional components that may be added later and still have *positive* "stations" in dimensional space. Examples include aircraft with bulbous nose modifications for radar, sensors, and additions. The AerospaceWeb.org (2013) notes that

[1]For a detailed text description of the Space Shuttle Coordinate System, refer to: http://science.ksc.nasa.gov/shuttle/technology/sts-newsref/sts_coord.html#sts_coord

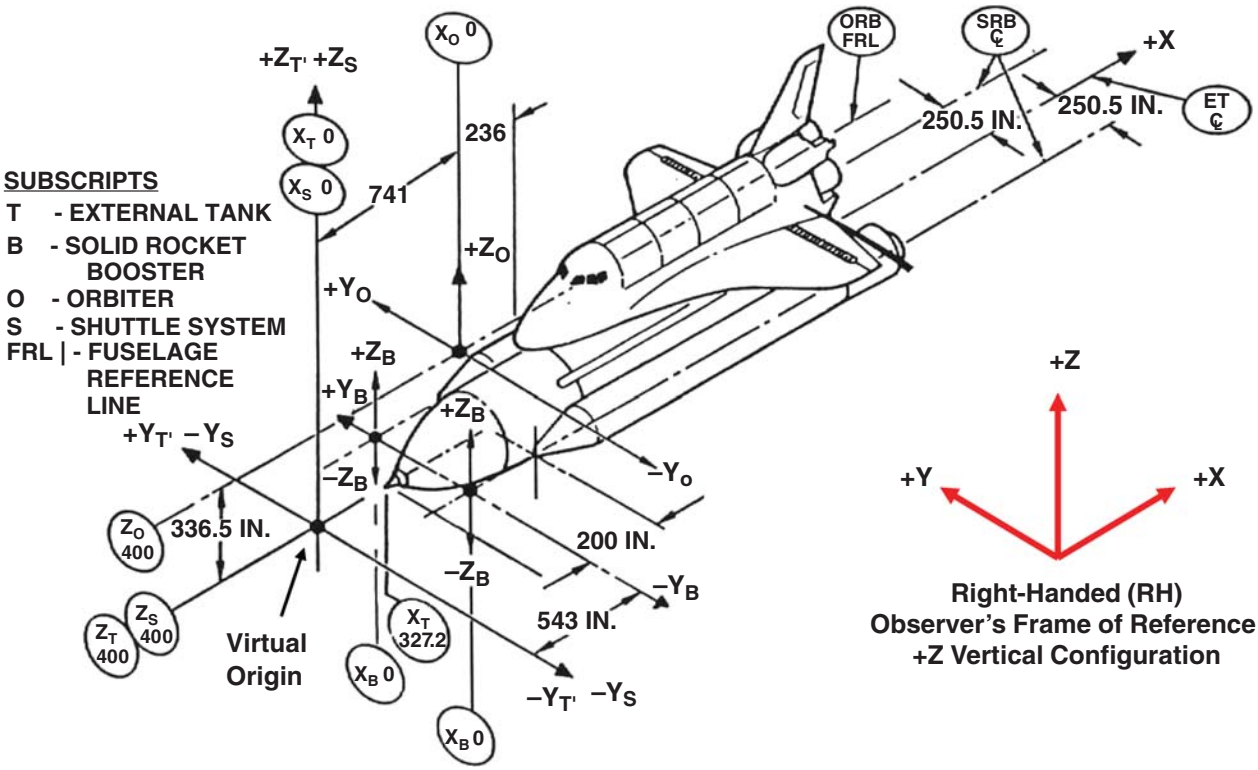


Figure 25.6 Space Shuttle Dimensional Coordinate Reference System. Source: Rogers Commission (1986), Figure 1: Challenger Report http://history.nasa.gov/rogersrep/v1p69.htm

the nose of an aircraft tends to grow while the wings and Aft section generally remain at the same station locations.

- The X-axis, which is referred to as the Fuselage Line (FS) in aircraft, extends through the nose and exits the Aft of section of aircraft (AerospaceWeb.org, 2013). Any point along this axis is considered *positive* by convention. Systems such as aircraft establish dimensional benchmarks or "stations" along the FS as a means of referencing component locations. For example, FS 100 to indicate a specific position along the FS relative to the origin. Using a longitudinal X-axis convention extending from the origin in front of the nose through the Aft section ensures that all "stations" along the FS are positive.
- The +Z-axis above the origin is referred to as the Water Line (WL) in aircraft (AerospaceWeb.org, 2013).
- The Y-axis, which extends out the right wing, is referred to as the Butt Line (BL) in aircraft. The name has limited use. Since aircraft wings have pylons and hard points for mounting sensors and weapons, the term *weapon stations* is typically used (AerospaceWeb.org, 2013).

Dimensional—Not Navigational—Coordinate Reference System

Author's Note 25.2 Please note that the Space Shuttle Coordinate Reference System discussion focuses on the physical design and integration of the "stack." This is different from a navigational coordinate reference system discussed later in this section.

25.5.3.3 Other Examples of Real-World Conventions

Engineering conventions occur in a number of forms. Consider the following examples:

Integrated Circuit (IC) Pin Layout Convention

Example 25.4 When viewing some IC devices with the notch at the top and pins on the left and right sides, Pin 1 is located in the upper left corner (Figure 16.2); the remaining pins are numbered *counterclockwise* around the perimeter of the device with the last ID assigned to the pin in the upper right corner. If a notch is not present, Pin 1 is usually designated by a small dot or impression imprinted on the device or impressed into the body of the chip next to the Pin 1 location.

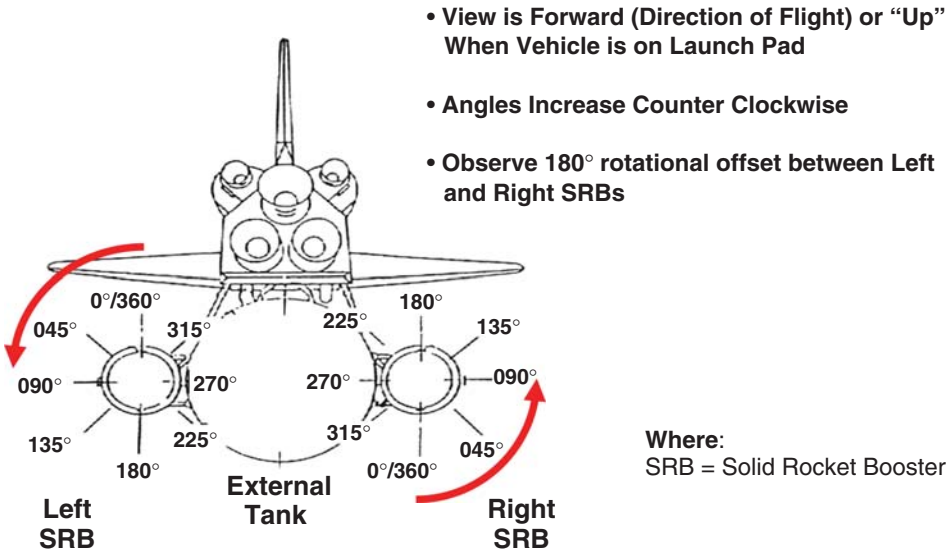


Figure 25.7 Angular Reference System for SRBs/Motors. Source: Rogers Commission (1986), *Report of the Presidential Commission on the Space Shuttle Challenger Accident*, Figure 26

Example 25.5 **Ingress (Entry)/Egress (Exit) Door Convention**

Office and residential doors are designated as LH or RH as viewed by an observer. When an observer approaches a door:

- **RH Door Convention**—When an Observer faces a door, if the handle or knob is positioned on the RIGHT side of the door and opens toward the Observer's LEFT side, it is by convention a RH Door.
- **LH Door Convention**—When an Observer faces a door, if the handle or knob is positioned on the LEFT side of the door and opens toward the Observer's RIGHT side, it is by convention a LH Door.

The RH and RH conventions are only part of the door considerations. Another key consideration includes which side of the wall constrains the door's movement in terms of "in-swing" into a room toward the observer or "out-swing" into the adjacent room away from the Observer.

The bottom line is: SEs, System Analysts, building Architects, and designers must analyze their systems to identify areas that may require conventions for identification and interfacing components to *avoid confusion* and *safety issues*. Then, establish, document, and communicate the details of those conventions to development teams.

25.5.4 Angular Displacement Reference Systems

Some systems also employ common components in LH and RH Observer's Frame of Reference conventions that require the need for a convention. Referring to Figure 25.6, *how did NASA uniquely identify the orientation of the Left and Right Solid Rocket Boosters (SRBs) relative to the ET*? Figure 25.7 provides the Solution.

In this figure, the Observer's *eyepoint* serves as the Frame of Reference origin and is positioned behind the Aft section of the Orbiter looking *forward* along the longitudinal X-axis in the vehicle's *forward* direction of travel. Relative to the observer's frame of reference, the SRB on the left is designated as LEFT and the other as RIGHT.

Now observe that the Left and Right SRBs each employ mirror image *angular displacement systems*. Note the location of the respective 0°/360° reference marks when the SRBs are attached and integrated to opposite sides of the ET. Configuration orientation diagrams such as this are developed by SEs in collaboration with Subject Matter Experts (SMEs) and are critical to design decision-making, models and simulations, and Engineering drawings.

25.5.5 Cylindrical and Polar Coordinate Systems

In addition to the RH Cartesian Coordinate System, manufacturing Computer Numerical Control (CNC), and medical devices employ other coordinate systems such as the Cylindrical and Polar Coordinate Systems. Groover (2013 p. 988) illustrates examples of various types of coordinate systems in Figure 25.8 for robotics and CNC manufacturing applications.

Robotics helped revolutionize manufacturing beginning in the 1980's with their ability to repeatedly produce predictable and consistent work. As the technology matured

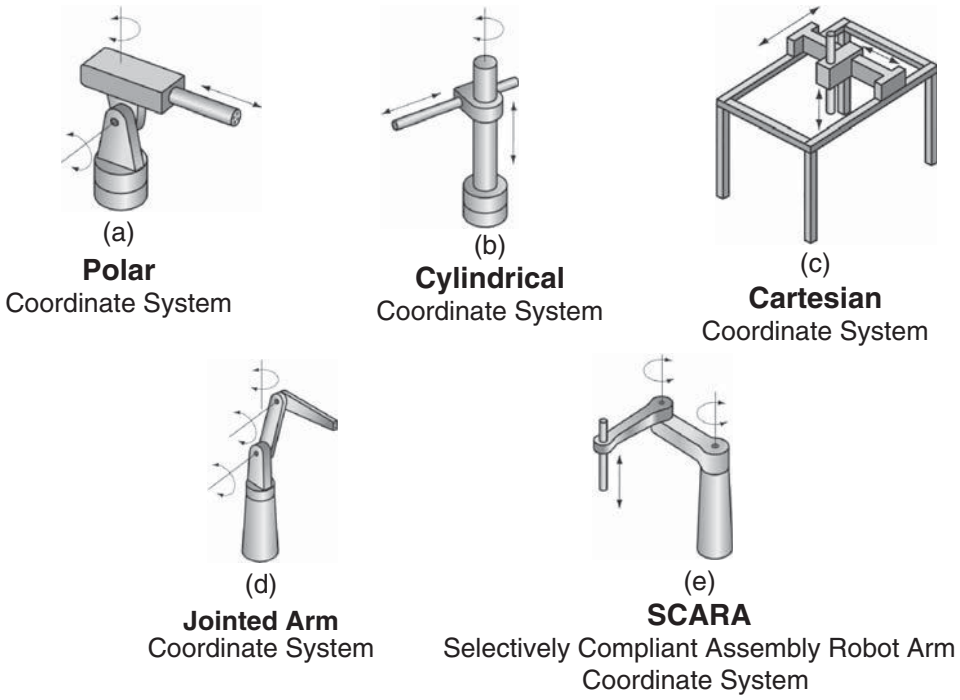


Figure 25.8 Polar, Cylindrical, and Cartesian Coordinate Systems; Jointed Arm and Selectively Compliant Assembly Robot Arm (SCARA) - Robotic and CNC Applications (Groover, 2013 p. 988). Used by permission

over the years, the medical industry began exploiting its capabilities. Although Figure 25.8 may appear overly simplistic, they provide the foundation for more complex real-world examples, especially in medical robotics. A key thrust of robotic surgeries is *minimally* invasive intrusions to the body. Figure 25.9 provides an example illustration of a surgical robot used in cardiovascular, urological, gynecological, colorectal, head and neck, and thoracic surgery. A separate console enables the surgeon to remotely Monitor, Command, and Control (MC2) the device.

25.5.6 Spherical Coordinate Systems

Systems such as satellites, aircraft, and ships, by virtue of their respective Earth, planetary, and galaxy missions are dependent on Spherical Coordinate Systems for pinpointing their spatial locations relative to a frame of reference. The Earth, for example, is characterized by a World Coordinate System (WCS) (World Geospatial-Intelligence Agency, 2005). The same is true for space exploration by organizations such as NASA in planning interplanetary travel. Consider the following example:

Author's Note 25.3

NASA Lunar Coordinate System

Although the US conducted manned space missions to the Moon, plans to return to the Moon prompted NASA to formulate a Lunar Coordinate System for use by the Lunar Reconnaissance Orbiter (LRO). Between April and July 2006, NASA conducted a series of meetings and biweekly teleconferences to establish a community consensus for the LCS. (NASA, 2008, p. 4)

Systems, such as land surveying, aircraft, and military troops, track their geospatial positions based on displacement of the origin of the Mission System relative to the Earth's surface. Two commonly used WCSs in Systems Engineering are the Earth-Centered, Earth-Fixed (ECEF) and Earth-Centered Inertial (ECI) coordinate systems. Let's explore a brief synopsis of each of these.

25.5.6.1 ECEF Orthogonal Coordinate System The ECEF orthogonal coordinate system, which is sometimes referred to as Earth Centered Rotating (ECR), is a Cartesian Coordinate system that is permanently fixed to and rotates with the Earth. The World Geodetic System 1984 (WGS84), for example, is an ECEF "terrestrial reference system and geodetic datum" (WGS84, p. 1). Figure 25.10 provides a graphical example representing a tangential point on the surface of the Earth.

The ECEF's:

- Origin is located at the Earth's Center of Mass (CoM).

Figure 25.9 Medical Robotics Employing Cylindrical and Polar Coordinate Reference Systems. (a) Surgeon's Console (Source: Intuitive Surgical, 2014) and (b) Patient-Side Cart Robotic Surgical Device (Source: Intuitive Surgical, 2014 Used by permission)

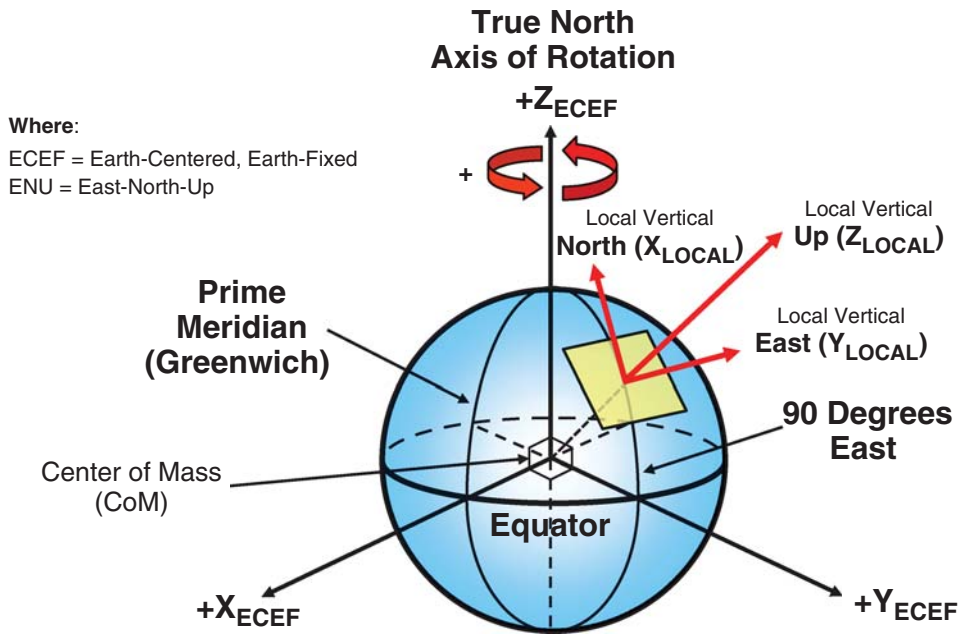


Figure 25.10 East-North-Up (ENU) Body Frame of Reference System Applied to an Earth-Centered Earth-Fixed (ECEF) Coordinate System

- X–Y axes plane coincide with the Earth's Equatorial Plane.
- X-axis extends from the Earth's CoM (origin) through the Equatorial Plane and exits at the Prime Meridian.
- Y-axis extends from the Earth's CoM (origin) through the Equatorial Plane and exits 90° East of the Prime Meridian.
- Z-axis extends from the Earth's CoM (origin) through the North Pole to the North Star.

25.5.6.2 ECI Coordinate System The ECI is also a Cartesian Coordinate System. Whereas the ECEF Coordinate System is *fixed* to the Earth's rotation, the ECI is *fixed* relative to the Celestial Sphere with the Earth rotating about the ECI frame of reference. The ECI's:

- Origin is located at the Earth's CoM (common with the ECEF).
- X-Y axes plane coincides with the Earth's Equatorial Plane (common with the ECEF).
- X-Axis has a directional heading to a position on the Celestial Sphere.
- Y-Axis in general, the Earth's equatorial plane rotates about the ECI X and Y Axes.

The ECI is often used for satellite applications to establish the satellite's position in space and is relative to the fixed ECI frame of reference. Computation of a satellite's Equations of Motion (EOM) is generally easier when using a fixed frame of reference such as the Earth's ECI versus its ECEF, which has a rotational frame of reference relative to the satellite.

The WCS enables us to reference a specific point on the surface of the Earth. But, *how do systems such as aircraft and spacecraft, which are free bodies in space, establish their geophysical location relative to the Earth, which is also in motion? How do we express their heading and rotational velocities and accelerations relative to the Earth's frame of reference*? Let's explore these questions further.

25.5.7 Free Body Frame of Reference Conventions

The orientation or perspective of the three-axis coordinate reference system depends on the Observer's *perspective* and challenges in dealing with the physical, spatial, and mathematical implications and consequences. This brings up two key questions:

1. *How do we express free body movements relative to the Observer's Frame of Reference*?
2. *How are integrated* EQUIPMENT *Element and physical component displacements and relationships characterized in terms of a three-axis frame of reference*?

The answer to the first question resides in establishing a coordinate reference system. The second question requires establishing a *convention* for application to the coordinate system.

The Earth is characterized by the ECEF or ECI Coordinate Systems that have a +Z-Axis that points upward toward True North. What we refer to as Vertical or Upward varies by the Observer's geophysical position on the Earth's surface. Vertical or Upward is a local context and can be represented by a local frame of reference with $\mathbf{Z}_{\mathbf{Local}}$ as its vertical axis. However, $\mathbf{Z}_{\mathbf{Local}}$ does equate to True North.

Any location on Earth's surface can be characterized in terms of a local frame of reference as illustrated in Figure 25.10. We create a Local RH (X, Y, Z) Cartesian Coordinate System consisting of:

- A virtual plane that is tangential to the Earth's surface at a specific geophysical location.
- A "Local vertical" *z-axis* orthogonal to the virtual plane that extends from the Earth's CoM through the Earth's surface location into free space.
- A Local East pointing *x-axis.*
- A Local North pointing *y-axis.*

Theoretically, this works fine as long as the Observer is on the surface of the Earth. However, from an Engineering perspective, air and space-based vehicles in flight are confronted with *gravitational effects* on their motion that require consideration of an alternative system with the Local vertical *z*-axis pointing *downward*. To solve these challenges, we introduce two coordinate system orientations. These are.

- Local East-North-Up (ENU) Orientation.
- Local North-East-Down (NED) Orientation.

Let's address each one separately.

25.5.7.1 Local North-East-Down (NED) Body Frame of Reference Orientation *Sea-based* applications such as surface and underwater vehicles, *air-based* applications such as aircraft and rotorcraft, and *spacecraft* applications such as satellites typically employ a NED Body Frame of Reference orientation as shown on the *right* side of Figure 25.11. The NED convention is *fixed* to a vehicle's frame as follows:

- The NED origin is located at the vehicle's Center of Gravity (CG) or Center of Mass (COM).
- The positive, longitudinal $\mathbf{X}_{\mathbf{LOCAL}}$ axis points from the vehicle's CG or COM in the direction of travel.
- The positive $\mathbf{Y}_{\mathbf{LOCAL}}$ axis points from the vehicle's CG out the right side of the vehicle.
- The positive $\mathbf{Z}_{\mathbf{LOCAL}}$ axis points *downward* from the vehicle's CG toward the Earth's CoM.

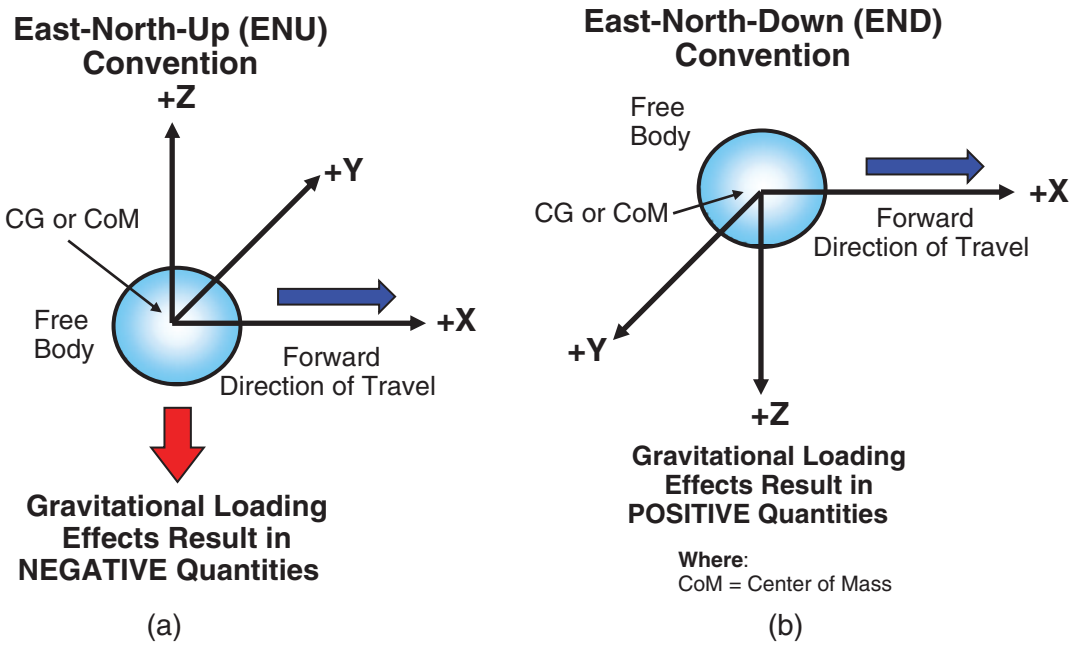


Figure 25.11 (a) East-North-Up (ENU) and (b) North-East-Down (NED) Body Frame of Reference Conventions

Koks (2006, p. 4) observes that since the Earth is "modeled as an oblate spheroid," the only time the $\mathbf{Z_{LOCAL}}$ axis line intersects the Earth's CoM occurs at the poles and the Equator.

A Word of Caution 25.3

CG versus CoM Location

Please note that CG or CoM are two separate concepts. CoM, which is fixed for both rigid and free bodes, does not change. CG, however, is subject to change due to the variances in gravitational forces on a satellite, for example. For fixed, rigid bodies on Earth, CG and COM are coincidental.

25.5.7.2 East-North-Up (ENU) Body Frame of Reference Orientation Land navigation on the Earth's surface typically employs an ENU Body Frame of Reference orientation as shown on the *left* side of Figure 25.11. The ENU convention is characterized as follows:

- The origin can be at any location on the Earth's surface.
- Positive X_{LOCAL} axis points *eastward* toward the Earth's geodetic East.
- Positive Y_{LOCAL} axis points northward toward the Earth's geodetic North.
- The X_{LOCAL}–Y_{LOCAL} plane is tangential to the Earth's surface at the ENU origin.
- Positive Z_{LOCAL} axis referred to as the *local vertical* points *upward* from the Earth's CoM through the location on the Earth's surface.

25.5.8 Navigational Coordinate Reference Systems and Conventions

The preceding discussions illustrate the basic concept of a standard RH Cartesian Coordinate Reference Systems as illustrated at the *left* side of Figure 25.11. For many SYSTEM or PRODUCT applications, this convention is acceptable. However, from an Engineering Mechanics perspective, the establishment of the *positive*, ENU orientation for a free body with the $\mathbf{Z_{LOCAL}}$-axis Vertical—pointing *upward*—means that gravitational loading effects result in *negative* $\mathbf{Z_{LOCAL}}$-axis components.

Now consider the added complexity in which the Space Shuttle maneuvers after entering orbit as illustrated in Figure 7.11 to fly *upside down* and *backward* during most of its mission to:

- Minimize the thermal and radiation effects of the sun by turning the vehicle's underside toward the Sun thereby allowing the thermal tiles to serve as a solar heat shield.
- Minimize the impact areas of orbital debris and meteorites that may cross its path that could damage thermal tiles on the vehicle's nose or leading edges of the wings that are crucial for reentry into the Earth's atmosphere and landing.

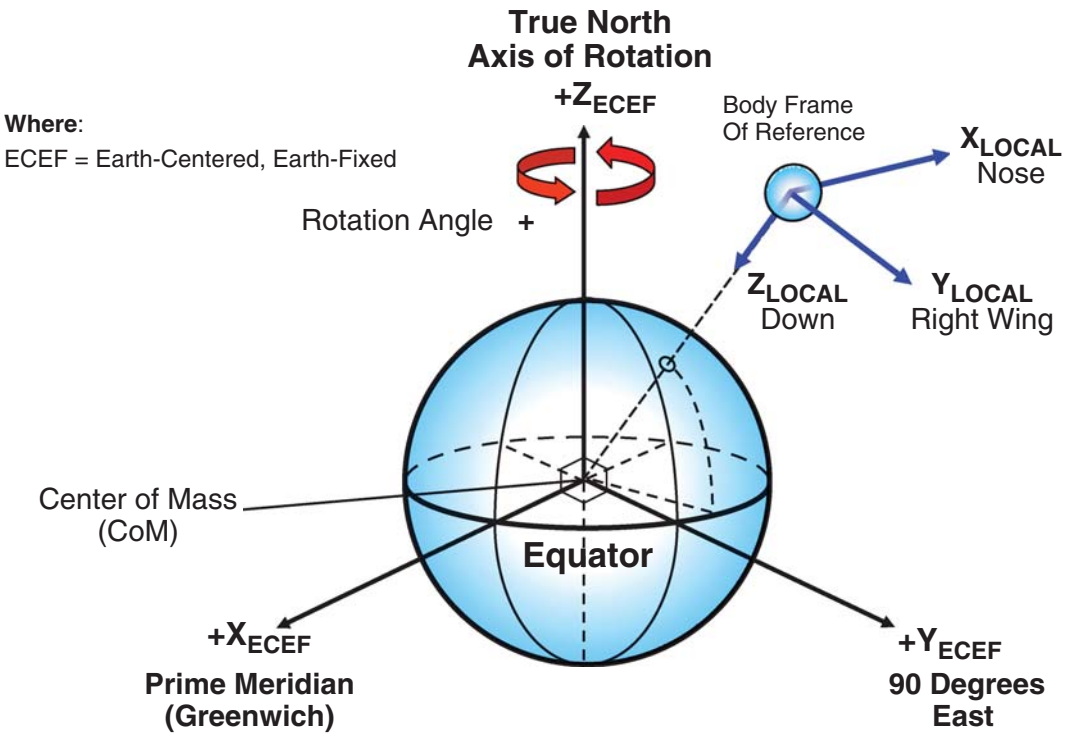


Figure 25.12 Free Body in Space Frame of Reference in Space Relative to the ECEF Coordinate System

- Shield Shuttle Cargo Bay equipment from impacts that could damage dangerous or fragile instruments
- Minimize the energy required to deploy payloads, where applicable.
- Slow the vehicle for reentry by firing small nozzles, which are located on the rear of the vehicle, in the direction of forward travel.

Compare and contrast the END reference system in Figure 25.11 with the Dimensional Reference System in Figure 25.6. This illustrates that a given system design may employ similar frames of reference but different *orientations* and *conventions*, each intended to support specific Stakeholder needs. Figure 25.11 facilitates Engineering Mechanics computations for system modeling; Figure 25.6 facilitates *dimensional* coordinate reference systems for product and manufacturing engineers and designers.

Heading 25.1 At this stage of our discussion, we have introduced the fundamental Cartesian (X, Y, Z), Polar, and Spherical Coordinate Reference Systems, illustrated applications to the Earth via the ECEF and ECI models descriptions; and described the ENU and NED Body Frame orientations (Figure 25.11). These Observer frames of reference enable us to establish the relationships among physical components (Figure 25.6), geophysical locations (Figures 25.10), and geospatial positions (Figure 25.12) relative to the Earth's surface.

Whereas dimensional reference systems are typically *stationary*, free body systems such as ships, aircraft, and spacecraft are *dynamic*. During their travel about the Earth from location to another, their operations require *dynamic* changes in their navigational headings and altitude. We need a means to characterize the trajectory of a free body in space in terms of its *position* and the *dynamics* concerning its rate of change – heading, velocity and acceleration - components.

25.6 DEFINING A SYSTEM'S FREE BODY DYNAMICS

Principle 25.8

State Vector Principle

Every free body moving through space at a specific instant in time is represented by a *state vector* that characterizes the body's geophysical or geospatial location – *position vector* - and rate of change - *velocity vector* - components.

The preceding discussions established the need to define a coordinate system as an *Observer's* Frame of Reference to facilitate Engineering collaboration, characterize *free body* movements, orientation, calibration, and alignment. Systems

such as land vehicles, ships and underwater craft, aircraft, and spacecraft exhibit motion-based dynamics. This requires understanding and modeling of their position relative to a frame of reference origin such as the Earth, their rate of positional change such as velocity, acceleration, and angular rotation about their principal axes. SEs in collaboration with SMEs must establish conventions for characterizing this motion.

The dynamic characteristics of a free body in space are characterized by its *state vector*. A state vector is a physics-based, mathematical expression that describes the trajectory of a free body in space in terms of its *geophysical* or *geospatial* position and its Six Degrees of Freedom (6-DoF).

To properly characterize an object such as a free body in geophysical space, there are three key attributes that need to be defined:

- An Inertial Frame of Reference relative to the Earth that remains stable.
- **Position vector** - that extends from the frame of reference CoM to the geophysical location or geospatial position of the free body's CoM.
- **Velocity vector** – that represents the free body's 6-DoF: (1) *translational movements* – forward/backward, upward/downward, and left/right directions and (2) *rotational* – angular - movements about its principal axes.

25.6.1 Position Vector

Geophysical or *geospatial* position characterizes the location of a ground, sea, air, or space-based object relative to a frame of reference such as the Earth. For example, land surveyors, land vehicles, surface and underwater vehicles, aircraft and spacecraft, employ a GPS to establish their geophysical and geospatial location relative to the Earth's surface or other celestial body.

Complex systems often require dynamic characterizations of a free body relative to another body such as the Earth that are assumed to be fixed as illustrated in Figure 25.12. For this illustration, we establish: (1) an ECEF Coordinate System located at the Earth's CoM to serve as a frame of reference and (2) an NED Local Coordinate System for a free body in space relative to the Earth. The Position Vector would extend from the Earth's ECEF origin to the CoM of the free body on the Earth's surface – *geophysical location* for vehicles or ships and *geospatial position* for missiles, aircraft, or spacecraft.

Now that we have established the location of the free body relative to a frame of reference, to complete our characterization of its state vector we need to characterize its 6-DoF.

25.6.2 6-DoF Translational Movements

Given a geophysical or geospatial location, we need to characterize the *translational movements* of an object in terms of its *heading* relative to a frame of reference. For example, a RH (X, Y, Z) Cartesian Coordinate System located at the Observer's *eyepoint* would be applied as follows to depict *translational* movements:

- The X, Y, and Z-axes are orthogonal to each other.
- The X-axis represents an object's *forward* or *backward* movements relative to the Observer's *eyepoint* (origin).
- The Z-axis represents an object's *upward* or *downward* movements relative to the Observer's *eyepoint* (origin).
- The Y-axis represents an object's *left* or *right* movements relative to the Observer's *eyepoint* (origin).

Translational movements enable us to characterize an object's *directional heading* and rate of travel in terms of its velocity, acceleration, and deceleration. However, objects deviate from directional headings. The question is: *how does the Observer express motion about those multiple axes*? This brings us to our next topic, Rotational Movement Characteristics.

25.6.3 6-DoF Rotational Movements

Principle 25.9

Rotational Conventions Principle

Rotation conventions about the principal +X, +Y, and +Z axes of an Observer's Frame of Reference are established by the Right-Hand (RH) Grip Rule.

Rotational *movements* enable us to characterize an object's dynamic movements in terms of multi-axis rotation. This requires that we establish a uniform rotation convention that can be applied to all principal axes. The solution is the Right-Hand (RH) Grip rule illustrated earlier in Figure 25.2. When we apply the RH Grip rule to the three principal axes, Figure 25.13 emerges. Table 25.4 provides descriptions of Pitch, Yaw, and Roll conventions applied to a free body axis system.

Observe that we can now characterize rotational movements in terms of their angular rates of change. This allows us to establish definitions, *clockwise/counterclockwise* rotational conventions, and angular rates of change for the three principal axes: ROLL along the Longitudinal Axis, PITCH along the +Y Lateral Axis, and YAW along the Vertical Axis. Figure 25.14 illustrates application to an aircraft or spacecraft such as the Space Shuttle.

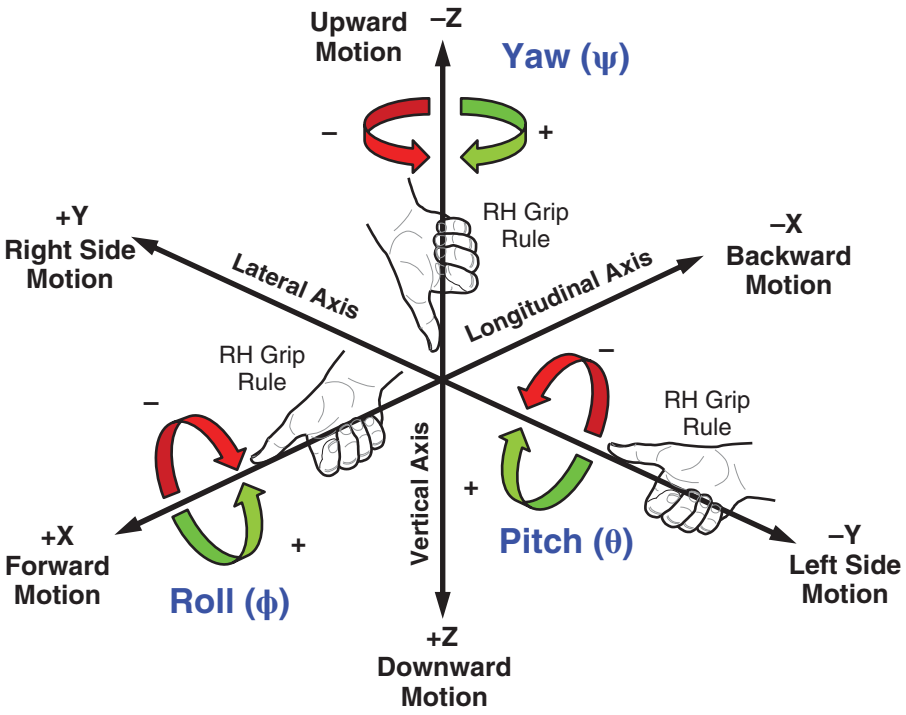


Figure 25.13 Illustration of a Free Body's Six Degrees of Freedom (6-DoF) Translational (Directional) Movements - Forward/Backward, Up/Down, Left/Right – and Rotational (Angular) Movements – Roll, Pitch, and Yaw (RPY)

TABLE 25.4 PITCH, YAW, and ROLL Conventions for a RH, Local NED Body Frame Coordinate Reference System

Parameter	Convention	Action Observed (North-East-Down (NED) Orientation
+X-axis (NED)	Definition	Body Frame *longitudinal* axis or fuselage centerline through the CG and extending in the direction of forward travel
+Y-axis (NED)	Definition	Body Frame *lateral* axis extending from the CG origin out the right-side wing
+Z-axis (NED)	Definition	Body Frame *vertical* axis extending from the CG origin *downward* toward the Earth's CoM
PITCH (θ)	PITCH (+)	RH Grip *clockwise* angular rotation about the *lateral* Body Frame +Y-axis
	PITCH (–)	RH Grip c*ounterclockwise* angular rotation about the *lateral* Body Frame +Y-axis
YAW (Ψ)	YAW (+)	RH Grip c*lockwise* angular rotation about the *vertical* Body Frame +Z-axis that points downward to the Earth's CoM
	YAW (–)	RH Grip c*ounterclockwise* angular rotation about the *vertical* Body Frame +Z-axis that points downward to the Earth's CoM
ROLL (Φ)	ROLL (+)	RH Grip c*lockwise* angular rotation about the *longitudinal* Body Frame +X-Axis
	ROLL (–)	RH Grip c*lockwise* angular rotation about the *longitudinal* Body Frame +X-Axis

25.6.4 Inertial Navigational Systems (INS)

The preceding sections defined various coordinate reference systems, their conventions, and spatial relationships as shown in Figures 25.12–25.15. Systems such as land, sea, air, and space-based vehicles need to be able to establish their location, attitude, headings, and velocities relative to a primary coordinate reference system such as the Earth or planetary body. *How do we accomplish this*?

To solve this problem, *free bodies* such as ships, aircraft, or spacecraft consist of an Inertial Navigation Unit (INU) that is calibrated to establish an initial condition for true North relative at a specific location. The objective is to maintain that setting with *minimal* drift and error throughout a mission to serve as a frame of reference for determining the vehicle system's attitude such as headings, velocity, and Roll-Pitch-Yaw (RPY) angles. Figure 25.15 provides an illustration.

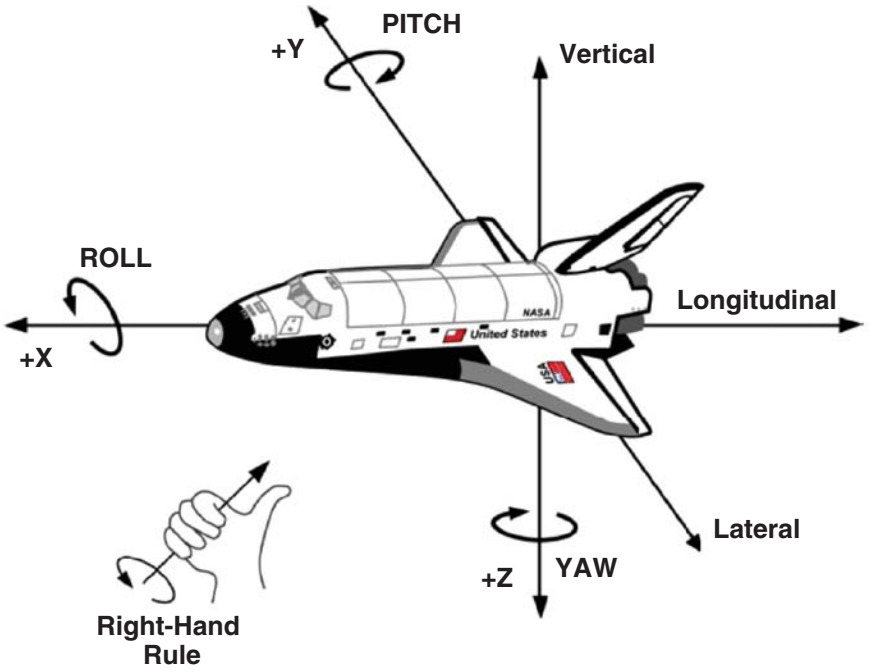


Figure 25.14 NED Application to the NASA STS Space Shuttle. Source: NASA (2010)

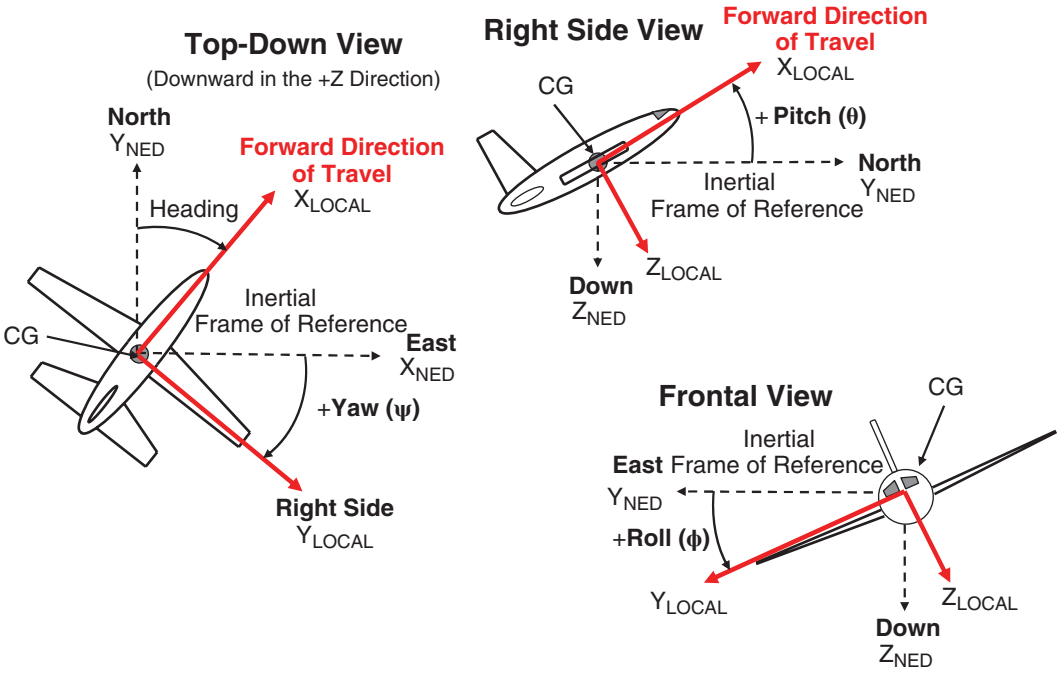


Figure 25.15 Example Illustrating Pitch, Yaw, and Roll Attitude of an Aircraft North-East-Down (NED) Body Frame of Reference Relative to its Local Inertial Frame of Reference

As the vehicle performs its mission and navigates as a free body, its Local NED Body Frame of Reference changes relative to its onboard Inertial Frame of Reference to provide information about its directional heading and attitude of flight. This information in combination with the GPS enables computation of its geophysical or geospatial location, altitude, heading, velocity, acceleration, and distance. The RPY angles are computed as *Euler Angles*. For a description of Euler angles, refer to CH Robotics, LLC (2013) and Wikipedia (2013b).

25.6.5 6-DoF Summary

In summary, an Observer's Frame of Reference may be fixed, stationary, or moving in terms of its 6DOF translational and rotational movements relative to another body's frame of reference. Collectively, a free body's *translational* and *rotational* movement characteristics are referred to as an object's 6DoF. Based on the 6DoF, we can apply Engineering statics, dynamics, and scientific principles to model the object's behavior, assess *what-if* questions and scenarios, and optimize its performance within its prescribed OPERATING ENVIRONMENT.

A System's, 6-DoF characteristics consist of:

- *Translational* movements
  - Forward or backward direction—distance and rates of change
  - Left or right direction—distance and rates of change
  - Upward or downward direction—distance and rates of change
- *Rotational* movements
  - Roll (Φ)— angle and angular rates of change about the Longitudinal Axis
  - Pitch (Θ)—angle and angular rates of change about the Lateral Axis
  - Yaw (Ψ)— angle and angular rates of change about the Vertical Axis

25.7 APPLYING ENGINEERING STANDARDS AND CONVENTIONS

As leads on a contract development effort, SEs must ensure that all standards and conventions used, applicable to a project, are well documented and explicitly communicated and well understood by all program personnel. So, *how are these documented?*

Standards and conventions are documented in work products such as specifications, plans, design documents, Interface Requirements Specifications (IRS), Interface Control Documents (ICD), and Interface Design Descriptions (IDDs). To understand some of the types of information to be communicated across these interfaces to ensure *compatibility* and *interoperability*, consider the following examples:

Example 25.6

- Weights and measures
- Coordinate system frame of reference and conventions
- Computational precision and accuracy
- Base number systems and conventions
- Units of conversion
- Calibration standards
- Engineering standards
- Data communication protocols
- Documentation guidelines, standards, and conventions

Finally, establish an official list of Project *Acronyms* and *Glossary of Key Terms*. Some refer to this as "languaging" the program. Although this simple task seems trivial, it will save many hours of personnel thrashing in chaos, each with different views of the terms. People need to share a common mindset of terminology.

25.8 ENGINEERING STANDARDS AND CONVENTIONS LESSONS LEARNED

Engineering standards and conventions involve a number of application and implementation lessons learned. Let's explore some of the more common lessons learned.

#1: Tailoring Organizational Standards
Organizational Standard Processes (OSPs) establish standards guidance to support a wide variety of system applications. Since every system is different, the OSP standards must include *normative* and *informative* tailoring instructions guidance for applying the standard to a program.

#2: Organizational Standards As Contract Requirements
If an Acquirer or User has requirements for Engineering standards and conventions, the formal Request for Proposal (RFP) solicitation or contract should explicitly specify these as requirements. Typically, the Acquirer has an obligation to provide authorized access to a copy for Offerors on-line or during normal business hours at a designated facility.

#3: Engineering Standards and Conventions Document
A common problem related to standards implementation is *failure to define* the standards to be used in developing system or entity interfaces, coordinate transformations, and software coding. Therefore, SEs

should establish, document, baseline, and release an *Engineering Standards and Conventions* document for application on each program and formally control changes to it.

#4: Assumptions about Conventions

One of the challenges confronting an SE is to define and document *assumptions* about SYSTEM / PRODUCT performance boundary *conditions* and *conventions* when specific aspects may be unknown. Inevitably, people assume everyone understands how to use the ABC convention (Principle 24.20).

A Word of Caution 25.4

Engineering Standards and Conventions

Always document the Standards of Units, Weights, and Measures, Coordinate Reference System conventions and configurations decision artifacts via project memorandums; *leave nothing to chance*! Every technical review should include a technical description of conventions such as a graphic, where applicable to ensure continuity, consistency, and completeness of SE design activities.

#5: Conflicts between Standards—Order of Precedence

Occasionally, conflicts occur within and between Engineering standards. Standards organizations work to ensure that conflicts are avoided. If, as a System Acquirer, you mandate that a system, product, or service meet specific standards, ensure specification requirements do not conflict and specify the order of precedence in the event of a conflict.

#6: Scope of Requirements

From an SE perspective, standards are often written for broader application to a variety of business domains. As an SE, your role is to collaborate with SMEs to ensure that the explicit provisions of the standard that are applicable to a contract or system development effort are referenced and identified by document number, title, date, version, and paragraph number. This *avoids* confusion and *unnecessary* verification expenses ferreting out *what is* and *is not* applicable.

#7: Application of Standards

When proposing or developing new systems, Acquirer SEs should ensure that all standards referenced in procurement packages or specifications are the most current, approved, and released version of the document. Likewise, System Developer SEs, via contract protocols, should request the Acquirer to *clarify* broad references such as "ANSI-STD-XXXX shall apply." Instead, specific requirements such as "ANSI-STD-XXXX-(version) para. X.X.X (Paragraph Title) shall apply."

#8: Coordinate System Transformations

Our discussions included a focus on designating a coordinate system for a free body such as an air vehicle. Aircraft, especially military aircraft, serves as a platform for other payload systems such as sensors and missile systems. Application of a RH Cartesian Coordinate System with a NED orientation convention that has the Z-Axis pointing *downward* does not necessarily mean interfacing components will also. Consider the following example:

Example 25.7

Various types of payloads such as missile, munitions, and sensors, are often used on a variety of aircraft platforms. Each aircraft platform that hosts these payloads may be developed for different Users; payloads may be developed by different System Developers If you were an aircraft manufacturer, do not assume the payload employs the same coordinate system as the aircraft. If not, you will either have to perform a coordinate transformation when exchanging data, which adds an element of complexity and risk, or pay to have the vendor transform the data to provide the required formatted outputs.

Where the coordinate systems of interfacing systems are different, *always* create a simple diagram that illustrates the coordinate systems and conventions employed; review, approve, baseline, and release the document; and communicate it across the project. Establish which interfacing system is required to perform the coordinate transformation in the respective specification.

#9: Standard Terminology

Successful system design and development requires that everyone on the project have a shared vision and mindset and communicate in a language that is consistent and mutually understood by everyone. This requires establishing *terminology*, *acronyms*, and definitions of *key terms* for universal application throughout all documentation.

Suggestions to establish standard terminologies are usually met with disparaging remarks from personnel about how terms are *intuitively obvious*. Then, when the project encounters major team-based work scope issues or interpretations caused by failures in application of terminology, everyone suddenly has one of those "why didn't we think of this sooner" responses. *How do you avoid this?*

The System Development Team (SDT such as a System Engineering and Integration Team (SEIT) must:

1. Establish ownership of the project's standard terms, definitions, and acronyms.

2. Communicate the list via online network drives or Web sites.
3. Communicate updates to everyone on the program.
4. Ensure compliance across all documentation including technical reviews, and audits.

25.9 CHAPTER SUMMARY

Our discussions of Engineering standards, coordinate system frames of reference, and conventions highlighted the need to establish project, corporate, national, and international standards for guiding the system development effort. We introduced the meanings of *normative* and *informative* clauses, and how they were to be implemented.

We also highlighted the need to establish interface standards and conventions such as English (BES) versus Metrics (SI) MKS systems and guidelines to support development work.

You may comment … w*e do not develop free body air vehicles. Why should we bother with considerations with coordinate systems*? The examples discussed previously are used to illustrate some of the complexities of getting coordinate-based systems correct. Similar concepts are employed for designing and using numerical control machines, aligning sensors and optical devices, land surveying, and so forth.

- Each project must establish and communicate a set of *conventions* for application to system configuration and interface applications, Engineering design processes and methods, and work products.
- Each project must conduct periodic assessments of Engineering standards compliance and review Engineering standards and conventions compliance as a key element of all technical reviews.

25.10 CHAPTER EXERCISES

25.10.1 Level 1: Chapter Knowledge Exercises

1. What is an Engineering standard?

2. What is an Engineering convention?

3. Why do we need Engineering standards?

4. What are the key subject matter categories of Engineering standards?

5. What are some types of Engineering standards?

6. What are some types of conventions?

7. Who establishes Engineering standards at the global, national, professional, and Enterprise levels?

8. What are Enterprise standards and conventions?

9. What are some examples of Enterprise standards and conventions?

10. How is the degree of compliance with Engineering standards verified?

11. What is the difference between an Observer's Frame of Reference and a Coordinate Reference System?

12. What are two conventions that apply to an Observer's Frame of Reference?

13. Why is an Observer's Frame of Reference convention is *necessary* but *insufficient* for defining a Coordinate Reference System.

14. What are the three possible multi-axis configurations for a Cartesian Coordinate Reference System?

15. How are Coordinate Reference Systems applied to A&D, medical, manufacturing, and business domains? Illustrate graphically.

16. What is an INS and how does it apply to a Body Frame of Reference Coordinate System?

17. What is the WCS and its two commonly used Earth models.

18. What is the Earth-Centered, Earth-Fixed (ECEF) model and how it is used?

19. What is the ECI model and how it is used?

20. What is the East-North-Up (ENU) and NED coordinate systems are, graphically explain their differences, and how they are used?

21. Why is application of the term "vertical" to some Coordinate Reference System configurations relative and how do you to deal with it?

22. What is a SYSTEM's 6-DoF?

23. What is the meaning, definition, and application of Euler Angles?

24. What is the meaning of a Free Body's Yaw, Pitch, and Roll, their rotational conventions, and applications?

25. Why it is important to establish a project's Standards of Units, Weights, Measures, and Coordinate Systems at the beginning of a project?

26. What is meant by a 6DOF's translational and rotational movements?

25.10.2 Level 2: Knowledge Application Exercises

Refer to www.wiley.com/go/systemengineeringanalysis2e

25.11 REFERENCES

AerospaceWeb.org (2013), *Aircraft Station Coordinate System*, (City): AerospaceWeb.org. Retrieved on 9/24/13 from http://www.aerospaceweb.org/question/design/q0289.shtml

CH Robotics, LLC (2013), *Euler Angles*, Payson, UT. Retrieved on 9/24/13 from http://www.chrobotics.com/library/understanding-euler-angles

DAU (2012), *Glossary: Defense Acquisition Acronyms and Terms*, 15th Edition. Ft. Belvoir, VA: Defense Acquisition University (DAU) Press. Retrieved on 3/27/13 from: http://www.dau.mil/publications/publicationsDocs/Glossary_15th_ed.pdf

DoD 5000.59-M (1998), *DoD Modeling and Simulation (M&S) Glossary*, Washington, DC: Department of Defense (DoD). Retrieved on 8/1/13 from http://www.dtic.mil/whs/directives/corres/pdf/500059m.pdf

Eash, Matthew (2013), *Correspondence - MRI Coordinate System Standards*, Milwaukee, WI: GE Healthcare MR Engineering.

Fleming, John Ambrose (1902). *Magnets and Electric Currents,* 2nd Edition. London: E.& F. N. Spon. pp. 173–174.

Groover, Mikell P. (2013), *Fundamentals of Modern Manufacturing*, 5th Edition, New York: John Wiley & Sons, Inc.

Intuitive Surgical (2014), *da Vinci Si System with Single-Site Instrumentation*, Sunnyvale, CA: Intuitive Surgical, Inc. Retrieved on 3/4/14 from http://www.intuitivesurgical.com/company/media/images/singlesite/SingleSite_PC_SC_Surgeon_head_in_console.jpg, http://www.intuitivesurgical.com/company/media/images/systems-si/000800_si_patient_cart_oblique.jpg

Johanningmeier, Bruce A. (2013), *Cartesian Coordinate System Right-Hand Rule*, (City): CNCexpo. Retrieved on 9/18/13 from http://cncexpo.com/Cartesian.aspx

Koks, Don (2006), *Using Rotations to Build Aerospace Coordinate Systems*, Department of Defence, Department of Science and Technology Organization (DSTO) Systems Science Laboratory, Retrieved on on 5/7/13 from http://www.dtic.mil/dtic/tr/fulltext/u2/a484864.pdf

MCO (1999), *Mars Climate Orbiter Failure Board Releases Report*, Washington, DC: NASA Headquaters. Retrieved on 9/24/13 from http://mars.jpl.nasa.gov/msp98/news/mco991110.html

MCO (2000), *Report on Project Management in NASA by the Mars Climate Orbiter Mishap Investigation Board*, Washington, DC: NASA Headquarters. Retrieved on 9/24/13) from http://mars.jpl.nasa.gov/msp98/misc/MCO_MIB_Report.pdf

NASA (1976), *The Standard Atmosphere Model*, Washington, DC: National Aeronautics and Space Administration (NASA). Retrieved on 5/6/13 from http://modelweb.gsfc.nasa.gov/atmos/us_standard.html

NASA (2008), *A Standardized Lunar Coordinate System for the Lunar Reconnaissance Orbiter*, LRO Project White Paper, Version 4, Greenbelt, MD: NASA Goddard Space Flight Center (GSFC). Retrieved on 9/17/13 from http://lunar.gsfc.nasa.gov/library/451-SCI-000958.pdf

NASA (2010), *Space Shuttle Roll Maneuver*, Washington, DC: NASA, Office of Procurement. Retrieved on 1/20/14 from http://www.nasa.gov/pdf/519342main_AP_ED_Phys_RollManeuver.pdf

NPG 5600.2B (1997 Elapsed) *Statement of Work (SOW): Guidance for Writing Work Statements*, NASA Procedures and Guidelines. Washington, DC: NASA, Office of Procurement. Accessed 9/23/13 http://www.hq.nasa.gov/office/procurement/newreq1.htm

Rogers Commission (1986), *Report of the PRESIDENTIAL COMMISSION on the Space Shuttle Challenger Accident*, Washington, DC. Retrieved on 9/23/13 from http://history.nasa.gov/rogersrep/genindex.htm

Taylor, Barry N. and Thompson, Ambler, Editors (2008a), *Guide for the Use of the International System of Units,* Special Publication 811, Washington, DC: National Institute of Standards and Technology (NIST), Retrieved on 3/16/13 from http://physics.nist.gov/cuu/pdf/sp811.pdf.

Taylor, Barry N. and Thompson, Ambler – Editors (2008b), *The International System of Units (SI),* Special Publication 330, Washington, DC: National Institute of Standards and Technology (NIST), Accessed 3/16/13 http://physics.nist.gov/Pubs/SP330/sp330.pdf

Wikipedia (2013a), *Cartesian Coordinate System*, San Francisco, CA: Wikimedia Foundation, Inc. Retrieved on 9/23/13 from: http://en.wikipedia.org/wiki/Cartesian_coordinate_system

Wikipedia (2013b), *Euler Angles*, San Francisco, CA: Wikimedia Foundation, Inc. Retrieved on 9/23/13 from: http://en.wikipedia.org/wiki/Euler_angles

Wikipedia (2013c), *Flemings Left Hand Rule for Motors*, San Francisco, CA: Wikimedia Foundation, Inc. Retrieved on 9/23/13 from: http://en.wikipedia.org/wiki/Fleming%27s_left-hand_rule_for_motors

Wikipedia (2013d), *Hans Christian Ørsted*, San Francisco, CA: Wikimedia Foundation, Inc. Retrieved on 9/23/13 from: http://en.wikipedia.org/wiki/Hans_Christian_Orsted

Wikimedia Commons (2013), *Human Body Planes*, US Government, San Francisco, CA: Wikimedia Foundation, Inc. Retrieved on 9/23/13 from: http://upload.wikimedia.org/wikipedia/commons/3/34/BodyPlanes.jpg

World Geospatial-Intelligence Agency (2005), *World Geodetic System 1984*, Version WGS 84 (G1674), Springfield, VA.

26

SYSTEM AND ENTITY ARCHITECTURE DEVELOPMENT

Living systems, by definition, are composed of interacting elements that enable the whole to accomplish more than the individual elements. Each element has its own unique identity, capabilities, and characteristics, integrated into a purposeful framework specifically arranged to accomplish the system's mission. The integrated, multi-level framework of elements represent the SYSTEM's architectural configuration or simply architecture.

Our discussions in this chapter establish the foundation for developing a System Architecture. Chapters 8 and 9 introduced *fundamental concepts* related to analyzing Systems and decomposing them into architectural structures. Chapter 26 builds on those concepts to formulate, select, and develop a SYSTEM or ENTITY's Architecture.

26.1 DEFINITIONS OF KEY TERMS

- **Architect** (System)—The person, team, or organization responsible and accountable for innovating and creating a system configuration that provides the best solution to User expectations and a set of requirements within technical, cost, schedule, technology, and support constraints.
- **Architecting**—The "process of conceiving, defining, expressing, documenting, communicating, certifying proper implementation of, maintaining and improving an architecture throughout a system's life cycle" (SEVOCAB, 2014, p. 17) (Source: ISO/IEC/IEEE 24765:2011. Copyright © 2012 ISO/IEC/IEEE. Used by permission.).
- **Architecture Description (AD)**—"A collection of products to document an architecture" (Copyright 2000, IEEE. Used with permission; IEEE Std. 1471-2000, 2000, p. 3).
- **Architecture**—A graphical model or representation that: (1) expresses the structural framework of a SYSTEM or ENTITY's components, their relationships, and operational, behavioral, and physical interactions *internally* and externally with its OPERATING ENVIRONMENT; (2) addresses the *views* and *viewpoints* of its Stakeholders; and (3) alleviates their *concerns*.
- **Architecture framework**—"Conventions, principles, and practices for the description of architectures established within a specific domain of application and/or community of stakeholders" (SEVOCAB, 2014, p. 18) (Source: ISO/IEC/IEEE 24765:2011. Copyright © 2012 ISO/IEC/IEEE. Used by permission.).
- **Architecture View**—A "work product expressing the architecture of a system from the perspective of specific system concerns" (SEVOCAB, 2014, p. 18) (Source: ISO/IEC/IEEE 24765:2011. Copyright © 2012 ISO/IEC/IEEE. Used by permission.).
- **Architecture Viewpoint**—A "work product establishing the conventions for the construction, interpretation, and use of architecture views to frame specific system concerns" (SEVOCAB, 2014, p. 18) (Source: ISO/IEC/IEEE 24765:2011. Copyright © 2012 ISO/IEC/IEEE. Used by permission.).
- **Design**—The process of: (1) analyzing SYSTEM or ENTITY specification requirements; (2) conceptualizing,

System Engineering Analysis, Design, and Development: Concepts, Principles, and Practices, Second Edition. Charles S. Wasson.

Companion Website: www.wiley.com/go/systemengineeringanalysis2e

formulating, and developing a physical solution; (3) selecting components from a set of viable candidates; (4) applying scientific and engineering principles to ensure compatibility and interoperability of component interfaces; and (5) documenting the design requirements—drawings, schematics, models, etc.—that are *necessary* and *sufficient* to Fabricate, Assemble, Integrate, and Test (FAIT) components for integration as a working system, product, or service that complies with specification requirements.

- **Centralized Architecture**—An architecture that "uses a central location for the execution of the transformation and control functions of a system" (Buede, 2009, p. 476).
- **Concerns**—"Those interests, which pertain to the system's development, its operation, or any other aspects that are critical or otherwise important to one or more stakeholders. Concerns include system considerations such as performance, reliability, security, distribution, and evolvability" (Copyright 2000, IEEE. Used with permission; IEEE Std. 1471-2000, 2000, p. 4).
  "A concern pertains to any influence on a system in its environment including developmental, technological, business, operational, organizational, political, economic, legal, regulatory, ecological and social influences" (SEVOCAB, 2014, p. 59) (Source: ISO/IEC/IEEE 24765:2011. Copyright © 2012 ISO/IEC/IEEE. Used by permission.).
- **Decentralized Architecture**—An architecture characterized by "multiple, specific locations at which the same or similar transformational or control functions are performed" (Source: Buede, 2000 p. 476).
- **Fault**—A latent defect, condition – incompatibility, degradation, or deterioration; or hazard that has the potential to materialize into a failure under certain types of operating scenarios and conditions if left undetected.
- **Open Standards**—Refer to the definition provided in Chapter 25, "Definitions of Key Terms."
- **Open System Architecture**—"A logical, physical structure implemented via well-defined, widely used, publicly-maintained, nonproprietary specifications for interfaces, services, and supporting formats to accomplish system functionality, thereby enabling the use of properly engineered components across a wide range of systems with minimal changes" (MIL-STD-499B DRAFT, 1994, p. 37).
- **Open Systems Environment (OSE)**—"A comprehensive set of interfaces, services, and supporting formats, plus aspects of interoperability of application, as specified by Information Technology (IT) standards and profiles. An OSE enables information systems to be developed, operated, and maintained independent of application-specific technical solutions or vendor products" (DAU, 2012, p. B-154).
- **Service-Oriented Architecture (SOA)**—"A software design and software architecture design pattern based on discrete pieces of software providing application functionality as services to other applications" (Wikipedia, 2013).
- **Viewpoint**—"A specification of the conventions for constructing and using a view. A pattern or template from which to develop individual views by establishing the purposes and audience for a view and the techniques for its creation and analysis" (Copyright 2000, IEEE. Used with permission; IEEE Std. 1471-2000, 2000, p. 3).

26.2 APPROACH TO THIS CHAPTER

Chapter 26 begins with a Mini-Case Study that serves as a backdrop for illustrating some of the architecting challenges in Enterprises due to the lack of Systems engineering courses. Our discussions begin with an introductory discussion of System Architecture Development. Topics include:

- What is an architecture and what are its attributes.
- What is an Architecture Description (AD) and what standards govern its application.
- What are the attributes of an architecture.
- What are User views, viewpoints, and concerns and how do they relate to an architecture.
- Architecting versus Engineering versus Designing.

Given this foundation, we shift to Developing the System Architecture. Our discussion focuses on an overlooked aspect of architecture concerning their capabilities to Monitor, Command, and Control (MC2) Mission, Mission System, and Enabling System performance. We build on that discussion with the introduction of a Conceptual Architectural Model that can be tailored to meet most System applications.

The final section addresses Advanced System Architecture topics. Discussions include:

- Centralized versus decentralized architectures.
- Operational, Cold or Standby, or "k-out-of-n" architectural configuration redundancies.
- Like and unlike component configuration redundancies.
- Architectural configuration redundancy versus component placement design redundancy.

- Other considerations—open standards; security and protection; fire detection and suppression; power and loss of power; Environmental, Safety, & Occupational Health (ES&OH), and System of Systems (SoS).

Let's begin with an Introduction to System Architecture Development.

26.3 INTRODUCTION TO SYSTEM ARCHITECTURE DEVELOPMENT

To facilitate our discussion and understanding of System Architecture Development, let's begin with a mini-case study.

Mini-Case Study 26.1

System Architecting by Presentation Charts

A project is planning to submit a proposal for a large System Development contract. The Project Engineer calls together a group of team leads for the proposal kick-off meeting. During the discussion, the team addressed the need to develop a System Architecture. The Project's SE boldly proclaims that they will "go to their office and will return with the System Architecture *within the hour*."

Later, they return with a presentation graphic depicting the System Architecture. During the review, one of the team leads asks *why this is a physical System Architecture portrayal*. Additionally, *why are there Integrated Circuits (ICs) such as Analog-to-Digital Converter (ADC), communications chips, et al shown in a* SYSTEM *Level Architecture Block Diagram (ABD) for a large complex system*. After some additional proclamations by the Project SE that IC functions are always included in an ABD, the group reluctantly decides in the interest of time to adapt and *live with it* rather than continue.

The project wins the contract and several weeks pass. In preparation for the event, the System Developer collaborates with the System Acquirer to issue invitations and develop an agenda for the SDR … The SDR arrives and the System Acquirer and User guests are welcomed.

One of the first agenda topics is a presentation about the System Architecture by the Project SE. Throughout the presentation, the System Acquirer and User Engineers pose challenging questions to the Project SE about the contents of the System Architecture – what's there, what's not there or apparent. Visibly perplexed by the Project SE's "spin", the System Acquirer and Users proceed with asking more pointed questions such as "We see no evidence in your architecture that the operators will be able to do *this*, the maintainers to be able do *that*, or the instructors to be able to teach the operators and maintainers to do *this* and *that*."

Additional requests for information focus on the need for information such as a Concept of Operations (ConOps) Document as well as the *operational* and *behavioral* aspects of the architecture. After all, the System Developer collected the Use Cases information from the User. *Was that a wasted exercise of our time and experiences*? Lacking a valid response, the Project SE responds that the System Architecture should be "apparent and self-explanatory," at least from their perspective.

The Project SE equally perplexed responds with "this is the first we have heard about that." This draws a response from the System Acquirer and User that these concerns were *implied* by System Requirements Document (SRD). The Project Engineer supporting the SE after verifying that the SRD was devoid of such requirements, inquires as to *why* those requirements were not in the SRD. The System Acquirer responds "we thought they were so obvious they didn't need to be documented."

Fictitious? Scenarios such as this happen every day, especially in Enterprises that employ the *ad hoc, endless loop*, Specify–Design–Build–Test–Fix (SDBTF) Design Process Model (DPM) Paradigm. Several points emerge from this mini-case study. System Architecture Development:

1. Requires more than someone returning to their office, quickly "dragging, dropping, and connecting" graphics in a presentation chart, and declaring it an Architecture Block Diagram (ABD).
2. Begins with an experienced System Architect who collaborates with the System Acquirer, Users, and System Developer—Engineers including Human Factors (HF) Engineers, Designers, Manufacturing Engineers, Testers, and others—well in advance of the release of an RFP to understand their User Stories, Use Cases (UCs), views, viewpoints, and concerns.
3. Should reflect, reconcile, and consolidate the *views* and *viewpoints* of its Stakeholders and alleviate their concerns about Critical Operational / Technical Issues (COIs/CTIs).
4. Ensure that capabilities or components for a given *level of abstraction* are appropriate for the architecture rather than *mixed* levels of components that range from SUBSYSTEMS to PART Level "nuts and bolts" such as ADCs.

It should be apparent from the Mini-Case Study 26.1 and the preceding points that a true SYSTEM or ENTITY Architecture is more than simply a 1-hour *quantum leap* (Figure 2.3) from Requirements to Physical Implementation in someone's office creating a presentation chart. To clarify the concept of System Architecting, let's begin with "What is an Architecture."

26.3.1 What is an Architecture?

Most people associate architecture with artistic renderings and fine buildings with ornamented façades that are traceable back to the Greek and Roman antiquity. Classically, you will hear people refer to the *form, fit, and function* of an architecture. In the case of building architectures, form, fit, and function are fine as artistic descriptors. However, Chapter 2 highlighted the shortcomings of the term *function* versus the more appropriate term, *capability*.

What is missing from the educational paradigm is a *universal* definition that encompasses building architectures, systems, products, and services. IEEE Std. 1471-2000 (2000, p. 3) defined an *architecture* as "The fundamental organization of a system embodied in its components, their relationships to each other, and to the environment, and the principles guiding its design and evolution" (Copyright 2000, IEEE. Used with permission). The author proposes the following definition:

- **Architecture**—A graphical model or representation that: (1) expresses the structural framework of a SYSTEM or ENTITY's components, their relationships, and operational, behavioral, and physical interactions *internally* and externally with its OPERATING ENVIRONMENT; (2) incorporates the *views* and *viewpoints* of its Stakeholders; and (3) alleviates their *concerns*.

An architecture *exposes* key features of a system, product, or service. It communicates those features via artistic renderings, drawings, or graphics that illustrate *how* those features interrelate within integrated framework of the SYSTEM and its OPERATING ENVIRONMENT. Note the term "exposes." Although an architectural element or object is visually *exposed* via a drawing or artist rendering, exposure *does not* infer frequency of usage. Some entities may be an integral part of a SYSTEM's phases of operation and in use 100% of the time; other entities may be only used 1% of the time. Depending on the mission or system application, the system, product, or service architecture can be *abstracted* to expose only those *essential capabilities* required by the mission.

This discussion raises several questions about a System Architecture development:

1. *Who is responsible for developing an architecture?*
2. *What is the difference between architecture, engineering, and design*?
3. *What is the relationship between an architecture and the* SYSTEM'S *Users*?
4. *Where is an architecture described*?

Let's address each of these questions.

26.3.2 System Architecting

In most professional domains, the classical building System Architect is expected to possess licensed credentials, preferably by some form of examination and registration accorded by a state of residency. Part of this process is to demonstrate to regulatory decision authorities that the System Architect has the experience, knowledge, and understanding of artistic, mathematical, and scientific principles to competently translate a User's vision into a system, product, or service within the constraints of performance standards, laws, and regulations established by society.

Traditionally, before System Architecting became a formal title, the architecture for most systems, products, or services was created by senior level engineers based on *Systems Thinking* (Chapter 1) and system design experience. Typically, these senior level Engineers originated from electrical, mechanical, civil, or chemical engineering disciplines.

Author's Note 26.1

The role of System Architects

In the discussion following Mini-Case Study 2.1, SDBTF-DPM Enterprises often have the following view of SE:

- Step 1 – Write the specifications.
- Step 2 – Analyze the specification requirements.
- Step 3 – Architect the system.
- Step 4 – Develop the design
- …

From a high-level project workflow perspective over time, this is true. However, SE requires *collaborative, analytical* development and decision-making of all of these steps as illustrated in Figure 11.2. When we address the role of a System Architect, *avoid* the notion that all project work is placed on hold until the Architect emerges with an architectural innovation.

As systems became more software intensive, the need for more specialized expertise lead to the need for a new job category for Software Architects, especially in the software domain. Today, the concept of a System Architect continues to *evolve* with recognition that professional certification standards must be established to certify competency in specific types of business domains. However, here is the challenge: *what does a System Architect "architect"*?

The answer gets resides in an analog to our earlier discussion in Chapter 1 concerning "Engineering the System" versus "Engineering the (Equipment Hardware/Software) Box." Based on our earlier discussion in Chapter 24 concerning Reason's Swiss Cheese Model (1990) (Figure 24.1), the

answer should be "Architecting the System" includes Human Factors (HF) considerations for the User. However, System Architecting often focuses on "Architecting the (Equipment Hardware/Software) Box."

26.3.3 Architecture, Engineering, and Design: What are the Differences

A question that people often have is: *what are the differences between architecting, engineering, and designing*? The answer is that they are *highly collaborative* and *interdependent* processes performed by a System Architect, multidiscipline SEs and Engineers, and Designers.

Author's Note 26.2

The sections that follow respond to the need to differentiate the interrelationships between Architecting, Engineering, and Designing in the context of creating a System Design Solution. This is just one aspect of what Architects, Engineering, and Designers are required to perform in their respective roles, for example. The role of multi-discipline SEs is to collaborate with the User to capture User Stories and develop Use Cases and scenarios. Then lead the multi-discipline development of specification requirements used by the System Architect for performing their role. On various size projects, an Engineer might perform any one, portions of, or all three roles – Architect, Engineer, and Designer shown in Figure 26.1 provides So, what is System Architecting? an illustration to support our discussion.

Architecting is the process of (1) formulating and conceptualizing a viable set of candidate structures—architectures, (2) collaborating with Engineering to select the optimal architecture, and (3) developing the AD.

Engineering is the process of applying mathematical and scientific methods to:

- Collaborate with the System Architect to (1) evaluate the System Architect's candidate architectures using models and prototypes, (2) perform an Analysis of Alternatives (AoAs – Chapter 32), and (3) select the optimal System Architecture.
- Collaborate with designers to select *compatible* and *interoperable* components to (1) create a System Design Solution and (2) develop a System Design Description.
- Develop test cases and procedures for System Verification and Validation (V&V) (Chapter 13).

A *system design* translates Engineering's System Design Solution into drawings and models of physical component layouts, connections, tolerances, parts lists, and so on suitable for procurement or Fabrication, Assembly, Integration, or Test (FAIT).

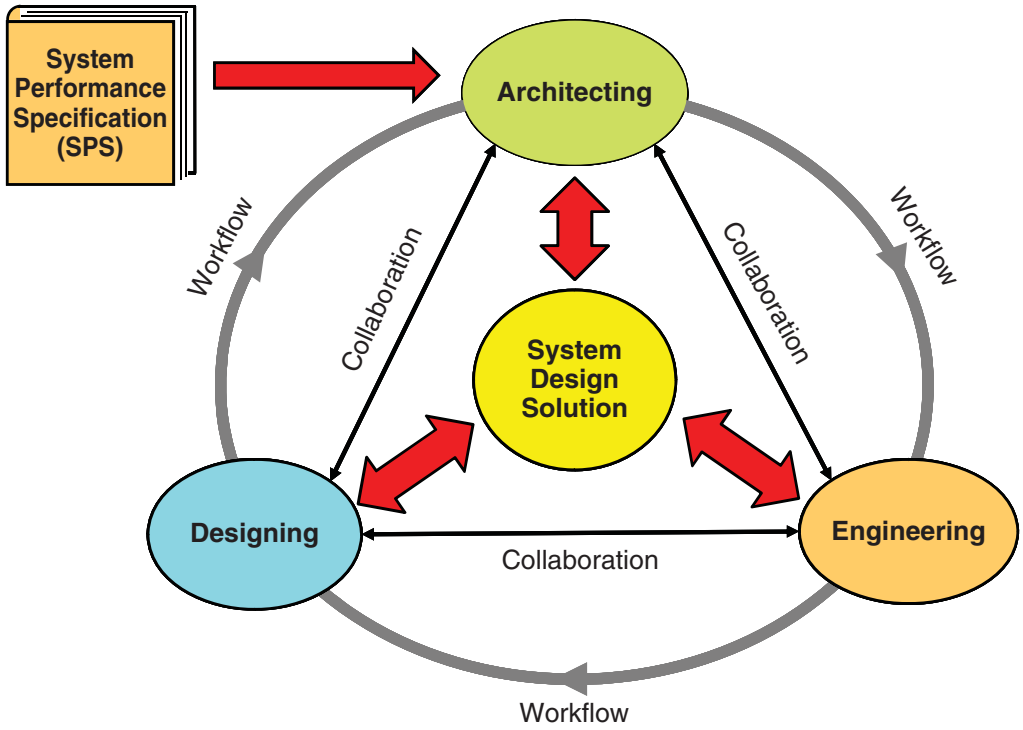


Figure 26.1 Relationships between Architecting, Engineering, and Designing to the System Design Solution

Author's Note 26.3

Based on the points earlier, it should be apparent that Architecting, Engineering, and Designing have an iterative, sequential workflow that is *highly collaborative* and *interdependent*. Unfortunately, simplistic descriptions of these processes would have you believe that a System Architecture is created first and then the SYSTEM or PRODUCT is designed in a manner similar to the Waterfall Development (Figure 15.1). This is a partial truth perpetuated as a myth!

When do Architecting, Engineering, and Designing begin and end in relation to each other?

- First, for small projects, an Engineer might perform all three roles. As project sizes become larger and more complex, then you will see these roles emerge in the form of roles assigned to various project personnel.
- Secondly, you would expect a System Architect to begin their conceptualization and collaborate with multi-discipline SE—hardware, software, testing, and so on. Multidiscipline SE activities might include Modeling and Simulation (M&S) (Chapter 33), scale-model prototypes and so on to support an AoA—Chapter 32—to select a preferred architecture from a set of viable candidates. Designers might be involved as part of the evaluation. The intent here is to identify, investigate, and mitigate any Critical Operational and Technical Issues (COIs/CTIs).

One of the rhetorical questions that emerges concerning Architecting, Engineering, and Designing is whether *design* is part of the *architecture* or is the *architecture* part of the *design*?

26.3.4 Formulating the AD

System Architectures originated hundreds of years ago via the need to express a conceptualization such as a building or ship worthy of resourcing a project and holding those developing the project accountable for the outcome. The expressions took the form of sketches, paintings, and drawings.

When people began to recognize the need for more detailed descriptions, System Architectures became a feature of the *design* documents such as a SYSTEM or ENTITY Design Description delivered at the completion of the project.

In the 1980s, as SYSTEMS and PRODUCTS became more software intensive, complex, and costly, System Acquirers, Users, and even System Developer personnel had a "need-to-know" *early* in System Development *how* the software was envisioned to operate, not at the end when the SYSTEM or PRODUCT was delivered. As SE and Software Development became more *closely coupled*, the concept of a Software Architect evolved into the need for a SYSTEM Architect.

One of the architectural work products that emerged from Software Architecture Development is a document referred to as the Architectural Description (AD). Figure 26.2 illustrates the context and Entity Relationships (ERs) of an AD to the System of Interest (SOI), its Architecture, Stakeholders, their Concerns, Architecture Rationale, and so on. So, *what is an AD*?

- IEEE Std. 1471-2000 (2000, p. 3) *Architectural Description for Software-Intensive Systems*, which has been replaced by ISO/IEC/IEEE 42010 (2011), generically defined an AD as simply "A collection of products to document an architecture" (Copyright, 2000, IEEE. Used with permission).
- ISO/IEC/IEEE 42010 (2011, p. 17), which is now titled *Systems and Software Engineering—Architecture Description*, defines an AD as a "(1) a collection of products to document an architecture" (SEVOCAB, 2014, p. 17) (Source: ISO/IEC/IEEE 24765:2011. Copyright © 2012 ISO/IEC/IEEE. Used by permission.). ISO/IEC/IEEE 42010:2011 (pp. 11–16) provides a description of what an AD should contain in terms of its outline and contents. In a separate ISO document, Hilliard (2012) provides an annotated *AD Template* based on ISO/IEC/IEEE 42010 :2011.

What should an AD contain? "A collection of products to document an architecture" (SEVOCAB, 2014, p. 17) as stated above? How do you determine when the "collection of products" is necessary and sufficient to "Engineer" and "Design" the product?

From an SE perspective, one would expect an AD at a *minimum* to contain work products such as the following:

1. Conceptual view of the proposed physical SYSTEM or PRODUCT at delivery. This might include various types of Engineering views such as Plan/Front View, Side Views, and Top Views.
2. Level 0 User's System ABD (Figure 8.4) - depicting the deliverable SOI - MISSION SYSTEM and ENABLING SYSTEM(S) – and their interfaces to other components within the User's System (Figures 9.1 and 9.2)
3. Level 1 System/Entity ABD - depicting the major components (Figure 8.13), their internal interfaces with each other (N2 Diagram - Figure 8.11), and interfaces to external systems (Figures 8.1, 9.3, and 10.1) in their OPERATING ENVIRONMENT.
4. A text description that describes *how* the SYSTEM/PRODUCT's architecture is conceived to address

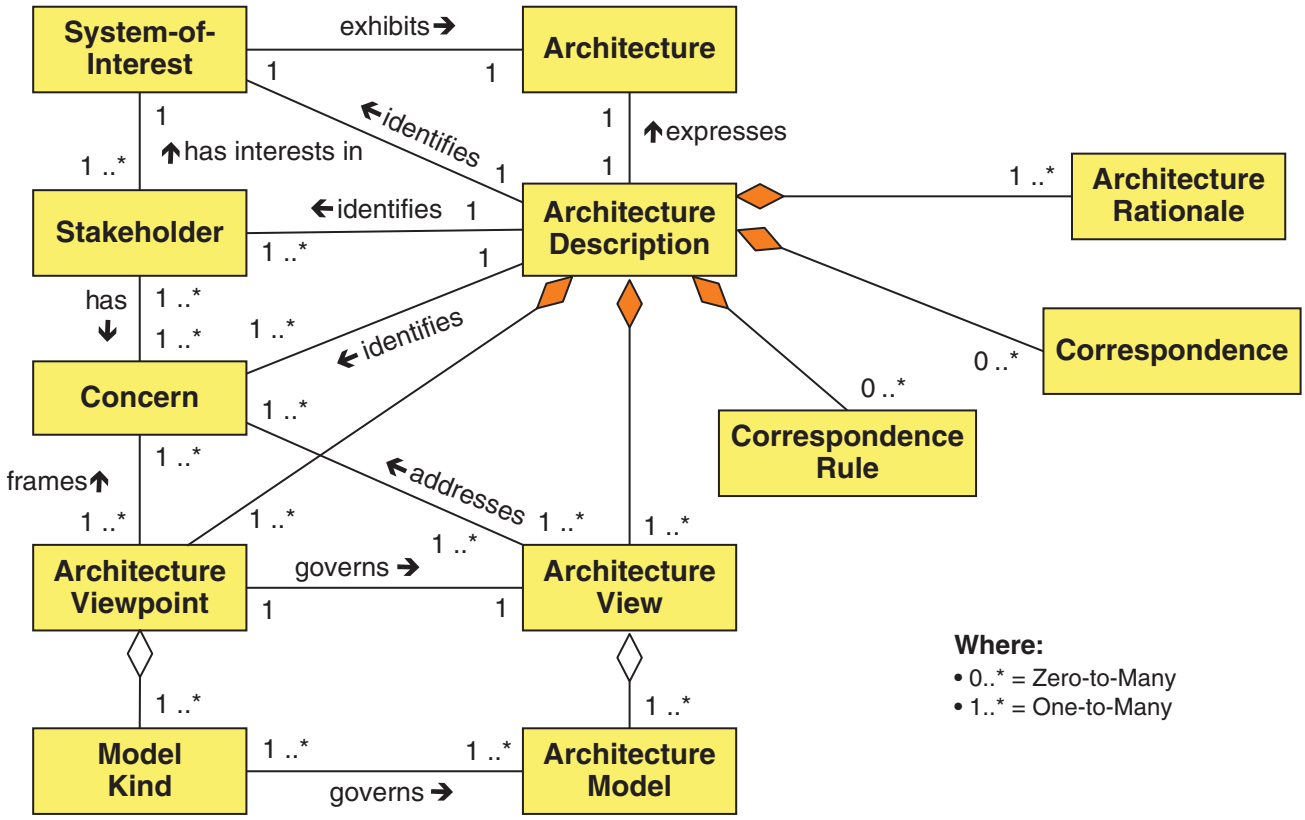


Figure 26.2 ISO/IEC/IEEE 42010:2011 Conceptual Model of an AD (Source: This excerpt is taken from ISO/IEC/IEEE 42010:2011, Figure 2 on page 5, with the permission of ANSI on behalf of ISO. (c) ISO 2014—All rights reserved.)

and satisfy User Stories, Use Cases (UCs), and scenarios including HFs (Chapter 24) for its Users operators, maintainers, trainers, and others during each of the SYSTEM/PRODUCT's Life Cycle Phases of Operation - Deployment; Operations, Maintenance, and Sustainment (OM&S); Retirement; and Disposal.

From a Software perspective, an AD is often considered a standalone document. From an SE perspective, an AD is an integral part of the ConOps document. In that context, the AD could either be presented in the ConOps or *incorporated by reference* as *referenced document*.

26.3.5 Stakeholder Views, Viewpoints, and Concerns

System Stakeholder Concerns and Views Principle

Principle 26.1 An architecture expresses System Stakeholder *concerns* via *views* that comply with *viewpoint* conventions for constructing the view.

Figure 26.2 introduces several ISO/IEC/IEEE 42010:2011 terms used in formulating a system, product, or service architecture that require definition. Specifically, the terms are *views, viewpoints,* and *concerns. Why are these important*?

If you revisit Mini-Case Study 26.1, System Acquirer and User participants in a technical program review include a cadre of Stakeholder roles. These roles include operators, maintainers, trainers, and so on as well as technical specialists such as Reliability, Maintainability, and Availability (RMA); Human Factors (HF); System Safety, System Security, and Technologists. The left side of Figure 15.11 provides an excellent example. Some specialists address COIs/CTIs such as global SYSTEM Level issues, SUBSYSTEM Level issues, or technology. Each of these roles and disciplines has *views* or *perspectives* of a SYSTEM or ENTITY based on their education, knowledge, and experience.

Given Stakeholder – User and End User - insights concerning a System Architecture, let's define *views, viewpoints, and concerns*.

- **View**—"An architecture view expresses the architecture of the system-of-interest in accordance with

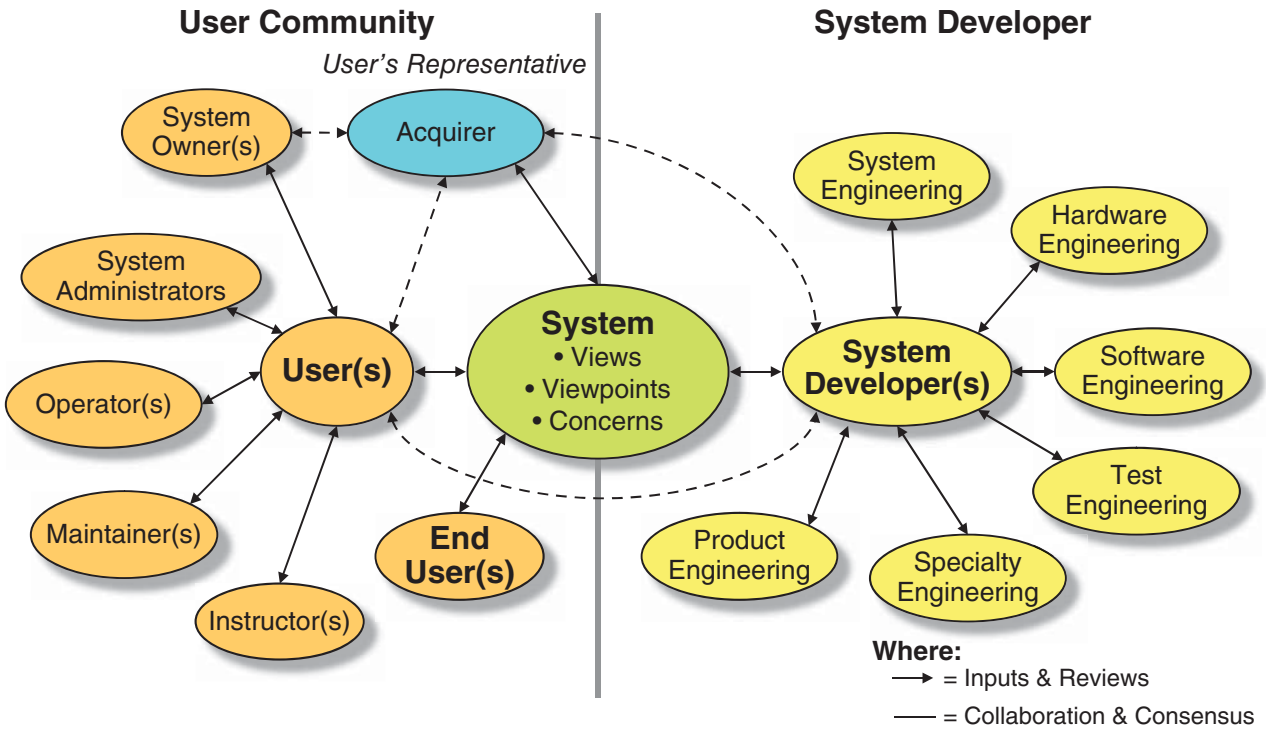


Figure 26.3 Illustration of the Challenges of Balancing Various Stakeholder Views, Viewpoints, and Concerns of a System

an architecture viewpoint (or simply, viewpoint)" (ISO/IEC/IEEE 42010:2011, Section 4.2.4, p. 6.)[1]
"A view is **governed** by its viewpoint: …" (ISO/IEC/IEEE 42010:2011, p. 6)[2]

- **Viewpoint**—"… the viewpoint establishes the conventions for constructing, interpreting and analyzing the view to address concerns framed by that viewpoint. Viewpoint conventions can include languages, notations, model kinds, design rules, and/or modeling methods, analysis techniques and other operations on views" (ISO/IEC/IEEE 42010:2011, p. 6)[3]
- **Concern**—An "interest in a system relevant to one or more of its stakeholders …
  Note: A concern pertains to any influence on a system in its environment, including developmental, technological, business, operational, organizational, political, economic, legal, regulatory, and ecological and social influences" (SEVOCAB, p. 59, Copyright, 2012, IEEE. Used with permission; Reference: ISO/IEC/IEEE 42010:2011 (2011 p. 6).

ISO/IEC/IEEE 42010:2011 (2011 p. v) provides examples of *concerns* such as "… the feasibility, utility and maintainability of the system."[4]

To better understand *how* views, viewpoints, and concerns influence a System Architecture, Figure 26.3 provides an illustration. Observe that the graphic consists of the User Stakeholders—System Owner, System Administrator, and various Users; and their End User on the left side and System Developer Stakeholders on the right side. Each of these Stakeholders has:

- A *view* or perspective of the system, product, or service based on their mission roles and assigned tasks. For example, Engineers such as EEs, MEs, SwE's, and others have their own unique perspectives of the electrical/electronic, mechanical, and software "system."
- *Viewpoints* expressed as conceptual mental models (Normal, 2013 Figure 1.11, p. 32)—Viewpoint Models—that represent how they believe the system, product, or service should be operated, maintained, etc. For example, an MRI medical device (Figure 25.5) Engineer (Figure 25.5) has a viewpoint concerning

[1]This excerpt is taken from ISO/IEC/IEEE 42010:2011, Section 4.2.4 on page 6, with the permission of ANSI on behalf of ISO. (c) ISO 2014—All rights reserved.
[2]Ibid
[3]Ibid
[4]This excerpt is taken from ISO/IEC/IEEE 42010:2011, Introduction on page 5, with the permission of ANSI on behalf of ISO. (c) ISO 2014—All rights reserved.

how the electromagnets of an MRI device (Figure 25.4) are calibrated and aligned. Specifically, how to achieve the *imaging isocenter* – the point at which all three gradient fields—+X, +Y, and +Z—are zero. (Eash, 2013)

- *Concerns* that represent COIs/CTIs that require resolution that is apparent in the Viewpoint Models.

As examples, here is a partial listing of Stakeholder concerns:

1. Single Use, Reusable, & Multi-Purpose
2. Reconfigurability
3. Effectiveness
4. Efficiency
5. Usability
6. Human-System Integration (HSI)
7. Physical Layout
8. Access Ports
9. Electrical Grounding and Shielding
10. Electromagnetic Interference (EMI)
11. Electromagnetic Compatibility (EMC)
12. Life Support
13. Environmental
14. Comfort
15. Communications
16. Illumination
17. Ingress and Egress
18. Consumables and Expendables
19. Waste Disposal
20. Safety
21. Compatibility and Interoperability
22. Growth and Expansion
23. Reliability
24. Maintainability
25. Availability
26. Producibility
27. Storage
28. System Integration
29. System Verification
30. System Validation
31. System Deployment
32. System Operation
33. System Sustainment
34. System Retirement
35. System Disposal
36. Transportability
37. Mobility
38. Maneuverability
39. Portability
40. Training
41. Security and Protection
42. Vulnerability
43. Survivability
44. Lethality

26.3.6 Attributes of an Architecture

If someone asked you to characterize an architecture, *what would be its attributes*? For many people, their mental picture of an architecture is simply an SBD or ABD. The reality is an architecture has a set of key attributes one should expect to see evident in an architecture.

26.3.6.1 Attribute #1: Assigned Ownership

Architectural Ownership and Accountability Principle

Principle 26.2

Assign *ownership* and *accountability* for the development of each SYSTEM or ENTITY architecture.

Every SYSTEM or ENTITY Architecture is assigned to an Owner accountable for its development and maintenance.

26.3.6.2 Attribute #2: Unique Document ID

Architectural Uniqueness Principle

Principle 26.3

Assign *a unique identifier* (ID) to each SYSTEM or ENTITY architecture.

Every SYSTEM or ENTITY Architecture document should be assigned a unique document ID that differentiates it from all others.

26.3.6.3 Attribute #3: Architectural Context

Architectural Context Principle

Principle 26.4

Every SYSTEM or ENTITY architecture begins with establishing its *context* within the User's Level 0 HIGHER-ORDER SYSTEM.

The third attribute of an architecture is to establish its context within the multi-level world of systems as viewed by its Stakeholders, Users, and End Users. To illustrate this point, consider a building architect commissioned to develop the architecture for a new office building. Imagine a white background with nothing but the building shown. Obviously, the building is the most important aspect of the artwork and is *necessary*. However, *is it sufficient in terms of satisfying Stakeholder visualizations* (e.g., its features

such as User access, parking, or environmental park-like setting) if this information is *suppressed* in the artwork? The answer is no! One of the important aspects of architecture is understanding a SYSTEM's context relative to its User's Level 0 (Figure 8.4) OPERATING ENVIRONMENT. Consider the following examples.

Example 26.1 **Home Architectural Context**

A home architect is commissioned to develop plans for a home to be located on the edge of a city, mountain lake, or arid desert. So, they create an artistic rendering of the envisioned home as the centerpiece surrounded by its environment. The intent is to captivate the customer's excitement and imagination and to show that it blends *unobtrusively* into its environment.

Example 26.2 **Military Aircraft: Battle Environment Context**

A System Developer of a new military aircraft creates an artistic rendering or ABD of the aircraft in an aerial battle to illustrate its *agility* and *air superiority*. Within the artwork, operational assets are depicted including graphically layered lines illustrating communications with satellites, ships, other aircraft, ground forces, and so on.

Example 26.3 **Medical Device Context**

A medical device manufacturer releasing a new product might create an artistic or a graphical view of the device in a patient's or operating room setting that includes the PERSONNEL —various medical personnel, EQUIPMENT, MISSION RESOURCES, PROCEDURAL DATA, SYSTEM RESPONSES, and FACILITIES (Figures 9.1 and 9.2).

These examples illustrate just one aspect of an architectural via an artistic rendering. From an SE perspective, we can create contexts with ABDs based on Stakeholder views and viewpoints.

26.3.6.4 Attribute #4: Architectural Framework

Principle 26.5 **Architectural Framework Principle**

Every SYSTEM or ENTITY architecture requires a Level 1 SYSTEM architecture that illustrates:

1. Its framework of System Elements—PERSONNEL, EQUIPMENT, MISSION RESOURCES, PROCEDURAL DATA, SYSTEM RESPONSES, and FACILITIES—their relationships, and interactions.
2. External interfaces and interactions with HUMAN SYSTEMS, the NATURAL ENVIRONMENT, and the INDUCED ENVIRONMENT SYSTEMS (Figures 9.1 and 9.4) that comprise its OPERATING ENVIRONMENT.

In summary, an architecture: (1) *exposes* a contextual view of a SYSTEM or ENTITY's Level 1 framework of System Elements—PERSONNEL, EQUIPMENT, MISSION RESOURCES, PROCEDURAL DATA, SYSTEM RESPONSES, and FACILITIES—(2) their interrelationships, and (3) interactions with external systems in their respective OPERATING ENVIRONMENTS. Figure 9.1 provides an example.

26.3.6.5 Attribute #5: Architectural Domain Views

Principle 26.6 **Architectural Domain Views Principle**

Each SYSTEM or ENTITY architecture consists of one or more *viewpoint models* that:

1. Represent views shared by one or more of its Stakeholders.
2. Express a conceptualization to guide Solution Space development.
3. Alleviate any *concerns.*

A SYSTEM or ENTITY's architecture consists of four types architectural views, one for each Solution Domain illustrated in the SE Process (Figure 14.1). From an SE perspective, your role is to develop a shared consensus of these SYSTEM Level Solution Domains views across the System Acquirer, User, and System Developer communities as shown in Figure 26.4. This graphic illustrates the deficiencies of the *ad hoc* "I will return in an hour with the System Architecture" approaches depicted in Mini-Case Study 26.1.

One of the *ambiguities* of SE and *deficiencies* of Enterprise training or the lack thereof occurs when System Architectures are developed. Referring to Mini-Case Study 26.1, you may hear a System or Product Development Team (SDT/PDT) member boldly proclaim they are going to "develop a system architecture." The problem is listeners are *thinking* one type of architecture and the individual creating the architecture has something different in mind—a single Physical System Architecture viewgraph. As a result, the architectural work product may or may not suit the development team's needs.

When someone *boldly proclaims* to develop a System Architecture, you should ask: *Which architectural views and viewpoints do they intend to capture and express?* The answer should be the timely sequence of Operational, Behavioral, and Physical Architecture artifacts from the SE Process (Chapter 14) collaboratively developed over time with the User and System Developer Stakeholders.

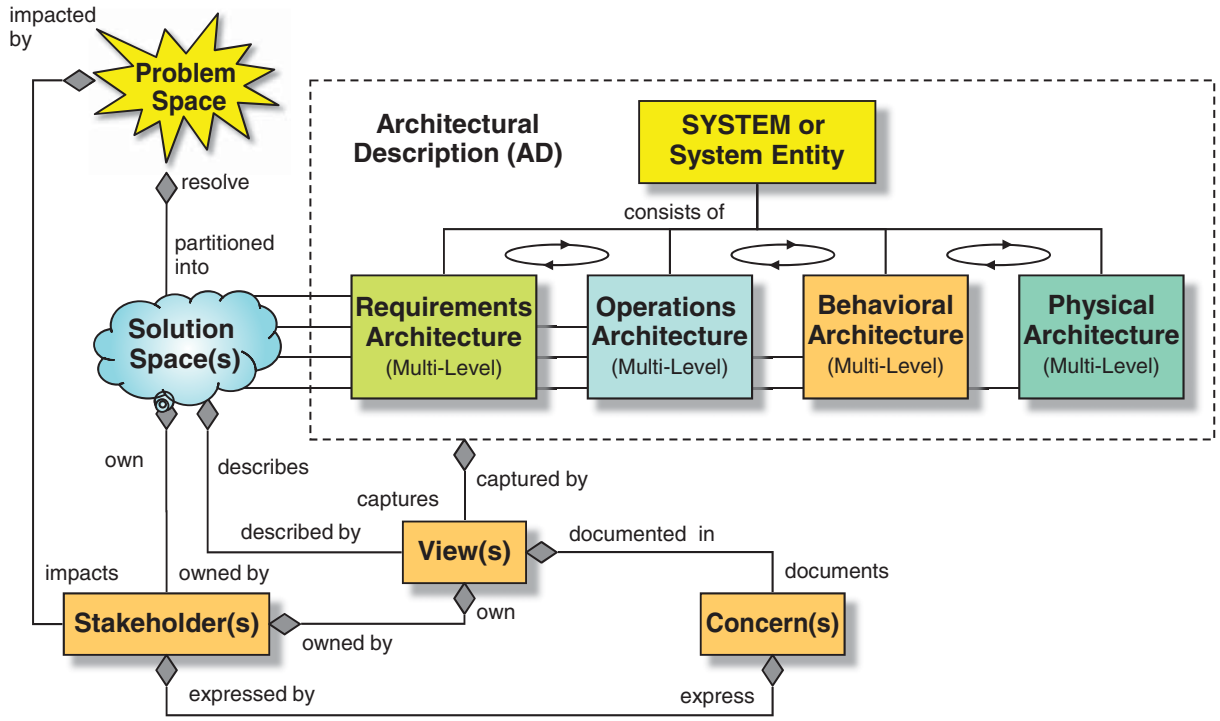


Figure 26.4 Establishing a Consensus of Stakeholder Views of a System's Four Domain Solutions

26.3.6.6 Attribute #6: Architectural Validity

Architectural Validity Principle

Principle 26.7

The *validity* of an architecture is determined by its ability to:

1. Configure and control its capabilities in response to UC-based User Command and Control (C2) during Deployment; Operations, Maintenance, and Sustainment (OM&S); and Retirement/Disposal Phases of the System/Product Life Cycle.
2. Deliver the required outcomes and levels of performance for each configuration.

Any type of architecture can provide a set of capabilities to produce an outcome. The real challenge is: *can it deliver unique sets of capabilities required to support the System UCs and Scenarios within each Mode and State (Chapter 7) for each of its System/Product Life Cycle Phases of operation (Figure 3.3)*?

26.3.6.7 Attribute #7: Architectural Completeness

Architecture Completeness Principle

Principle 26.8

A SYSTEM OR ENTITY architecture is *complete* when it:

1. Complies with Stakeholder specification requirements.
2. Clearly communicates a solution that alleviates Stakeholder concerns and mitigates operational, technical, technology, cost, and schedule risks.
3. Contains *essential* information that is *necessary* and *sufficient* for design.
4. Has been *verified*, *validated*, and *accepted* by its Stakeholders.

Since a SYSTEM or ENTITY architecture is an integral part of its design, *completeness* evolves to *maturity* through design reviews; System Integration, Test, and Evaluation (SITE); system V&V; and User experience in the field of operation.

26.3.6.8 Attribute #8: Architectural Design Consistency

Architectural Consistency Principle

Principle 26.9

A SYSTEM or ENTITY ARCHITECTURES and Engineering designs must be *consistent* in nomenclature, mnemonics, and terminology with all other architectures internal and external to the SYSTEM.

Multi-Level System Architectures, especially for large, complex systems, are often developed by different people

and contractors. One of the complexities of System Design integration is ensuring that the architectures be developed as if a *single source* created them. This means that *Viewpoint Models* are based on established standards such as SysML™ to ensure *consistency* (1) between peer-level models and hierarchically between levels of abstraction and (2) in nomenclature, mnemonics, and terminology. In this context, it is critical for SE to create a *Glossary of Terms and Acronyms* that is accessible, reviewed, approved, baselined, and maintained for everyone working on the project.

26.3.6.9 Attribute #9: Architectural Selection Rationale

Architecture Selection Rationale Principle

Principle 26.10 Every SYSTEM/ENTITY architecture should include a listing of selection criteria and supporting rationale for its final selection.

System Architectures are often selected for a variety of reasons. Later, when issues surface, questions emerge concerning what criteria were used as the basis for its selection. Therefore, document the criteria and supporting rationale used to substantiate the selection.

26.3.6.10 Attribute #10: Architectural Traceability

Architectural Traceability Principle

Principle 26.11 Every SYSTEM or ENTITY ARCHITECTURE must be *traceable* to HIGHER-ORDER SYSTEM architectures and to User requirements via the System Performance Specification (SPS).

SYSTEM /ENTITY Architectures must be *traceable* to each other and User *source* or *originating* requirements via the SPS (Principle 19.13).

26.3.6.11 Attribute #11: Architectural Capabilities We stated earlier in Principle 26.7 that the validity of a System Architecture is determined by its ability to respond, configure, and control its capabilities in response to User MC2 during all phases, modes, and states of its life cycle. A challenge question for System Architects is: *how do you assess the validity and completeness of an architecture to comply with Principle 26.7*? *How good is "good enough"*? *Completeness from whose perspective*? The architect, engineer, designer, or User? Obviously, the answer becomes known through field usage. Although that may be *necessary*, it is *insufficient* in terms of developing the SYSTEM or PRODUCT.

Actually, the answer to the question resides in three supporting questions:

- **Question #1:** *What are the capabilities the User(s) expect the* SYSTEM *or* ENTITY *to provide?*
- **Question #2:** *Based on seasoned System Architecting experience, are there essential capabilities that have not been addressed by the User?*
- **Question #3:** *Does the AD answer this question?*

The answer to Question #1 is documented in the SPS. Question #2 could be a variable dependent on the System Architect's experience and business domain. However, there is an easier way of answering the question by simply applying some *Systems Thinking* (Chapter 2).

If you analyze SYSTEMS or PRODUCTS of various domains such as transportation, medical, energy, or communications, nine *essential* capabilities emerge, each with some specified level of performance. Even though the implementations may be: (1) fixed installations, mobile, or portable or (2) operate from electrical, mechanical, solar, wind, or other power sources, they share these *essential* capabilities. In general, most Engineered Systems capabilities include:

1. Send and Receive Messages and Data.
2. Accept, Transform/Convert, Distribute, and/or Store Energy.
3. Monitor Environmental Conditions—Internal and External.
4. Maintain and Report Situational Assessment Status.
5. Respond to User C2 Inputs.
6. Maintain Mission Standard Time.
7. Stow and Retrieve Personnel, Tools, PROCEDURAL DATA, and MISSION RESOURCES—data and Ancillary Equipment.
8. MC2 System Performance.
9. Record Mission and System Performance Data.
10. Store and Retrieve Mission and System Data.
11. Sense Operational Health and Status (OH&S).
12. Provide Security and Protection.
13. Produce Performance-Based Outcomes.

Using these capabilities as a *general checklist*, a reviewer should be able to review an AD and be able to clearly identify and trace from Input to Output how the capability is accomplished *operationally*, *behaviorally*, and *physically*.

26.3.7 Architecture Representation Methods

For most applications, System Architectures are communicated via three mechanisms: (1) three-dimensional or two-dimensional artistic renderings of buildings, (2) block diagrams such as SBDs or ABDs, and (3) hierarchy trees. Most SE applications employ block diagrams such as various types of SySML™ Structural Diagrams—Block Definition, Internal Block, and Package (Appendix C, Figure C.2) (OMG SysML™, 2012); System Block Diagrams (SBDs) or

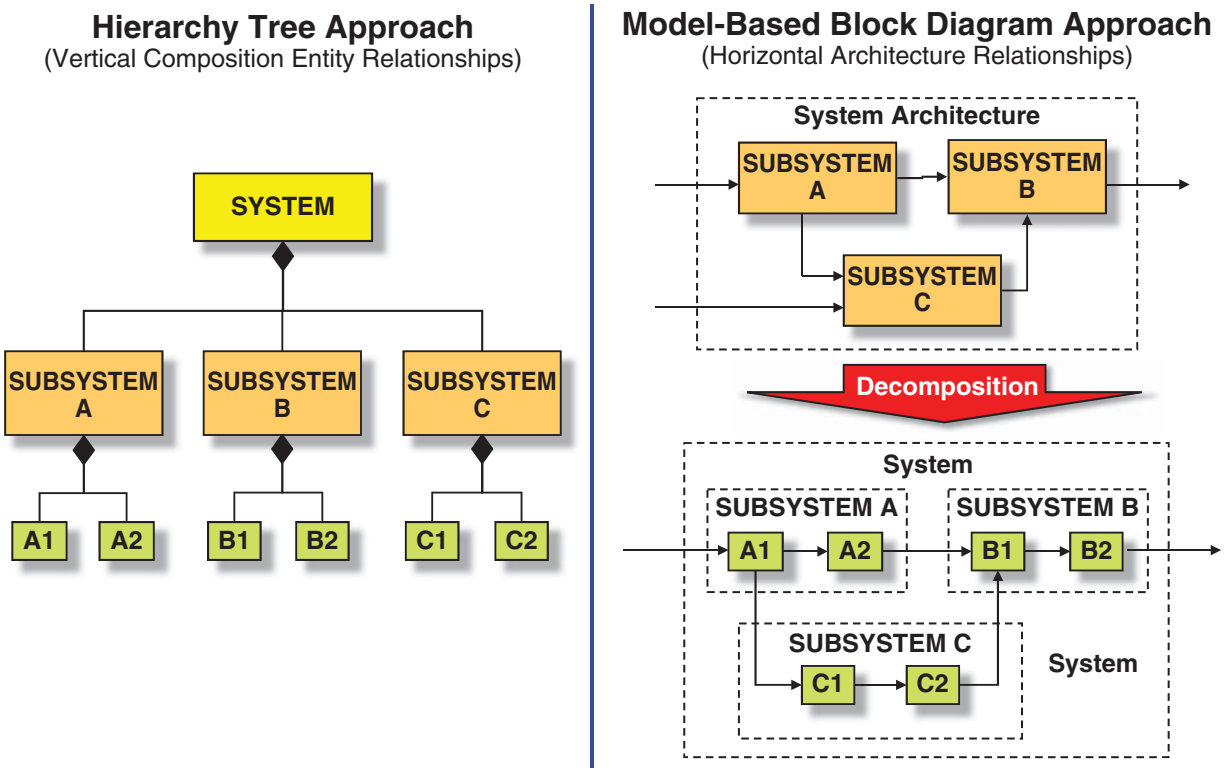


Figure 26.5 Presentation Methods Used to Depict System Architecture Structures—(Left) Hierarchy Tree, (Top Right) Architecture Block Diagram (ABD), and (Lower Right) Multi-Level Decomposition

more specifically ABDs as the primary mechanism for communicating a SYSTEM or ENTITY's architecture:

- *Block diagrams* such as SBDs and ABDs, in general, depict horizontal, peer-level, and external relationships within a given level of abstraction. Vertical linkages to higher-level *parent* or lower-level *child* entity architectures (Figure 21.2) are referenced by symbols such as connectors between sheets of a drawing or a document. For example, a tool may annotate a SYSTEM or ENTITY in a diagram with a *symbol* to denote that a lower level of abstraction exists.
- *Hierarchy diagrams* enable us to depict vertical, hierarchical relationships as levels of abstraction. They *do not* communicate, however, direct relationships and interactions among peers.

Figure 26.5 illustrates these two approaches.

26.4 DEVELOPMENT OF SYSTEM ARCHITECTURES

Given an introductory understanding of System Architecture Development, how do you develop a System Architecture? Since a System Architecture serves as the central infrastructure of a System Design Solution, its success at least conceptually is *unknown* until the System is placed into field operation and has proven mission performance. That represents a challenge that we need to remedy early in the project. *How do we reduce architectural risk*? As with any type of system, the more you know and understand about a SYSTEM or PRODUCT's operations, behavior, and physical implementation, your chances of success increase significantly. So, models, simulations, prototypes, and analyses approaches provide a way to derisk the architectural design.

One of the first things we learn about Systems or Products is that we need to be able to control their outcomes and performance. Performance control, in turn, requires feedback concerning *actual* versus *planned* performance.

26.4.1 MC2 System Performance and Outcomes

Principle 26.12

Measurement and Control Principle

Unless you can measure the performance of a system or process, you cannot control it. (Adapted from Demarco, 2009, p. 96).

If you contemplate a system, product, or service's architecture, there are four core capabilities required for Users to operate and maintain it:

1. Configure the SYSTEM's capabilities to accomplish performance-based outcomes during all phases, modes, and states of operation.
2. Command the SYSTEM to perform specific actions.
3. Control the SYSTEM's performance.
4. Monitor the SYSTEM's performance.

Analytically, the first three items represent C2; the third item requires Situational Awareness of the SYSTEM or PRODUCT's performance—Situational Assessment.

Figure 26.6 provides a high-level illustration of the relationship between a SYSTEM's Situational Assessment and C2 capabilities. Each of the boxes highlights key aspects of its scope.

In performing Situational Assessment and C2, a system, product, or service must be able to respond to its *authorized* Users—operator, maintainer, or instructors. Three questions emerge:

1. What level of control does the User need to exercise to safely achieve mission objectives?
2. How does the User know when the desired SYSTEM Level of performance is achieved, is optimal, or encroaches into a potential Caution or Warning (Principles 24.16 and 24.17) safety area (Figure 30.1) that may impose *unnecessary* stress on the SYSTEM, jeopardize the mission, and potentially result in injury or loss of life?
3. If SYSTEM performance is not at an expected level of performance due to an *unknown* component failure, how does the User know this and confirm its Operational Health & Status (OH&S) especially when conditions preclude inspection? For example, in the case of the NASA Space Shuttle, the astronauts were unaware of the Challenger (1986) addressed later in Mini-Case Study 26.2 or Columbia (2003) in Mini-Case Study 26.3.

The answer to these questions requires that the User make timely, *informed* Situational Assessment and *corrective action* decisions. *Informed* decisions require presentation of current, *essential* information at the User's interface based on an assimilation of OH&S data. As a result, the User should be able to apply training and experience to take preemptive, corrective actions via the SYSTEM's C2 capabilities.

The point here is captured in the Demarco's (2009) quote (Principle 26.12) concerning the need to be able to measure performance if you expect to control it (Demarco, 2009,

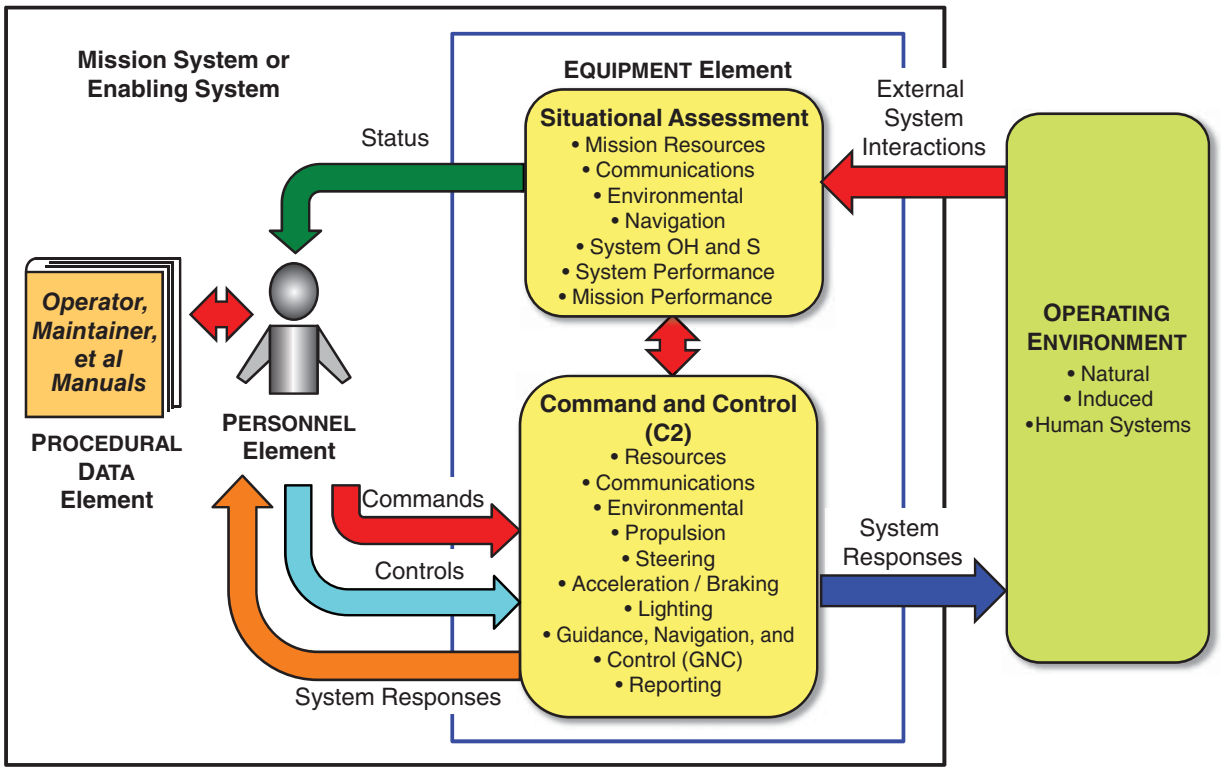


Figure 26.6 High-Level System Construct Illustrating the Situational Assessment and C2 Interfaces with the User

p. 96). The context of the quote was software development; however, it is relevant to System Engineering and Development especially in formulation of System Architectures. *How do we provide the User with the essential information required to manually control the System or have it automatically control itself*? The answer to that question requires the Situational Assessment to provide only essential information.

Principle 26.13

Measuring the Unmeasurable Principle

Avoid demands to "measure the unmeasurable" (Leveson and Turner, 1993, p. 39).

Leveson and Turner (1993, p. 39) in their paper "An Investigation of the Therac-25 Accidents" observe that management and others often pressure Engineers to *quantity* risk. They note that Engineers should *exercise caution* and *insist* that any risk assessment numbers are "meaningful." Since these numbers are based on *conditional* probabilities, they should be treated with caution.

Although the context of the Leveson and Turner quote relates to risk management, it applies to Situational Assessment data collection as well. From an HF perspective, humans under stress need only specific information that is *essential* to a decision, not information overloads. If Situational Assessment information is *nonessential*, *why spend resources—time, cost, weight, and other factors—instrumenting and measuring data that contributes additional technical risk, reduces reliability, increases weight, and is non-value added*? This brings us to another principle in System Architecture Development.

Principle 26.14

Situational Assessment Information Principle

Essential, real-time Situational Assessment information should always be *current* and *readily available* for Users to make informed decisions.

The importance of this principle is highlighted in Mini-Case Study 26.2 of a current, real-time Situational Assessment in an Enterprise System and an Engineered System.

Author's Note 26.4

The case studies in the following text address the NASA Space Shuttle Challenger and Columbia accidents. As an Engineering case study, the point here is not to highlight problems. NASA has made tremendous contributions to advancing our knowledge of space and its exploration and development of new technologies that have improved our lives. The best way we can honor the brave men and women astronauts who flew these missions is to learn from the accidents and mitigate the conditions that lead to their occurrence.

Mini-Case Study 26.2

NASA Space Shuttle Challenger Accident

On January 28, 1986, a decision was made to launch NASA Space Shuttle Challenger on it STS-51-L mission. At 73 seconds after liftoff, the External Tank ruptured from a SRB interaction causing the liquid fuel to explode leading to the Challenger breakup.

"The consensus of the Commission and participating investigative agencies is that the loss of the Space Shuttle Challenger was caused by a failure in the joint between the two lower segments of the right Solid Rocket Motor. The specific failure was the destruction of the seals that are intended to prevent hot gases from leaking through the joint during the propellant burn of the rocket motor. The evidence assembled by the Commission indicates that no other element of the Space Shuttle system contributed to this failure" (Rogers Commission, 1986, p. 40).

"The decision to launch the Challenger was flawed. Those who made that decision were unaware of the recent history of problems concerning the O-rings and the joint and were unaware of the initial written recommendation of the contractor advising against the launch at temperatures below 53° F and the continuing opposition of the engineers at Thiokol after the management reversed its position. They did not have a clear understanding of Rockwell's concern that it was not safe to launch because of ice on the pad. If the decision-makers had known all of the facts, it is highly unlikely that they would have decided to launch 51-L on January 28, 1986" (Rogers Commission, 1986, Chapter 5, p. 82).

Mini-Case Study 26.2 illustrates the human decision making flaws in an Enterprise System concerning the flow of current, real-time Situational Assessment information to key decision-makers. Forrest (2013) characterizes the Challenger accident as "A failure in decision support system and human factors management." The mishap exemplifies Reason's (1990) Accident Trajectory Model (Figure 24.1).

Now, consider another context involving the Space Shuttle Columbia accident.

Mini-Case Study 26.3

Space Shuttle Columbia Accident

In the morning hours of February 1, 2003, the NASA Space Shuttle Columbia accident materialized as a result of wing leading edge damage from ice breaking off at liftoff

several days earlier. The Columbia Accident Investigation's Board's Report (2003, pp. 11–12) stated:

> "The de-orbit burn to slow *Columbia* down for re-entry into Earth's atmosphere was normal, and the flight profile throughout re-entry was standard. Time during re-entry is measured in seconds from "Entry Interface," an arbitrarily determined altitude of 400,000 feet where the Orbiter begins to experience the effects of Earth's atmosphere. Entry Interface for STS-107 occurred at 8:44:09 a.m. on February 1. Unknown to the crew or ground personnel, because the data is recorded and stored in the Orbiter instead of being transmitted to Mission Control at Johnson Space Center, the first abnormal indication occurred 270 seconds after Entry Interface."

Unfortunately, at that point in reentry, there was nothing that could be done. As a result of the Columbia accident, NASA (2005, p. 50) noted "In addition to improved cameras on the ground and on the Space Shuttle, Discovery's astronauts will conduct close-up, in-flight inspections with cameras, lasers, and human eyes."

These examples illustrate the importance of Enterprise and Engineered Systems providing adequate means for the Users to MC2 the SOI. The objective is to provide decision-makers with timely Situational Assessments to enable corrective actions that mitigate a potential hazard from becoming a problem. *How do we create an architecture to accomplish this objective*?

The answer to the question is every System Architecture is business domain and application dependent based on specialized expertise. It would be impractical to create every type of architecture. For example, SYSTEMS and PRODUCTS that include broad ranges of consumer goods; offices and public buildings; transportation—land, sea, air, and space based; medical; and many others. There is, however, model-based solution that could be used to identify key considerations when developing an architecture.

If you analyze a broad spectrum of SYSTEMS or PRODUCTS, you can identify some high-level classifications of capabilities that a typical architecture should exhibit. The classifications are starting point for architectural thinking that have to be tailored based on domain expertise:

- In some cases, a User such as an airline pilot is an integral part of an aircraft's performance and success that involve End User lives.
- In other cases, the User may be *external* to the SYSTEM OR PRODUCT such as a tablet computer.

How the User is integrated into the operation of the SYSTEM or PRODUCT, mission duration, system safety, or HF drives different architectural frameworks and decisions. What an analysis does reveal is a set of capabilities that may be common to most SYSTEMS or PRODUCTS as well as some domain-specific and application dependent capabilities. The general list includes the following classifications at a *minimum* for architectural considerations:

- User ingress and egress portals
- User displays
- User controls
- Ancillary equipment and storage
- Accommodations
- Life support
- Data
- Consumables
- Expendables
- Monitoring
- C2
- Communications
- Sensors
- Discrete Input and Output (I/O)
- Exterior lighting
- Electrical
- Stabilization
- Directional control
- Emissions
- Propulsion
- Energy conversion
- Mission deliverables
- Cargo storage
- Maintenance portals

Observe how this list evolves from Architectural Attribute #1 (Section 26.3.6.11). Two key points about this list are as follows:

- First, since highly complex systems are part of the analysis, capabilities such as life support have visibility as a unique capability for some systems such as medical or space-based SYSTEMS. Obviously, a tablet computer does not require Propulsion or Life Support. However, it does require Life Support to replenish its battery power.
- Secondly, it is much easier to consider a broad range of SYSTEMS and PRODUCTS that include complex systems and employ a process of elimination to rule out those that are not applicable than to "rule in" missing capabilities.

26.4.2 The Conceptual Architecture Model

To better understand the context of the list, we can create a Conceptual Architectural Model as shown in Figure 26.7.

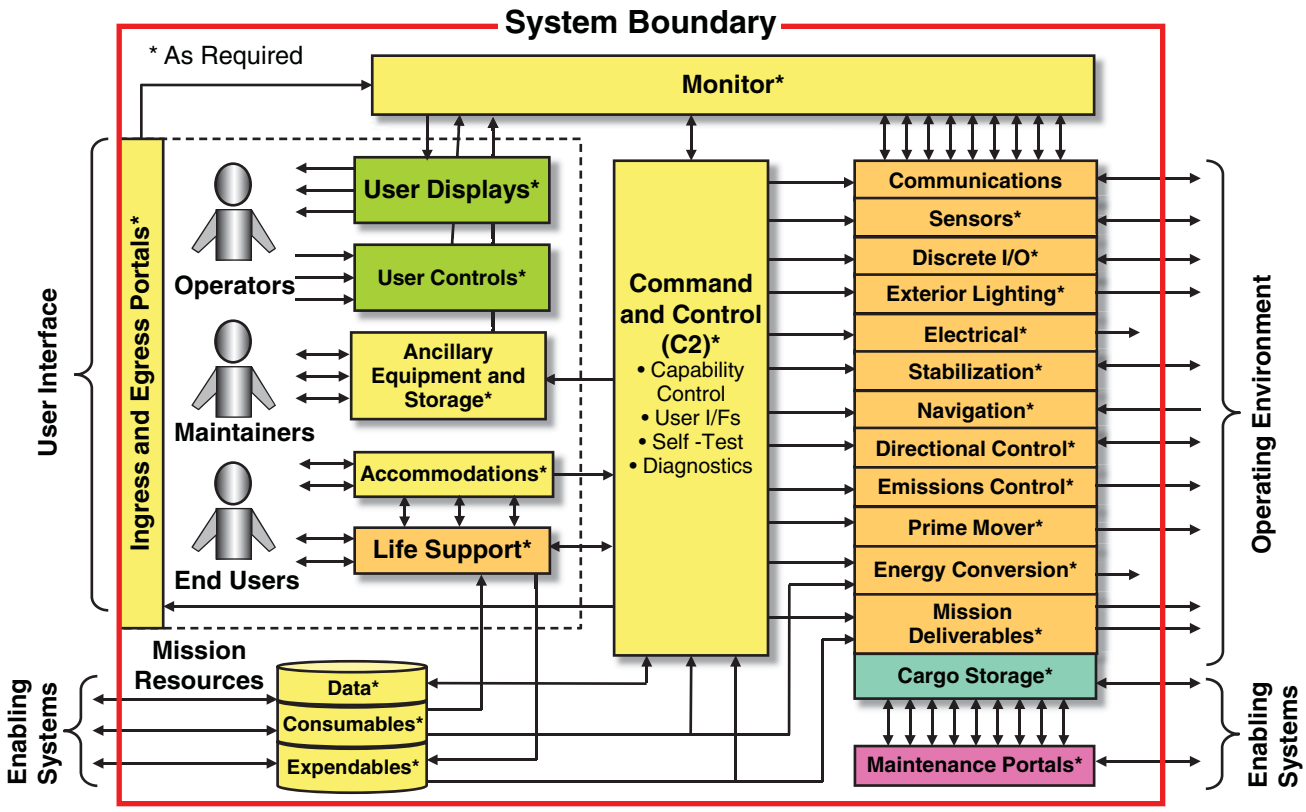


Figure 26.7 Conceptual Capability Architecture Model for System and Application-Dependent Tailoring

There are numerous ways of illustrating this information; Figure 26.7 is just one example.

Several points concerning the model are as follows:

1. The System Boundary is depicted by the thick outer rectangle.
2. Observe that the User and End Users are enclosed by "cabin" bounded by the dashed line with Ingress and Egress portals for entry and exit. For example, a commercial airline flight or your automobile with passengers. If your system or application does not require a User cabin, tailor it out and move the Users outside the System Boundary (Figure 8.1).
3. Prior to a mission, MISSION RESOURCES such as Data, Consumables, and Expendables are loaded into the System by ENABLING SYSTEMS.
4. During the mission, the User operator employs the Monitor capability that provides a current Situational Assessment of SYSTEM or PRODUCT operations and performance. Based on that information in conjunction with a mission plan, they can C2 SYSTEM or PRODUCT performance.
5. The right side includes a vertical set of blocks that are used to control SYSTEM performance. This is a simplified list created for graphical convenience with no specific order or relationship connections shown. That is another layer of detail that you should be able to create using an N x N (N2) chart such as Figure 8.11.
6. In an MC2 environment, Monitor continuously collects information such as OH&S for presentation to the User operator or maintainer. The vertical arrows extending out of the bottom right side of Monitor are an example. Likewise, C2 must control each of the capabilities on the right side of the figure. We will discuss these later.
7. The connection from C2 to Ingress and Egress Portals represents separate Monitor and C2 capabilities. For example, an automobile's C2 capability might lock the doors—Ingress and Egress—when the vehicle is placed in DRIVE or reaches a specific velocity. Hypothetically, Monitor might report the Open or Closed status of the door to C2 as a precondition for locking the door.

In summary, use the Conceptual Architecture Model shown in Figure 26.7 as a reference and initial starting point for considerations in System Architecture Development. The discussion earlier focuses on the SYSTEM Level. The reality is the model is analogous to the SE Process Model shown earlier in Figure 14.1. The analogy resides in its *recursive*

characteristic in which it can be applied to entities at any level of abstraction—PRODUCT, SUBSYSTEM, ASSEMBLY, SUBASSEMBLY, and so on. For example, it can be applied and tailored for large, complex systems or a Single Board Computer (SBC) within the SYSTEM.

A Word of Caution 26.1

Remember that you and your Enterprise are wholly responsible and accountable for your decisions in tailoring this model to meet the business domain and User needs of your SYSTEM or PRODUCT and its intended application. Employ the services of a qualified, competent System Architect that knows how to "Architect the System" rather than "Architect the Equipment Box" to advise, review, or create a System Architecture that may be appropriate for your application and situation.

26.4.3 Special Considerations: Conceptual Architecture Model

Now that we have established the Conceptual Architecture Model, there are some nuances that require special consideration.

26.4.3.1 Monitor versus Command and Control (MC2) Considerations

MC2 Capability Principle

MC2 is a capability, not a physical implementation.

Principle 26.15

Our discussions up to this point have highlighted the importance of MC2 in SYSTEMS or PRODUCTS. Remember that MC2 is a capability, not a physical implementation. Some SYSTEMS may use a single processor to provide both capabilities within an application. In contrast, complex systems may dedicate processors specifically to performing one of the capabilities such as an automobile's networks addressed later by Bannatyne (2009) concerning *Centralized* versus *Decentralized* Control Architectures.

26.4.3.2 Passengers versus Mission Deliverables versus Cargo Considerations

Regarding the Conceptual Architecture Model's Mission Deliverables capability, this title may appear to conflict with Passengers in the User Cabin. Mission Deliverables is an *abstracted* label representing items such as cargo, munitions, missiles, or countermeasures devices for military aircraft or prescription drugs for medical infusion devices. We elevate the importance of Passengers, who help generate revenue, rather than equate them to *inanimate* objects such as Mission Deliverables or Cargo. If there are terms that more appropriately describe the context in your business domain, consider using them.

26.4.3.3 Monitor versus C2 Operator Interfaces Considerations

Depending on the task workload of a processor, it may be debatable whether Monitor should: (1) interface directly with the User via the User Displays and User Controls capabilities or (2) pass this information to C2 to perform. Conceptually, C2 performs C2 of key *mission critical* capabilities such as Propulsion, Stabilization, Navigation, and so on. Multitasking and diverting C2 to perform what some perceive as lower-priority tasks such as displays instead of real-time, automatic, closed loop control of critical capabilities is a *misplaced* priority. The decision depends on a number of factors such as system optimization, cost, performance, and risk.

26.4.3.4 Self-Test and Diagnostics Accountability Considerations

A key question arises as to whether Monitor or C2 should be accountable for Self-Test and Diagnostics; it depends on the SYSTEM or PRODUCT. For example, Propulsion may have its own internal Self-Test and Diagnostics that run when initiated or continuously in the background to provide periodic OH&S updates to a common memory location for Situational Assessment Monitoring. The periodic updates alleviate the C2 task workload of having to interrogate a device by command for its status. Since Self-Test and Diagnostics often require initiation and C2 is accountable for commanding and controlling capabilities, this would be the logical choice.

26.4.3.5 Directional Control Considerations

Direction Control, like Mission Deliverables, is an abstracted title. The challenge is: Users steer ships and ground vehicles. Pilots control aircraft via Roll, Pitch, and Yaw (RPY) C2. Tablet computer Users navigate World Wide Web via links in lieu of "steering" the web via forward and backward links. *What term fits*? Ultimately, it comes down to Directional Control being the System objective.

26.5 ADVANCED SYSTEM ARCHITECTURE TOPICS

System Architecture Development requires more than simply selecting an *optimal* architecture (Figure 14.8) to provide capabilities to support User phases, modes, and states of operation. In highly advanced and complex systems such as space travel, military equipment, commercial transportation, medical devices, and others in which the realization of these may injure or endanger people's lives, SYSTEM Reliability becomes a major issue.

Fault Isolation & Containment Principle

Principle 26.16

When appropriate, every SYSTEM/ENTITY architecture should provide the capability

to detect, isolate, neutralize, and contain faults to *prevent* or *minimize* propagation if their effects to internal components or external systems.

How does System Reliability relate to SYSTEM *or* ENTITY *Architecture Development*? Ultimately, a SYSTEM or ENTITY's Architecture establishes the framework that will determine its:

- **System Reliability**—System reliability to complete its mission without disruption and successfully achieve its performance-based objectives.
- **Failure Detection and Containment**—Ensure System Safety through *detection* and *containment* or corrective action and recovery from a failure during a mission to prevent propagation. Prevent propagation of failure effects (Figure 34.17) into other parts of the SYSTEM or external systems such as the public and the environment. (Principle 26.16).

Several key points about System Reliability and Fault Tolerance are as follows:

1. **System Reliability**
   a. Observe that the term *failure* represents an outcome *condition*. Remember that a *failure*, by definition (Chapter 34), is a condition in which a component is *noncompliant* with its specified requirements. It does not mean that the component has *failed* in terms of self-destructed or destroyed. A component failure such as an engine may be *repairable* and *restorable* to a fully compliant state in terms of meeting its specification requirements.
   b. A *failure* is a state in which a potential *hazard* or *fault* such as a *latent defect*—design errors, flaws, or deficiencies; component integrity; and workmanship issues - has materialized with *consequences* such as degraded performance or destruction. Reason's (1990) Accident Trajectory Model (Figure 24.1) illustrates the materialization as an *incident* or *accident*.
2. **Fault Detection, Isolation, and Containment**—Although intended for a different context, as a "what if" exercise, imagine if the bold arrows – C2 and Responses - in Figure 9.6 penetrated each higher *level of abstraction* represented a *fault* propagating and proliferating into other areas of the SYSTEM.

Chapter 34 will address SYSTEM architectural redundancies network configurations (Figures 34.13 – 34.15).

This discussion brings us to a key concept in System Architecture Development concerning Fault-Tolerant Architectures, our next topic

26.5.1 Fault-Tolerant Architectures

Fault-Tolerant Architectures Principle

Principle 26.17 Fault-tolerant architectures are intended to improve the *robustness* of a System Design Solution to cope with *unanticipated* faults and scenarios, not as a substitute for a lack of System Development discipline in *eliminating latent defects*.

The challenge in developing any type of system is creating a System Architecture that is sufficiently *robust* to tolerate and cope with various types of internal and external faults. When confronted with these conditions, the design must be able to continue without significant performance degradation or catastrophic failure.

26.5.1.1 Understanding Faults and Fault Conditions

To illustrate the *fault tolerance* concept, Heimerdinger and Weinstock (1992) provide the illustration shown in Figure 26.8. The authors categorize faults in terms of:

- Observability—A fault is either known (*discovered*) or unknown (*undiscovered*).
- Propagation—A fault is either *propagating* or not.

Within the context of their SYSTEM Boundary, assume fault, f_1, is *undiscovered* and *not propagating*. In contrast, faults f_2 and f_5 are *undiscovered* and *propagating* beyond the boundary of their component. Observe that:

- Fault f_6 is *undiscovered* and has *propagated* beyond the System Boundary.
- Fault f_8 has been *discovered* and *propagated* beyond the System Boundary.

These scenarios represent the *conditions* and *pathways* that Fault-Tolerant architectures and design must address to preserve the integrity of the SYSTEM or ENTITY's mission, assuming its success is critical in terms of safety and mission completion.

Depending on SYSTEM design objectives, there are numerous methods of developing fault-tolerant architectures. Regardless of the method, a Failure Modes and Effects Analysis (FMEA)—Chapter 34—is critical for evaluating the SYSTEM or ENTITY Architecture for potential failure modes and effects including Single Failure Point (SFPs).

A more comprehensive expansion of the FMEA Concept includes a Failure Modes and Effects Criticality Analysis (FMECA) to identify and prioritize specific components for close attention. FMEAs and FMECAs assess and recommend *compensating provisions* such as design modifications and operating procedures (Figure 34.16) to mitigate failure conditions (Chapter 34).

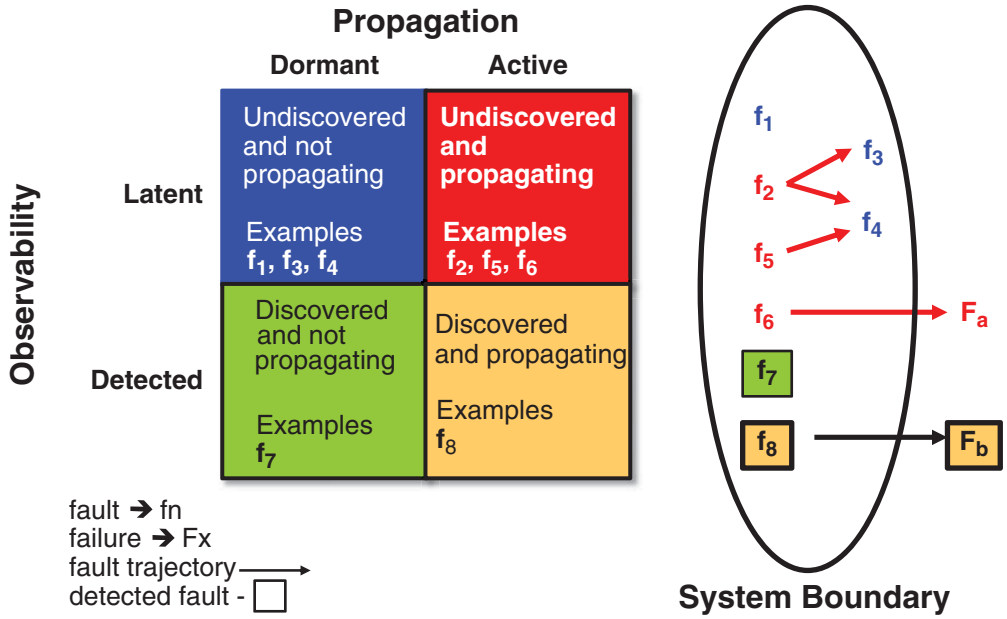


Figure 26.8 The Importance of Fault Containment: Observability and Propagation of Faults within a System and Beyond Its Boundary. Source: Heimerdinger and Weinstock (1992) Figure 3–2, p. 20 Fault Attributes. Used with Permission

Which architectural capabilities or components should be considered for fault-tolerant design? The answer to this question is dependent on answering four other critical questions:

1. Which components are *mission critical* (Chapter 34)?
2. What is the *reliability* (Chapter 34) of each mission critical component for a specified number of missions between maintenance?
3. What is the marginal cost for redundant components?
4. What are the trade-offs concerning cost, volumetric space, power, weight, and performance to create a fault-tolerant design for those components?

A word about *mission critical* components. An automobile's engine and tires are considered *mission critical*. However, it would be cost, space, and weight *prohibitive* to attempt to make those components *fault tolerant*.

Now, contrast these automobile components with a rocket launching astronauts into space. Although it may not have a fault-tolerant design, an automobile can pull over to the side of the road and wait for service; a rocket with ignited solid fuel has no such option. The result is a spacecraft launch vehicle architecture consists of a *fault-tolerant* design that includes multiple engines to improve its reliability.

There are several methods for accomplishing fault-tolerant architectural *frameworks*. These include approaches such as *centralized* versus *decentralized* control architectures and architectural redundancies. Let's begin with Centralized versus Decentralized Control Architectures. Advanced topics such as these topics illustrate why the "1 hour" System Architectures addressed in Mini-Case Study 26.1, are often *ill conceived* and *ineffective*.

26.5.1.2 Fault Detection and Containment A key question often arises: *if a fault is detected, how does the System contain it*? The answer depends on the SOI and the type of fault. First, let's delineate Enterprise Systems from Engineered Systems.

- *Faults* in Enterprise Systems include vulnerabilities in physical, operational, and communications security systems and detection of unauthorized entry and intrusions.
- Ground-based Engineered Systems – the PERSONNEL Element should be capable of responding to faults indicated by the EQUIPMENT Element via Situational Assessments such visual display caution and warning alerts or notifications and audible alarms. In those cases, the operator powers down the EQUIPMENT or safely pulls over to the side of the road.
- Airborne Engineered Systems, the User cannot exit the aircraft except in specific types of aircraft and emergency situations. In those cases, engine fire suppression systems, for example, are employed.

At this juncture in our discussion, you should recognize and appreciate the relationship between the Situational Assessment Monitor capability and mission success!

- Step 1 is dependent on the EQUIPMENT Element *detecting* a fault or the User observing faults such as smoke, blaze, leaks, odors, vapors, fumes, or wires burning.
- Step 2 requires containing, neutralizing, and preferably eliminating the fault to prevent it from propagating to other parts of the system.

Whatever the condition may be, Enterprise Systems need to ensure that all System Elements—PERSONNEL, EQUIPMENT, MISSION RESOURCES, PROCEDURAL DATA, SYSTEM RESPONSES, and FACILITIES—have robust *barriers* or *safeguards* in place to preclude the fault - hazard - from becoming an *incident* or *accident* using Reason's (1990) Accident Trajectory Model shown in Figure 24.1.

In general, faults in the EQUIPMENT Element might include overheating, broken wires or cabling, electrical shorts, fuel leaks, propulsion fires, computing errors, and so on. Faults might be induced by operator or maintainer errors such as slips, lapses, and mistakes (Reason, 1990). A later section on Other Architectural Considerations addresses some of these issues.

Once a fault has been detected, *containment* mechanisms include *quarantine* and *elimination* of computer viruses, *removal* of power and fuel sources, fire suppression, and other methods. For situations such as computing errors and other malfunctions, System Architectures should include provisions for User-initiated or automatic recovery actions (Figure 10.17) such as reboots or restarts, software reloads, hardware resets, and other methods.

26.5.2 Centralized Versus Decentralized Control Architectural Configurations

One of the key decisions in formulating, evaluating, and selecting an optimal architecture from a set of viable candidates by an AoA (Chapter 32) is the need to consider how processing workloads will be accomplished. Referring to Mini-Case Study 26.1, the reality is these conceptual decisions – presentation charts *unsubstantiated* by insightful analysis - often evolve into a "hope for the best outcome" when the SYSTEM or PRODUCT is designed and implemented. The AoA should be supported by Decision Support activities (Chapters 30–34) such as analyses, M&S, prototypes, and other methods that provide *quantitative* data to make *objective*, not *subjective*, decisions.

One of the key aspects of this Decision Support Process (Chapters 30 – 34) concerns system optimization, future growth and expansion, and other architecture objectives and requirements. In general, *how many computers are required to achieve these objectives*? From a workload and system optimization perspective, the answer resides in a concept concerning Centralized versus Decentralized Control Systems. Consider Example 26.4.

Example 26.4 **Centralized Control Example**

Imagine for a moment that you go into an airport ticketing area that has a dozen ticketing stations, not kiosks, for accepting passenger baggage and issuing boarding passes. There is a very long line of passengers waiting to be processed to meet their flights. Consider the following scenarios:

Scenario #1:

- *Situational Assessment*—Only one ticket agent is available to service 12 ticketing stations and determine who was next for service.
- *C2 Response*—A lone ticket agent has to run frantically back and forth from one ticketing station to another in an effort to service passengers lined up at each station.

Scenario #2:

- *Situational Assessment*—Each of the 12 ticketing stations is staffed with a ticketing agent with oversight performed by a supervisor.
- *C2 Response*—All 12 ticketing agents are simultaneously processing baggage and issuing boarding passes; each line is moving fast. If one ticketing agent is unable to perform, the passenger line is impacted but only by an 8.3% slowdown. Although each ticketing station may represent an SFP, overall, there are no SYSTEM-level SFPs per se with the exception of the computer system network that should have a backup.

Several key points emerge from these two scenarios:

1. Scenario #1 represents a poorly planned business model in which one ticket agent is attempting to keep up with an overwhelming workload that is simply impractical. The passengers are not happy; the ticketing agent is not happy. *What happens if the ticketing agent collapses due to exhaustion or the computer system fails* ? Centralized control is totally vested in the success or failure of one ticketing agent, an SFP.
2. Scenario #2 represents a better business model that unleashes the power of decentralized C2 processing. Ticketing agents process baggage and issue boarding passes at a reasonable rate. The passengers are satisfied with the workflow performance; the ticketing agents are happy as well.

The point is: where is the *optimal* trade-off (Figure 14.8) between cost, performance, and other factors?

- Scenario #1 represents highly *centralized* control with an SFP at reduced cost with *unhappy* customers and personnel.

- Scenario #2 represents *decentralized* control with no SFP but at increased labor expense with happy customers and personnel.

This discussion brings us to our next topic, Centralized versus Decentralized Control Architectures.

26.5.2.1 Centralized Control Architectural Configurations *Centralized control* architectures, as illustrated on the left side of Figure 26.9, consist of a single processor. For most applications, the mechanism directly controls capabilities by repetitive cycling from one task to another (Figure 31.3). Consider the following examples.

Example 26.5 **Centralized Control Example**

Video surveillance systems route multichannel, real-time camera video streams to a central command center staffed with a security guard to monitor the various channels of video. On detection of an intruder, the security guard vacates the post—*system vulnerability*—to investigate the event.

Example 26.5 illustrates an application of centralized MC2 vested in a single security guard, an SFP, tasked with "doing it all." From a security perspective, centralization can be *efficient* and *effective* if the Monitor capability remains in place and dispatches security guards to investigate incidents. Centralized control architectures are fine for many applications such as the examples cited previously. However, they do have limitations that represent SFPs.

As an SFP, some applications may require vast lengths of wiring to remote sensors that increase weight. For applications such as aircraft where an SFP may be a critical risk, additional weight, assuming it can be accommodated, translates into increased fuel consumption, increased fuel tank capacity, and reduces payload weight.

There are several approaches to solving this *problem space*. Example *solution spaces* include:

1. Avoid performance degradation and provide for expansion and growth by decentralization of processing functions.
2. Reduce weight by deploying decision-making mechanisms at key locations and interconnecting the mechanisms via networks.
3. Avoid risk due to potential SFPs by implementing appropriate types of redundancies.

26.5.2.2 Decentralized Control Architectural Configurations *Decentralized control architectures* allocate key control capabilities and deploy them via remote processing mechanisms that service Input/Output (I/O) requests as illustrated at the right side of Figure 26.9. The deployment may require:

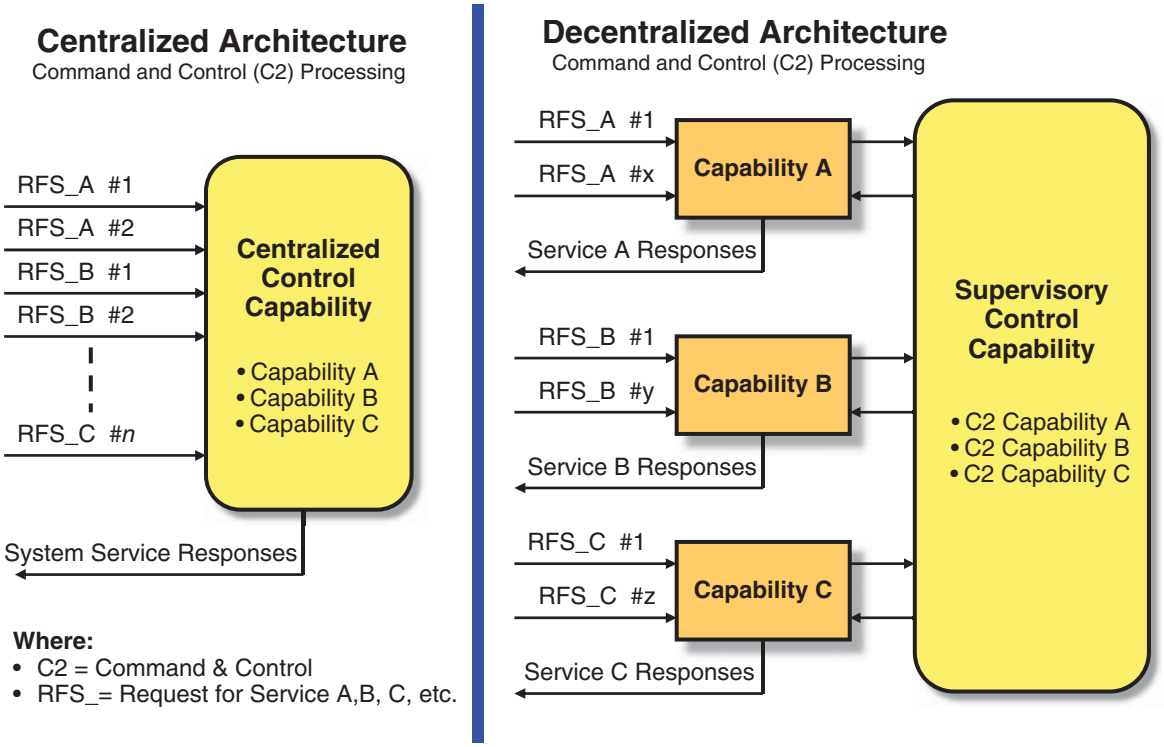


Figure 26.9 Comparative Examples of a Centralized versus Decentralized Architectures

1. Remote, dedicated processing to support a specific sensor or suite.
2. Retaining a central supervisory function to C2 each of the decentralized computing capabilities.

Depending on the mission and system application, *decentralized* capabilities might be: (1) physically located throughout a building or vehicle or (2) geographically separated across a country or globally.

The *decentralized* capabilities may be allocated to EQUIPMENT – HARDWARE OR SOFTWARE - hardware or software entities or dynamically assigned based on processing loads. As a result, overall system performance is improved but at the expense of adding more processors, which increases cost and risk. Consider the following automobile example.

Example 26.6 **Microcontrollers in Automobiles**

Automobile developers are often challenged to deal with the need for specialized capabilities to control various types of components while reducing gross weight to meet fuel economy regulations. For example, Pretz (2013) notes that the automobile cabling and wiring harnesses rank *third* in terms of component *cost* and *weight* behind the chassis and engine.

To meet these challenges, the Electrical System of an automobile consists of a distributed architecture of Microcontroller Units (MCUs) to control vehicle components such as Antilock Braking Systems (ABS), emission controls, actuators, and so on, which are then interconnected via networks to a central computer. Pretz (2013) notes that mid-priced cars today contain about 50 MCUs; higher-end cars contain about 140 MCUs.

As an example of a *decentralized* control system architecture, Bannatyne (2009) makes several observations concerning key features for present and future automobile network architectures:

- Multiple communications networks operate on a vehicle.
- Each network may consist of "subbus" networks connecting to various types of I/O devices such as sensors and actuators.
- Gateways are established to enable networks to communicate and share information.
- Critical capabilities such as "chassis control" are performed via a *redundant*, *fault-tolerant* network restricted to safety critical data.
- Passenger safety critical capabilities such as airbags need their own independent, highly robust network.
- Autonomous communications interface to minimize the need for the central computer to manage the interactions.

26.5.3 Architectural Configuration Redundancy

Another approach to solving the SYSTEM reliability issue is through a concept referred to as *architectural configuration redundancy*. *Uninformed* System Acquirers and Users will often state the following specification requirement:

> "Component X shall have a minimum of 'n' redundant backup systems."

In response to the requirement, System Developers will respond with *double* or *triple* redundancies for Component X. Their view is: "That's what the customer requested; that's what they get … end of story." The requirement illustrates three key points:

- First, you should readily recognize and appreciate. specifications specify *what* is to be accomplished, not *how* to physically implement components of a SYSTEM (Principle 19.6).
- Secondly, although RMA is not addressed until Chapter 34, component *redundancy* is a design action to meet a specification requirement. From an *academic* Engineering design perspective, unless it is shown by analysis that the SYSTEM or ENTITY Physical Design Solution *fails* to comply with a specification reliability requirement or is at best *marginal*, architectural configuration redundancy is not required. For marginal solutions, seasoned Engineering knowledge, wisdom, and experience must prevail.
- Thirdly, "redundant backup systems" is a case of naively connecting similar terms—*redundant* and *backup*—that refer to two different concepts to form a phrase. Redundant components are back-ups.

Redundancy is a critical part of space-based missions. NASA JPL's (2013) *Basics of Spaceflight* notes that most spacecraft contain "redundant transmitters, receivers, tape recorders, gyroscopes, and antennas." Redundancy includes software components as well. As an example, NASA's JPL (2009) described the Stardust Spacecraft as:

> "Virtually all spacecraft subsystem components are redundant with critical items cross-strapped. The battery includes an extra pair of cells. A software fault protection system is used to protect the spacecraft from reasonable, credible faults but also has resiliency built into it so many faults not anticipated can be accommodated without taking the spacecraft down."

Recognize that these descriptions represent implementations of redundant components. Redundant components exist due to the need for architectural configuration redundancy to achieve a level of performance for System Reliability.

So, *what is Architectural Configuration Redundancy and how does it solve the specification reliability requirement issue*?

Architectural configuration redundancy consists of configuring redundant components either as fully operational "on-line" or "off-line" devices during all or portions of mission phases or operation. There a three primary types of architectural redundancy: 1) *Operational*, 2) *Cold or Standby*, and 3) *k-out-of-n systems*.

26.5.3.1 Operational Redundancy *Operational redundancy* configurations employ backup device capabilities that are *energized* and/or *enabled* throughout the operating cycle of the SYSTEM or PRODUCT. For this type of redundancy, primary and redundant elements operate simultaneously for a total of "n" elements. Some people refer to this as *hot* or *active redundancy*. Consider the following example.

Example 26.7

Aircraft Systems Redundancy

An aircraft may require a *minimum* number of engines to be operating within specified performance limits for take-off, cruising, or landing; independent Inertial Navigation Systems (INS) required to continue a mission; or multiple, tandem train locomotives for specific geographic and loading conditions.

During SYSTEM operations, redundant capabilities or physical items can also be configured to operate *concurrently* and even share the loads. If one of the redundant items fails (fault detection), the other item(s) assumes the workload performed by the *failed* item and continues to perform (Principle 26.17) the required capability(ies). In most cases, the failed component is left in place and disabled, *powered down,* or *disabled*, assuming it does not interfere or create a safety hazard (Principles 26.16 and 26.17) until *corrective maintenance* (Chapter 34) is available and can be performed.

26.5.3.2 Cold or Standby Redundancy Backup Systems *Cold or standby redundancy* consists of items that are not *energized, activated,* or *configured* into the SYSTEM unless the primary item fails. If the primary item fails, the *standby* item is connected automatically or manually through direct operator intervention or by default within a period of time. In reality, *cold* or *standby* redundancies are effectively *backup* systems. NASA JPL (2011), for example, described its Dawn Spacecraft launched in 2007 as having "Automated onboard fault protection software (that) will sense any unusual conditions and attempt to switch to backups." Additional examples include the following.

Example 26.8

Cold or Standby Redundancies

Cold or standby redundancies include an emergency brake on an automobile, activating additional mass transportation vehicles (trains, buses, etc.) to support surges in consumer demand, emergency backup lighting switched on during a power failure, and backup power generation equipment.

26.5.3.3 "k-out-of-n" Systems Redundancy

t*"k-out-of-n" redundancy* is a hybrid redundancy approach whereby the SYSTEM consists of a total of "*n*" elements but only "*k*" elements (*k* out of *n*) are required to operate during specific phases of a mission. For example, an automobile requires a *minimum* of four tires to be operational *and* the spare tire available as a contingency.

You may ask: *What is the difference between operational redundancy and k-out-of-n redundancy?*

- *Operational redundancy* assumes that all physical components are integrated into the SYSTEM, may be operational—standby—but require *connection* and *engagement*. For example, a retail business may reassign on-site staff to specific customer service roles during peak business hours.
- *"k-out-of-n" redundancy* assumes that a minimum quantity of criteria or components that must be fully operational and performing for a specific phase of operation of a SYSTEM's mission as a prerequisite for launch, take-off, or landing. For example, a governing body that employs parliamentary procedures may require a *quorum* of its members defined by its constitution or charter to be present to conduct official business.

26.5.3.4 Redundancy Summary To illustrate the combinations of these types of redundancies, Mini-Case Study 26.4 addresses fault tolerance on the former Space Shuttle (NASA JPL's *Computers in Spaceflight*, 2013).

Mini-Case Study 26.4

Space Shuttle Fault Tolerance

"Fault tolerance on the Shuttle is achieved through a combination of redundancy and backup. Its five general-purpose computers have reliability through redundancy, rather than the expensive quality control employed in the Apollo program [61]. Four of the computers, each loaded with identical software, operate in what is termed the "redundant set" during critical mission phases such as ascent and descent. The fifth, since it only contains software to accomplish a "no frills" ascent and descent, is a backup. The four actuators that drive the hydraulics at each of the aerodynamic surfaces are also redundant, as are the pairs of computers that control each of the three main engines."

26.5.4 Component Redundancy Approaches

Once an architectural configuration redundancy concept is selected, the next step is to determine *how* to physically implement the concept. There are two approaches to component configuration redundancy: (1) *identical* components or (2) *unlike* components that are similar in function.

26.5.4.1 Like Redundancy Implementation *Like redundancy* is implemented with identical components such as vendor product models that are separately or both configured in an *operational* or *standby* redundancy configuration.

26.5.4.2 Unlike Redundancy Implementation Unlike redundancy consists of items acquired from and designed and developed by different vendors that provide identical capabilities, interfaces, and form factor constraints that comply with the System Acquirer's specification requirements.

Both implementation approaches offer advantages and disadvantages. If the items used for *like redundancy* are sensitive to certain OPERATING ENVIRONMENT characteristics, having identical items may not be a solution. If you purposely choose *unlike redundancy* and the same situation occurs, redundancy may only exist over a limited operating range if one item has higher reliability. If items identical only in function and performance are qualified over the required operating range, *unlike redundancy*, they may offer advantages.

26.5.5 Reducing Component Single Point of Failure (SPF) Risk

One method of reducing the item SPF risk requires operating *identical* or *redundant* items in several types of configurations. Redundancy type examples for electronic systems include:

- Processing redundancy
- Voted *k*-out-of-*n* redundancy
- Data link redundancy
- Service request redundancy

Let's explore each of these topics further.

26.5.5.1 Processing Redundancy Detection of a hardware processor failure or software fault may require dynamic reallocation of processing tasks to another processor to achieve fault tolerance (Principle 26.17). For example, SUBSYSTEMS A and B in Figure 26.10 both include redundant processing Components A and A' and Components B and B'. When a fault or failure occurs, processing switches over the back-up A' or B'-component.

26.5.5.2 Voted "1-out-of-n" Component Redundancy Some systems have redundant, peer-level processors that employ an *operational hot or active redundancy* configuration.

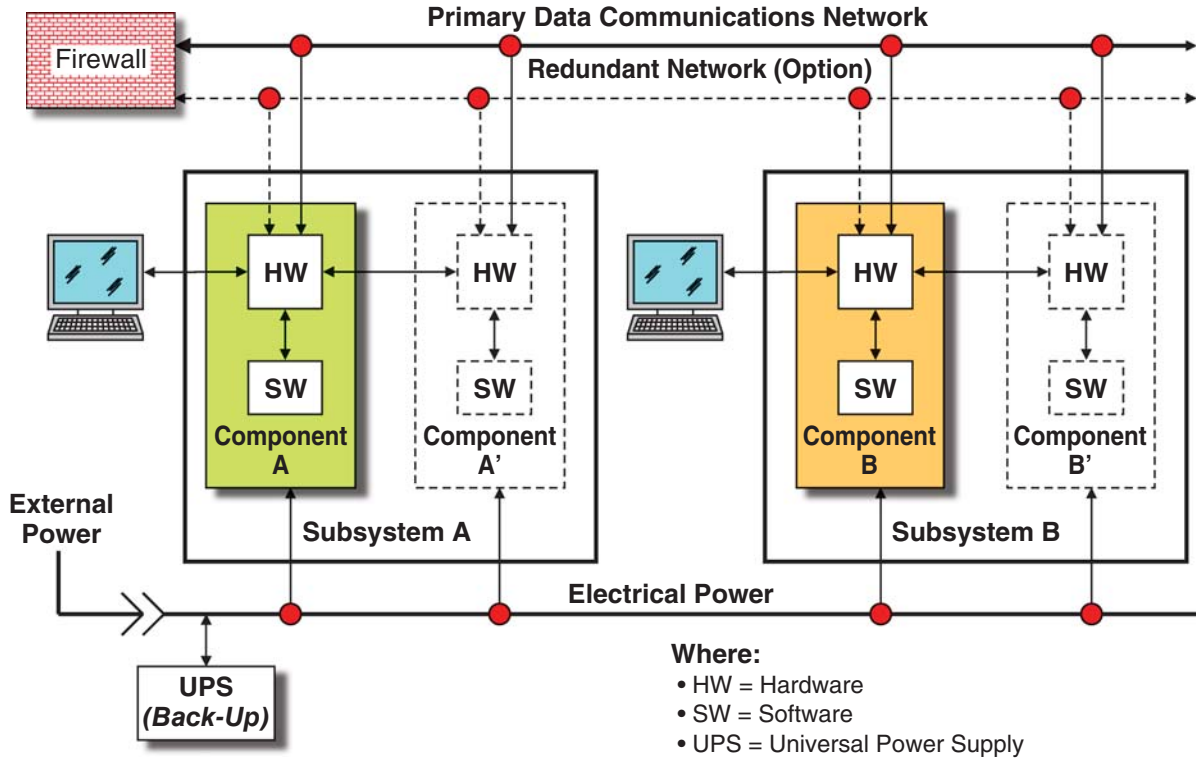


Figure 26.10 Redundant Components and Networks Example

Referring to Figure 26.10, SUBSYSTEMS A and B processing results are routed through a central decision-making mechanism (Figure 26.9 right side) such as software that determines if "*k*" out of "*n*" results agree. If "*k*" out of "*n*" results agree, transmit the results to a specific destination.

26.5.5.3 Data Link Redundancy There is an old adage: "Systems break at their interfaces." From an interface reliability perspective, this adage holds true. System developers often take great pride in creating *elegant* system designs that employ redundant processing components. Then, they connect the redundant components to a single external interface that is an SPF.

One way to avoid this problem is to employ redundant networks as shown in Figure 26.10. Obviously, if interconnecting items such as cables are placed in a stable/static position, not subjected to stressful OPERATING ENVIRONMENT conditions, and interfaced properly, there is a good chance that additional independent connections are *unnecessary* and can be *avoided*.

For applications that employ satellite links or transmission lines that may be switched periodically, it may be necessary to employ backup links, such as landlines or other telecommunications media, as a contingency.

26.5.5.4 Service Request Redundancy Some systems may be designed to transmit messages one or more times when requested. Using Figure 26.10 as an example, SUBSYSTEMS A and B automatically *retransmit* messages to each other. Others may issue service requests to repeat messages, acknowledgments, or data responses. As illustrated in Figures 10.5 and 10.6, this example is equally applicable to external systems.

26.5.5.5 Physical Connectivity Redundancy One of the ironies of developing architectural configurations and component redundancies is *violation* and *invalidation* of the redundancy concept via *poor* physical design. Figure 26.11 provides an illustration.

The upper part of the graphic illustrates a SYSTEM or ENTITY having item #1 with a redundant item #1 Backup. Observe the independent sets of I/O lines. Now, observe the lower part of the figure. Here, we have a similar scenario. Notice that we have a single connection into and out of the SYSTEM or ENTITY. The connection represents an SFP, which invalidates claims of having a true redundancy.

26.5.5.6 Differentiating Architectural Redundancy from Component Placement Redundancy Based on the preceding discussions, redundancy may appear to be an ideal solution. However, there is a difference between (1) architectural configuration redundancy and (2) component placement redundancy. You can create redundant systems. However, if you physically locate them next to each other and a major problem or catastrophic event occurs that destroys the backup system, *redundancy* is *irrelevant*. As an example, Figure 26.12 illustrates how physical design *placement* of components can *invalidate* architectural redundancy when a major event occurs. (NTSB/AAR-SO/06, 1990, p. 34). This point also illustrates the critical importance of collaborative integration across System Architecting, Engineering, and Design addressed in Figure 26.1.

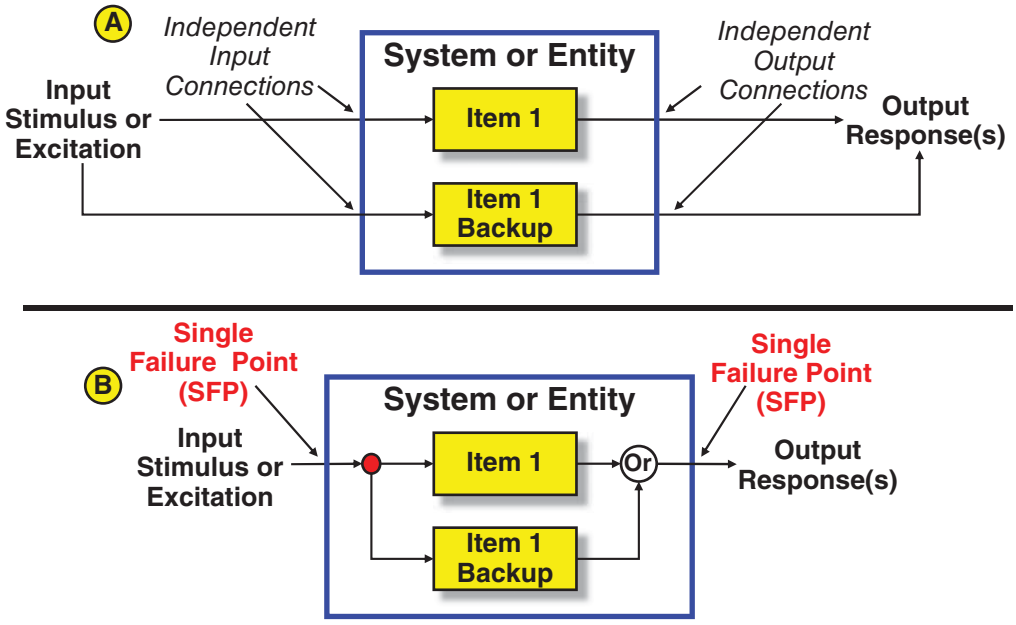
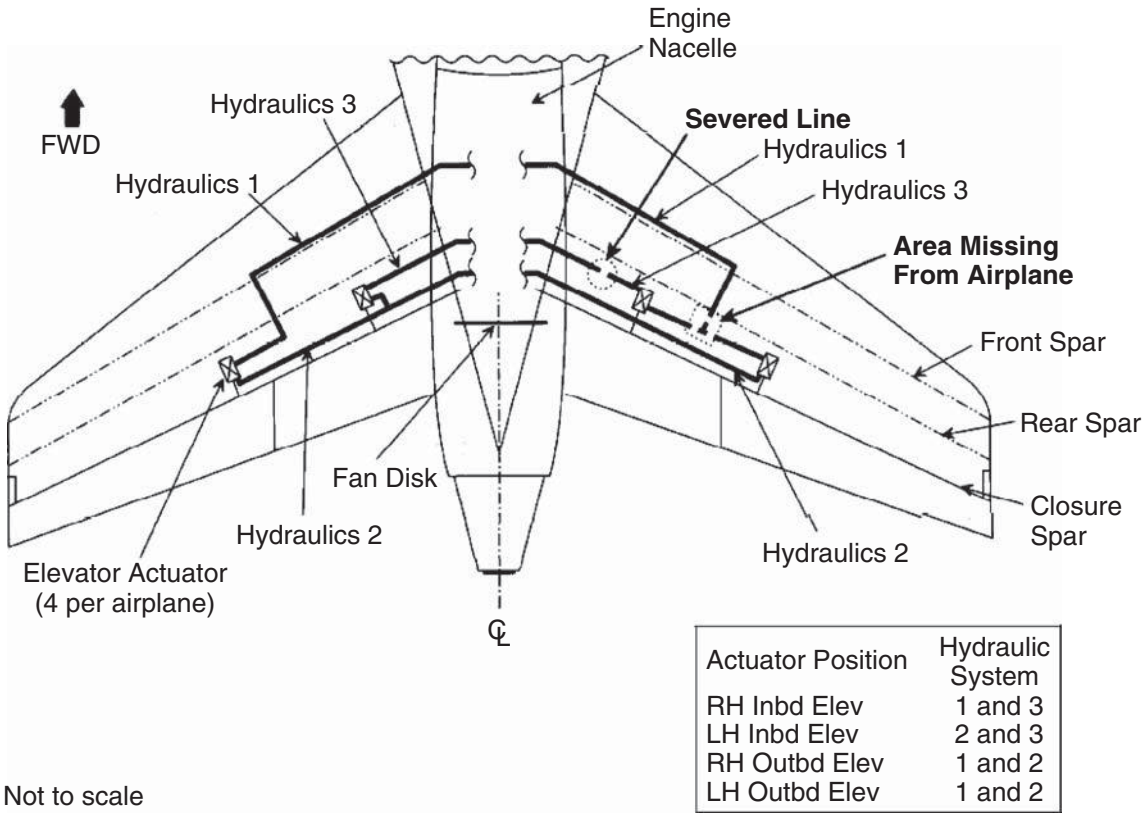


Figure 26.11 Comparative Examples Illustrating the Fallacy of Redundant Components with Single Point of Failure (SPF) Interface versus Redundant Interface Connections



Actuator Position	Hydraulic System
RH Inbd Elev	1 and 3
LH Inbd Elev	2 and 3
RH Outbd Elev	1 and 2
LH Outbd Elev	1 and 2

Figure 26.12 United Airline Flight 232 Accident—Architectural Redundancy versus Design. Source: NTSB/AAR-SO/06 (1990), National Transportation Safety Board (NTSB) Report, Figure 14 N1819U, planform of horizontal stabilizer hydraulic system damage

26.5.6 Network Architectures

Prior to the arrival of modern-day technologies, automobile, aircraft, ship, office buildings, and other types of SYSTEMS consisted of large bundles of copper wires that crisscrossed throughout the vehicle or facility. Although this was considered state of the art, the weight, installation, and maintenance expenses; performance of parallel versus serial communications; and other factors related to of copper wiring became increasingly prohibitive. Fortunately, the advancement of new technologies and standards related to serial, twisted-pair wiring, subsequently fiber-optic cables, and wireless communications overcame many of these issues, specifically through the *elimination* of non-standard, dedicated, specialized interfaces that were widely used.

Principle 26.18

Network Architecture Principle

Network architectures represent *potential solutions* to SYSTEM problems, not necessarily *the* only solution!

Today, networks are central to implementing System Architectures. Enterprises that employ the SDBTF-DPM Engineering Paradigm succumb to a cultural mind-set that everything requires a network. For example, we need to:

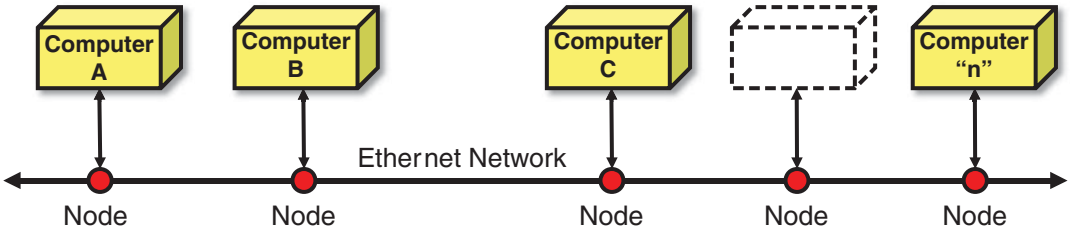
- Measure outdoor temperature—let's use a network!
- Know if the sun is shining—let's use a network!

The point here is: taking the quantum leap from requirements to physical implementation illustrated in Figure 2.3 … *before* … they understand *what* the system is expected to accomplish and whether a network is even appropriate. As an illustration of this mind-set, consider Mini-Case Study 26.5.

Mini-Case Study 26.5

Networks as Solutions for Electronics Problem

Corporation XYZ has won a System Development contract. During the first technical review with the System Acquirer and Users, Engineers parade a large wall-sized chart such as Figure 26.13 into the meeting conference room to impress *uninformed* customers. Verbose presentations describe how



What are the relationships and interactions between Computers A – D?

From / To	Computer A	Computer B	Computer C	Computer D
Computer A		??	??	??
Computer B	??		??	??
Computer C	??	??		??
Computer D	??	??	??	

Figure 26.13 Fallacy of Pre-Mature Leaps (Figure 2.3) to Physical Network Architecture Solutions Before Determination of the Optimal Solution

Computer C analyzes data collected from sensors by Computer D, sends it to Computer A for processing and formatting. Then, Computer A sends its results to Computer "n" for reporting the results to an operator.

In frustration, a User interrupts the presenter and asks: *Can anyone of you please explain*:

- Why are there so many computers?
- What the relationship is between Computers, A, C, F, and 10 others?
- Why do they exist?
- What capabilities does each provide?
- Where are they located?

Lacking a logical explanation, the System Developer's participants launch into a discussion that is often *conflicting* and *confusing*. Then, comes the "cover all cases" comment such as "If you understood networks, you would know the answers to those questions." Imagine making that statement to your customer as a confidence builder for your SYSTEM!

It's up to you and your project to provide a logical solution and explanation with supporting materials that makes sense! It's your *responsibility* to *articulate* and *communicate effectively* so your customer *understands*!

The point here is this: Figure 26.13 is fine as an introductory overview graphic for illustrating component networks. The problem is this graphic is *the* only architecture considered and presented without supporting rationale justifying its selection. Therein lies the problem as indicated by the customer's frustration.

26.5.6.1 Client-Server Architectures For system applications that require desktop or Web-based access to a central repository of information, client-server architectures are employed. In this case, a processor is dedicated to processing client requests for entering or searching for data, retrieving the data from a central repository, and disseminating the data to the client. Applications such as this, which include internal Enterprise intranets and Web-based sites, are helpful for contract projects that need to provide access to project and contractor data to authorized Acquirer/Users, System Developers, subcontractors, and vendors.

26.5.6.2 Service-Oriented Architectures (SOAs) SOAs represent a Software Engineering (SwE) architectural approach based on (1) platform independence, (2) reusable components that may originate from different vendors, and (3) implemented via Information Technology (IT) components. Due to the specialization unique to SwE, please refer to texts on the subject matter.

26.5.7 Other Architectural Considerations

Developing a System Architecture to provide capabilities to support all phases of the mission is only part of what is required. The architecture must also include other key considerations. These include:

- Open standards
- Power source architectural considerations
- Data storage during power loss

- Sustainment of operations and equipment resulting due to power loss
- Power quality considerations
- ES&OH architectural considerations
- Fire detection and suppression architectural considerations
- System security architectural considerations
- Remote visual inspections

26.5.7.1 Open Standards One of the key decisions in System Architecture development is usage of *open standards* versus dedicated or proprietary standards, especially concerning development and life cycle cost considerations. Every proprietary interface requires specialized expertise, tools, and equipment that may not be readily available. Exploit open standards such as file formats, hardware, protocols, and programming languages.

26.5.7.2 Power System Architectural Considerations

Architectural Power and Backup Principle

Principle 26.19 Factor energy and backup power source considerations for *normal*, *abnormal, and emergency* operations (Figure 19.5) and conditions into every System Architecture.

The preceding discussions highlight methods to enhance the fault tolerance of systems and products. If they are electrically powered, no matter how elegant the redundancy solution, it only works when power is applied. A *loss of power* involves several issues:

- **Issue #1**—Safe evacuation and egress of PERSONNEL from facilities to designated gathering points some distance away from the facility prevents injury or loss of life.
- **Issue #2**—Safe storage of *critical mission* and *system data* immediately following the event prevents data loss.
- Issue **#3**—Sustain power to *critical processes* that must process to completion and place the system in a *safe* mode.

26.5.7.3 Sustaining Operations and Equipment Resulting from Loss of Power When a power loss event occurs, systems require a finite amount of time to store mission and system data. To ensure a continuation of power for a specified time, rechargeable batteries or an Uninterruptible Power Supply (UPS) offers potential solutions. Depending on mission and system application, alternative power solutions include external fuel-based generators, solar panels, fuel cells, and other technologies.

26.5.7.4 Data Storage During Power Loss When critical operations are being performed and the SYSTEM experiences a power failure, it is important to be able to complete any processing operations and Mission and System performance data storage. Depending on space and weight margins, Uninterruptable Power Systems (UPS) may need to be considered.

26.5.7.5 Sustainment of Operations and EQUIPMENT Resulting Due to Power Loss Some sensitive EQUIPMENT may have moving parts or appendages that may be vulnerable to mechanical damage related to the power failure or power-up when power is restored. Consideration should be given to use of the UPS as a power source to place this EQUIPMENT into a Safe Mode to prevent consequential damage.

26.5.7.6 Power Quality Considerations Another factor that requires architectural consideration is power quality. Power surges, brownouts, overvoltage, noise, and stability conditions wreak havoc with some systems that require power conditioning. So, ensure these considerations are fully addressed by the architecture within available resource constraints.

26.5.7.7 ES&OH Architectural Considerations

Architectural ES&OH Principle

Principle 26.20 Factor considerations for PERSONNEL and the public's Environmental Safety and Occupational Health (ES&OH) into every System Architecture.

System architecting, in general, tends to focus on EQUIPMENT Element Architectures rather than the effects of the EQUIPMENT Element on the Users, the public, and the NATURAL SYSTEMS ENVIRONMENT. Therefore, when evaluating a SYSTEM Architecture, the System Architect and others should factor in ES&OH considerations.

ES&OH Consideration Examples

Example 26.9 At a *minimum,* considerations of the effects of EQUIPMENT Element by-products on the User(s), public, and environment should include the following considerations such as:

- Toxic and hazardous materials.
- Ingress and egress access.
- Thermal effects
- Atmospheric effects
- Environmental spills and remediation

26.5.7.8 Fire Detection and Suppression Architectural Considerations

Principle 26.21

Fire Detection and Suppression Principle

Where appropriate, every system architecture should incorporate safety features for fire detection and suppression.

Another key architectural consideration is fire detection and suppression systems. Since PERSONNEL, EQUIPMENT, and FACILITY Elements safety are paramount, the System Architecture should include features that enable a rapid response when smoke or fires are detected including suppression systems that eliminate the source of the fire following PERSONNEL Element evacuation. Consider the following example.

Example 26.10

Aircraft Fire Suppression Methods

SKYbrary (2013) notes fire suppression considerations for aircraft such as:

- Portable fire extinguishers in the cabin and flight deck
- Cargo hold fire extinguishing systems
- Fire bottles in engine compartments
- Automatic toilet waste bin fire extinguishers

Architectural considerations require more than simply remembering to include these. Operational issues become a factor in every mission phase of operation.

As expressed in Principle 26.21, System Architecture considerations should include safety considerations such as cautions and warning notifications, alerts, and alarms (Principles 24.16 and 24.17).

26.5.7.9 System Security Architectural Considerations

Principle 26.22

Architectural Vulnerabilities Principle

Assess SYSTEM vulnerabilities and factor SYSTEM security and protection considerations into every system architecture.

For systems that involve sensitive or classified data, system security should be a key architectural consideration. This includes *reasonable measures* for physical security, operational security, communications security, personnel security, and data security.

26.5.7.10 Remote Visual Situational Assessments

Principle 26.23

Architectural Remote Inspection Principle

Every system architecture should include Situational Assessment considerations not only for *normal* but also *abnormal* operations via remote inspections.

System Architectures often focus on the ideal world of conducting missions and the importance of redundant architectural configurations to ensure the mission is completed. Typically, the system is instrumented with electronic sensors and software to indicate if there are problems. However, *what is the value of an error code if you cannot correct the problem other than knowing that it occurred and logging the event*? Obviously in the case such as aircraft, or train "black boxes," they provide insights into key parameters leading up an incident or accident. Unfortunately, that is an "after the fact" analysis after injury or harm has been done to those involved. What is often neglected is: *what happens if the system encounters a problem that cannot be resolved by on-site inspection and maintenance*? For example, NASA JPL sends a probe or rover to Mars.

Space-based systems often have cameras such as NASA's Space Shuttle and the International Space Station. In general, cameras are used to facilitate astronaut spacewalks and repairs. Unfortunately, this was a lesson learned addressed in Mini-Case Study 26.3 following the Space Shuttle Columbia accident. *But suppose in the case of the Mars rover there is no astronaut onboard to conduct maintenance inspections and repair services*?

One of the fascinating features of the Mars Curiosity Rover is the recognition by JPL of the need for cameras to perform a Situational Assessment. For example, a notional architecture and design would use cameras to "look ahead" to drive over terrain. However, suppose one of the Rover's wheels encounters an obstruction or has a mechanical wear problem. You need to know "why" and certainly more than "look ahead" in the distance views. Mini-Case Study 26.6 provides a brief description of how NASA's JPL solve the architectural and design problem.

Mini-Case Study 26.6

Mars Rover – Remote OH&S

Study Figure 26.14 - This view of the three left wheels of NASA's Mars rover Curiosity combines two images that were taken by the rover's Mars Hand Lens Imager (MAHLI) … The camera is located in the turret of tools at the end of Curiosity's robotic arm.

The main purpose of Curiosity's MAHLI camera is to acquire close-up, high-resolution views of rocks and soil at the rover's Gale Crater field site. The camera is capable of focusing on any target at distances of about 0.8 inch (2.1 centimeters) to infinity, providing versatility for other uses, such as views of the rover itself from different angles.

In summary, ensure that System Architectures include provisions for remote inspection in harsh environmental conditions where applicable.

26.5.7.11 SoS Architectures This text addresses the conceptualization, formulation, and development of Enterprise

Figure 26.14 Photo Illustrating How the Mars Hand Lens Imager (MAHLI) Camera is Used to Assess the Condition of the NASA JPL Mars Science Laboratory Curiosity Rover Wheels (NASA JPL, 2012)

and Engineered Systems deeply rooted in a multi-level, analytical, SYSTEM decomposition strategy. Today, SE involves integration of autonomous systems into higher-level Systems of Systems (SoS). Several key points are as follows:

1. Observe the two different concepts—SYSTEM development through a process of (1) traditional *top-down*, iterative, analytical *decomposition* from the abstract to the physical versus (2) SYSTEM development through a process of *integrating existing and new systems into an SoS*.
2. Whereas Enterprise and Engineered Systems discussed in this text have physical implementations and configurations, an SoS can range from a loose federation of business entities to social, political, economic, academic, and other types of complex organizations that are in a state of continuous change and evolution.
3. An SoS, which is comprised of existing, *autonomous* systems, exemplifies the concept of *emergence* (Chapter 3) by leveraging the capabilities of lower-level systems through integration to create a HIGHER-ORDER SYSTEM to accomplish missions that are beyond individual system capabilities.
4. If the higher-level SoS or one of its autonomous systems ceases to exist, the federation continues operate and perform their Enterprise missions.

SoS is a specialized topic and evolving field of study. To learn more about SoS, refer to textbooks dedicated to the topic.

26.6 CHAPTER SUMMARY

In summary, we introduced key practices in SYSTEM and ENTITY Architecture Development. Key points from our discussions include:

1. Recognition that System Architecture Development requires:
   a. More insight than simply creating a 1-hour presentation block diagram (Mini-Case Study 26.1)
   b. Capture and reconciliation of Stakeholder *views, viewpoints*, and *concerns* (Figure 26.4) that are addressed in an Architectural Description (AD).
   c. Formulation and development of a set of candidate architectures to be evaluated with models, simulations, prototypes, and final selection of an optimal architecture by an AoA (Chapter 32).
   d. Collaboration with Engineering and Design.
2. Understanding:
   a. What a SYSTEM or ENTITY Architecture is.
   b. That a SYSTEM or ENTITY Architecture expresses a framework of interconnected capabilities or components without regard to frequency of usage that can be configured into combinational sets that may be unique to and support specific phases, modes, and states of operation.
3. Understanding the System Architect's role and who performs it in small, medium, and large projects.
4. Understanding and differentiating the interrelationships between *architecting* versus *engineering* versus *designing* a system, product, or service.

5. An AD requires evolution of Four Domain Solution architectures—Requirements, Operations, Behavioral, and Physical (Figure 26.4).
6. An architecture is characterized by the following attributes:
   a. Assigned ownership
   b. Unique document ID
   c. Context
   d. Framework
   e. Domain Views—Requirements, Operations, Behavioral, and Physical
   f. Validity
   g. Completeness
   h. Consistency
   i. Traceability
   j. Capabilities
7. Every SYSTEM or ENTITY Architecture (Figure 26.6) should express how it will:
   a. Monitor Mission and System performance to present a Situational Assessment for its Users for operational decision-making
   b. Enable Users to C2 Mission and System performance
8. The Conceptual Architectural Model (Figure 26.7) serves as a general template for developing and assessing SYSTEM or ENTITY Architectures.
9. System Architecting requires insightful consideration of: 1) System Reliability and 2) fault detection, isolation, neutralization, and containment (Figure 26.8) and possible recovery (Figure 10.17).
10. The SYSTEM or ENTITY Architecture establishes the configuration framework for computing its reliability (Chapter 34).
11. System reliability can be improved via Architectural Configuration Redundancy Approaches such as:
    a. Operational redundancy
    b. Cold or standby redundancy
    c. "*k*-out-of-*n*" redundancy
12. Two types of Component Redundancy Implementation Approaches:
    a. Like component redundancy
    b. Unlike component redundancy
13. Key SYSTEM or ENTITY Architecture considerations that are often neglected include:
    a. Open standards
    b. Power source architectural considerations
    c. Data storage during power loss
    d. Sustainment of operations and EQUIPMENT resulting due to power loss
    e. Power quality considerations
    f. ES&OH architectural considerations
    g. Fire detection and suppression architectural considerations
    h. System security architectural considerations
    i. Remote visual Situational Assessment inspections
14. Analytical redundancy is different from physical implementation redundancy, especially when redundant components are physically located next to each other leaving both vulnerable to propagation of the other's failures (Figure 26.8) effectively nullifying the concept (Figure 26.12).

26.7 CHAPTER EXERCISES

26.7.1 Level 1: Chapter Knowledge Exercises

1. What is an architecture?

2. What is an AD?

3. What are the differences between User *views*, *viewpoints*, and *concerns* and provide examples?

4. What roles do the System Architect and SE perform regarding User *views*, *viewpoints*, and *concerns?*

5. What are the differences and interrelationships between architecting versus engineering versus designing?

6. Is System Architecting a 1-hour block diagram exercise? If not, explain why?

7. What is a fault-tolerant architecture and what is its purpose?

8. Who performs a System Architect role on small, medium, and large projects?

9. Pick one of the following SYSTEMS or PRODUCTS listed below. Describe: 1) what information the Monitor capability provides as a Situational Assessment to the User and 2) what controls are available for the User to Command & Control (C2) it:
   a. Aircraft
   b. Automobile
   c. Desktop, laptop, or tablet computer
   d. Medical blood pressure device
   e. Home heating and cooling system.

10. Identify three examples of each of the types of system architectures listed below:
   a. Centralized architecture
   b. Decentralized architecture

11. Identify an example of each of the types of redundancies listed below:

a. Operational or Hot Standby redundancy
b. Cold or standby redundancy
c. "*k*-out-of-*n*" redundancy

12. Identify an example of each of the types of component implementation redundancies listed below:
a. Like redundancy
b. Unlike redundancy
c. Data link redundancy
d. Physical connectivity redundancy (Figure 26.12)

26.7.2 Level 2: Knowledge Application Exercises

Refer to www.wiley.com/go/systemengineeringanalysis2e

26.8 REFERENCES

Bannatyne, Ross (2009), "Microcontrollers for the Automobile," Transportation Systems Group, Motorola Inc., City: *Micro Control Journal*.

Buede, Dennis M. (2009), *The Engineering Design of Systems: Models and Methods*, 2nd Edition, New York: Wiley & Sons, Inc.

Defense Acquisition University (DAU) (2012), *Glossary: Defense Acquisition Acronyms and Terms*, 15th ed. Ft. Belvoir, VA: Defense Acquisition University (DAU) Press. Retrieved on 3/27/13 from http://www.dau.mil/pubscats/PubsCats/Glossary%2014th%20edition%20July%202011.pdf.

Demarco, Tom (2009), "Software Engineering: An Idea Whose Time Has Come and Gone," *IEEE Software*, New York: IEEE Computer Society.

Forrest, Jeff (2013) *The Space Shuttle Challenger Disaster: A failure in decision support system and human factors management*. Denver, CO: Metropolitan State College of Denver. Retrieved on 11/5/13 from http://dssresources.com/cases/spaceshuttlechallenger/index.html.

Heimerdinger, William L. and Weinstock, Charles B. (1992), *A Conceptual Framework for System Fault Tolerance*, Technical Report CMU/SEI-92-TR-033/ESC-TR-92-033, Software Engineering Institute (SEI), Pittsburgh: Carnegie-Mellon University (CMU).

Hilliard, Rich (2012), *Architectural description template for use with ISO/IEC/IEEE 42010:2011*, Geneva: International Organization for Standardization (ISO). Retrieved on 10/30/13 from http://www.iso-architecture.org/ieee-1471/templates/42010-ad-template.pdf.

IEEE 1471-2000 (2000), *IEEE Recommended Practice for Architectural Description of Software-Intensive Systems*, *New York: Institute of Electrical and Electronic Engineers (IEEE).*

ISO/IEC/IEEE 42010:2011 (2011), *Systems and Software Engineering - Architecture Description*, Geneva: International Organization for Standardization (ISO).

Leveson, Nancy and Turner, Clark S. (1993), "An Investigation of the Therac-25 Accidents," *IEEE Computer*, Vol. 26, No. 7, New York, NY: IEEE Computer Society.

MIL-STD-499B Draft (1994), Military Standard: *Systems Engineering*, Washington, *DC: Department of Defense (DoD).*

NASA (2003), *Columbia Accident Investigation Board (CAIB) Report*, Volume 1, Washington, DC: NASA. Retrieved on 11/2/13 from http://spaceflight.nasa.gov/shuttle/archives/sts107/investigation/CAIB_medres_full.pdf

NASA (2005), "In-Flight Inspection and Repair," *Identifying & Repairing Damage In-Flight*, Washington, DC: NASA. Retrieved on 11/2/13 from http://www.nasa.gov/pdf/186088main_sts114_excerpt_inflight_repair.pdf.

NASA (2013), *Computers in Spaceflight: The NASA Experience*, Chapter Four: Computers in the Space Shuttle Avionics System, Washington, DC: NASA. Retrieved on 11/8/13 from http://history.nasa.gov/computers/Ch4-4.html.

NASA JPL (2009), *Stardust Spacecraft* web page, Pasadena, CA: NASA Jet Propulsion Laboratory. Retrieved on 11/8/13 from http://stardust.jpl.nasa.gov/mission/spacecraft.html.

NASA JPL (2011), *Dawn Spacecraft - Mission Overview* web page, Pasadena, CA: NASA Jet Propulsion Laboratory. Retrieved on 11/8/13 from http://www.nasa.gov/mission_pages/dawn/spacecraft/index.html.

NASA JPL (2012), Curiosity Rover "Wheels and a Destination (photo)," Mars Science Laboratory, Image Credit: NASA/JPL-Caltech/Malin Space Science Systems, Sept. 10, 2012, Pasadena, CA: Jet Propulsion Laboratory (JPL). Retrieved on 3/27/15 from http://mars.jpl.nasa.gov/msl/images/wheel_image-br2.jpg

NASA JPL (2013), *Basics of Space Flight*, Section II Flight Projects – Chapter 11 Typical On-Board Systems, Pasadena, CA: NASA Jet Propulsion Laboratory. Retrieved on 11/8/13 from http://www2.jpl.nasa.gov/basics/bsf11-4.php.

NTSB/AAR-SO/06 (1990), Aircraft Accident Report: *United Airline Flight 232, McDonnell Douglas DC-1040, Sioux City Gateway Airport, Sioux City, IA*, November 1, 1990, Washington, DC: National Transportation Safety Board (NTSB)

OMG SysML™ (2012), *Systems Modeling Language (OMG SysML™) Specification*, Version 1.3, Needham, MA: Object Management Group (OMG). Retrieved on 11/1/13 from http://www.omg.org/spec/SysML/1.3/PDF.

Pretz, Kathy (2013), *Fewer Wires, Lighter Cars: IEEE 802.3 Ethernet standard will reduce the weight of wires used in vehicles* web page, New York, NY: IEEE the Institute.

Reason, James (1990), *Human Error*, Cambridge, UK: Cambridge University Press.

Rogers Commission (1986), Report of the PRESIDENTIAL COMMISSION on the Space Shuttle Challenger Accident, Washington, DC. Retrieved on 9/23/13 from http://history.nasa.gov/rogersrep/genindex.htm.

SEVOCAB (2014), Software and Systems Engineering Vocabulary, New York, NY: IEEE Computer Society. Accessed on 5/19/14 from www.computer.org/sevocab

SKYbrary (2013), *Aircraft Fire Extinguishing Systems* web page, Retrieved on 11/11/13 from http://www.skybrary.aero/index.php/Aircraft_Fire_Extinguishing_Systems.

Wikipedia (2013), *Service-Oriented Architecture* web page, San Francisco, CA: Wikimedia Foundation, Inc. Retrieved on 11/11/13 from http://en.wikipedia.org/wiki/Service_Oriented_Architecture.

27

SYSTEM INTERFACE DEFINITION, ANALYSIS, DESIGN, AND CONTROL

Principle 27.1

Interfaces as System Components Principle

Interfaces, as components of every Engineered SYSTEM or PRODUCT, enable emergent behavior to reveal itself.

Principle 27.2

System-Interface Design Priority Principle

In some cases, a SYSTEM or ENTITY's design has priority and drives interface design; in other cases, interface design has priority and drives SYSTEM or ENTITY design.

Interface definition, analysis, design, and control in some Enterprises are often treated as *secondary* activities to SYSTEM or ENTITY design. Unfortunately, this is an Engineering paradigm based on physical components being represented by boxes. In contrast, an interface is often viewed as "nothing more than a thin line on a drawing"; they are just … interfaces. In fact, interfaces such as cables, wiring, and mechanical linkages, which interconnect ENTITIES are Engineered System components. If we model and simulate a SYSTEM or PRODUCT, each interface cable or wiring is modeled as an ENTITY with a characteristic transfer function.

Since interfaces are an integral part of System Architecting (Chapter 26), Chapter 27 complements and explores interface definition, analysis, design, and control. Engineers often think of interfaces as just *lifeless* wires and cables, mechanical connections or restraints, and thermal insulating or conductive materials, which serve no function. Interface design is just another example of Figure 2.3 in action. From a Systems Engineering (SE) perspective, Engineers *ignore* the fact that interfaces are characterized by transfer functions that have *operational, behavioral,* and *physical* attributes that enable electrical and mechanical connectivity, data communications, and so forth. As a result of the mindset, Engineers take a *quantum leap* from specification requirements to a physical interface solution without due consideration of the operational and behavioral of the connection.

To overcome these shortcomings, our discussions introduce an interface methodology for defining, analyzing, and designing interfaces. Systems Engineers (SEs) often think of the SE Process (Figure 14.1) and the System Capability Construct (Figure 10.17) as applying only to a SYSTEM or ENTITY; they apply to interfaces as well. Every SYSTEM or ENTITY has phases of operation, modes and states, as well as behavioral and physical characteristics. Through *inheritance* of derived requirements, interfaces support achievement of those operational, behavioral, and physical characteristics.

Our discussions explore some of the project aspects such as: *Who owns and controls an interface*? *How do you determine ownership and control*? Where interface ownership, control, and accountability are ill-defined, the truth emerges on entry into System Integration, Test, and Evaluation (SITE) in two ways as follows:

1. ASSEMBLIES, SUBSYSTEMS, PRODUCTS fail to integrate due to the lack of *compatibility* and *interoperability*—poor interface design and coordination.

System Engineering Analysis, Design, and Development: Concepts, Principles, and Practices, Second Edition. Charles S. Wasson.

Companion Website: www.wiley.com/go/systemengineeringanalysis2e

2. Missing cables and wiring—Engineering focuses on the design of the SYSTEM, PRODUCT, SUBSYSTEM, ASSEMBLY, including interfaces. Yet, no performance accountability for delivery of those interfacing cables.

The preceding paragraphs mimic the traditional SE theme of "…everyone define your interfaces." In other words, make sure to "connect lines to all the architectural boxes."

Chapter 27 sifts to a new paradigm concerning how we view, think about, Engineer, and orchestrate the development of system or product interfaces. Interfaces are more than "lines connecting boxes." The reality is: interfaces are actually "components" of a system or product. As you will discover, interfaces have transfer functions that model the performance-based outcome of an event such as an automobile crash.

Traditional Engineering mindsets think of interfaces in terms of Normal Operations (Figure 19.5) such as Cockburn's "Main Success Scenario" (Principle 5.21) in which compatibility and interoperability are fully achieved. For example, supporting structures under tension or compression, communicating data, providing security, electrical shielding, and so forth. However, what happens when external forces cause the interface to fail resulting in injury or death?

In the case of an automobile, traditional Engineering mindsets tend to think of "reactive" measures such shock absorbers and air bags to buffer passenger compartments as a means of withstanding impacts. Observe that we said "reactive" measures. "What if" Engineering:

1. Applied Systems Thinking (Chapter 1) to interface problem-solving and solution development and took a "proactive" approach to controlling how an interface performed during: (1) Normal Operations and (2) Emergency or Catastrophic Operations failure conditions (Figure 19.5)?
2. Shifted the traditional Engineering "reactive" shock absorber approach and purposely designed interfaces to (1) fail and (2) fail in a specific manner under specific Operating Environment conditions - fault diversion.

These are exciting and challenging topics we will explore later in the chapter.

In summary, Chapter 27 introduces interface definition, analysis, and control concepts and principles seldom addressed in most text.

27.1 DEFINITIONS OF KEY TERMS

- **Analysis Paralysis** A condition exhibited by an analyst that has become preoccupied or immersed in the details of an analysis while failing to recognize the marginal utility and diminishing returns of continual investigation.
- **Compatibility** The ability for the physical characteristics of two or more interfacing Entities to fit within specified dimensions, parameters, and tolerances.
- **Coupling** "The manner and degree of interdependence between software modules" (SEVOCAB, 2014, p. 72–73) (Source: ISO/IEC/IEEE 24765:2011. Copyright © 2012 ISO/IEC/IEEE. Used by permission.).
- **Interface control** "The process of: (1) identifying all functional and physical characteristics relevant to the interfacing of two or more items provided by one or more organizations; and (2) ensuring that proposed changes to these characteristics are evaluated and approved prior to implementation" (MIL-STD-480B, 1988, p. 10).
- **Interface ownership** The assignment of accountability to an individual, team, or organization for the identification, specification, development, control, operation, and support of an interface.
- **Interoperability** "Degree to which two or more systems, products or components can exchange information and use the information that has been exchanged" (SEVOCAB, 2014, p. 162) (Source: ISO/IEC/IEEE 24765:2011. Copyright © 2012 ISO/IEC/IEEE. Used by permission.).
- **Line Replaceable Unit (LRU)** Refer to the definition provided in Chapter 16.
- **Redundant design** The selection of redundant components and physical placement and spatial separation of components to ensure survival.

27.2 APPROACH TO THIS CHAPTER

Most people believe that interface design consists of connecting two points together—point-to-point interfaces. Analytically, this is true. Analysis, by definition, breaks an abstract problem down into successively smaller pieces that can be solved (Principle 4.17).

Our approach to Chapter 27 evolves from simple, point-to-point interface concepts into more complex interface discussions. Our discussion consists of the following sequence of topics:

- Interface Ownership, Work Products, and Control
- Interface Definition Methodology
- Interface Design—Advanced Topics
- Interface Definition and Control Challenges and Solutions

Let's begin with Interface Ownership, Work Products, and Control, which serve as the foundation for interface definition success in system development.

27.3 INTERFACE OWNERSHIP, WORK PRODUCTS, AND CONTROL CONCEPTS

As the SYSTEM or PRODUCT Context Diagram (Figure 8.1) and System Architecture (Figure 20.5) evolve and mature, interfaces emerge as Entity Relationships (ERs) between the System Elements—PERSONNEL, EQUIPMENT, MISSION RESOURCES, PROCEDURAL DATA, SYSTEM RESPONSES, and FACILITIES (Figure 8.13)—and their respective operational, behavioral, and physical entities. As each interface emerges, *accountability* must be established to oversee and ensure the integrity of its evolution. Accountability requires the following:

- A charter that assigns interface ownership, objectives, resources, and work products.
- Work product documentation—specification requirements, designs—that bounds and defines the interface and agreements.
- Control of the work product to ensure its integrity and compliance with technical, cost, and schedule requirements.

On inspection, your initial response may be that this is Project Management, not SE. However, SE in its leadership role has to establish accountability for the technical work to be performed. If this accountability is left unresolved by weak project or technical leadership, *chaos* and *conflicts* will distract the focus and energy that should be concentrated on the definition of the interfaces. To better understand this point, let's elaborate on each one.

27.3.1 Interface Ownership

Interface Ownership and Accountability Principle

Principle 27.3 Assign ownership and accountability for every interface within the SYSTEM or PRODUCT based on the highest-level architecture that identifies the interface.

As Principle 27.3 states, interface *ownership* and *accountability* within a system, product, or service must be assigned to an individual or team. This includes the specification, design, and control of the interface. Observe that we said assign "ownership and accountability for every interface … to an individual or team."

Everyone assigned to an interface must be accountable for specification, design, implementation, and its control. However, there is a *major difference* between ownership, accountability, and interface work task performance. Why? The objective is to preclude *ad hoc* changes to an interface that can impact the capability and performance boundaries other interfacing Entities. You simply cannot have an individual or team deciding *unilaterally* to make changes to an interface unless all Stakeholders in the interface specification are part of the decision.

In general, there are two approaches to interface ownership and control:

- The *Ad hoc* Engineering Approach (Mini-Case Study 26.1).
- The Architecture-Based Approach (Figure 20.5).

27.3.1.1 Ad Hoc Engineering Interface Ownership Approach Technical directors and project engineers view interface ownership as exercising their managerial role of "delegation." Their "task and forget" approach to interface ownership expects two or more interfacing parties to "work it out on their own." Depending on the situation and personalities, this approach works in some cases; in other cases, it is very *ineffective*. This approach is common to Enterprises that employ the *ad hoc*, Specify-Build-Test-Fix (SBTF)-Design Process Method (DPM) Engineering Paradigm (Chapter 2).

Several potential problems can arise from the *ad hoc* SDBTF-DPM approach.

- Conflicts occur when the interfacing parties are *unable* or *unwilling* to agree on how to implement the interface.
- One personality dominates the other, thereby creating technical, cost, or schedule issues that favor the dominating party.

If this *chaos* or *dominance* continues, the dominating party may and probably will *suboptimize* the SYSTEM (Figure 14.8). There is, however, a better approach to interface ownership and control that resolves the problems created by the *ad hoc* engineering ownership approach. That is Architecture-Based Ownership and Control Approach.

27.3.1.2 Architecture-Based Ownership and Control Approach To avoid the problems of the *ad hoc* SDBTF-DPM Engineering approach, assign interface ownership and control to the individual or team accountable for the System Performance Specification (SPS) or Entity Development Specification (EDS) and the associated architecture. The interfacing parties represent entities *within* the architecture.

The discussion to this point focuses on interfaces within the SYSTEM. *What about SYSTEM external interfaces*? External interfaces that require both programmatic and technical solutions between Enterprises. This brings us to our next topic, Interface Control Working Groups (ICWGs).

27.3.1.3 Interface Control Working Groups (ICWGs) Definition and control of SYSTEM or PRODUCT *external* interfaces generally occurs in two forms as follows:

- **Scenario #1**—Interface definition between a new SYSTEM or an upgrade to a legacy system that must integrate to an existing external system.
- **Scenario #2**—Agreement between Enterprises to define a new interface.

In Scenario #1, the existing system may dictate the interface requirements for the new SYSTEM or upgrade. The Enterprise that owns the external system *may or may not* be willing to discuss changes due to costs and other factors.

In Scenario #2, both Enterprises may mutually agree to charter an ICWG with representatives from both Enterprises. The ICWG's mission is to work constructively together to define interface requirements and control changes that serve the interests of both parties.

One of the key questions that arises is: *who should chair the ICWG*? The answer is system and situational dependent.

- If System A is an existing system with an established interface that a new System B under development must connect, System A's project probably has no interest in participating in an ICWG. Simply live with the established interface unless System A's project agrees to changes. When that situation occurs, System A assigns the ICWG chair.
- If Systems A and B are both new and being developed simultaneously, then Projects A and B need to make the decision, each with a co-chair of the ICWG.

27.3.2 Interface Definition Work Products

From an SE perspective, interface definition require several types of work products to guide the development of an interface as follows:

1. Specification requirements—bound and specify *what* an interface is to accomplish and *how well*.
2. Operational Concept Description (OCD)—Within a Concept of Operations (ConOps) Document (Chapter 6) conceptualizes the "*who, what, when, where,* and *how*" for each interface based on the vision for its employment, its Use Cases (UCs) and scenarios (Chapter 5), and operational constraints.
3. Trade study Analysis of Alternatives (AoA)—Chapter 32—Selects the optimal interface approach interface from a set of viable candidates.
4. Design solution requirements—Define the physical implementation of an interface such as architecture, drawings, schematics, wiring lists, cabling, and connector pin-outs.
5. Design description—Describes how the interface is to be installed, operated, maintained, and sustained.

The discussions that follow refer to Interface Requirements Documents (IRDs) as a general classification label. In this context, IRDs consist of a number of different types of interface documents such as:

- Interface Requirements Specifications (IRSs) document interface requirements, in general. Some view an IRS as documenting SYSTEM *external* interface requirements.
- Interface Control Documents (ICDs) document internal *hardware* interface details as *design requirements*.
- Interface Design Descriptions (IDDs) document internal *software* design details as *design requirements*.

Given this overview, let's explore these topics further.

27.3.2.1 Specifying SYSTEM/ENTITY Interface Requirements

The specification of interface requirements employs the same methods as discussed in Chapters 19–23. There are three key aspects that should be addressed:

- Interface specification approaches.
- Interface specifications.
- The structure of the interface requirements within a specification.

27.3.2.1.1 Interface Requirements Specification (IRS) Approaches

Interface Requirements Specification (IRS) Approaches

Interface requirements are typically specified in Section 3.4 Interfaces section (Table 20.1.) of an SPS or EDS. Beyond this statement, the format of an IRS varies by industry, Enterprise, and Engineering discipline.

When we specify interface requirements, there are two approaches in general use as follows:

1. **Approach #1**—Specify both *external* and *internal* interface requirements for every SYSTEM or ENTITY.
2. **Approach #2**—Specify only *external* interface requirements for a SYSTEM or ENTITY.

Observe that Approaches #1 and #2 focus on *what* has to be accomplished, not *how*. The word specification is not mentioned.

Approach #1, which is common in some Enterprise specification outlines, has two major fallacies that result in a duplication of requirements and effort.

- **Fallacy #1**—If we begin with a SYSTEM Level architecture and decompose it into lower levels of abstraction, *internal* interfaces within each architecture become *external* interfaces for those same Entities at the next lower level. Using Figure 20.5 as

an example, *why* would you want to specify internal interfaces at the SYSTEM Level SPS—A2–B1, B1–D1, C2–D1, A4–C2 – and:

- Use the Performance Specification Approach (Figure 20.4), which assumes the SYSTEM or PRODUCT to be a "box" with a To Be Determined (TBD) architecture?.
- Redefine them as external interfaces in SUBSYSTEM A–D specifications?
  Remember - duplication of specification requirements is a violation of Principle 22.1, which states that there should be *one and only one instance* of a requirement within a SYSTEM.

- **Fallacy #2**—If we employ a Performance-based Specification Approach (Figure 20.5) that: (1) treats each Entity as a "box" with inputs, outputs, and capabilities and (2) specifies *what* has to be accomplished but not *how*, then any interfaces internal to the "box," by definition, are To Be Determined (TBD)—*unknown*. Identification and definition of those interfaces will *evolve* and *mature* over time as part of the SE analysis, design, and development of the SYSTEM or ENTITY.

By a process of elimination, Approach #2 – specify external interface requirements only - is the only logical approach for specifying external interface requirements.

27.3.2.1.2 Specifying Interface Requirements

Principle 27.4 **Single Information Source Principle**

Everything a User "needs to know" about a SYSTEM or ENTITY such as its specification requirements, UCs and scenarios, operational concepts, design, interfaces, testing, verification, or validation should be encapsulated within a *single* document unique to that subject.

Exceptions include separation of the information into different documents due to: (1) the quantity of pages or (2) Sensitive Information Protection (SIP) containment of proprietary, IP, competition sensitive data, privacy, government security, and data on a justified "need-to-know" basis Warning 17.2.

In compliance with the Single Information Source Principle (Principle 27.4), a SYSTEM's or an ENTITY's requirements reside in Section 3.0 of the SPS or EDS (Table 20.1). Principle 27.4 notes that exceptions are acceptable as long as there are *compelling reasons* to do otherwise … that are documented and approved. Examples of exceptions include the following scenarios:

- Scenario #1—A contract or task mandates development and delivery of separate IRSs.
- Scenario #2—Two or more Enterprises develop and specify requirements for an *unknown* interface that will evolve over time after the specification has been approved.
- Scenario #3—There is a need to *isolate* and *breakout* requirements unique to a specific interface for a vendor that does not have a "need to know" about the contents of the SPS or EDS for Intellectual Property (IP), proprietary, or security classification reasons.
- Scenario #4—The quantity of pages required to specify SYSTEM or ENTITY interfaces is overwhelming and requires creation of a separate IRD.

27.3.2.1.3 Interface Requirements Specifications (IRSs)

In the case of Scenario #2 mentioned previously, interfaces must be specified by a literal name and incorporate a separate IRD such as an IRS by reference. Consider the following example:

Example 27.1 **Incorporation by Reference of an IRS 3.3.X (XYZ) Interface Requirements**

Assume we have a requirement:

"*The* SYSTEM/ENTITY ***shall*** *provide a bi-directional data communications interface with external System XYZ.*"

"*The* SYSTEM/ENTITY XYZ INTERFACE ***shall*** *comply with the data format specified in Table _ of Interface Requirement Specification (IRS) Revision _".*

In the interest of keeping the requirement statement brief and concise, you can debate the need for the terms "bi-directional" and "data communications" in this high-level requirement. However, it is often useful to declare in a *parent*-level requirement (Figure 21.2) what is to be accomplished—an *abstract* "communications interface" versus a *more explicit* "bi-directional communications interface." You could make those *child*-level requirements. In addition, the IRS "version" information is critical due to the potential risk of the SYSTEM/ENTITY capabilities being developed to an IRS external to the project that is updated to a new version before delivery. (Principle 23.4).

Observe that the context of an IRS here is simply *generic*. Some Enterprises such as the US DoD have specific requirements for an IRS. Consider the following US DoD IRS example:

Example 27.2 **IRS Description**

DoD Data Item Description (DID) DI-IPSC-81434A (1999, p. 1) describes the IRS as follows: "The IRS specifies the

requirements imposed on one or more systems, subsystems, Hardware Configuration Items (HWCIs), Computer Software Configuration Items (CSCIs), manual operations, or other system components to achieve one or more interfaces among these entities. An IRS can cover any number of interfaces. . . . The IRS can be used to supplement the System/Subsystem Specification (SSS) . . . and Software Requirements Specification (SRS) . . . as the basis for design and qualification testing of systems and CSCIs."

IRS TOPICS When specifying interface requirements, there are key topics that must be addressed. Contextually, Engineers often think that the methodologies expressed by the SE Process Model (Figure 14.1) and the System Capability template (Figure 10.17) apply to the *design* of a SYSTEM or ENTITY as a "box." The focus shifts to the Engineering of interface *compatibility*—mechanical parts, electrical power, and signal wires and cables; interface *interoperability* is secondary. They fail to recognize that the SE Process Model and System Capability methodologies apply to the *interoperability* exchanges of energy, forces, and data that occur across the interface via the mechanical parts, wires, and cable. *How do the SE Process and the System Capability methodologies apply to interfaces*?

- The SE Process methodology instills the need to define the Requirements, Operational, Behavioral, and Physical Domain Solutions. This applies to the SYSTEM- and ENTITY-Level architectural components … and … their interfaces.
- The System Capability methodology instills the need to define how the phases of a capability—Pre-Capability, Capability, and Post-Capability—will be implemented.

These two points provide the foundation for identifying the key topics to be addressed in an IRS. The following outline serves as an example within the framework of an SPS or EDS Section 3.4 (Table 20.1) Interface Requirements.

Author's Note 27.1

Two key notes concerning the follow-on discussion:

1. The topics listed below cover a diverse set of common interfaces such as mechanical, thermal, hydraulic, optical, electrical, and data. Other examples include nuclear, biological, and chemical. Tailor the topics to meet the specific attributes of each interface. Renumber accordingly.
2. AR = As Required.

Section 3.4:	System External Interfaces
Section 3.4.1:	Interface #1 Name
Section 3.4.1.1:	(Directional Source–Destination Interface Name)
Section 3.4.1.1.1:	Operational Characteristics and Constraints
Section 3.4.1.1.2:	Behavioral Characteristics and Constraints
Section 3.4.1.1.3:	Physical Characteristics and Constraints
Section 3.4.1.1.3.1:	Mechanical Characteristics and Constraints (AR)
Section 3.4.1.1.3.2:	Thermal Characteristics and Constraints (AR)
Section 3.4.1.1.3.3:	Hydraulic Characteristics and Constraints (AR)
Section 3.4.1.1.3.4:	Electrical Characteristics and Constraints (AR)
Section 3.4.1.1.3.5:	Optical Characteristics and Constraints (AR)
Section 3.4.1.1.3.6:	Data Characteristics and Constraints (AR)
Section 3.4.1.1.3.X:	(Label) Characteristics and Constraints (AR)

If the interface has bi-directional characteristics, specify the *reverse* direction requirements.

STANDARD IRS TEMPLATES To ensure consistency in System Development, create standard templates for documenting interface requirements and descriptions. For example, the US DoD DID DI-IPSC-81434A provides a standard template for documenting IRS interface requirements including HWCIs and CSCIs.

27.3.2.2 Interface Control Documents (ICDs)

Heuristic 27.1 ICDs

As a general rule, especially in Aerospace and Defense (A&D) applications, ICDs are typically used to document HARDWARE interface details such as drawings, wiring lists, and cable diagrams. An ICD:

- Ranges in length from a single-page or multi-page document, such as interface wiring diagrams, mechanical drawings, mechanical hole-spacing layouts, and physical connector pin layouts.
- Serves as a detailed *design* solution response to SPS, EDS, or IRS requirements.

When all interface Stakeholders agree with the contents of the ICDs via reviews, each ICD is approved, baselined, placed under configuration control, and released for formal decision-making (Principle 20.8).

In general, ICDs are controlled by the project's Configuration Control Board (CCB). An exception may be external interfaces in which two or more Enterprises must agree on the initial baseline but also any formal changes to it. In cases such as this, a joint ICWG composed of representatives from all Enterprises may be assigned as members.

ICDs are often developed in *free form* unless your contract or Organizational Standard Process (OSP) mandates a specific format.

There are numerous ways to structure an ICD, depending on the application. Perhaps, the most important aspects of the outline can be derived from the traditional specification outline (Table 20.1) such as:

- Section 1.0 Introduction
- Section 2.0 Referenced Documents
- Section 3.0 Requirements—refer to outline provided previously

Since ICDs contain detailed information regarding the physical implementation of a system, product, or service, their contents are typically referred to as *design requirements*. These design requirements are incorporated into Engineering drawings for physical components. Each physical component is verified by Quality Assurance (QA) for compliance.

ICDs should be written with a focus on *who* the Users—readers—are and *how* they want to use the document. Most ICDs are used simply to document design parameters such as wiring and cable lists, connector function pin-outs, and polarity conventions.

Observe that we mentioned ICDs are often used to "document design parameters." If an ICD's communicates requirements, one would expect there to be a Section 3.0 Requirements that consists of at least one or more "shall" statements making compliance mandatory. As a best practice, employ "shall"-based interface capability requirements in the same manner as specifications (Principle 22.7).

27.3.2.3 IDDs

Heuristic 27.2 IDDs

As a general rule, especially in Aerospace and Defense (A&D) applications, IDDs are used to document software interface descriptions.

The subject of IDDs is one that has a number of contexts and meanings.

- Context #1—By title, one expects the document to describe *what you need to know* about the PERSONNEL and EQUIPMENT - HARDWARE, and SOFTWARE - implementation of an interface—it's operation, behavioral responses, and physical implementation details.
- Context #2—The software community, especially on US DoD projects, views an IDD as a document unique to Software.

Software interfaces are typically documented in an IDD. This approach is often used for CSCIs where there is a need to isolate CSCI-specific IDDs as a separate document. IDDs, however, are not limited to *software*. They are equally applicable to HARDWARE applications as well. DI-IPSC-81436A (2007) serves as the DID for developing an IDD.

27.3.2.4 How Many ICDs/IDDs?

One of the early challenges in identifying interfaces is determining how many ICDs/IDDs are required. Engineers take great pride in the act of designing interfaces. Yet, one of their shortcomings is the lack of confidence in defining how many interface documents—ICDs or IDDs—will be required to develop the SYSTEM OR PRODUCT. In many of these cases, there is often a lack of SE or weak SE presence to guide technical decision-making. As stated earlier in Chapter 1 (McCumber & Sloan, 2002), SE is responsible for "maintaining intellectual control" of the System Design Solution. This entails understanding what types and specific instances of documents must be developed.

Central to this issue are the two options as illustrated in Figure 27.1:

- **Option A**—Do we create a single interface document—ICD and/or IDD—that defines all of PRODUCT A's HARDWARE and SOFTWARE interfaces?
- **Option B**—Do we create individual interface documents—ICDs and/or IDDs—for each PRODUCT A's internal interface? For example: an A1–A2 ICD, an A2–A3 ICD, or an A1–A3 ICD?

There are no rules that prevent a single interface document from covering both the HARDWARE ICD and SOFTWARE IDD details. For simplicity, readers can rationalize the need to have all details about an interface in a single document (Principle 27.4) to avoid having to research multiple hardware and software interface documents. Begin with a single interface document for the SYSTEM. If impractical, justify the need to (1) break it into separate interface documents for each ENTITY's *external* interface at every level of abstraction—Interface A–B, Interface C–D. Where appropriate, create separate Hardware ICDs and/or Software IDDs. So, *what is the answer* and *how do you decide*? There are several points to consider in answering these questions.

Principle 27.5 Interface Disclosure Principle

To prevent disclosure of Intellectual Property (IP) or Competition Sensitive information, *isolate* requirements for a specific interface into a separate for coordination with external Enterprises that have a justified "need to know."

First, recognize that each additional document you create *increases* the cost of maintenance. As a general rule, *avoid* creating separate ICDs/IDDs unless:

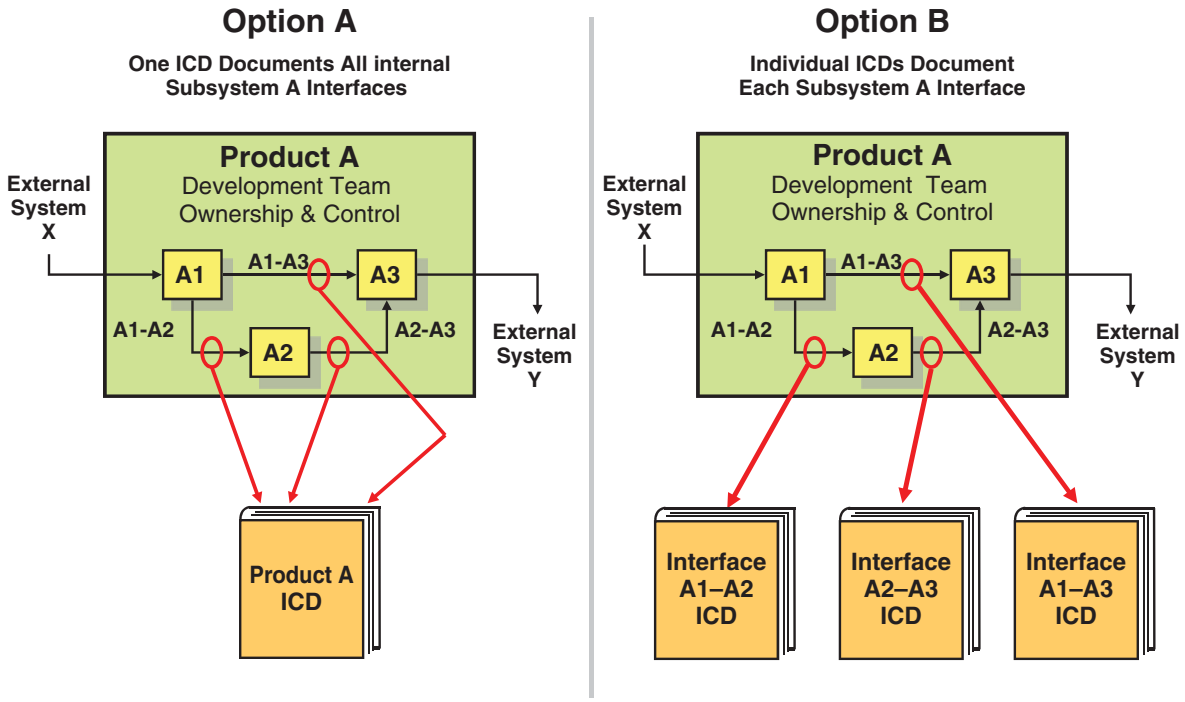


Figure 27.1 Interface Control document (ICD) Implementation Options—Single or Multiple ICDs

1. The amount of information becomes unwieldy for the reader due to the page count.
2. There is a need to isolate details due to IP, proprietary data, and security reasons to limit authorized access to a specific interface based on a "Need-to-Know" justification.

Secondly, based on the first point:

- A single SYSTEM ICD may be appropriate for small projects in which interface information is shared across all team members. Observe that we said SYSTEM—for the whole project, not SYSTEM Level ICD.
- For larger projects, *isolate* and *create* separate ICDs/IDDs for interfaces to be developed by external vendors for the project. To reduce the need for separate ICDs and IDDs, procure external components that have standard connector or data interfaces such as RS-232, Ethernet.

27.3.3 Interface Control

Interface definition, as a part of SYSTEM/ENTITY design, requires that SE "maintains intellectual control of the (evolving and maturing interface) problem solution" (Principle 1.3 - McCumber and Sloan (2002) (Principle 1.3). Interface design control consists of two stages of control based on a trade-off of the maturity of the interface design solution and the cost to perform formal change management. They are as follows:

- Stage 1—Semi-formal Engineering control.
- Stage 2—Formal Configuration Management (CM) control.

Using stages of semiformal and formal CM control makes processing changes more efficient and effective. The approach is not perfect; it is still dependent on the collective experience, wisdom, and informed judgment of the Product Development Team (PDT) such as an Integrated Product Team (IPT) stakeholders.

27.3.3.1 Stage 1—Semi-formal Engineering Control

Early in the development of an interface, *flexibility* to accommodate changes to either side of the interface is important. As the interface definition *evolves* and *matures*, there is a point which *discretionary* changes become *disruptive*. Although the interface is *owned* and *managed* preferably by a team such as an SDT or PDT, the team's SE should initiate efforts to establish a consensus that changes require team approval via team discussions, not formal requests. This process continues until it is time to *approve, baseline, and release* interface design work products (Principal 20.8) under Stage 2 Formal CM Control.

27.3.3.2 Stage 2—Formal CM Control As interface work products such ad specifications, OCDs, trade studies, and designs reach a point where formal control is required, they are formally reviewed by their respective Stakeholders—Users and End Users, approved, and submitted to CM. CM places each one under formal change management by establishing a baseline and releasing the work product for decision making.

Heading 27.1

Given an understanding of interface ownership, accountability, and control, our next task is to define each interface. We need a methodology-based strategy that enables us to define each interface. This brings us to our next topic, the need for an Interface Definition Methodology.

27.4 INTERFACE DEFINITION METHODOLOGY

When we identify, specify, and design an interface, there are key technical issues that must be considered. For example:

1. *Who* are the interface stakeholders—operators, maintainers, or instructors?
2. *Where and* under *what* conditions is the interface to be employed?
3. *What* is to be exchanged or transferred across the interface—data, forces, energy, and directional flow?
4. *When* and *how frequently* is the interface to be employed—synchronous or asynchronous data transmissions, door handles and opening/closing, switch usage?
5. *How* will the interface be controlled for security and privacy reasons—encryption, decryption?
6. *How* will the interface be protected from natural threats in its OPERATING ENVIRONMENT—weather, rodents?
7. How will interface *compatibility* and *interoperability* be assured?

For Enterprises that employ the *ad hoc* Plug & Chug … SBTF Paradigm (Chapter 2), answers to these questions are often ambiguous, incomplete, and often addressed in the incorrect order or not at all. Note in Figure 2.3 these Enterprises leap from requirements to a physical solution while *ignoring* the operational and behavioral aspects.

Answering these questions requires methodology that allows us to logically derive interface information. The following interface definition methodology allows us to define interfaces.

- Step 1—Identify each SYSTEM or ENTITY interface.
- Step 2—Define the interface's purpose and objectives.
- Step 3—Define what is to the transferred, exchanged, or communicated.
- Step 4—Identify each interface's Users and End Users.
- Step 5—Define the interface's UCs and scenarios.
- Step 6—Identify the interface's modes of operation.
- Step 7—Derive and model *operational* requirements and constraints.
- Step 8—Develop the OCD.
- Step 9—Specify and model *behavioral* interactions and constraints.
- Step 10—Specify *physical* requirements and constraints.
- Step 11—Standardize across other interfaces.
- Step 12—Label and color-code interface cables and connectors.

Let's elaborate each of the steps in more detail.

27.4.1 Step 1—Identify SYSTEM or ENTITY Interfaces

The first step in identifying SYSTEM interfaces is to simply recognize and acknowledge its existence externally as specified in the SPS. This begins with the System Context Diagram (Figure 8.1) and N2 Diagram (Figure 8.11). Apply these methods at all levels of abstraction beginning with the System of Interest (SOI) followed by the MISSION SYSTEM and ENABLING MISSION. Within the multi-level System Architecture, both types of diagrams are useful; however, N2 Diagrams provide a more explicit reference concerning interface Input/Output (I/O) interaction directions of the SYSTEM or ENTITY.

Principle 27.6

Interface Identification Principle

Assign a unique Source-Destination mnemonic and title to each SYSTEM/ENTITY interface.

Once an interface is identified, establish a *naming convention* that assigns a *unique* identifier to distinguish the interface from all others within the SYSTEM. A typical naming convention consists of a Source-to-Destination naming convention with the interfacing SYSTEM or ENTITY names represented by short names, acronyms, or abbreviations. Consider the following example:

Example 27.3

Automatic Teller Machine (ATM) Example

Interface ID: USER—ATM Interface

- User to ATM ID: USER—ATM (Source-to-Destination)
- ATM to User Interface ID: ATM—USER (Source-to-Destination)

Two key points concerning the aforementioned example:

1. Remember that interface IDs should be *unique* labels and appear *once and only once* within the SYSTEM. To avoid duplication of names, create a tool such as a database to track current names.
2. The Source–Destination naming convention—USER—ATM—can be expanded to identify transactions that represent *unique* interactions such as USER—ATM ____ #1 RESP (Response) or ATM—User ____ #6 REQ (Request).

To *avoid* quantum leaps from requirements to physical solutions (Figure 2.3), our approach to interface definition will be to apply the SE Process (Figure 14.1) to each physical interface to identify the sequence of operational, behavioral, and physical characteristics.

27.4.2 Step 2—Define the Interface's Purpose and Objectives

Once the Users and End User's have been identified, define the interface purpose and objective(s) of what it is expected to accomplish. For example,

- Source electrical power available "on demand" to remote devices such as motors or lights.
- Transmit and receive asynchronous/synchronous data commands and messages to/from remote devices.
- Continuously monitor remote analog or digital sensors at a 10-Hz rate.
- Change I/O Device XYZ's current mode of operation.
- Send and receive audio, video, or data messages signals on command.
- Display or sound audio and/or video cautions or warnings via alerts, alarms, when conditions warrant.

27.4.3 Step 3—Define What is to be Transferred, Exchanged, or Communicated

Define:

- *What* is to be transferred, exchanged, or communicated such as energy—electrical, mechanical, optical, thermal; mechanical—forces, pressure; and data—commands and messages or data - commands or messages.
- The *directionality—unidirectional* or *bidirectional*—nature of the transfer, exchange, or communications.

27.4.4 Step 4—Identify Each Interface's User(s) and End User(s)

Define *who* the interface's Users and End Users are, their "need to know" access privileges, and projected frequency of usage.

- Are there *restrictions* on use of the interface and when?
- Do Users require *authorized* accounts with User IDs and passwords?
- Are there Users or End Users that require an interface that may not have been specified in the SPS or EDS?

Each side of an interface consists of User (Actor) and End User (Actor) roles (Figure 5.10). You should recall from our definitions of the User and End User roles introduced in CHAPTER 3.

- An interface's User *Monitors, Commands, and Controls (MC2)* the SYSTEM/ENTITY and its capabilities - energy, raw materials, forces, data commands, and messages, for one direction of an interface. For example, a User *transmits* data commands and messages.
- An interface's End User, as *recipient*, benefits from source energy, raw materials, forces, and data, transferred across the interface. For example, an End User *receives* data commands and messages.

Consider the following examples:

Example 27.4 **Bi-Directional Phone Conversation**

A phone conversation, as a bi-directional interface, consists of Actors that alternate roles - both User and End User - as performing entities. A User Role communicates audio messages to an End User Role that benefits from receiving and vice versa the communication.

Example 27.5 **Voice Mailbox Users and End Users**

A caller, as an End User role, calls a phone number and leaves a recorded voice mail message in a voice mailbox.

Later, the owner of the voice mailbox performs a User Role to MC2 the playback of recorded messages. As messages are played, the owner is an End User that *benefits* from message(s) content.

27.4.5 Step 5—Define the interface's UCs and Scenarios

For each User or End User, identify and define their respective UCs and scenarios (Chapter 5).

27.4.6 Step 6—Identify the Interface's Modes of Operation

On the basis of an interface's UCs, identify its *modes* of operation (Chapter 7). For example, a data communications device might have the following modes of operation:

- COMMUNICATIONS Mode—Send and receive *unencrypted* data commands or messages.
- TEST Mode—Send and receive "loop back" test data messages.

27.4.7 Step 7—Derive Operational Requirements and Constraints

Using the interface's purpose and UCs, specify and bound its operational capabilities and constraints. *Operational constraints* may include weather conditions, Time-of-Day (TOD), geophysical location, and modes of operation. Then, define the interface's operational timeline, as required, within time constraints. If a response to an interface data request is required within 10 ms, so state.

27.4.8 Step 8—Develop the Operational Concept Description (OCD)

Based on the interface's *operational requirements*, develop an OCD for each interface that describes *how* the interface interactions between Actors are envisioned to occur—discrete data driven, interrupt driven, and so forth.

Interface operational characteristics are derived by identifying and scoping: w*ho* interacts with *whom*, *when*, *how* frequently, under *what* constraints and operating conditions, and *what* are the expected outcomes. For example, mechanical, electrical, optical, data, responses—and levels of performance. Consider the following example of what might be described by an interface OCD:

Example 27.6

ATM Bank Card System OCD

An ATM bank card system consists of remote, multi-location, electronic teller devices located at kiosks at bank and business locations. The devices, which are available 24 hours per day, 7 days a week for card transactions such as obtain cash, make deposits, or pay bills without having to go to a bank or other business. Geographical access to remote ATMs of other banks is greatly expanded when the User's bank and other banking institutions have agreed to process credit or debit transactions with the bank's ATM card.

To initiate a bank card transaction at an ATM device:

Step	Actor	Action Performed
1.	ATM Device	Displays a "Welcome" greeting for the next User and requests the User to insert their bank card.
2.	User	Reads display message and inserts bankcard into ATM Device.
3.	ATM Device	Reads bank card data, verifies the account, and requests the User to enter a password to verify authorization.
4	User	Enters password.
5.	ATM Device	Verifies the password, requests authorization from a central computer, and receives an authorization number to conduct the transaction.
6.	ATM Device	Requests the User to select the type of transaction desired.
7.	User	Selects the transaction type.
8.	ATM Device	Requests the User to enter the amount based on the transaction type.
9.	User	Enters the transaction amount.
10.	ATM Device	Reads User transaction amount, processes the transaction, and dispenses cash.
11.	User	Removes cash from the ATM Device.
12.	ATM Device	Inquires if the User would like another transaction.
13.	User	Selects Yes or No for another transaction.
14.	ATM Device	Cycles for another transaction or User according to the Yes/No answer.
15	ATM Device	Thanks the User for their business and returns to Step 1.

27.4.9 Step 9—Specify and Model Behavioral Interactions and Constraints

Interface *behavioral* characteristics characterize how a SYSTEM or ENTITY responds to stimuli, excitations, or cues in its OPERATING ENVIRONMENT. Figure 10.12 (Fishbone) provides a general illustration. Chapter 10 provides graphical illustrations (Figures 10.13 – 10.16) of key interface decisions that drive behavioral interface characteristics. Example decisions include the following:

1. When an external system in the OPERATING ENVIRONMENT *directly* or *indirectly* interacts with a SYSTEM or ENTITY performing its mission, is a response required? Yes or No?

2. If so, what type of system response is required? A non-response? An acknowledgement? Final response only?
3. What is the required format of the response?
4. On the basis of the interface's operational requirements, what time constraints are levied on the response back to the source?

One of the best tools for defining these interactions is a SysML™ Sequence Diagram (Figure 5.11). Sequence diagram interactions and events must be synchronized with a Mission Event Timeline (MET) (Figure 5.5) that provides analytical insights into the temporal—time-dependent—aspects of *how* the interfacing entities are expected to interact. It is important to note that each interaction in Example 27.7 below represents an interface capability that can be translated into an SPS or EDS requirement.

Step 1 Identify SYSTEM or ENTITY Interfaces - enables us to assign a *unique* label to each interface such as ATM-User, User-ATM. Since there are scripted behavioral interchanges—messages and responses—between the ATM Device and the User, you can expand the IDs to include what is exchanged such as requests and responses. Consider the following example in which REQ = Request and RESP = Response:

USER—ATM Transaction Interactions

Example 27.7

Step	Request/Response Mnemonic	Action to be Performed
1.	ATM—USER REQ_#1:	"Enters User Card"
2.	USER—ATM RESP_#1:	User inserts card correctly into ATM
3.	ATM—USER REQ_#2:	"Enter password"
4.	USER—ATM RESP_#2:	User enters Password

Behavioral characteristics involve more than simply defining the step sequence of interactions. The question is: *how does the* SYSTEM OR ENTITY *process and develop responses to external interactions*?

SYSTEM or ENTITY responses may require actions such as:

- Evasive actions—non-responses, measured responses, or changing location.
- Initiation of counter-measures or counter-countermeasures.

The type and scope of interactions—for example, cooperative, benign, adversarial, harsh, hostile—determine the appropriate responses. Chapter 10 describes how the SYSTEM or ENTITY Modeling and Simulation (M&S) is used to represent these interactions. Definition of the behavioral aspects of an interface interaction established *what* actions and sequences have to be accomplished; however, we did not define *how* the interface will be physically implemented. This brings us to the need to define the physical characteristics of an interface.

27.4.10 Step 10—Specify Physical Interface Characteristics and Constraints

On the basis of what is to be transferred, exchanged, or communicated, select the transfer medium that will be required. For example, electrical copper cables, Radio Frequency (RF)—wireless; Fiber Optics (FO)—channels; mechanical mechanisms, connections; thermal conductors, insulation. Conduct a trade study AoA to assess the best approach for implementing and maintaining the interface.

When the transfer medium has been selected, specify the physical characteristics of the interface such as:

- Mechanical characteristics—compression/tension, thermal, pressure.
- Electrical characteristics—analog/digital signals; voltages; current; digital discrete bits—On/Off, Open/Close; protocol—Ethernet, IEEE-488; analog, digital, and power grounding and shielding; connector pin-outs.
- Optical characteristics—luminance, spectral frequency, intensity, attenuation.
- Data characteristics—data command and data message formatting, packing, encoding.

Analysis of interacting systems requires investigation of a variety of classes of physical interactions that may be required. For most systems, the classes of interfaces include: electrical, mechanical, optical, acoustical, nuclear, chemical, biological, environmental, and human. In general, a SYSTEM's physical response characteristics are dependent on a SYSTEM or ENTITY's mechanical design and *mass properties* (Chapter 3), optical characteristics, or design controlled behavioral response actions such as electrical, mechanical, or optical. Examples include:

- Physical movements such as 6-Degrees of Freedom (6-DoF) (Figure 25.13).
- Thermal expansion or contraction.
- Pressure increases or decreases.
- Electrical signals, response times.
- Energy conversion and storage.

One of the challenges of physical interface analysis is that SEs and System Analysts suffer from *analysis paralysis* and become intrigued and immersed by a specific class of interaction. They overlook or ignore other classes that may become showstoppers. This point illustrates the need for multi-discipline teams such as an SDT or PDTs to participate in the identification, definition, and control of their respective levels of interfaces.

If the interface has any physical constraints such as weight, size, technology, color-coding, polarity or keying conventions, OPERATING ENVIRONMENT conditions, specify and bound those requirements.

Finally, *understand how* the interface will interact with external systems in its OPERATING ENVIRONMENT. Figure 27.2 provides an analytical matrix of potential types of interactions for consideration between a MISSION SYSTEM and the NATURAL, INDUCED, AND HUMAN SYSTEMS ENVIRONMENTS.

27.4.11 Step 11—Standardize Across Other Interfaces

Interface design, as is the case with any form of System Design, focuses on meeting the specified requirements while minimizing cost, schedule, technical, technology, and support risks. Every time you design a new interface solution—dedicated interface, you must be prepared to mitigate the risks of an *unproven* interface. In addition, the cost of development and maintenance of *non-standard* interfaces by specialized expertise increases labor costs significantly.

One way of reducing the impacts of these risks is to employ design solutions that are already *proven*. In addition, any technology solution you choose may be subject to becoming obsolete in a short period of time. In sharp contrast, consumer products in the marketplace, especially computers, demand that systems be designed to accept technology upgrades to maintain system capabilities and performance without requiring new systems as replacements.

One of the ways industry addresses this marketplace need is with standard interfaces that promote development and selection of LRUs that have industry standard interfaces such as RS-232, IEEE-488, Ethernet protocols or USB, SCSI, DB, RJ, BNC coax, connectors.

27.4.12 Step 12—Label and Color-Code Interface Cables and Connectors Connections

Interface definition and design activities focus almost exclusively on the SYSTEM or PRODUCT'S physical interactions—force, data, energy, energy exchanges—with external systems. However, interfaces require not only Installation and Checkout (I&CO), but also *preventive* and *corrective* maintenance (Chapter 34). Therefore, investigate ways of ensuring that maintenance can be performed easily, properly, and correctly. This includes consideration of authorized access—physical, security; color-coding and labeling of cables and connectors; caution and warning labels.

Heading 27.2

The preceding discussion of an Interface Definition Methodology provides a technical strategy that will enable us to define and design an interface. While we tend to think of interfaces as discipline

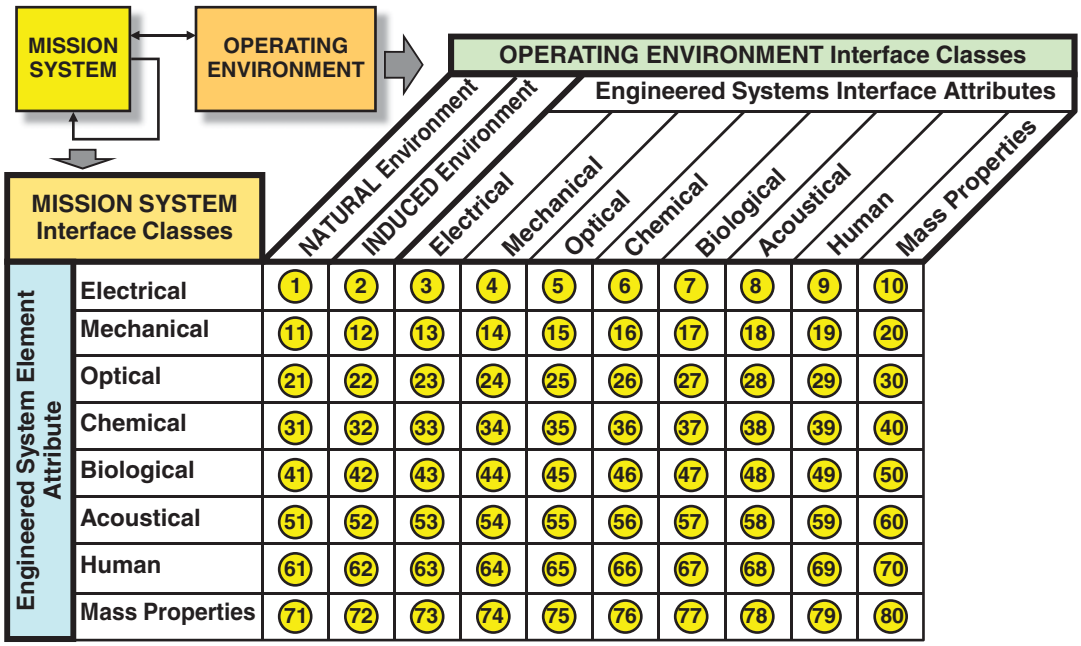


Figure 27.2 Analytical Interface Interactions Matrix

centric—mechanical, electrical, software, they are often multi-faceted and have their own sets of complexities and trade-offs. This brings us to our next topic, Interface Design—Advanced Topics.

27.5 INTERFACE DESIGN—ADVANCED TOPICS

Educationally, to keep things simple, we focus on *boundary condition problems* that are unique to an Engineering discipline. In general, figure out how to connect Point A to Point. In today's world of technology, faster computational devices and M&S methods such as Model-Based Systems Engineering (MBSE) enable us to study and design more complex multi-discipline interfaces and issues.

Our discussion in this section introduces some advanced interface definition and design topics that involve more than simply connecting Point A to Point B. They require more robust insights through *Systems Thinking* (Chapter 1). This section introduces the concept of purposely *designing* and *exploiting* interface characteristics to influence a specified *outcome* such as an automobile collision. Since a significant amount of interface design occurs in digital computing and data communications and mechanical design, let's explore the application of interface design to a couple of examples.

27.5.1 Data Communications Case Study

High-speed audio, video, and data communications requires investigating the following:

- Physical implementation and Medium—wire cables, optical fiber or Fiber Optic (FO) cables, wireless, infrared, RF—VHF, UHF, microwave.
- Data transmission and formatting methods—data transfer protocols, message formatting, and synchronous/asynchronous transmissions.

Fiber optic data communications are common today, especially in office buildings, land line telecommunications, and military applications. Advantages include features such as high bandwidth, longer transmission distances, smaller size and reduced weight, data security, Electromagnetic Interference (EMI) immunity, elimination of ground loops, and no sparks. Figure 27.3 provides an illustration of a fiber application with FO Transceivers. In this illustration, each transceiver:

- Consists of a semiconductor laser (transmitter) and a photo detector (receiver).
- Interfaces with electronics in the respective side of the interface and the FO cable

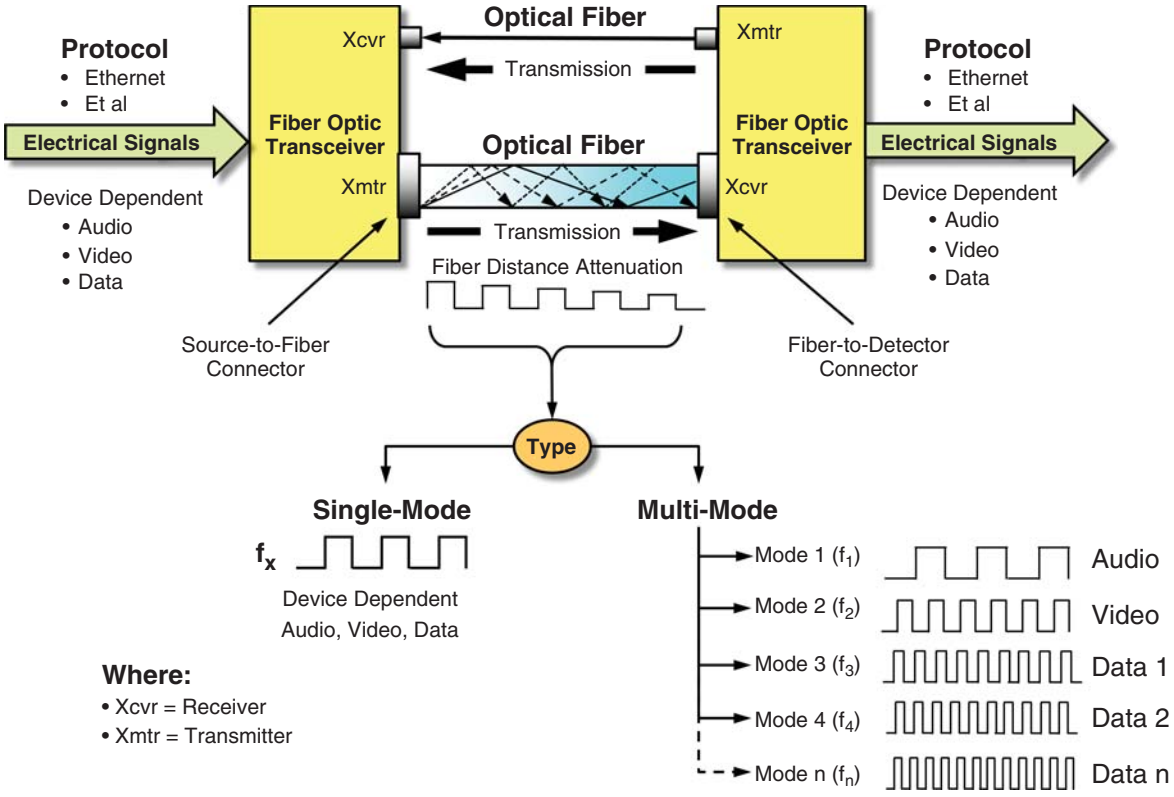


Figure 27.3 Fiber Optic (FO) Interface Data Communications Example

Optical fibers, in general, support two types of data transmission: Single Mode or Multi-mode. A *mode* in fiber terminology equates to a *channel*. A method referred to as a Wavelength Division Multiplexing enables multiple modes (channels) to be transmitted over a single fiber. As shown in the example (Figure 27.3), the following modes have been reserved for specific types of transmission: Mode 1—Audio Channel, Mode 2—Video Channel, and Modes 3–5—Data Channels #1–*n*.

Now, let's assume that we have a requirement to monitor and record measurements from several remotely located analog sensors. The physical design of the remote Sensor System consists of a Command and Control (C2) Processor and a Telemetry Processor to transmit data messages back to the Central Computer System as shown in Figure 27.3. A decision is made to use an FO cable to telemeter the data.

For this application, a C2 processor reads the sensor data from I/O devices that perform an Analog-to-Digital (A–D) conversion and stores the data in a buffer—designated shared memory location. The Telemetry Processor reads the data and formats it into a data packet or frame structure as shown in Figure 27.4. Data are read and formatted into a message structure that consists of the following:

- **Preamble**—protocol-defined data.
- **Start of Frame** (SOF)—protocol-defined data.
- **Message Header**—Message_Start (unique set of alpha-numeric and control characters), unique Record_ID, Data_Time_Stamp, Message_Type such as command or data, and Message_Length (words or bytes).
- **Message Body**—Data Items #1, #2, and so forth—*signed* or *unsigned* integers of different word lengths with pre-defined unit values, Digital Discretes #1–#*n*—single bits representing binary values such as On/Off, Open/Closed.
- **End of Message**—Message_End (unique set of alpha-numeric and control characters) and Check_Sum for error checking and correction.
- **End of Packet or Frame**—protocol-defined data.

Each packet is formatted with data in accordance with the Software IDD using the structure such as the one shown in Figure 27.5.

Each data packet or frame is transmitted as a serial data message stream at a 10-Hz rate via a Single Mode fiber cable (Figure 27.3) back to a Central computer.

When designing a data communications interface, consider the following:

- Magnitude of the data to be transferred or exchanged—the number of bits, bytes, or words,

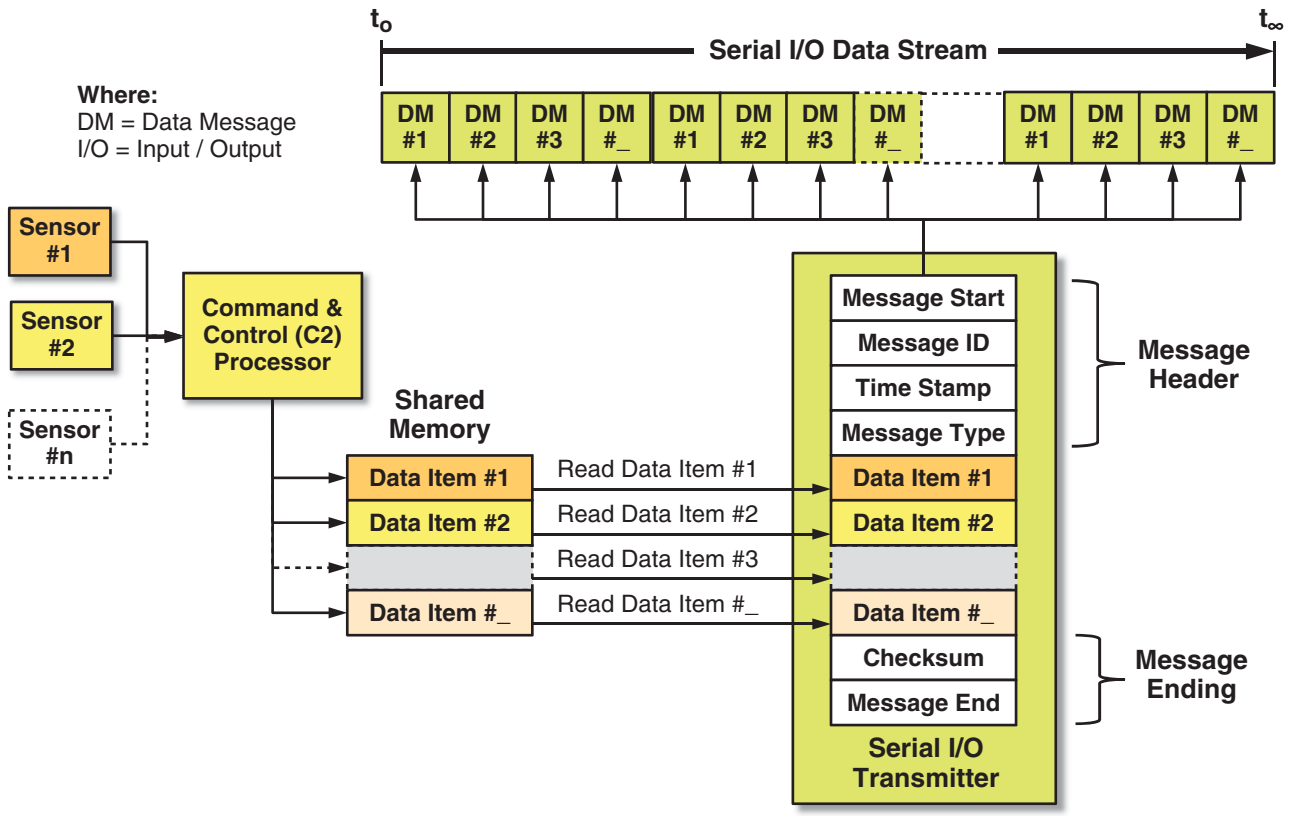


Figure 27.4 Telemetry Data Stream and Data Command/Message Packing Example

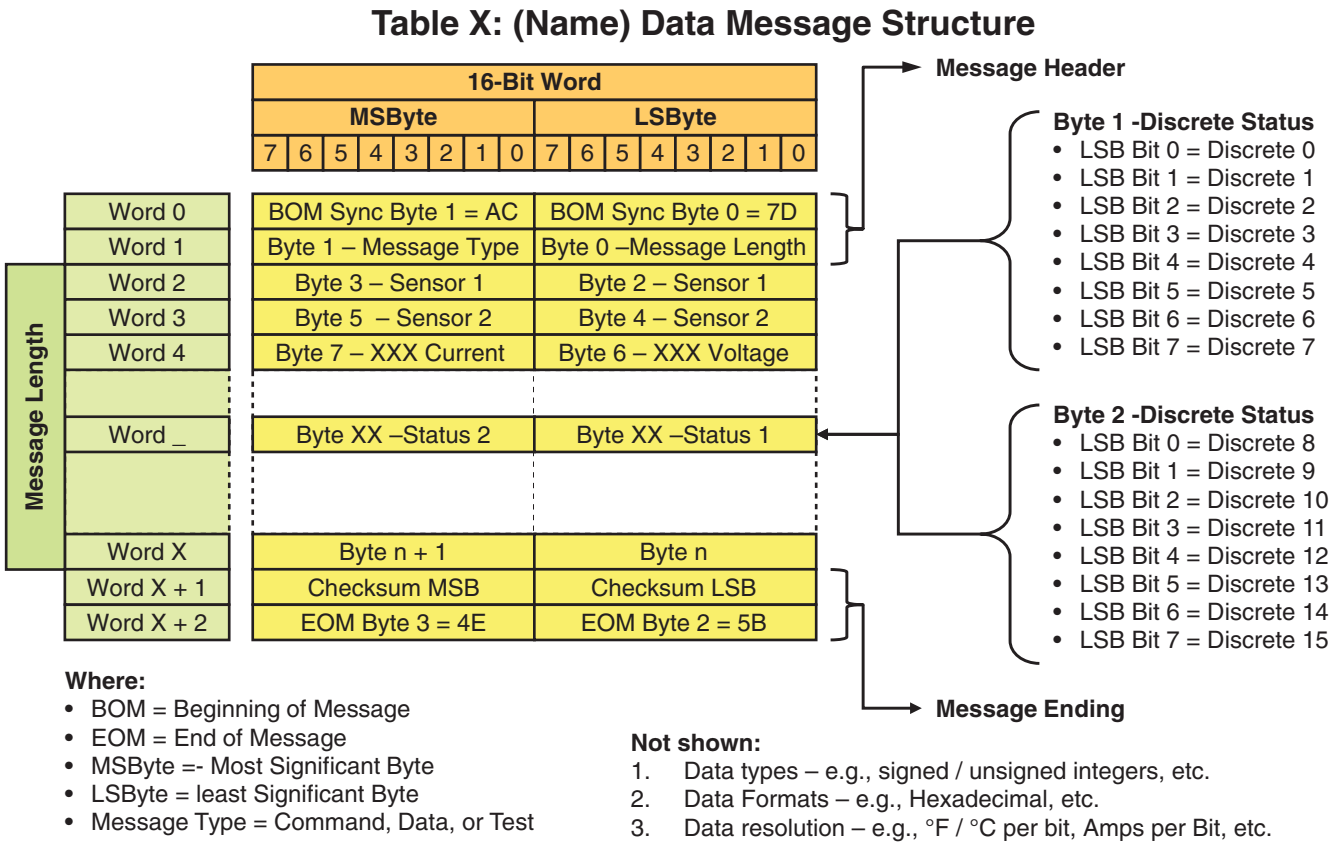


Figure 27.5 Telemetry Data Command/Message Packet Format Structure

data precision and accuracy, and word sizes required for formatted command or data messages.

- Bandwidth, messaging time constraints, and the flexibility to accommodate growth and expansion. For example, a design goal of 50% of transmission capacity to meet current needs.

This example illustrates several key points as follows:

- Select a transfer medium—Single Mode FO cable (Figure 27.3).
- Select a data communications protocol such as Ethernet for transmitting packets of information.
- Format data within each protocol packet according to a defined data table (Figures 27.4 and 27.5).
- Transmit the data packets at a *synchronous* rate such as 10 Hz or 1 KHz.

27.5.2 Automobile Case Study

Principle 27.7

Power of Engineering Principle

The power of Engineering resides in its application of "… judgment to develop ways to utilize economically the materials and forces of nature for the benefit of mankind."

We tend to think of interface design as the Engineering between two entities such as the transceivers shown in Figure 27.3. By definition, *analysis* enables us to break down complex interfaces into successive levels of discrete segments that facilitate piecewise (interface) physical and mathematical analysis and models. Finite Element Analysis (FEA) is a good example based on its wire mesh models.

Engineers tend to think of the Power of Engineering (Principle 27.7) in terms of the "Engineering the Box" such as PRODUCTS, SUBSYSTEMS, in a static configuration sense rather then the "Engineering of Systems" (Chapter 1). Components such as automobile's radio, body, exist in their present configuration throughout their accident-free useful service lives. Traditional interface design between these components—cables, wiring, mechanical linkages—occurs *after* the components are identified and located. However, there are reverse situations in which interface design drives the design of components, especially in safety-related situations. Case in point: you can react to energy in a *passive* sense—absorb energy via shock and vibration—or you can *proactively* manage it via interface design. Such is the case with automobiles during an accident.

An objective of automobile design is to reduce injury to passengers by applying various types of interfaces to reduce the impact on passengers. From a Power of Engineering

(Principle 27.7) perspective: harness the forces of (human) nature, *absorb, dissipate,* and *divert* them and reduce their impact on passengers.

Beginning as early as in the 1950s, automobile developers created specialized interfaces such as front and rear crumple zones and a Passenger Compartment as a *safety cell*. The purpose of this integrated chain of interfaces is: (1) to *absorb* and *dissipate* as much Kinetic Energy (KE) on impact, and (2) *divert* and *distribute* it away from the Passenger Compartment (Grabianowski, 2013).

Figure 27.6 illustrates an example of automobile crumple zones surrounding a Passenger Compartment. The crumple zone *absorbs* the impact as a means for *harnessing* and *diverting* KE forces away from the passenger area. Those forces are *directed* and *distributed* into the structure around the Passenger Compartment via the roof, floor, and engine firewall bulkhead as shown in Figure 27.7.

On inspection, this may appear to be Engineering of mechanical interfaces to get them to crumple under certain operating conditions. This is true. However, there is another dimension that requires consideration … *time*.

During a collision, Newton's Second Law of Motion mathematically expresses the magnitude of the force exerted on a vehicle at impact as a function of its mass and acceleration.

$$\text{Force} = \text{mass} * \text{acceleration} \tag{27.1}$$

Substituting acceleration with its velocity components:

$$\text{Force} = \text{mass}\left[\frac{\text{Final Velocity} - \text{Initial Velocity}}{T_{\text{Final}} - T_{\text{Initial}}}\right] \tag{27.2}$$

where: $T_{\text{Final}} - T_{\text{Initial}}$ represents the duration of the event—impact until stopped.

Since Final Velocity = 0, the acceleration component is *negative,* therefore reflecting *deceleration*. Differentiating Eq. 27.2 to derive the rate of change of F, force, as a function of the rate of change in velocity results in:

$$dF = \text{mass} * \frac{dv}{dt} \tag{27.3}$$

On inspection of Eq. 27.3 if we can control or reduce the rate of *deceleration*, the *instantaneous* force exerted on the Passenger Compartment (safety cell) as a function of time is *reduced*.

In general, a collision of a vehicle traveling at a velocity of 55 mph striking a stationary object requires approximately 0.7 seconds to come to a complete stop. Mathematically, slowing the *rate of deceleration* by a few tenths of a second can significantly reduce the *instantaneous* force to be distributed and countered by the Passenger Compartment structure, air bags, seat-shoulder belts. The mechanism for controlling the rate of deceleration of this force via *deformation* is the automobile's *crumple zone*. As a compensating

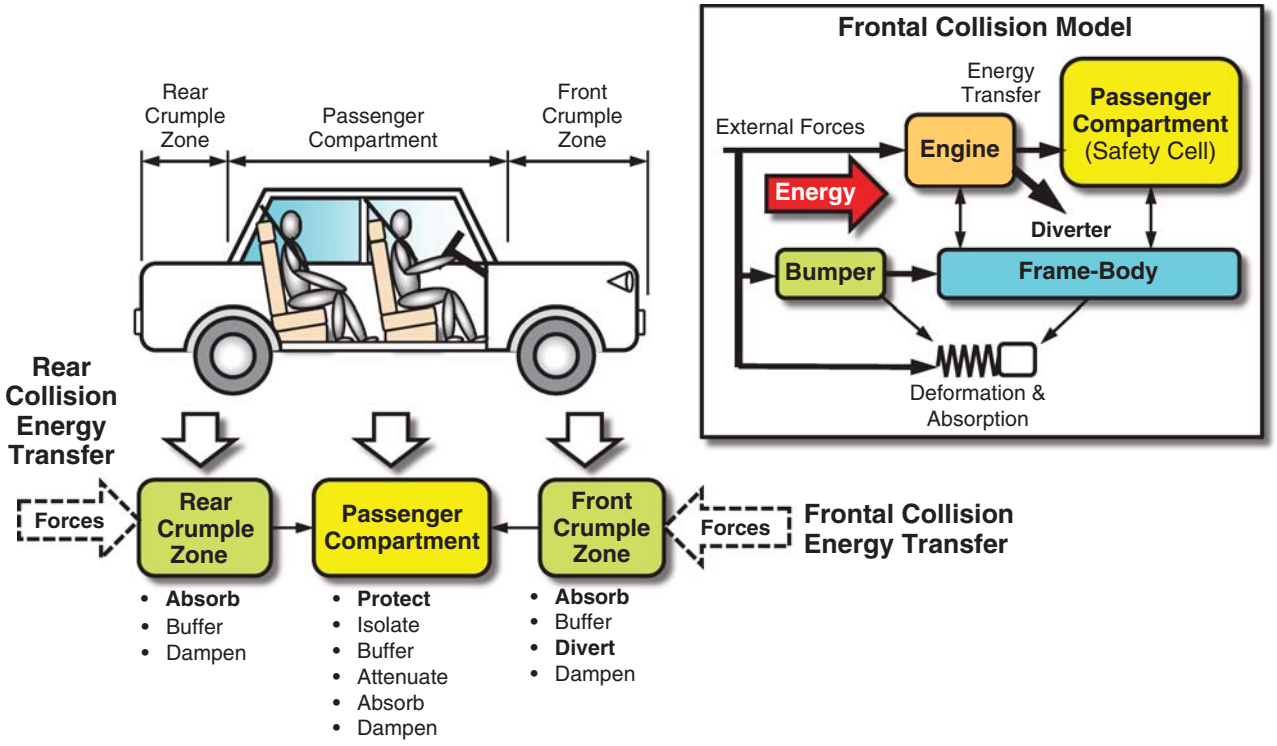
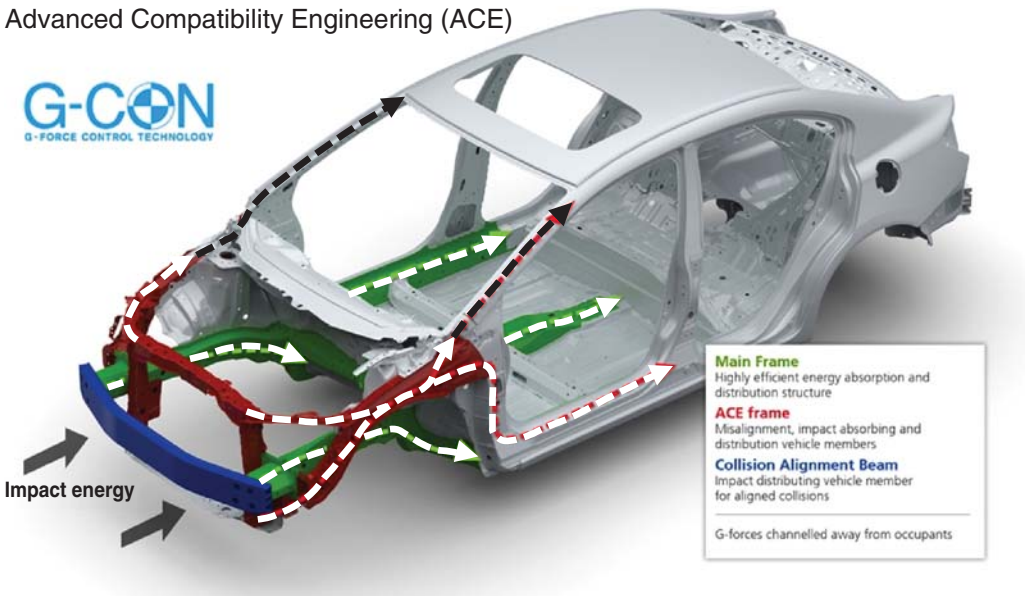


Figure 27.6 Automobile Crumple Zone and Passenger Compartment Modeling Example



Original Source: Honda Motor Company. Used by permission and adapted.

Figure 27.7 Example of Dispersal of Collision Kinetic Energy (KE) Forces Around an Automobile Passenger Compartment to Reduce Injury. Source: Honda Motor Company. Used by permission and adapted.

action, automobile designers engineer a chain of components and interfaces to collapse in a controlled manner for a given set of conditions.

The key points of this discussion are:

1. Interfaces are more than "line connectors on a drawing" connecting components via wire, cables, and mechanical linkages. They represent "components" of a SYSTEM or PRODUCT that perform a *function*. From a System Design perspective, interfaces should receive an equivalent level of importance.
2. As "components" of a SYSTEM or PRODUCT, Engineers design interface structures and components to include weak points that enable it to collapse in a controlled deformation manner (AIP, 2004). Grabianowski (2013) notes that automobile design creates special structures "to be damaged, crumpled, crushed, or broken."
3. On the basis of the Power of Engineering (Principle 27.7), if we apply "… judgment to develop ways to utilize economically the materials and forces of nature for the benefit of mankind," we can engineer an interface to control the rate of deceleration of a collision to reduce injury and save lives.

Heading 27.3

The preceding sections provide a foundation for identifying, analyzing, and specifying an interface based on a methodology. Advanced topics addressed special considerations for the implementing interfaces. Our final topic, Interface Definition and Control Challenges, introduces a series of challenges that SEs must be prepared to address concerning the definition, design, and development of interfaces.

27.6 INTERFACE DEFINITION AND CONTROL CHALLENGES AND SOLUTIONS

Interface definition, design, development, operations, and support activities often face challenges that are common across many systems. Let's identify and discuss some of these key challenges.

- Challenge 1: Lack of external interface commitments.
- Challenge 2: Failure to assign interface ownership and control.
- Challenge 3: Identification of and vulnerability to threats.
- Challenge 4: Environment, Safety, and Occupational Health (ES&OH) Risks.
- Challenge 5: Availability on DEmand.
- Challenge 6: Interface reliability.
- Challenge 7: Interface maintainability.
- Challenge 8: Lack of compatibility and interoperability.
- Challenge 9: Mitigating interface integrity issues.

- Challenge 10: External electrical power availability, quality, and backup.
- Challenge 11: Power, analog, and digital signal grounding and shielding.
- Challenge 12: RF and EMI Emissions.
- Challenge 13: Component Failure Modes and Effects.
- Challenge 14: Fault Containment.
- Challenge 15: Reduction in Total Number of Interfaces.

27.6.1 Challenge 1: Lack of External Interface Commitments

Contracts are awarded every day in which the Acquirer states in the SPS … "The system **shall** report Sensor #1 status to External System XYZ at a 10 Hz rate."

On investigation, the Acquirer should have ensured that a Memorandum of Agreement (MOA) between the User and the owner(s) of external System XYZ for the interface was in place. This should have occurred *prior to* the release of a formal solicitation such as a Request for Proposal (RFP). What happens in these situations is that User abdicates responsibility and *transfers the risk* to the System Developer. In some cases, this approach may be acceptable, especially if the System Developer has existing relationships and rapport with the interfacing parties. However, in an "open" procurement, the Acquirer and System Developer *should not* assume this. Consider the following Mini-Case study.

Mini-Case Study 27.1

The Uncooperative Interface Owner

Assume that a User's System ABC is new and is required to interface to System XYZ. Coincidentally, the User and System XYZ's owner are both members of the same Enterprise. If relations between the User acquiring the System and System XYZ's owner are *poor*, in an effort to avoid dealing directly with System XYZ's owner, the User's contract requires System ABC to interface directly to System XYZ. However, System XYZ's owner has no interest in supporting the interface. What the System Developer believed to be easy to accomplish during the proposal phase turned into an *uncomfortable* and *avoidable* situation.

A System Developer should never be placed in a position of negotiating agreement between two parties of a User's Enterprise; however, it happens every day. Many times even the System Acquirer serving as the User technical representative is *unaware* – and *should be* - of these situations. If the System Developer proactively attempts to deal with the issue in a "positive, professional manner," Systems ABC and XYZ management could turn and accuse the System Developer of "managing their Enterprise," another uncomfortable situation. Thoroughly investigate what agreements and commitments the Acquirer or User have made with external system owners concerning interfaces *before* the contract is signed.

27.6.2 Challenge 2: Failure to Assign Interface Ownership and Control

Accountability for each System or Entity interface must be assigned an owner(s) such as an individual, Enterprise, or ICWG. Those accountable *must* control interface definition, design; development; SITE; System Operation, Maintenance, and Sustainment (OM&S); phase-out; or disposal. As the *accountable* owner, the individual or Enterprise is responsible for reviewing and approving changes to the interface design baseline, but also the transference of interface operations and maintenance knowledge to System Users via User's guides, manuals, and training.

Interface ownership and control issues are sometimes manifested in *who* is accountable for developing and providing interface wiring and cabling. Specifically, if two or more interfacing Enterprises are designing an interface, *who provides the interconnecting cables and wiring?* Many times, each Enterprise *assumes* the other is developing and providing the wiring and cabling. Establish Enterprise MOAs that clearly define *who* is accountable for building and providing the wiring cables and when, where, and to whom they are to be delivered.

27.6.3 Challenge 3: Identification of and Vulnerability to Threats

System interface design is based on a pre-defined set of interfaces that must interoperate. The reality is that some operational interfaces, such as military, financial, medical, educational, and transportation systems, are vulnerable to external threats and attacks. These systems must contend with what are referred to as *unknowns* and *unknown-unknowns*. Acquirers and System Developers must work with the Users and their supporting Enterprises to thoroughly:

- Understand and anticipate a System or Entity's potential threats and threat conditions.
- Define *how* the System or Entity's interfaces will cope with those threats.

After a System or Entity is delivered, the Users must *continuously monitor* the performance (Chapter 29) of the System's mechanisms and processes used to operate the interface. In addition, the Users must assess the *susceptibility* and *vulnerability* of the interface to evolving or potential threats in its System's Operating Environment.

Where appropriate, reasonable measures should be required and implemented such as *specialized* solutions that include: encryption/decryption, security, firewall, and antivirus software.

27.6.4 Challenge 4: Environment, Safety, and Occupational Health Risks

Some SYSTEMS OR ENTITIES pose potential threats to the Environment, Safety, and Occupational Health (ES&OH) to MISSION SYSTEM and ENABLING SYSTEM PERSONNEL, the public, property, or the environment. When designing SYSTEM interfaces, thoroughly analyze potential ES&H scenarios, such as exhaust emissions, toxic chemicals, hazardous materials, and leaking fluids, which may pose ES&H and Environment, Safety, and Occupational Health (ES&OH) risks and require mitigation to acceptable levels.

Author's Note 27.2

For Challenges 5 – 8 that follow, recognize that their sequencing is based on a series of questions.

1. Is an interface *available* "on demand" when required?
2. If it is available, is it *compatible* and *interoperable* with external systems?
3. If so, is it *reliable*?
4. If so, can it be easily *maintained*?

27.6.5 Challenge 5: Availability on Demand

Users performing missions expect the MISSION SYSTEM or ENABLING SYSTEM to perform when required—System Availability (Chapter 34). Interface availability, as a contributory enabler to SYSTEM/ENTITY performance, is a critical issue both *internally* and *externally*.

- Internally, each interface must be available "on demand" and in a *state of readiness* when configured and activated to support the mission.
- Externally, how will your SYSTEM respond or adapt to an external system interface failure (Principle 19.22).

An interface's *availability* is a function of its *reliability* and *maintainability* (Chapter 34).

27.6.6 Challenge 6: Interface Reliability

If an interface capability is available when required, the question is: can the interface reliably perform its intended mission in the prescribed OPERATING ENVIRONMENT conditions to the level of performance required by the SYSTEM or ENTITY specification? Each interface must be designed to achieve a level of reliability that will ensure accomplishment of mission capabilities and throughout the mission. Two options are available as follows:

- Option #1—Select higher reliability components.
- Option #2—Consider interface design redundancy (Chapter 34)

Either of these two options requires AoA trade-offs in terms of System Development and System/Product Life Cycle cost considerations such as additional electrical power and weight for redundant components, maintenance, and thermal build-up from redundant components.

27.6.7 Challenge 7: Interface Maintainability

To minimize SYSTEM or PRODUCT *downtime*, you must ensure that the interfaces are maintainable (Chapter 34) with a specified level of skills and tools commensurate with the Pre-Mission, Mission, and Post-Mission phases of operation and operational tasks. Consider how interfaces will be inspected or monitored and maintained such as use of lockable access plates and ports; video monitoring; thermal imaging and temperature sensors for overheating; need for cautions, warnings, and alerts, sensing over/under voltage, over current; methods.

27.6.8 Challenge 8: Interface Compatibility and Interoperability

When two or more SYSTEMS or PRODUCTS developed by different Enterprises require integration into a Level 0 User's System (Figure 8.4), interface *compatibility* and *interoperability* become potential risk items. If poorly performed or not at all, the risk could become a problem … a major problem resulting in a lot of finger-pointing between Enterprises. *How do we mitigate the risk*? There are two aspects: interface *compatibility* versus *interoperability*. In addition, if the interface consists of mechanical, electrical, and data attributes, the problem becomes multi-faceted. The challenge question becomes:

- Assume that two very large systems being developed in different countries or parts of a country share common mechanical, electrical, and electrical/software data interfaces that may be stand-alone or combined. For example, a data communications interface has mechanical, electrical, and data characteristics. To reduce technical risk during System Integration, Test, & Evaluation (SITE) how would you propose verifying and validating interfaces *before* one or the other is moved for SITE?

In lieu of providing a solution here, we will defer this question for the Level 2 Chapter Exercises.

27.6.9 Challenge 9: Interface Integrity Issues

The integrity of an interface is dependent on how well its design performs under specified OPERATING ENVIRONMENT conditions. Typically, a *safety factor* (Chapter 31) such as 2× or 3× (Chapter 31) may be used to mitigate interface component *failures* that may compromise the SYSTEM or PRODUCT. Examples include bulkhead pressure leaks, clogged filters, contamination, lack of data packet, or frame error checking.

27.6.10 Challenge 10: External Electrical Power—Availability, Quality, and Backup

Engineers often focus on the internal design of their SOI—the SYSTEM, PRODUCT, SUBSYSTEM Levels, and so forth. They procrastinate on researching external interfaces such as electrical power sources, their attributes, and quality. In general, they assume that 110 vac or +28 vdc electrical power will "always be available and all we have to do is plug our device in."

Given these examples of power types, Engineers often overlook the subtleties such as 50 Hz versus 60 Hz versus 400 Hz, as well as tolerances placed on the magnitudes and frequencies. Power quality factors are also a consideration. Finally, a key question is: *During peak hours of operation, will the power source be reliably continuous, sporadic, or experience periodic hours of operation or blackouts?* This requires risk assessment, planning, handling, and control. You need a Contingency Plan.

Do not assume that the external system power source will always be available when you need it until you have thoroughly investigated and analyzed the interface. In addition, establish a documented agreement such as an MOA that represents a commitment by the power source's owner to allocate and budget power resources to allow your System to operate when required. The same is true for *power quality* and *filtration*.

Finally, assess the need for:

- Back-up power to mitigate data loss during a mission.
- Electrical power failures alerts and displays.
- Electrical power automatic or manual *shutdown* and *start-up* procedures to *minimize* data loss or damage to EQUIPMENT.

27.6.11 Challenge 11: Power, Analog, Digital Signal Grounding and Shielding

Analog signal and *digital* grounding as well as interface shielding are critical topics that must be addressed up front. One of the most overlooked problems is failure to research and understand the electrical power grounding system of the User's higher level—Level 0 SYSTEM (Figure 8.4) —in which your SYSTEM or PRODUCT is being integrated. There is often a complacent attitude by Engineers to defer decisions about electrical power ground and assume that because 110 vac, 60 Hz, or +28 vdc with power ground is readily available at a connector. Why so?

The reality is that you may not know what spurious signals other systems are inserting into the electrical power *ground* conductors either directly via *ground loops, surges, or transients* or indirectly through coupling from adjacent conductors—transients, noise conditions. Until you research and verify how clean the power ground conductors are electrically, these are *unknown-unknowns*.

Sources of these problems include high power switching circuits, components switching On/Off, or EMI coupling from mixing high and low power cables in the same bundle. Unfortunately, the Level 0 SYSTEM owner or their System Integrator may be *unaware* of these problems. This is particularly problematic when your SYSTEM of PRODUCT is required to collect sensor measurement data, especially low-voltage data. Thoroughly investigate the following:

- What types of external analog, digital, and power grounding systems are available such as Single Point ground, Star-Point ground?
- How the external system implements power and signal ground including limitations, constraints, and availability?
- What other external Systems Developers experienced and discovered when interfacing to this power source?

27.6.12 Challenge 12: RF and EMI Emissions

Electronic power and signal interfaces often emit RF spectrum and EMI signals that: (1) may couple to or disrupt data-sensitive devices or (2) are tracked by external security surveillance systems. In today's world, wireless Internet and EMI emissions can pose major risks not only to security, but also to privacy. Investigate robust encoding of data messages that require special security consideration. Reduce electronic emissions to comply with regulatory and security requirements.

27.6.13 Challenge 13: Component Failure Modes and Effects

Redundancy versus Redundant Design Principle

Principle 27.8

Redundancy is a design method to eliminate Single Failure Points (SFPs), thereby increasing SYSTEM or ENTITY's reliability. *Redundant design* is the selection and physical placement and separation of components to ensure survival.

Depending on the criticality of the interface, Engineers often think of *design redundancy* as a solution. However, you can design *redundant* systems to avoid SFP issues and still result in a potentially catastrophic situation. There is a difference between mandated component redundancy in specifications, architectural design redundancy as a reliability solution (Chapter 34), and physical placement of redundant components (Figure 26.12); recognize the difference. To better understand this point, refer to the accident investigation of United Airlines Flight 232 on July 19, 1989 (Wikipedia, 2013).

Failure Modes & and Effects Analysis (FMEA) Principle

Principle 27.9 Failure Modes & Effects Analysis (FMEA) requires more than simply analyzing failures and effects of components *fixed* in place or connected. Analyze the *direct* or *indirect* collateral effects on adjacent Entities.

You may ask: *how does the UA Flight 232 event (Figure 26.12) relate to interfaces and fault containment*? Obviously, when a fault occurs, it must be contained (Figure 26.8). However, faults involve more than simply components overheating or failing. The traditional Engineering paradigm *assumes* that an interface is a fixed based on physical connections. Consider the following scenarios:

- **Scenario #1**—When component failures break away from their physical connection and become projectiles that impact and *interface* with redundant systems causing them to fail?
- **Scenario #2**—When component failures overheat or catch on fire without breaking their physical connection. Their effects *interface*—ripple or spread—to redundant or other components causing them to fail or produce faulty data?

Recognize the implications of design decisions, in general, and their consequential effects, especially on redundant systems. When conducting a FMEA (Chapter 34), *avoid* restricting the analysis to fixed component failures. Investigate the impact of their failure modes and effects on nearby, upstream, and downstream components (Figure 26.12). For example, overheating causing other components to fail, electrical shorts causing a System to catch fire and fail, failure of a heater to maintain constant environmental conditions in a cold or outer space environment, and so forth. Leverage the results of a FMEA to mitigate failure effects with design compensating actions.

27.6.14 Challenge 14: Fault Containment

Interface Fault Containment Principle

Principle 27.10 When *faults* lead to *failures*, isolate and contain them to prevent or minimize their consequential effects on nearby Entities—Equipment; Personnel—operators, maintainers; the public, and the environment.

Systems inevitably experience failures due to *faults* or *defects.* Failures originate from *faults* that manifest themselves in the form of *latent defects* due to component composition or degradation integrity issues, poor assembly workmanship practices, design flaws, and errors, which can lead to overheating, electrical shorts.

Chapter 26 introduced the concept of *fault detection*, isolation, and *containment* (Figure 26.8). In general, every system should have some form of *fault detection* to sense conditions as indicators that a fault is about to occur. If a failure occurs, it can disrupt Mission and System operations. Even worse, faults can create a chain of effects that could lead to catastrophic results. For example, a cable breaks loose or a high-voltage wire chaffs and shorts to chassis ground. The potential ripple effects include a fire and destruction of a power supply or a short to an external system power source.

Once a fault occurs, *fault containment* becomes the first line of defense to prevent proliferation within the System or to external systems via interfaces. In general, when a component fails, they lack any means to contain their consequential effects. *So, how do we contain faults*?

Fault containment requires some form of interface boundary that limits and subsequently eliminates or quarantines the problem source to prevent a reoccurrence or later proliferation of effects. SEs employ FMEA. Then, identify and implement *compensating actions* (Chapter 34) to eliminate or minimize the effects. Compensating actions include design considerations that may involve one or more of the System Elements—Equipment, Personnel, Mission Resources, or Procedural Data. Consider the following example:

Fault Containment Methods

Example 27.8 Examples include power removal, fire suppression and sprinkler systems, building fire doors that close automatically, circuit breakers and fuses, sandbag walls or dikes to prevent flooding, antivirus software quarantine area.

What if there are failures with no destructive consequences or results? Suppose that a communications component simply fails or wiring connections break. Obviously, this causes a disruption of data commands and messages within the System or to external systems. If this occurs, you haven SFP. *How do we avoid this*? The solution is to create redundant communication interfaces.

27.6.15 Challenge 15: Reduction in Total Number of Interfaces

Principle 27.11

System Complexity Principle

The *complexity* and *risk* of a SYSTEM or PRODUCT is a function of the quantity of its interfaces.

Principle 27.12

Interface Reduction Principle

Apply analytical and risk reduction methods to minimize the quantity of interfaces in a SYSTEM or PRODUCT.

Every interface added to a SYSTEM or PRODUCT increases its complexity and probability—risk—of a potential fault or latent defect that can become a *failure point*. The left side of Figure 27.8 provides an illustration. Let's assume a System performs a sequence of computational operations performed by SUBSYSTEM A. Observe the absence of external interfaces required to perform the computation.

Now, observe how another team of designers *incorrectly* approached the implementation. They allocated capabilities across SUBSYSTEMS A and B with physical interfaces connecting back and forth with four interface crossings. Capabilities A and C are allocated to SUBSYSTEM A; Capability B and the Decision are allocated to SUBSYSTEM B. Each one of the four interface crossings represents a potential *failure point*. Analyze how to consolidate capabilities within the physical boundaries of a Configuration Item (CI) to minimize connecting cables and wiring that may break or fail.

27.7 CHAPTER SUMMARY

During our discussion of system interface definition, design, and control we:

- Introduced and demonstrated that interfaces are a critical component of any System or Product and should not be treated as secondary "afterthought" design activities.
- As a component of a System or Product, the SE methods and concepts such as the SE Process (Figure 14.1); Phases, Modes, and States of Operation (Chapter 7); and System Capability Construct (Figure 10.17) apply to interface design.
- Described how interfaces are identified and documented in an IRS, ICD, and IDD.
- Emphasized the importance of assigning interface ownership and control including the need to charter an ICWG for external interfaces between Enterprises that own different systems or products.
- Suggested a methodology for identifying and defining system interfaces.
- Introduced electronic/data and mechanical design case studies that illustrated how to design complex system interfaces.

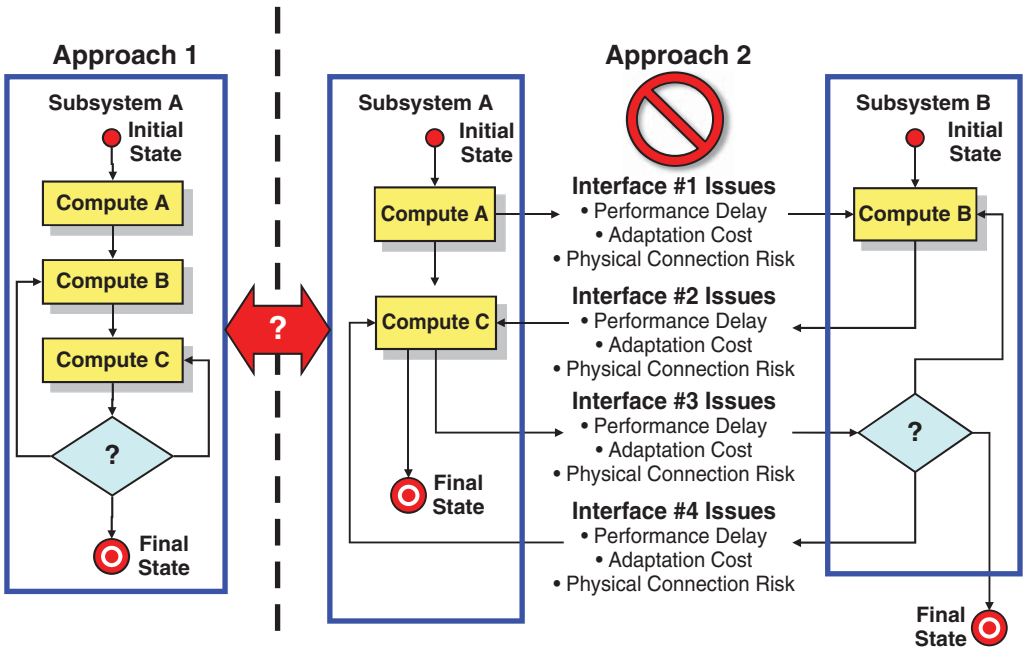


Figure 27.8 Example Illustrating the Importance of Reducing and Containing Interfaces within an Entity to Reduce the Risk of Interface Failures.

- Identified common challenges and issues in interface definition and control.

27.8 CHAPTER EXERCISES

27.8.1 Level 1: Chapter Knowledge Exercises

1. How are System interfaces identified?
2. How do SEs analyze interface interactions?
3. How does the SE Process (Figure 14.1) apply to interface design?
4. How does the System Capability Construct (Figure 10.17) apply to interface design?
5. What is the methodology for defining interfaces?
6. Who owns and controls System or Entity interfaces at various levels of abstraction?
7. What is an IRS and what is its relationship to interface definition?
8. What is an ICD and what is its relationship to interface design?
9. What is an IDD and what is its relationship to interface design?
10. How to determine whether to develop an IRS, ICD, and/or IDD?
11. What is an ICWG and what is its relationship to a System Development project?
12. How is the membership composition of an ICWG determined technically?
13. Who chairs an ICWG?
14. What are some of the challenges of analyzing, designing, and controlling System or Entity interfaces?

27.8.2 Level 2: Chapter Knowledge Application Exercises

Refer to www.wiley.com/go/systemengineeringanalysis2e

27.9 REFERENCES

AIP (2004), *Safer SUVs for Everyone: Automotive Engineers Design Bumpers That Absorb Impact Better*, College Park, MD: American Institute of Physics (AIP).

DI-IPSC-81434A (1999), *Interface Requirements Specification (IRS), Data Item Description (DID)*, Washington, DC: Department of Defense (DoD).

DI-IPSC-81436A (2007), *Interface Design Description (IDD)*, Washington, DC: Department of Defense (DoD).

Grabianowski , Ed (2013), *How Crumple Zones Work*, Atlanta, GA: How Stuff Works.com. Retrieved on 9/2/13 from http://auto.howstuffworks.com/car-driving-safety/safety-regulatory-devices/crumple-zone.htm

ISO/IEC/IEEE 24765–2010 (2012), *IEEE Systems and Software Engineering – Vocabulary*, New York: IEEE Computer Society.

ISO/IEC 25010:2011 (2011) *Systems and Software Engineering–Systems and Software Quality Requirements and Evaluation (SQuaRE)–System and Software Quality Models*, New York: IEEE Computer Society.

McCumber,William H. and Sloan, Crystal (2002), *Educating Systems Engineers: Encouraging Divergent Thinking*, Rockwood, TN: Eagle Ridge Technologies, Inc. Retrieved on 8/31/13 from http://www.ertin.com/papers/mccumber_sloan_2002.pdf.

MIL-STD-480B (1988), *Military Standard: Configuration Control - Engineering Changes, Deviations, and Waivers*, Washington, DC: Department of Defense (DoD).

SEVOCAB (2014), *Software and Systems Engineering Vocabulary*, New York, NY: IEEE Computer Society . Accessed on 5/19/14 from www.computer.org/sevocab

Wikipedia (2013), *United Airlines Flight 232*, Retrieved on 8/19/13 from http://en.wikipedia.org/wiki/United_Airlines_Flight_232.

28

SYSTEM INTEGRATION, TEST, AND EVALUATION (SITE)

As each system component or item completes the Component Procurement and Development Process (Figure 12.2) of the System Development Process, it is ready for System Integration, Test, and Evaluation (SITE). Planning for SITE begins during the SE Design Segment and continues through system delivery and acceptance.

This chapter introduces SITE practices for implementing the right side of the V-Model (Figure 15.2) illustrated in Figure 12.6. Our discussions address what SITE is and how it is conducted. We begin with a discussion of the fundamentals of SITE. Next, we explore how SITE planning is performed and describe the test organization.

Given a basic understanding of SITE, we introduce key tasks that capture how developers incrementally test and verify compliance of multi-level test articles in accordance with their performance or development specifications. We also explore some of the challenges of test data collection and management. We conclude with a discussion of common SITE issues.

28.1 DEFINITIONS OF KEY TERMS

- **Acceptance Testing** "(2) formal testing conducted to enable a user, customer or other authorized entity to determine whether to accept a system or component (SEVOCAB, 2014, p. 3)(Source: IEEE 829-2008. Copyright © 2012 IEEE. Used by permission.)"
- **Acceptance Test Procedures (ATPs)** Formal procedures that script the process and method(s) of verification for applying a stimulation, excitation, or cue as an input to a SYSTEM or ENTITY to produce a performance-based outcome response such as test measurements, results, or observations.
- **Anomaly** An unexplainable event or observation that cannot be easily replicated based on current knowledge or facts of the original occurrence.
- **Automatic Test Equipment (ATE)** "Any automated device used for the express purpose of testing prime equipment; usually external to the prime device (e.g., support equipment)." DAU, 2012, p. B-18
- **Compatibility Testing** "The determination of a product's ability to substitute for another similar product without a major difference in form, fit, or function (F^3) parameters." (FAA, 2012, p. G-1)
- **Compliance Testing** "The determination of a system, product, or service's ability to comply with specified performance characteristics." (FAA, 2012, p. G-1)
- **Destructive Tests** Tests that result in stressing a test article to failure beyond repair. Destructive tests usually destroy, damage, or impair a test article's form, structure, capabilities, or performance beyond refurbishment at a practical and economically feasible level.
- **First Article** "First article includes pre-production models, initial production samples, test samples, first lots, pilot models, and pilot lots; and approval involves testing and evaluating the first article for conformance with specified contract requirements before or in the initial stage of production under a contract." (DAU, 2012, p. B-85)

System Engineering Analysis, Design, and Development: Concepts, Principles, and Practices, Second Edition. Charles S. Wasson.

Companion Website: www.wiley.com/go/systemengineeringanalysis2e

- **Formal Testing** "(1) testing conducted in accordance with test plans and procedures that have been reviewed and approved by a customer, user, or designated level of management (SEVOCAB, 2014, p. 128)(Source: ISO/IEC/IEEE 24765:2011. Copyright © 2012 ISO/IEC/IEEE. Used by permission.)
- **Functional Testing** " (1) testing that ignores the internal mechanism of a system or component and focuses solely on the outputs generated in response to selected inputs and execution conditions (SEVOCAB, 2014, p. 134)(Source: ISO/IEC/IEEE 24765:2011. Copyright © 2012 ISO/IEC/IEEE. Used by permission.)
- **Integration Point (IP)** – A node at higher levels of integration in which a component is integrated with others that have already been verified as compliant with their respective specification requirements..
- **Non-Destructive Tests** Tests that subject a test article to a prescribed set of input conditions and operating environment to demonstrate compliance to requirements. Non-destructive tests do not destroy, damage, or impair a test article's appearance, form, structure, capabilities, or performance other than minor refurbishment.

Example 28.1 **Non-Destructive Tests**

Non-Destructive tests include Qualification Tests (QTs) concerning temperature, humidity, shock, vibration, and so forth that do not destroy the test article.

- **Qualification Test (QT)** "Simulates defined operational environmental conditions with a pre-determined safety factor, the results indicating whether a given design can perform its function within the simulated operational environment of a system." (DAU, 2012, p. B-184)

Author's Note 28.1 **Simulated Operating Environments**

Please note that although a simulated Operating Environment can subject a test article to discrete *worst case conditions* such as shock and vibration, Electro-Magnetic Interference (EMI), environmental, the ultimate QT requires fielding the system, product, or serve and allowing its Users to perform exercises under a range of realistic conditions.

- **Regression Testing** Tests repeated to validate previous test results that may be suspected due to a test event that required a corrective action such as rework, repair, redesign, and so forth.
- **Switchology** A colloquial expression used by testers to represent an understanding of a test article and its human interface such as switches, buttons, and so on required to command and control (C2) the test article.
- **Test** The process of subjecting, measuring, and evaluating a system or entity's responses to a prescribed and controlled set of Operating Environment conditions, verification methods, and stimuli and comparing the results to a set of specified capability and performance requirements.
- **Test and Evaluation (T&E)** The *informal* or *semiformal* process of evaluating the behavioral response and reaction time performance of a system entity within a prescribed operating environment to a controlled set of stimuli for purposes of assessing entity functionality and eliminate defects.
  T&E establishes that an article is free of latent defects or deficiencies subject to acceptable discrepancies—blemishes, non-critical component issues, and so on—and is ready for formal verification. T&E should include a "dry run" of the formal verification ATPs prior to formal verification. T&E activities are generally informal contractor activities and may or may not be observed by the Acquirer.
- **Test and Evaluation Working Group (TEWG)** A team consisting of User, Acquirer, System Developer, Subcontractor, and vendor or supplier personnel stakeholders formed to plan, coordinate, implement, monitor, analyze, and evaluate test results.
- **Test Article** An initial unit of a system or product or one randomly extracted from a production lot to be used for conducting non-destructive or destructive tests.
- **Test Case (TC)** One instance of a series of Use Case (UC) scenario-based tests that employ combinations of test inputs and conditions to verify an item's ability to *accept* or *reject* ranges of inputs, perform value-added processing and to produce only *acceptable* performance-based outcomes or results and minimize or eliminate unacceptable outputs.
- **Test Configuration** An operator-controlled architectural framework capable of representing a System or entity's Operating Environment conditions—Natural, Induced, or Human Systems via simulation, stimulation, or emulation to verify that the test article satisfies a specific requirement or set of requirements.
- **Test Coverage** "(1) the degree to which a given test or set of tests addresses all specified requirements for a given system or component (SEVOCAB, 2014, p. 323)(Source: ISO/IEC/IEEE 24765:2011. Copyright © 2012 ISO/IEC/IEEE. Used by permission.)
- **Test Criteria** "Standards by which test results and outcome are judged." (DAU, 2012, p. B-229)

- **Test Data** Documented operator observations or instrument-produced readouts or printouts that serve as objective evidence of test results.
- **Test Discrepancy (TD)** Actual test measurement results—performance-based outcomes—that do not comply with a specification capability requirement.
- **Test Environment** The set of System Elements—Equipment, Personnel, Facilities, Procedural Data, Mission Resources, simulated Natural and Induced Environments, and so on—configured to represent a test article's Operating Environment conditions.
- **Test and Measurement Equipment** "The peculiar or unique testing and measurement equipment that allows an operator or maintenance function to evaluate operational conditions of a system or equipment by performing specific diagnostics, screening, or quality assurance effort at an Enterprise, intermediate, or depot level of equipment support." (MIL-HDBK-881C, 2011, p. 229)

Example 28.2

Examples include the following:
"Test measurement and diagnostic equipment, precision measuring equipment, automatic test equipment, manual test equipment, automatic test systems, test program sets, appropriate interconnect devices, automated load modules, tapes, and related software, firmware and support hardware (power supply equipment, etc.)" (MIL-HDBK-881C, 2011, p. 229)

- **Test Incident Report** "(1) document reporting on any event that occurs during the testing process which requires investigation (SEVOCAB, 2014, p. 324)(Source: ISO/IEC/IEEE 24765:2011. Copyright © 2012 ISO/IEC/IEEE. Used by permission.)"
- **Test Instrumentation** "Test instrumentation is scientific, Automated Data Processing Equipment (ADPE), or technical equipment used to measure, sense, record, transmit, process, or display data during tests, evaluations, or examination of material, training concepts, or tactical doctrine. Audio-visual is included as instrumentation … " in some domains. (Adapted from DAU, 2005, p. B-19)
- **Test Range** An indoor or outdoor facility that provides a safe and secure area for evaluating a system or entity's capabilities and performance under near Natural Environment or simulated conditions.
- **Test Repeatability** " (1) an attribute of a test, indicating that the same results are produced each time the test is conducted (SEVOCAB, 2014, p. 325)(Source: ISO/IEC/IEEE 24765:2011. Copyright © 2012 ISO/IEC/IEEE. Used by permission.)
- **Test Resources** "A collective term that encompasses all elements necessary to plan, conduct, and collect/analyze data from a test event or project." (DAU, 2012, p. B-19)
- **Test Results** *Objective evidence* of the performance-based outcomes from a test performed to a formal test script such as an ATP that serve as *quality records* for assessing compliance to specification requirements.
- **Testing** "(1) activity in which a system or component is executed under specified conditions, the results are observed or recorded, and an evaluation is made of some aspect of the system or component (SEVOCAB, 2014, p. 326)(Source: IEEE 829-2008. Copyright © 2012 IEEE. Used by permission.)
- **Transient Error** "(1) an error that occurs once, or at unpredictable intervals (SEVOCAB, 2014, p. 333)(Source: ISO/IEC/IEEE 24765:2011. Copyright © 2012 ISO/IEC/IEEE. Used by permission.)"

28.2 SITE FUNDAMENTALS

To better understand the SITE tasks discussed later in this chapter, let's introduce some of the fundamentals that provide the foundation for SITE.

28.2.1 What Is System Integration, Test, & Evaluation (SITE)?

SITE is the sequential, bottoms-up process of:

1. Incrementally interfacing previously verified System Configuration Items (CIs) and items and integrating them beginning with the Part level into higher levels of abstraction - Part, Subassembly, Assembly, Subsystem, and Product-Level – via a series of Integration Points (IPs).
2. Conducting functional and QTs of the integrated test article to verify all capabilities comply with specification and design requirements.
3. Evaluating the test results for compliance and optimizing test article performance.

Completion of SITE at each Integration Point (IP) should occur as a formal verification test with formal Acceptance Test Procedures (ATPs). On completion of each ATP, conduct compliance assessments based on specification Section 4.0 (Table 20.1) verification requirements and methods for the Entity being verified. Each test provides *objective evidence* that proves the test article's compliance with a requirement to perform over the prescribed operating range of inputs and environmental conditions (Principle 19.20). For some

projects, lower level SITE verification tests are conducted *informally* or *semi-formally* with and without witnesses. This raises a key technical integrity and compliance issue: *lacking documented objective evidence, how does the project know that the test article:*

1. Is ready to be integrated into the next higher-level system?
2. Will seamlessly integrate without compliance issues from previous testing that waste valuable time?

In other cases, the tests:

- Are formal.
- Employ approved ATPs derived from specification requirements and verification methods.
- Are witnessed by all stakeholders including the Acquirer and User.

Depending on the contract, the Acquirer and User, at the request of the Acquirer, are invited to "witness" the SITE and verification activities. The System Developer is *accountable* for notifying the Acquirer regarding SITE in accordance with the Terms and Conditions (Ts&Cs) of the contract. Under normal contracting protocol (Principal 18.7), the Acquirer invites the User community to participate in multi-level acceptance testing User unless the Acquirer has made special arrangements with the System Developer to do so. Typically, this only occurs at the System Level. Conversely, when the Acquirer and User have *open* invitations to observe and witness SITE activities, as a professional courtesy to the System Developer, the Acquirer should inform the System Developer's Project Manager in advance concerning who, how many people, and what Enterprises(s) will participate.

28.2.2 SITE Objective

The *objective* of SITE is to:

1. Subject a test article of a SYSTEM, item, or Configuration Item (CI) to a range of Test Cases (TCs), input *stimuli*, *excitation*, and/or *cues*, and conditions representative of a prescribed OPERATING ENVIRONMENT.
2. Collect *objective evidence* such as documented test results and observations of the results of formal tests performed in accordance with approved ATPs.
3. Provide the objective evidence to the Verification Process to verify specification requirements for compliance.

Observe the third point "Provide the objective evidence to the Verification Process …" There are several nuances as to how this is accomplished.

1. In most Enterprises, SITE and the Verification Process are conducted *simultaneously* by personnel attending the ATP.
2. Recognize that SITE is sometimes conducted by testers that who have been certified to execute the ATP scripts and collect data but may lack the engineering expertise required to assess compliance.
3. Near the end of project, a Functional Configuration Audit (FCA) is conducted to formally audit and reverify the specification compliance results.

Engineers often have misperceptions of what is to be accomplished by SITE. The *erroneous* view is that a SYSTEM or ENTITY is subjected to a set of conditions bounded by *minimum* and *maximum* performance requirement thresholds. The fallacy in this view is: *how does the item perform when subjected to conditions below or above these limits?* Obviously, you could test an item beyond its physical limits such as environmental conditions, electrical overload, or shorts, but *what about improper data formats, magnitudes that are under/over range*? If the SYSTEM/ENTITY is designed properly, it should accommodate these OPERATING ENVIRONMENT conditions without failure.

The objective of SITE should be to exercise and assess the test article's capability to cope with *acceptable* and *unacceptable* input conditions (Figure 3.2) and OPERATING ENVIRONMENT conditions. Likewise, for those input conditions, produce only *acceptable* SYSTEM RESPONSES (Chapter 8) such as behavior, products, by-products, and services.

SITE, which is part of Developmental Test & Evaluation (DT&E) (Chapters 12, 13, and 16), is conducted following the Component Procurement and Development Process of the System Development Phase as illustrated in Figure 18.1. Throughout the SITE Segment, the System Developer conducts Test Readiness Reviews (TRRs) (Chapter 18) as *entry criteria* prior to testing to ensure that the:

1. Test architecture is properly configured and documented.
2. Test environment is controllable.
3. Test logs are readily available to logging tests.
4. Approved ATP and emergency procedures are in place.
5. Tests can be performed safely without injury to personnel or damages the test article, test environment, or facility.
6. System Design documentation is readily available for reference.

SITE activities culminate in a formal System Verification Review (SVR) prior to SYSTEM acceptance.

28.2.3 What Do SITE Activities Prove?

We can create marketing phrases about SITE such as *verify compliance* with specification requirements, but what do SITE activities really accomplish. In very simple terms, SITE answers the following five key questions:

1. Can the SYSTEM/ENTITY's design and physical components *interoperate* with external systems in its OPERATING ENVIRONMENT in compliance with specification interface requirements?
2. Does the SYSTEM/ENTITY *predictably* and *responsively* function as planned?
3. Can the SYSTEM/ENTITY's design and components survive the stresses of the prescribed OPERATING ENVIRONMENT conditions for the duration of the mission/operating cycle limits and perform in accordance with specification requirements and tolerances?
4. Is the quality of *workmanship* and *construction* sufficient to ensure the *integrity* of component interfaces?
5. As applicable, can the SYSTEM/ENTITY be easily deployed, operated, maintained, and sustained (Figure 6.4)?

SITE activities involve testing systems, product, or services that employ semantics need to be addressed. This brings us to our next topic.

28.2.4 SITE Test Article Semantics

SITE terminology includes terms such as *Engineering model, first article, test article*, and Unit-Under-Test (UUT). Let's explore each term briefly.

An Engineering Model, which may or may not be a contract deliverable, is typically an initial prototype. The model is used in collecting data to validate system models and simulations or in demonstrating technologies or proofs of concept. From a Spiral Development (Figure 15.4) perspective, Engineering Models of a SYSTEM, SUBSYSTEM, ASSEMBLY, or SUBASSEMBLY may be developed in *iterative* sequences as risk mitigation devices over a period of time to understand and drive out technical risk as pre-cursors to the initial First Article(s).

The term "*first article*" refers to the initial units of the approved Developmental Configuration (Chapters 12 and 16) that are available for verification testing and subsequent delivery in accordance with the Terms & conditions (Ts&Cs) of the contract. The term is sometimes a misnomer since several first articles may be produced from the Developmental Configuration. Collectively, a set or lot of devices is referred to as *first articles*. Each initial set of *first articles* is subjected to various *non-destructive* and *destructive* tests. On successful completion of the first article(s) SYSTEM Verification Test (SVT), the resulting Developmental Configuration is established as the Product Baseline (Chapter 18).

A first-article SYSTEM progressing through SITE is referred to as a *test article* during SITE. When the test article is integrated into the test configuration for verification testing, the item is referred to as a UUT.

The Developmental Configuration represents *first articles* that comply with System Performance Specification (SPS) requirements. However, that does not mean that the Development Configuration design can be *cost effectively* produced in terms of large-scale production.

When a decision is made to issue a production contract, the Developmental configuration may have to go through a series of Pre-Planned Product Improvements (P3I) to achieve cost-effective production solution. During this period, *production* versions of *first articles* may be verified as test articles and UUTs. Once the production design is reverified and baselined, mass produced items are submitted only for *functional testing* to ensure each item will perform and is free of workmanship and component issues.

28.2.5 Types of Testing

When Enterprises discuss Developmental Configuration SITE activities, you will often hear terms such as *functional, environmental, qualification, destructive, and non-destructive* testing. Let's define the context of each of these terms.

In general, SITE activities include two categories of testing: *Functional Testing* and *Environmental/Qualification Testing*.

- **Functional testing** refers to component tests—SYSTEM, SUBSYSTEM, ASSEMBLY, or SUBASSEMBLY. Once a design has been *verified* based on Environmental Qualification Testing (EQT), there is no need to reverify the design with each production article. As a result, each production article may be powered up to assess its fundamental capabilities—functions and performance to produce specified performance-based *outcomes* - and *interoperate* as planned with no errors under ambient conditions such as a laboratory. Some Enterprises refer to this as "functional testing."
- **Environmental/Qualification Testing (E/QT)** is the next higher level and focuses on *how well* the item performs in its prescribed OPERATING ENVIRONMENT conditions. Qualification testing, in general, includes tests performed as part of DT&E verification activities and, if applicable, during Operational Test & Evaluation (OT&E) validation activities. E/QT exposing the test article to actual field operating conditions such as temperature, humidity, shock, vibration, EMI, and so forth.

Although the operative term in SITE is *test*, SITE involves more than test activities, especially from a SYSTEM verification perspective. It may require technical demonstrations such Proof of Concept, Proof of Technology, and Proof of Principle technical demonstrations.

Technical demonstrations are sometimes performed as part of the DT&E procedures through the use of prototypes, technology demonstrations, and proof of concept demonstrations. These activities provide an excellent opportunity to assess and evaluate SYSTEM performance and obtain more insight into SYSTEM requirements and their refinements. In addition, sometimes, the results of technology demonstrations *are not* the actual test articles but the set of requirements and data derived from the technical demonstration.

28.2.5.1 Non-Destructive and Destructive Testing During Functional and Environmental/Qualification Testing (E/QT), test articles may be subjected to a wide range of tests that may destroy, blemish, or alter the test article's appearance, structural integrity, capability, or performance. In general, these are *non-destructive* tests may not harm the test article which may be refurbished for delivery, assuming that it is permissible by contract and safety practices.

In contrast, *destructive testing* often results in partial or complete destruction of the test article. Typically, the test article undergoes *non-destructive testing* to collect verification data. On completion of the *non-destructive* testing, the test article is subjected to *destructive testing*. In general, destructive testing includes drop tests from humans, test towers, or aircraft; crushing; exploding, and so forth.

Example 28.3 **Qualification Tests (QTs)**

QTs subject system and component test article(s) to *harsh* environmental test conditions in a NATURAL ENVIRONMENT or controlled laboratory or field test environments. Conditions may include shock, vibration, EMI, temperature, humidity, and salt spray to qualify the design for its intended application (in space, air, sea, land, etc.).

E/QTs are typically followed by a Formal Qualification Review (FQR) (Chapter 18), which assesses the verification results of the E/QTs.

During the System Production Phase of a project, production sample test articles may be randomly selected for a test project to assess the quality of materials, workmanship quality, SYSTEM capabilities and performance, shelf-life degradation, and so forth.

Commercial System Developers also conduct formal SYSTEM, PRODUCT, and SERVICE verification testing via field trials in the marketplace. Systems, products, and services evolve through a series of design iterations based on feedback from Users in field trials or test markets. Ultimately, marketplace *supply and demand* determine the public's response in terms of SYSTEM acceptability.

When all verified components have been integrated into the SYSTEM Level, formal System Verification Tests are performed on SYSTEMS and PRODUCTS at a designated facility and formally witnessed by the Acquirer and System Developer QA. A User representative(s) is typically invited by the Acquirer in accordance with contract protocol to participate (Principle 18.2). The purpose of the SVT is to prove that the SYSTEM OR PRODUCT fully *complies* with and meets its SPS requirements.

In preparation for the SVT, the System Developer prepares formal ATPs for review and approval by the Acquirer, depending on contract requirements. Prior to the SVT, a TRR (Chapter 18) may be conducted to determine the *state of readiness* of the test article(s) and supporting test environment—including EQUIPMENT, FACILITIES, PERSONNEL, PROCEDURAL DATA and MISSION RESOURCES Elements as well as processes, methods, and tools.

Acceptance Tests (ATs) are formal tests that fulfill the technical and legal criteria for acceptance of the SYSTEM or PRODUCT by the Acquirer for the User based on compliance with specification requirements. Compliance results provide a pre-requisite for the Acquirer to formally accept the SYSTEM.

In general, ATs are conducted with a set of formally approved and released ATPs that have been agreed to by the Acquirer, and the User, and the System Developer as applicable and by the System Developer. SYSTEM Level ATs:

1. Are derived from the SPS Section 3.0, Requirements.
2. Apply verification methods (Principle 22.4) identified in the SPS Section 4.0, *Qualification Provisions*, to verify accomplishment of each Section 3.0, Requirement(s).

For some projects, the term AT is synonymous with the SVT. In other cases, an AT may utilize a subset of the SVT ATPs after SYSTEM installation at a User site as a final verification prior to formal SYSTEM acceptance. ATs, as a generic term, are also used by Product Development Teams (PDTs) to demonstrate entity compliance to a higher-level team such as a System Development Team (SDT).

ATs are accomplished by *procedure-based* or *scenario-based* ATPs that have been approved typically by the System Acquirer.

28.3 KEY ELEMENTS OF SITE

Planning and implementation of SITE require an understanding of its key elements—PERSONNEL, EQUIPMENT,

FACILITIES, and other System Elements (Figure 8.13)—and *how* to orchestrate these elements during the test. We represent these System Elements via the SITE Architecture Block Diagram (ABD) shown in Figure 28.1.

A test article - UUT - serves as the System of Interest (SOI) for SITE. Surrounding the Test Article is the ENABLING SYSTEM Test Environment, which consists of the test operator, test procedures, test log, test equipment and tools, controlled test environment, and design documentation, all contained within the test facility.

Author's Note 28.2

Controlled OPERATING ENVIRONMENT

Note how the Controlled OPERATING ENVIRONMENT such as a thermal vacuum facility, shaker table, and EMI facility is *abstracted* as an *entity* rather than shown as surrounding the test article. Analytically, both methods are acceptable. Abstracting the Controlled OPERATING ENVIRONMENT as a box with interfaces *explicitly* reminds you of the need to identify and specify the interfaces. If we encompassed the test article within the Controlled OPERATING ENVIRONMENT (i.e., a box within a box), the interface relationships would be less explicit and may go undefined.

28.3.1 Guiding Philosophy of SITE

Principle 28.1

SITE Incremental Verification Principle

Integrate and incrementally verify *one and only one* test article and its capabilities at a time into a previously verified configuration.

The guiding philosophy of SITE can be summarized in a few words: *keep it simple*! Simplicity means verify each ENTITY – PRODUCT, SUBSYSTEM, ASSEMBLY, SUBASSEMBLY, or PART - to its specification requirements and then *incrementally* integrate other verified items one at a time. To illustrate this point, consider the following mini-case study.

Mini-Case Study 28.1

Failed SITE Test – How to Make a Problem Even Worse

Engineers often have the *misperception* that System Entities—PRODUCTS, SUBSYSTEMS, ASSEMBLIES, and so forth can be *lashed together* all at once into a massive test configuration and begin testing! If you do this, you may and probably will be confronted with a large number of *simultaneous* problems with no clue as to where the sources of the problems originate. This is can be *potentially dangerous* and *catastrophic*.

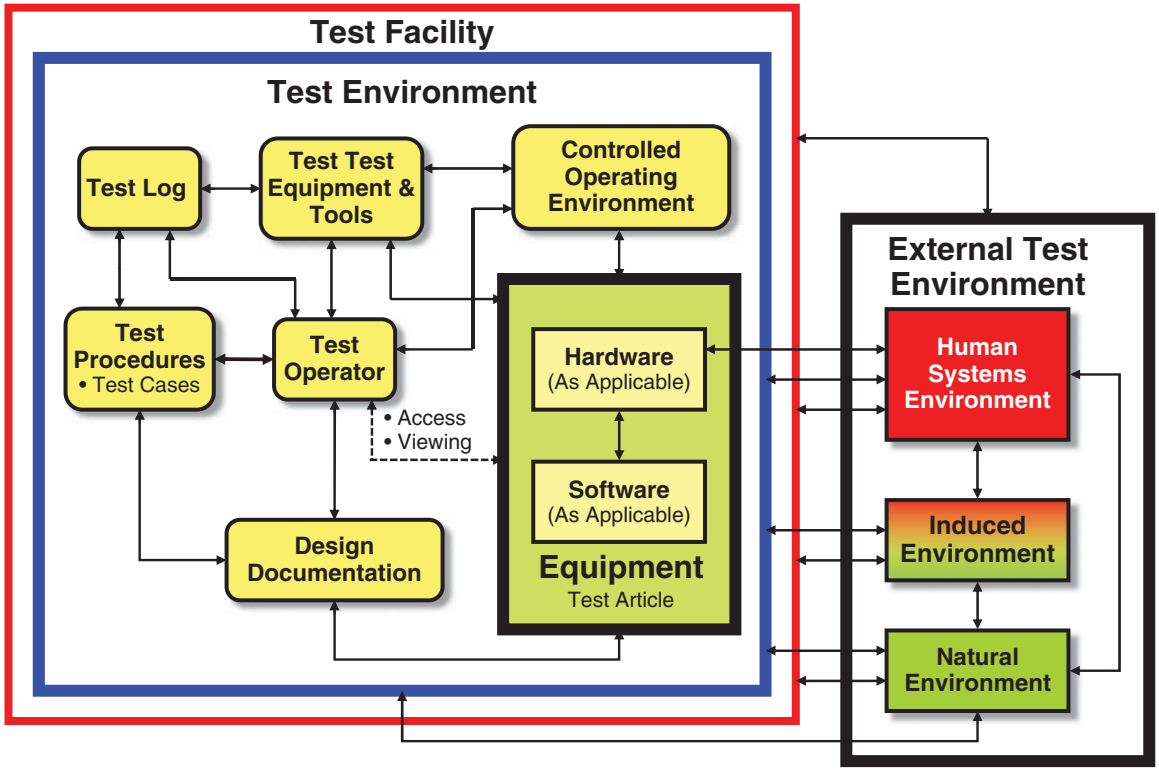


Figure 28.1 Example System Integration, Test, & Evaluation (SITE) Architecture and its Elements

If you attempt to introduce more than one *unproven* capability simultaneously, you can *naively* destroy the test article or even worse result in *injury* to the Test Operators or *damage* to facilities – the Law of Unintended System Consequences (Principle 3.5). This requires Systems Thinking (Chapter 1) and Test Readiness Reviews (TRRs) to assess risk how the test and its consequences will be conducted (Principle 26.15).

If a test fails to perform properly, *how will you ever sort out and determine where the source(s) of the problem(s) originate?* The first thing Engineers often do when something does not work is to start disturbing the test configuration. Specifically, they physically disturb the "scene of the incident" by disconnecting cables, removing components, or making adjustments. Then, the problem and challenge gets even larger and more complex. Suppose the source of the problem was a mistake made by the Test Operator in executing the test procedures. Now you have *invalidated* the test configuration and results. The problems just keep getting worse.

You will soon discover that you need to *disassemble* everything and start with a simple, proven, and verified item, integrate another other verified items with it, and incrementally integrate the SYSTEM. Remember, SITE fault isolation wisdom (Principle 28.1) says 'integrate one and only *one verified* test article and capability at a time! At least you will know *where* and *when* the problem entered the test article configuration and thereby can simplify troubleshooting and fault isolation. For that test article, inhibit, disable, or disconnect as many capabilities as you can and incrementally test and enable one new capability at a time until all capabilities have been verified.

28.3.2 Fault Isolation

Principle 28.2

Fault Isolation Principle

To isolate a fault in a serial configuration chain, employ the Half Split Method and iterate until the fault is located.

As stated numerous times beginning with Chapter 3, SYSTEMS/ENTITIES are stimulus-response mechanisms that produce performance-based behaviors and outcomes as a function of their inputs – *stimuli, excitations,* or *cues* (Principle 3.3). Mathematically, we can express the relationship between its outputs and inputs as a *transfer function* (Chapter 10). Implementation of the transfer function requires User Command & Control (C2) of Modes of operation (Chapter 7) into chains of serial, parallel, and serial-parallel strings (Figure 34.12) of PRODUCTS, SUBSYSTEMS, ASSEMBLIES, and so forth. During SITE when an expected outcome fails to be produced, *how do you isolate the fault and its location within the serial, parallel, and serial-parallel strings*?

First, if the test article being integrated has already been verified as a UUT, the logical starting point for trouble shooting might be the interface that has been introduced, assuming that there have been no internal component failures on either side of the interface. If this checks out, the next question is: *which side of the interface is the source of the problem*? This point illustrates a key concept in SITE - referred to as the Half Split Method (Principle 28.2) for isolating a fault.

The Half Split Method is based on a Divide and Conquer approach. Specifically in a chain of components, drop back to a halfway point in the chain and verify that the outcome at that test point is correct. This enables you to determine if the fault is occurring "upstream" or "downstream" from that test point. Once you know which side of the test point is the region in question, repeat the exercise until the fault is isolated.

28.3.3 Preparing Items for SITE

During the System Design Process (Figure 12.2) of the System Development Phase, we *partitioned* and *decomposed* the System Architecture into successively lower levels of abstraction and Entities to the PART Level. During the Component Procurement and Development Process (Figure 12.2), each component is procured/fabricated, coded, assembled, and tested. Incrementally, we verify each component to ensure that it complies with its *design requirements* such as drawings and schematics. Levels of integrated components – SUBASSEMBLIES → ASSEMBLIES → SUBSYSTEMS → PRODUCTS – are verified for compliance to its respective specification requirements (Figures 12.3 and 15.2).

System integration is actually a verification exercise whereby components are verified for *form, fit, and function* (Chapter 3) to evaluate their (1) *compatibility* - physically integrated - "form and fit" checking and (2) *interoperability* – "function" - is demonstrated or tested and verified based on the integrated set of *capabilities* and *interfaces*. Consider the following example:

Example 28.4

Compatibility and Interoperability Testing

Assume we integrate two Level 2 SUBSYSTEMS that have been *verified* as compliant to their Entity Development Specifications (EDSs). The integration forms a Level 1 SYSTEM (Figure 28.2) that will be verified for compliance to its SPS. When we physically integrate the SUBSYSTEMS – *compatibility* – and conduct *interoperability* tests for the SYSTEM, we are verifying the integrated set of capabilities and interfaces. This

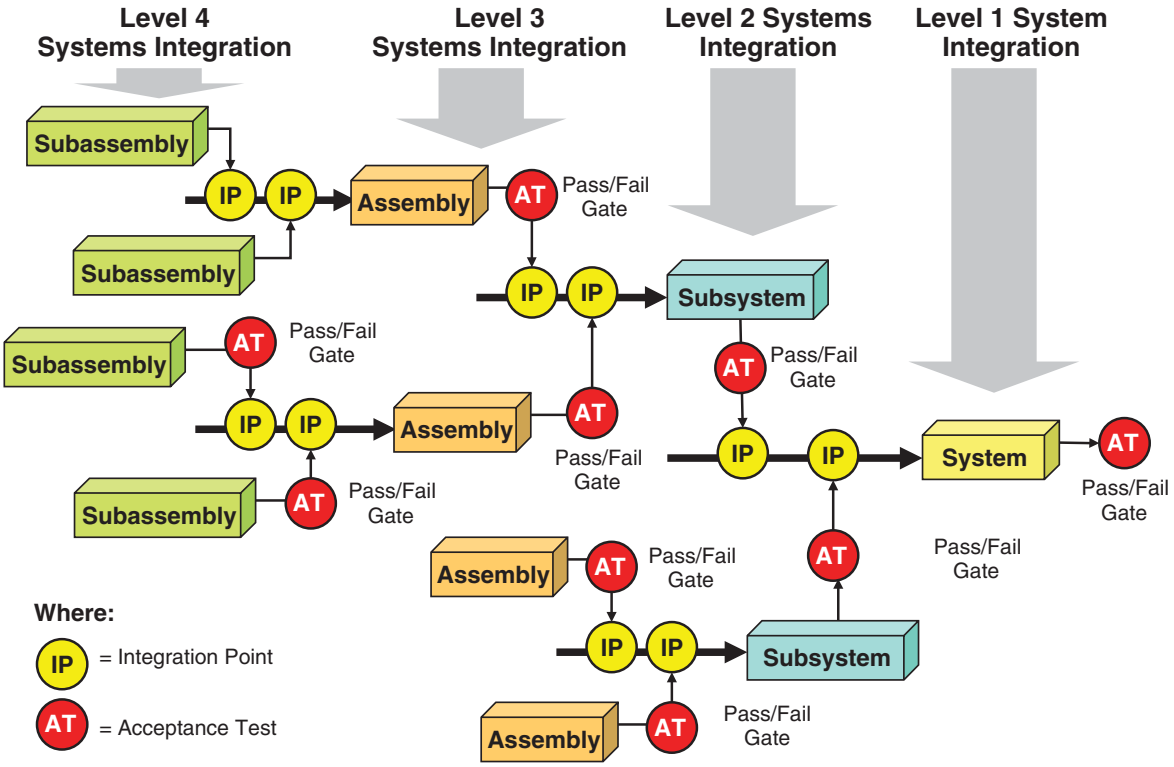


Figure 28.2 Test Discrepancy (TD) Fault Isolation Tree for Use in Analyzing Discrepancy Reports (DRs)

occurs at all levels of integration and the SYSTEM's or PRODUCT's integration at the User's Level 0 SYSTEM.

In each of these cases, the objective is to create an interface that is representative or realistic of the external OPERATING ENVIRONMENT through simulation, emulation, or stimulation (Chapter 33), the test article is unable to discern the test from reality. This point brings us to our next topic.

28.3.4 Creating the Test Article's OPERATING ENVIRONMENT

When we perform SITE, we subject the test article to OPERATING ENVIRONMENT conditions and scenarios. This requires creating the HIGHER ORDER SYSTEMS (Figure 9.3) and PHYSICAL ENVIRONMENT Elements (Figure 9.4). These include the HUMAN SYSTEMS, INDUCED, and NATURAL ENVIRONMENTS. *How do we do this?*

There are several options for creating the OPERATING ENVIRONMENT as illustrated in Figure 28.3. We can *simulate, stimulate, or emulate* entities within the OPERATING ENVIRONMENT or employ combinations of these options.

- **Simulate** To create a virtual model interface that represents the performance-based behavioral responses and characteristics of an external system.
- **Stimulate** To create an interface using actual equipment or a test set that has identical physical performance characteristics of the external system.
- **Emulate** To create an interface that identically mimics the actual operations, processing sequences, and performance characteristics of an external system.

Once we have created the Test Environment and ready for testing, we need to establish some *ground rules* for ensuring *consistency* and *continuity* of testing operations. This brings us to our next topic, SITE conduct Operating Constraints.

28.3.5 SITE Conduct Operating Constraints

Two types of operating constraints govern SITE conduct: (1) Standard Operating Practices and Procedures (SOPPs) and (2) ATPs.

28.3.5.1 Standard Operating Practices and Procedures (SOPPs) Testing SOPPs are Enterprise command media—policies and procedures—that apply to test conduct in test facilities and ranges. SOPPs focus on the protocols and safe and proper handling of EQUIPMENT – test articles, UUTs, and tools, human and environmental safety, laboratory equipment/test range procedures, security procedures, emergency procedures, as well as the prevention of hazards.

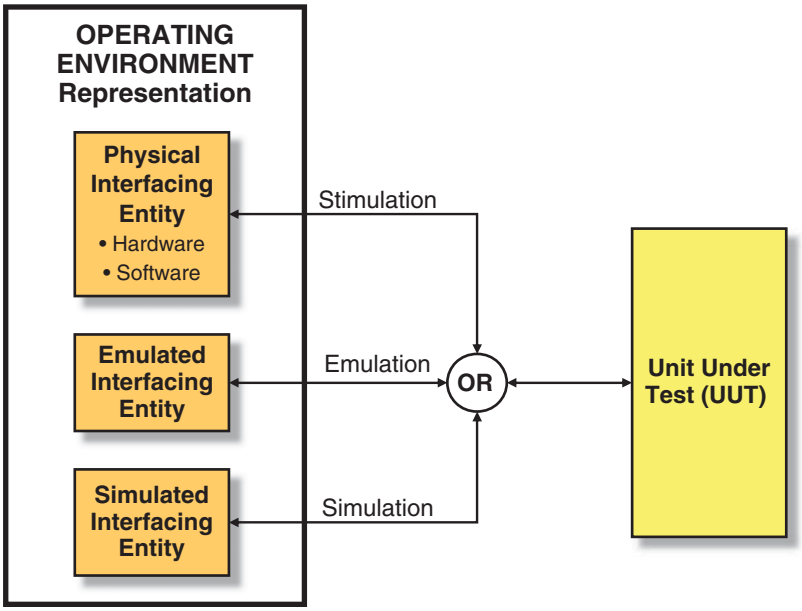


Figure 28.3 Example of a System Integration & Test Tree Featuring Entity Integration Points (IPs) and Acceptance Tests (ATs)

28.3.5.2 Acceptance Test Procedures (ATPs)

Principle 28.3

Single Test-Multiple Requirements Principle

Strive to verify as many specification requirements as practical with a single test, if appropriate.

ATPs are derived from each SPS or EDS Section 3.0 requirements using their Section 4.0 Qualifications Provisions verification methods. Since a key objective is to *minimize* SITE costs and effort (Principle 22.4), some SPS or EDS Section 3.0 Requirements should be verified simultaneously with a *single test*. ATPs should reflect this approach and reference the requirements being verified.

28.3.6 Who Performs Formal Tests?

Questions often emerge regarding who should perform the PART, SUBASSEMBLY, ASSEMBLY, SUBSYSTEM, PRODUCT, and SYSTEM-Level tests. Two issues emerge from these questions (1) Testing Personal Work Products and (2) Certifying Test Personnel.

Issue 1: Testing Personal Work Products

Principle 28.4

Testing Conflict of Interest Principle

Testing and verification of personal work products, especially contract deliverables, is viewed as a *conflict of interest* and should be objectively performed by an independent tester and verifier.

Test activities range from *informal* to *highly formal*. SYSTEM designers *should not* test their own designs, provided that someone else is capable. The underlying philosophy of this principle is that it represents a potential *conflict of interest* in objectively testing and checking off your own work. In most Enterprises, SE designers develop and may perform *informal* testing of their work products, typically with peer level review scrutiny. Some Enterprises assign Independent Test Teams (ITTs) and assign them to perform testing at all levels of integration. Some contracts also employ Independent Verification and Validation (IV&V) (Chapter 13) contractor teams to perform testing and verification, especially for software.

Issue 2: Certifying Test Personnel

Testing requires knowledge, discipline, observational skills, adherence to safety practices, integrity, accuracy, and precision in reporting test results. In general, due to the legal implications, formal ATP testing is not for amateurs; it requires training and experience, and often certification. Therefore, Enterprises should establish in-house command media policies that only testers, who have been trained and certified, are authorized for a defined period of time to perform tests.

28.3.7 Simultaneous Testing Strategy for Multiple Requirements

Testing can rapidly become very expensive and consume valuable schedule resources. Remember, you only employ

Verification by Test where Verification by Inspection, Analysis, and Demonstration is *insufficient* (Figure 22.2) to demonstrate full compliance with a requirement.

There are ways to *efficiently* and *effectively* perform tests to reduce costs and schedule Principle 22.4. Some people *erroneously* believe that you conduct one test for each requirement. In general, analyze SPS or EDS requirements to identify dependencies and sets of requirements that can be verified simultaneously.

Author's Note 28.3

Multiple Requirements Verification Strategy

Figure 21.7 illustrates a key point concerning the need to leverage *one* test to *verify* several specification requirements. We can do this by formulating a single test – Test Case - that will enable us to verify a series of specification requirements (Principle 28.3). As a result, we employ a (1) *single* test to verify Requirements A_111 → A_112 for SUBSYSTEM A's EDS and (2) *single* test to verify Requirements B_111 → B_112 → B_113 for SUBSYSTEM B's EDS. When SUBSYSTEMS A & B are integrated at the SYSTEM Level, we can then employ a single test to verify the Requirements A_11 → B_11 thread as accomplishment of SPS Requirement SYS_11.

28.3.8 What Is Regression Testing?

Recognizing that you want to minimize the amount of rework and retest, the challenge question is: *if you discover a latent defect—design flaw, deficiency, error, poor workmanship, or component defects—that requires corrective action, what test results may have been impacted by the failure must be repeated before you can resume testing at the test sequence where the failure occurred*? The answer resides in Regression Testing.

During Regression Testing, only those aspects of the *Entity's* design that were affected by a *corrective action* are re-verified. Previously performed formal tests of the UUT that were successfully completed and not affected by any corrective actions typically are not re-verified.

28.3.9 Discrepancy Reports (DRs)

Principle 28.5

Discrepancy Reporting Principle

Discrepancy Reports (DRs) document a *discrepancy* found during test article verification, not the source of the problem.

Principle 28.6

Corrective Actions Principle

Disposition Discrepancy Reports (DRs) quickly for analysis to determine *root cause* and *corrective actions*; allocate resources based on the *level of significance* and *priority*. Where appropriate, collaborate with the Acquirer Technical Representative (ATR).

Inevitably, discrepancies occur between *actual test data* and *expected results* specified in the SPS, EDS, or design requirements such as drawings, schematics, and parts lists. We refer to the occurrence of a discrepancy as a *test event* or *incident*.

When test events such as *anomalies* or *failures* occur, a Discrepancy Report (DR) is documented by the Test Operator as a Test Event. The Test Director should establish ground rules for what qualifies as a *test event* in the System Integration and Verification Plan (SIVP) and *event criteria* for documenting a DR. At a *minimum*, DRs document:

1. The test event, date, time.
2. Tester's name and ID.
3. Test article name and identification by Model and Serial Number.
4. ATP document title, number, revision, and date.
5. Conditions and prior sequence of steps preceding a test event.
6. Test article identification.
7. Test architecture and environment configuration.
8. Reference documents and versions.
9. Specification requirement and expected results.
10. Results observed and recorded.
11. DR author and witnesses or observers.
12. Degree of significance requiring a level of urgency for corrective action.

DRs have levels of significance that affect the test schedule. They range from safety issues, data integrity issues, isolated tests that may not affect other tests, and cosmetic blemishes in the test article. As standard practice, establish Classification Systems such as the one shown in Table 28.1 to facilitate real-time decision-making concerning DRs.

28.3.10 SITE Work Products

SITE *work products* that serve as ISO 9000 *objective evidence* for a SYSTEM OR ENTITY, regardless of level of abstraction, include the following:

1. A set of dated and signed entries in the Test Log that identify and describe:
   a. Test Team—the name of the responsible Engineering team and lead.
   b. Test Article—What level of test article was created by integrating what version of lower level test articles - identify by Model Number and Serial Number.

TABLE 28.1 Example Test Event DR Classification System

Priority	Test Event	Event Description
1	Emergency condition	All testing must be **TERMINATED IMMEDIATELY** due to imminent **DANGER** to the Test Operators, test articles, or test facility.
2	Test component or configuration failure	Testing must be **HALTED** until a corrective action is performed. Corrective action may require redesign or replacement of a failed component.
3	Test failure	Testing can continue if the failure does not diminish the integrity of remaining tests; however, the test article requires corrective action reverification prior to integration at the next higher level.
4	Cosmetic blemish	Testing is permitted to continue, but corrective action must be performed prior to SYSTEM acceptance.

   c. Test(s) cases and procedures performed.
   d. Test environment configuration.
   e. Test results—where recorded and stored.
   f. Problems, conditions, or anomalies encountered and identified.
2. Discrepancy Reports (DRs).
3. Hardware Trouble Reports (HTRs).
4. Software Change Requests (SCRs).
5. State of readiness of the test article for scheduling formal verification.

In summary, when a Test Discrepancy (TD) occurs, the Test Operator logs the test event and submits a Discrepancy Report (DR) that does not assess the source of the problem. Later, analysis of the TD may reveal the source of the problem to be Hardware, Software, or both, for example. As a result, a Hardware Trouble Report (HTR) or a Software Change Request (SCR) is submitted for approval and subsequent corrective action. This completes our overview of SITE fundamentals. Let's shift our attention to Planning for SITE.

28.4 PLANNING FOR SITE

SITE success begins with insightful planning to identify the test objectives; roles, responsibilities, and authorities; tasking, resources, and facilities; and schedule. Testing, in general, involves two types of test plans for new system development:

- The Test and Evaluation Master Plan (TEMP).
- The System Integration and Verification Plan (SIVP).

28.4.1 The Test and Evaluation Master Plan (TEMP)

In general, a TEMP is a User's document that:

1. Identifies Critical Operational or Technical Issues (COIs/CTIs) concerning integration of the System being acquired into the User's Level 0 SYSTEM (Figure 8.4) to conduct missions.
2. Expresses how the User or an Independent Test Agency (ITA) representing the User plans to *validate* a SYSTEM, PRODUCT, OR SERVICE.

From a User's perspective, development of a new system raises COIs/CTIs that could become *showstoppers* to validating satisfaction of an Enterprise's operational need. Therefore, the scope of the TEMP covers the OT&E period and establishes objectives to verify resolution of COIs/CTIs.

The TEMP is structured to answer a basic question: *does the* SYSTEM, PRODUCT, *or* SERVICE, *as delivered, satisfy the User's validated operational needs in terms of resolving the problem or issue space* (Figure 4.3)? Answering this question requires formulation of a set of scenario-driven test objectives and TCs based on UCs and scenarios.

28.4.2 The System Integration and Verification Plan (SIVP)

The SIVP is written by the System Developer and *expresses* their approach for integrating, testing, and verifying the SYSTEM or PRODUCT. The scope of the SIVP, which is contract dependent, covers the DT&E period from Contract Award through the formal SVT (Figure 13.6), typically at the System Developer's facility.

The SIVP identifies objectives, Enterprise roles and responsibilities, tasks, resource requirements, integration strategy for sequencing testing activities, and schedules. Depending on contract requirements, the SIVP may include delivery, Installation and Checkout (I&CO) at a User's designated job site.

28.4.3 Developing the System Integration, Test, and Evaluation (SITE) Strategy

The strength of SITE requires "up front" Systems Thinking (Chapter 1) to ensure that the vertical integration occurs Just-in-Time (JIT) in the proper sequences. Therefore, the first step is to establish a SITE strategy.

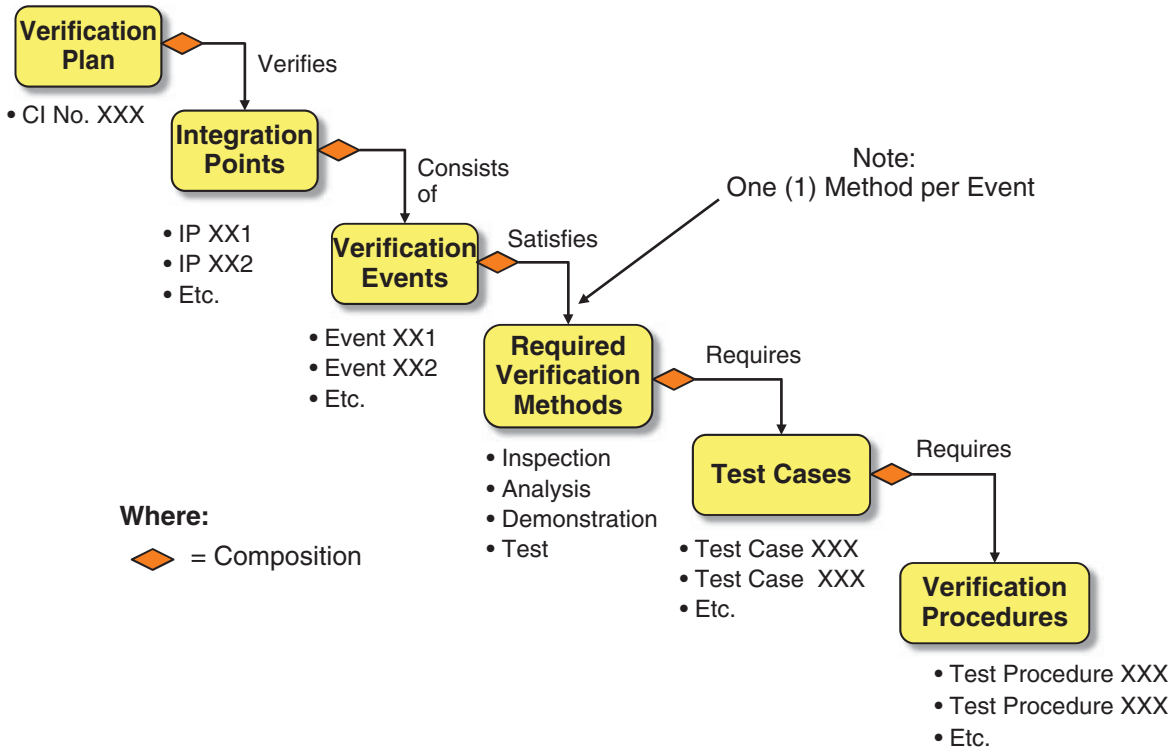


Figure 28.4 Integration Point (IP) Entity Relationships (ERs)

One method is to construct a System Integration and Test Concept graphically via a series of IPs into an integration decision tree as shown in Figure 28.4. The test concept should reflect the following:

1. What component integration dependencies are critical?
2. Who is responsible and accountable for the integration?
3. When and in what sequence they will be integrated?
4. Where is the integration to be performed?
5. How verified components will be integrated into higher level systems?

The SITE Process may require a single facility such as a laboratory or multiple facilities within the same geographic area, or integration across various geographical locations via the Internet or some form of secure Intranet.While engineers can design systems well and understand the basic concept of system integration, a common shortcoming is the ability to develop a strategy for planning the logical and sequential steps integrating and testing multi-level entities such as Figure 28.2. It is as if they are taking a test and afraid of being wrong. Make a decision! Then, validate it with your colleagues and take corrective action, as necessary.

Beginning with a foundation of one verified item in Figure 28.2, observe how lower level entities such as a SUBASSEMBLY is integrated into the existing level of integration at an IP. An AT is performed on the integrated set of entities. On successful completion of this series of IPs and ATs for SUBASSEMBLIES, ASSEMBLIES, and SUBSYSTEMS, a SYSTEM-Level ATP is performed.

Once the general SITE strategy of IPs and ATPs is established, the next step is to detail how the sequential steps will be orchestrated and documented. One approach is to establish an Entity Relationship Diagram (ERD) such as the one shown in Figure 28.4.

Each SIVP should define the approach for SITE IPs and ATPs based on ERs such as:

- Each IP may consist of one or more Verification Events such as Event XX1 and Event XX2, representing a specific specification requirement or sets of specification requirements to be verified.
- Each Verification Event may require one or more Verification Methods such as Inspection, Analysis, Demonstration, and Test assigned as verification requirements to be performed for each specification requirement or sets of specification requirements.
- Each Verification Method may require one or more TCs representing how a specification requirement should be tested in terms of *Acceptable* and *Unacceptable* Input and Output ranges illustrated in Figure 3.2.

- Each Test Case may require one or more ATPs that have been approved for verifying each specification requirement.

28.4.4 Destructive Test Sequence Planning

During the final stages of SITE DT&E, several test articles may be required. The challenge for SEs is: *how and in what sequence do we conduct non-destructive tests to collect data to verify design compliance prior to conducting destructive test that may destroy or damage the test article? Think* through these sequences carefully. Once the SITE plans are in place, the next step requires establishing the test organization.

28.5 ESTABLISHING THE TEST ORGANIZATION

One of the first steps following approval of the SIVP is establishing the test organization and assigning test roles, responsibilities, and authorities. Key roles include Test Director, Lab Manager, Tester, Test Safety Officer or Range Safety Officer (RSO), Quality Assurance (QA), Security Representative, and Acquirer/User Test Representative.

28.5.1 Test Director Role

The Test Director is a member of the System Developer's project and serves as the key decision authority for planning and orchestrating testing activities. SITE activities bear the impact of (1) poor specification requirements that have multiple *interpretations* (Principle 19.9) that may result in potential conflicts with the Acquirer or User and (2) failure to accommodate test ports and test points to collect test data, especially after test articles are sealed following verification. Therefore, the Test Director role should be assigned early and be a key participant in developing the SE Design Process strategy (Figure 12.3, 12.6, and 13.6).

At a *minimum*, primary responsibilities for the Test Director include:

1. Develop and implement the SIVP.
2. Chair the TEWG, if applicable.
3. Plan, coordinate, and synchronize Test Team task assignments, resources, and communications.
4. Exercise authoritative control over the test configuration and environment.
5. Identify, assess, and mitigate test risks.
6. Review and approve test conduct rules, TCs, and test procedures.
7. Account for SITE Occupational, Environmental, Safety, and Health (OES&H).
8. Train and certify test personnel.
9. Collaborate with the User or their Acquirer Test Representative (ATR), when applicable, to prioritize and expeditiously disposition DRs and test issues.
10. Verify DR corrective actions.
11. Accomplish contract test requirements.
12. Archive and preserve test data and results.
13. Conduct test failure investigations.

28.5.2 Lab Manager Role

The Lab Manager is a member of the System Developer's project and supports the Test Director. At a *minimum*, the primary Lab Manager responsibilities include:

1. Participate in the development of the SPS and EDSs to identify SITE test equipment for acquisition, calibration, and alignment.
2. Implement the test configuration and environment.
3. Acquire test tools and equipment.
4. Establish laboratory SOPPs for the project.
5. Ensure test tools and equipment are properly *calibrated* and *aligned*.
6. Create, review, and maintain the Test Log.
7. Plan and implement Test Operator training can certification.

28.5.3 Test Safety Officer (TSO) or Range Safety Officer (RSO) Role

Since testing often involves unproven designs and test configurations, safety is a critical issue, not only for test personnel, but also for the test article and facilities. Therefore, every project should designate a Test Safety Officer (TSO).

In general, there are two types of Test Safety Officers (TSOs): SITE Test Officer and Range Safety Officer (RSO).

- The Test Safety Officer is a member of the System Developer's Enterprise and supports the Test Director (TD) and Lab Manager.
- The RSO is a member of a test range, if applicable.

In some cases, RSOs have authority to destruct test articles such as rockets and missile should they become *unstable* and *uncontrollable* during a test flight or mission and pose a threat to PERSONNEL, FACILITIES, EQUIPMENT, and/or the public.

28.5.4 Test Operator Role

As defined by Principle 28.4, SYSTEM, PRODUCT, or SERVICE designers should not test their own designs due to the potential COI and objectivity. However, at lower levels of abstraction, projects often lack the resources to adequately train Test

Operators. So, System Developer personnel often perform their own informal testing. For some contracts, Independent Verification & Validation (IV&V) Teams (Chapter 13) *internal* or *external* to the project or Enterprise may perform the testing.

Regardless of *who* performs the Test Operator role, they must be trained in *how* to safely perform the test, formally record and document results, and deal with anomalies. Some Enterprises formally train personnel and refer to them as Certified Test Operators (CTOs).

28.5.5 QA Representative

At a *minimum*, the System Developer's Quality Assurance (QA) representative is responsible for assuring compliance with contract and specification requirements, Enterprise and project command media, the SIVP, and ATPs. For software-intensive SYSTEM development efforts, a Software Quality Assurance (SQA) representative may also be assigned to the project.

28.5.6 Security Representative

At a *minimum*, the System Developer's Security Representative, if applicable, is accountable for the assuring compliance with contract security requirements, Enterprise and project command media, the project's security plan, and ATPs.

28.5.7 Acquirer Test Representative (ATR)

Acquirer Technical Representative ATR Principle

Principle 28.7 Every System Acquirer should appoint one Test Representative who serves with authority as *the* single Voice of the Customer (VOC) representing the Acquirer-User community concerning SITE decisions.

Throughout SITE, test issues emerge that require an Acquirer decision. In addition, some Acquirers represent several Enterprises, many with *conflicting* opinions and agendas. This presents a challenge for a System Developer.

One solution is for the Acquirer Project Manager (PM) to appoint *one* individual to serve (1) as their *on-site Acquirer Technical Representative (ATR) at the System Developer's facility and (2) single voice representing all Acquirer viewpoints and decisions. Primary responsibilities of the ATR are to:*

1. Serve as the *single, technical point of contact* for all Acquirer and User technical and contract interests and communications.
2. Collaborate with the Test Director to resolve any SITE-related COIs/CTIs that affect Acquirer-User interests.
3. Where applicable by contract, collaborate with the Test Director to assign priorities for DRs resolution and corrective action.
4. Where required by contract, review and coordinate approval of ATPs.
5. Where appropriate, provide a single set of ATP comments that represent a consensus of the Acquirer-User Enterprises.
6. Witness and formally approve ATP results.

28.6 DEVELOPING TEST CASES (TCs) AND ACCEPTANCE TEST PROCEDURES (ATPs)

In general, ATPs provide the scripted strategy to verify compliance with SPS or EDS requirements. In Chapter 21 (Figures 21.6 and 21.7), we discussed that an SPS or EDS is derived from SYSTEM or ENTITY UCs and scenarios task-based capabilities. These are based on *how* the User envisions deploying, operating, maintaining, sustaining, retiring, and disposing of the SYSTEM, PRODUCT, or SERVICE. SPS and EDS capability requirements address this range of usage.

ATPs should be written for a specific specification requirement or more preferably sets of requirements (Principle 28.3) specification requirements. Although it is true ATPs have linkages to specification requirements, the linkages are indirect via TCs. Those who understand SE methods recognize that the UCs provide a foundation for developing TCs that stress the System's capabilities for a specified OPERATING ENVIRONMENT. One UC, for example, might lead to the identification of one to "n" TCs that represent test scenarios that stress the System/Entity's inputs and outputs over *acceptable* and *unacceptable* ranges (Figure 3.2 and 20.4). When identified, TCs become the basis for development of ATPs.

In general, the ATPs script the prescribed steps required to demonstrate that the system, product, or service provides the specified set of capabilities documented in the SPS or EDS. To establish a test strategy, we begin with TCs that verify these capabilities as shown in Table 28.2.

ATPs are generally of two types: (1) *procedure based* and (2) *scripted scenario* based (ATs).

28.6.1 Procedure-Based ATPs

Procedure-based ATPs provide detailed, scripted instructions that describe test configurations, environmental controls, expected results and behavior, and other details. Test Operators are required to follow prescribed, multi-step scripts to establish a specific test configuration, set switch

TABLE 28.2 Derivation of TCs

Phase of Mission Operation	Mode of Operation	Use Cases (UCs)	UC Scenarios	Required Operational Capability (ROC)	Test Cases (TCs)
Pre-Mission Phase	Mode 1	UC #1.0		ROC #1.0	TC #1.0
			Scenario #1.1	ROC #1.1	TC # 1._
				…	TC # 1._
			Scenario #1.*n*	ROC #1.*n*	TC # 1._
				…	TC # 1._
	Mode 2	UC #2.0		ROC #2.0	TC # 2.0
			Scenario #2.1	ROC #2.1	TC # 2._
				…	TC # 2._
			Scenario #2.*n*	ROC #2.*n*	TC # 2._
				…	TC # 2._
		UC #3.0		ROC #3.0	ROC #3.0
			Scenario #3.1	ROC #3.1	TC # 3._
				…	TC # 3._
			Scenario #3.*n*	ROC #3.*n*	TC # 3._
				…	TC # 3._
Mission Phase	Etc.				
Post-Mission Phase	Etc.				

Note: TCs shown in the table symbolically represent a range of TC variations. As previously discussed, employ a variety of TCs to test the SYSTEM/ENTITY's inputs/outputs over *acceptable* and *unacceptable* ranges as illustrated in Figures 3.2 and 20.4.

positions, and control inputs to stimulate or enter into the SYSTEM and document the results. Each test procedure step identifies the expected results for verification against specification requirements. Consider the example shown in Table 28.3 for a secure Web site log-on test by an authorized user.

28.6.2 Scenario-Based ATPs

Scenario-based ATPs provide general guidance in the form of operational scenarios for use during DT&E or OT&E (Chapter 12) field exercises. Detail actions such as switch settings – *switchology* - and configuring the software required to operate the SYSTEM are left to the SYSTEM Test Operator.

Scenario-based ATPs employ objectives or missions as key drivers for Test or Demonstration verification methods. Thus, the ATP tends to involve very-high-level statements that describe the operational mission scenario to be accomplished, objective(s), expected outcome(s), and performance. In general, scenario-based ATPs defer to the Test Operator's knowledge of an aircraft, for example, as a surrogate User of the SYSTEM to determine which sequences of switches and buttons — switchology —to use based on operational familiarity with the test article.

Scenario-based ATP data sheets generally include a field for recording the actual measurements and observations. Designated witnesses from the System Developer, Acquirer, and User Enterprises as well as System Developer QA/SQA representatives witness and authenticate the ATP results as Quality Records (QRs).

28.7 PERFORMING SITE TASKS

SITE activities are more than conducting tests. As a "SITE System," they are mission-oriented, provide capabilities, and consist of three phases of operation: Pre-Testing Phase, Testing Phase, and a Post-Testing Phase (Figure 10.17). Each phase consists of a series of tasks for integrating, testing, evaluating, and verifying the design of a SYSTEM or ENTITY. *What occurs during each of the phases*? Remember, every system is unique. The discussions that follow represent *generic* test tasks that apply to every level of abstraction. These tasks are *highly interactive* and may iterate numerous times within a Mission Phase of Operation, especially in the Testing Phase.

Task 1.0: Perform Pre-Test Activities

- Task 1.1: Develop TCs and ATPs
- Task 1.2: Configure the Test Environment (Figure 28.1).
- Task 1.3: Prepare and instrument the test article(s) for SITE.
- Task 1.4: Integrate the Test Article into the Test Environment (Figure 28.1).
- Task 1.5: Perform a test readiness inspection and assessment.

Task 2.0: Test and Evaluate Test Article Performance

- Task 2.1: Perform informal testing.
- Task 2.2: Evaluate informal test results.

TABLE 28.3 Example Procedure-Based AT Results.

Test Step	Test Operator Action to be Performed	Expected Results	Measured, Displayed, or Observed Results	Pass/Fail Results	Operator Initials and Date	QA
Step 1	Using the left mouse button, double click on the web site icon	Web browser is launched to selected web site	*Web site appears*	*Pass*	*JD 4/18/XX*	QA 208 *KW 4/18/XX*
Step 2	Using the left mouse button, click on the "Logon" button.	Logon access dialog box opens up.	*As expected*	*Pass*	*JD 4/18/XX*	QA 208 *KW 4/18/XX*
Step 3	Position the cursor within the User Name field of the dialog box.	Fixed cursor blinks in field	*As expected*	*Pass*	*JD 4/18/XX*	QA 208 *KW 4/18/XX*
Step 4	Enter user ID (max. of 10 alphanumeric characters)	Field displayed	*User ID entered*	*Pass*	*JD 4/18/XX*	QA 208 *KW 4/18/XX*

- Task 2.3: Optimize design and test article performance.
- Task 2.4: Prepare test article for formal verification testing.
- Task 2.5: Perform a "dry run" test to check out ATP.
- Task 2.6: Collaborate with the ATR to invite Acquire and Users (Principle 18.2) to witness formal acceptance tests.
- Task 2.7: Coordinate with internal QA /SQA to witness test.

Task 3.0: Verify Test Article Performance Compliance

- Task 3.1: Conduct a Test Readiness Review (TRR) (Figure 18.1).
- Task 3.2: Formally verify the test article.
- Task 3.3: Authenticate and test results and data.

Task 4.0: Perform Post-test Follow-up Actions

- Task 4.1: Prepare item Verification Rest Reports (VTRs).
- Task 4.2: Archive test results and data.
- Task 4.3: Disposition Discrepancy Report (DR) analysis and corrective actions.
- Task 4.4: *Refurbish/recondition* test article(s), if permissible, for delivery.

Formal tests are scripted using approved ATPs, which identify test articles, define Test Environment configurations, TCs, test procedures, and so forth. Tests often have problems with cabling, test article failure, ATP issues, and so on. Therefore, informal "dry runs" are encouraged to ensure everything is working properly prior to conducting a formal ATP, especially if the System Acquirer, their TR, or User will be attending.

Despite careful planning, *unplanned* test events do occur during formal ATPs as a result of Test Discrepancies (TDs). When a TD occurs, a DR must be logged and submitted indicating a *non-compliance* issue. A DR *should not* identify the root cause of the discrepancy (Principle 28.5), only its occurrence.

Principle 28.8

Root Cause(s) Principle

Identify the root, most likely, or probable cause(s) of discrepancy, incident, or accident events via a *process of elimination* of contributing factors.

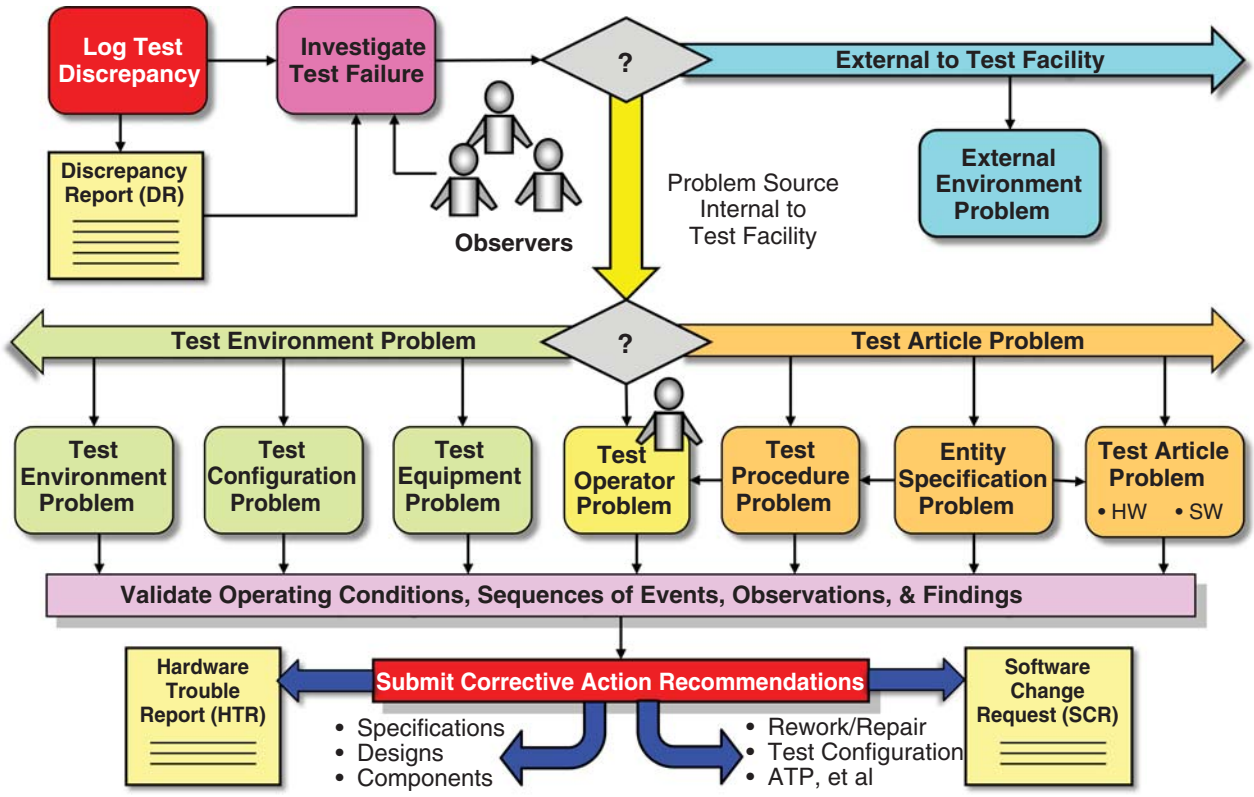


Figure 28.5 Test Discrepancy Source Isolation Tree

When a test *anomaly* or *failure* occurs and a DR is recorded, a determination has to be made concerning the significance of the problem on the test article and test plan as well as isolation of the problem source. While the general *tendency* is to focus on the test article due to its *unproven* design, the *source* of the problem can originate from any one of the Test Environment elements shown in Figure 28.1 such as a Test Operator or Test Procedure error; test configuration or Controlled OPERATING ENVIRONMENT problem; or combinations of these. From these contributing elements, we can construct a Test Discrepancy Fault Isolation Tree such as the one shown in Figure 28.5 to rule out by a *process of elimination* until the *root* or *probable cause* is identified. Our purpose during an investigation such as this is to *assume everything is suspect and logically rule out potential sources* by a *process of elimination*.

A Word of Caution 28.1

Discrepancy, Incident, or Accident Root Causes

Regarding the "root cause(s)" theme of Principle 28.8, recall our discussion in Chapter 24 of Reason's (1990) Swiss Cheese Model that most *incidents* and *accidents* are often the result of multiple contributory causes – accident waiting to happen – due to latent defects, not one single cause (Principle 24.4).

Once the DR and the conditions surrounding the failure are understood, our first decision is to determine if the problem originated *external* or *internal* to the Test Facility, as applicable. For those problems *originating* within the facility (Figures 28.1 and 28.5), decide if this is a Test Operator, Test Article, Test Configuration, Test Environment, or Test Equipment problem. Observe how this methodology exploits the Half Split Method (Principle 28.2) to determine the root cause(s).

Since the scripted ATP is the mechanism for orchestrating the test, start with it and its test configuration. Example TD investigation questions include:

1. Is the Test Configuration correct?
2. Was the test environment controlled at all times?
3. Did the operator perform the steps correctly
4. In the proper sequence without bypassing any?
5. Is the test procedure flawed? Does it contain errors?
6. Was a "dry run" conducted with the test procedure to verify its logic and steps?

If so, the test article may be suspect.

A Word of Caution 28.2

Test Discrepancy (TD) Scene Protection

One of the shortcomings of formal ATP TD Events is that Engineers instinctively start dismantling the Test Configuration to isolate the problem. DO NOT TOUCH THE TEST CONFIGURATION! Formal TD Events are analogous to Crime Scene Investigations (CSI) that must be cordoned off and remain untouched until an investigative team has arrived on scene to perform the investigation. Disturbing a formal ATP Test Configuration is amateurish and unprofessional.

While people tend to rush to judgment and take corrective action, *validate* the problem source by reconstructing the "scene of the event." This includes test configuration, operating conditions, sequence of events, test procedures, and observations as documented in the test log, DR, and test personnel interviews.

If the problem originated with the Test Article, retrace the development of the *test article* in reverse order to the appropriate System Development Process (Figure 12.2). *Was the item built correctly per its design requirements*? *Was it properly inspected and verified*? If so, *was there a problem due to component, material, process, or workmanship defect(s) or in the verification of the Entity*? If so, determine if the test article will have to be *repaired*, *scrapped*, *re-acquired*, or retested. If not, the design or specification may be suspect.

Audit the design. *Is it fully compliant with its specification*? *Does the design have an inherent flaw*? *Were there errors in translating specification requirements into the design documentation*? *Was the specification requirement misinterpreted*?

- If so, *redesign* to correct the *flaw* or *error* will have to be performed.
- If not, since the specification establishes the compliance thresholds for verification testing, you may have to consider (1) revising the specification and (2) reallocating performance budgets and design safety margins. On the basis of your findings, recommend, obtain approval, and implement the corrective actions.

A Word of Caution 28.3

Ad hoc Revisions to Specification Requirements

Regarding the revision of the specification or design, these should be informed decisions that are documented with supporting rationale, not *ad hoc* decisions

Then, perform *regression testing* starting with the last test that was *unaffected* by the discrepancy.

28.8 COMMON INTEGRATION AND TEST CHALLENGES AND ISSUES

SITE practices often involve a number of SE challenges and issues. Let's explore some of the more common ones.

Challenge 1: SITE Data Integrity

Deficiencies in establishing the test environment, poor test assumptions, improperly trained and skilled test operators, and an uncontrollable test environment compromise the integrity of Engineering test results. Ensuring the *integrity* of test data and results is crucial for downstream decision making involving formal acceptance, certification, and accreditation of the system, product, or service.

Warning 28.1

Distortion and Misrepresentations of Test Data

Purposeful actions to *distort* or *misrepresent* test data are a *violation* of professional and business ethics as well as Federal law. Such acts are subject to *serious* criminal penalties that are *punishable* under federal or other statutes or regulations.

Challenge 2: Biased or Aliased SITE Data Measurements

When instrumentation such as measuring devices are connected or "piggybacked" to "test points," the resulting impact can *bias* or *alias* test data and/or degrade system, product, or service performance. Test data measurements should not *load or degrade* SYSTEM performance. Thoroughly analyze the impact of potential effects of test device *alias* or *bias* SYSTEM performance *before* instrumenting a test article. If there are aliasing or biasing problems, investigate other analytical methods to derive the data implicitly from other data. Decide:

1. How critical the data is needed.
2. If there are alternative data collection mechanism or methods.
3. Whether the data "value" to be gained is worth the technical, cost, and schedule risk.

Challenge 3: Preserving and Archiving Test Results and Data

The purpose of SITE and system verification is to prove that a system, product, or service fully complies with its SPS or EDS. The *validity* and *integrity* of the compliance decision reside in the formal ATP QRs recorded as *objective evidence* of test and compliance verification results. Therefore, test data results recorded during a formal ATP must be *witnessed*, *authenticated*, and *preserved* by archiving it in a

permanent, safe, secure, and limited access facility. This data may be required to support:

- A Functional Configuration Audit (FCA) and a Physical Configuration Audit (PCA) (Chapters 12 and 13), if required, prior to SYSTEM delivery and formal acceptance by the Acquirer for the User.
- Analyses of SYSTEM failures or problems in the field.
- Legal claims.

Most contracts have requirements and Enterprises have policies that govern the storage and retention of contract data, typically several years after the completion of a contract.

A Word of Caution 28.4

Retention of Data Records

Always consult your contract concerning requirements for the storage and retention of data records.

Challenge 4: Test Data Authentication

When formal test results and data are recorded, the *validity* of the verification data should be authenticated, depending on end usage. *Authentication* occurs in a number of ways. Generally, the authentication is performed by an ITA or individual within the System Developer's Enterprise's QA organization. This individual should be trained and authorized to stamp formal test results and data (Table 28.3) in accordance with prescribed policies and procedures. Authentication may also be required by higher level bonded, external Enterprises. At a *minimum*, authentication criteria include witnessed affirmation of the following:

- Test article and test environment configuration.
- Test operator qualifications and methods.
- Test assumptions and operating conditions.
- Test events and occurrences.
- Compliance verification of expected results.
- Pass/Fail decision for each test.
- TDs.

Challenge 5: One Test Article and Multiple Integrators and Testers

Because of the expense of developing large complex systems, multiple integrators may be required to work in shifts such as 8 hours to meet development schedules. This potentially presents problems when integrators on the next shift waste time uninstalling *undocumented* "patches" to a configuration "Build" (Figures 15.5 and 15.6) such as software and cable/jumper wire changes from a previous shift. Therefore, each SITE work shift should begin with a joint coordination meeting of persons going "off-shift" and coming "on-shift." The purpose of the meeting is to make sure everyone communicates and understands the current configuration "build" that transpired during the previous shift.

Challenge 6: Deviations and Waivers

When a SYSTEM, PRODUCT, or SERVICE fails to meet its performance, development, and/or design requirements, the item is tagged as *non-compliant*. For hardware, a Non-Conformance Report (NCR) documents the discrepancy and dispositions it for corrective action by a Material Review Board (MRB). For software, a software developer submits a Software Change Request (SCR) to a Software Configuration Control Board (SCCB) for approval. *Non-compliances* are sometimes resolved, depending on the circumstances, by issuing a *deviation* or *waiver* (Chapter 18), rework, or scrap without requiring a CCB action.

Challenge 7: Equipment and Tool Calibration and Certification

Principle 28.9

Certified Tools & Equipment Principle

To avoid *invalidating* verification test results, ensure that all required test tools and equipment are certified to be calibrated and aligned to source standards prior to each test.

The *credibility* and *integrity* of a Verification and Validation (V&V) effort are often dependent on ENABLING SYSTEM:

1. Test FACILITIES and EQUIPMENT to establish a controlled OPERATING ENVIRONMENT for SYSTEM modeling, simulation, and testing.
2. Special tools used to make precision adjustments in SYSTEM/ENTITY capabilities and outputs.
3. Instrumentation used to measure and record the SYSTEM's environment, inputs, and outputs.
4. Tools used to analyze the SYSTEM responses.

All of these factors:

1. Require *calibration, alignment,* or *certification* to national and international standards for weights, measures, and conversion factors.
2. Must be traceable to national/international standards.

Therefore, *avoid* rework by ensuring that V&V activities have technical *credibility* and *integrity*. Begin with a firm foundation by ensuring that test equipment and tools are calibrated, certified, and traceable to source standards within a prescribed time period as marked on the item.

Challenge 8: Failure to Provide Test Data Access Points

Test Data Access Principle

Principle 28.10 Anticipate what types of access ports HARDWARE test points, and SOFTWARE mnemonic data will be required from Entities at each level of integration to collect test data required for specification requirements and design verification.

During specification development and System/Entity Design, one of the characteristics exhibited by the ad hoc, SDBTF-DPM Engineering Paradigm Enterprises is a lack of *Systems Thinking* (Chapter 1) concerning *how* SITE test data will be acquired. To maintain the integrity of Design Verification (Chapter 13) for a SYSTEM at various levels of integration, verified test article UUTs are typically sealed by QA to prevent *unauthorized* opening, entry, tampering, or intrusion. This presents a *major problem*, especially if a test at a higher level of integration requires test data via HARDWARE or SOFTWARE test points within the sealed unit. Beginning with specification development, this point illustrates *why* it is so critically important to do more than write "shall" statements into a specification outline. The Test Director (TD) is a key collaborating decision-maker that should challenge System Development Team (SDT) and Product Development Teams (PDTs) preparing specifications. For every specification requirement, document a one or two sentence verification plan for how the requirement will be verified, what data, test ports and text points, and test instrumentation and equipment will be required and at what IPs (Figure 28.2).

Challenge 9: Insufficient Time Allocations for SITE

Perhaps, one of the most *serious* challenges is allocating the right amount of time for SITE activities due to poor project planning and implementation. When Enterprises bid on System Development contracts, executives bid *aggressive* schedules based on some perceived "clever strategy" to win the contract. This occurs despite the fact that the Enterprise may have had a long-standing history of poor contract performance such as cost/schedule overruns.

Heuristic 28.1 SITE Time Allocation

As a *rule of thumb*, a project should dedicate at least 40% of its schedule to SITE.

Theoretically, the more testing you perform, the more latent defects you can discover due to design defects such as deficiencies, flaws, or errors. While every system is different, as a "starting point" allocate at least 40% of the project schedule as a starting point for perform SITE. Most contract projects fall behind schedule and compress that 40% into 10%. As a result, they rush SITE activities, *inadequately* test the system, and deliver it with latent defects that could have been discovered with more time for SITE. Figure 13.2 and Table 13.1 illustrate how failure to properly perform adequate testing to remove latent defects only results in more costly problems later.

There are several reasons for this:

1. Bidding aggressive, unrealistic schedules to win the contract.
2. Allowing the project performance to get off schedule, beginning with Contract Award, due to a lack of understanding of the *problem* and *solution spaces* (Figures 4.3 and 4.7) or poor data delivery performance by external Enterprises.
3. Rushing *incomplete* designs through the Component Procurement and Development Process to and into SITE (Figure 12.2) "check the box" and *boldly proclaim* that SITE was entered "on time" and then attempt to finish the designs during SITE.
4. Assigning project management that understands meeting schedules and making profits but *does not* understand or appreciate the magnitude of the technical problem(s) to be solved nor *how to* orchestrate successful contract implementation and completion.

To better appreciate the significance of this challenge, refer to the Epilog discussion. Specifically, Jet Propulsion Laboratory (JPL) Chief Engineer Brian Muirhead's observations concerning the *importance* and *criticality* of System Validation through design mitigation testing and a robust test program of the working system.

Challenge 10: Discrepancy Reporting Obstacles to SITE

One of the challenges of SITE is staying on schedule while dealing with TDs. Establish a DR Priority System to delineate DRs that do not:

1. Affect PERSONNEL or EQUIPMENT System Elements safety.
2. Jeopardize or invalidate higher-level test results.

Establish Go/No-Go DR criteria for proceeding to the next level of SITE.

Challenge 11: DR Implementation Priorities

Humans, by nature, like to work on "fun" things and the easy tasks. So, when DR corrective actions must be implemented, developers tend to work on those DRs

that give them "instant credit" for completion. As a result, the challenging DRs get pushed to the bottom of the stack. Then, progress report metrics proudly proclaim the large quantity of DR corrective actions completed.

In an Earned Value Management (EVM) context, you have a condition in which 50% of the DRs recompleted the first month. Sounds great! Lots of productivity! Wrong! Here's the problem. Only 50% of the quantity of DRs that represent 10% of the total work effort to be performed has been completed. Therefore, project technical management, who also contribute to this decision making, need to do the following:

- Prioritize and schedule DRs for implementation.
- Allocate resources based on those priorities.
- Measure EV progress based on relative importance or value of DRs.

In this manner, you tackle the challenging problems first. Executive management and the Acquirer need to understand and commit their full support to the approach as "the right thing to do!" As Stakeholders, they need to be participants in the prioritization process.

Challenge 12: Paragraph versus Singular Requirements

In addition to the SE Design Process challenges, the consequences of paragraph-based specification requirements arise during SITE. The realities of using a test to demonstrate several dependent or related requirements scattered in paragraphs throughout a specification can create many problems. *Think smartly* "up front" when specifications are written and create *singular* requirements statements that specify one and only one performance-based outcome (Principle 22.9) that can be easily checked off as completed.

Challenge 13: Credit for Requirements Verified

The issue of *paragraph* versus *singular* requirements also presents challenges during verification. The challenge is paragraph-based requirements cannot be checked off as verified until *all* of the requirements in the paragraph have been verified. Otherwise, if the paragraph has 10 embedded requirements and you have completed nine, the requirement is considered to be *unverified* until the tenth requirement is verified. Create singular requirement statements!

Challenge 14: Refurbishing/Reconditioning Test Articles

SVT articles, especially expensive ones, may be acceptable for delivery under specified contract conditions such as after refurbishment and touch-up. Establish acceptance criteria *before* the contact is signed concerning *how* SVT articles will be dispositioned *after* the completion of SYSTEM testing.

A Word of Caution 28.5

Refurbishment of Test Articles as Deliverables

Always consult your contract and your Enterprise Project, Legal, or Contracts organizations for guidance concerning refurbishment of test articles as deliverables.

Challenge 15: Calibration and Alignment of Test Equipment

Testing is very expensive and time consuming. When the SVT is conducted, most projects are already behind schedule. During SITE, if it is determined that a test article is *misaligned* or your test equipment is deemed *out of calibration* by the date stamp on the EQUIPMENT item, you have test data *integrity* issues to resolve.

Do yourself a favor. Ensure that *all* test equipment and tools are *certified* as calibrated and aligned *before* you conduct formal tests. Since calibration certifications have expiration dates, plan ahead and have a contingency plan to replace test equipment items with calibration due to expire prior to and during the SITE. Tag all EQUIPMENT and tools with EXPIRED calibration notices that are highly visible; secure the *expired* equipment until calibrated.

Challenge 16: Test "Hooks"

Test hooks provide a means to capture data measurements such as test points and software data measurements. Plan for these "hooks" during the SE Design Process (Figures 12.3 and 12.6) and ensure they do not *bias* or *alias* the *accuracy* of hardware measurements or degrade software performance. Identify and visibly tag each one for test reporting and easy *removal* later. Then, when test article verification is completed, make sure all test hooks are removed, unless they are required for higher-level integration tests.

Challenge 17: Anomalies

Anomalies can and do occur during formal SITE. Ensure that Test Operator PERSONNEL and EQUIPMENT properly *log* the occurrence and configuration and event *sequences* when anomalies occur to serve as a basis to initiate your investigation.

Anomalies are particularly troublesome on large complex systems. Sometimes, you can isolate anomalies by luck; other times they are elusive, so you find them unexpectedly. In any case, when anomalies occur, record the sequence of events and conditions preceding the event. What may appear to be an *anomaly* as a single event may have patterns of recurrences over time. Tracking *anomaly* records over time may provide clues that are traceable to a specific root or probable cause.

Challenge 18: Technical Conflict and Issue Resolution

Technical conflicts and issues can and do arise during formal SITE between the Acquirer's Test Representative (ATR) and the System Developer, particularly *over interpretations* of instrument readings or data.

- Firstly, ensure that the ATPs are explicitly stated in a consistent manner that *avoids* multiple interpretations. Where test areas may be questionable, the TEWG should establish prior to or during the Test Readiness Review (TRR) (Chapter 18) how conflicts will be managed.
- Secondly, establish a Conflict and Issue Resolution Process between the Acquirer (role) and System Developer (role) prior to formal testing. Document the Conflict and Issue Resolution Process in the SIVP.

Challenge 19: Creating the Real-World Scenarios

During SITE planning, there is a tendency to "check-off" individual requirements via individual tests during verification. From a requirements compliance perspective, this has to be accomplished. However, verifying each requirement as a discrete test may not be desirable. There are two reasons: (1) cost and (2) real world.

- Firstly, just because a specification has separately stated requirements does not mean that you cannot conduct a single test that verifies a multiple number of requirements to minimize costs (Principles 22.4 and 28.4). This assumes that the test is *representative* of SYSTEM usage rather than a *random* combination of *unrelated* capabilities.
- Secondly, when Users employ systems to conduct missions, there are typically multiple capabilities operating simultaneously. In some cases, interactions between capabilities may conflict. This can be a problem, especially if you discover later *after* individual specification requirement tests that indicated the SYSTEM is compliant with the requirements. This point emphasizes the need for UC and scenario-based tests and TCs to *exercise* and *stress* combinations of SYSTEM/ENTITY capabilities to *expose* potential interaction conflicts while verifying one or more specification requirements.
- Conceptually, Engineers test one or sets of requirements in sequential order beginning with SPS or EDS Section 3.0 - verify Requirement 3.1.1, then 3.1.2, and so forth. Test completed!! Then, when the User operates the Equipment with odd combinations of capabilities operating simultaneously, problems occur. Examples include mechanical clearance problems, Electromagnetic Interference (EMI), radiant heat sources, and so forth interact with "components" of the system while it is operating.
- Remember, Users do not use an SPS or EDS to operate a system or product; Users operations and SDS/EDS requirements compliance are two different concepts but interrelated! So, the challenge is two fold: (1) verify SPS and EDS requirements for compliance and (2) exercise and stress the system or product with various combinations of capabilities operating simultaneously to understand their interactions. Referring to the quote by Dr. Michael Griffin (Warwick & Norris, 2010) in Chapter 2: "What is needed is a new view that the core systems engineering function 'is not primarily concerned with characterizing the interactions between elements and verifying that they are as intended.' What's more important, he says, is understanding the dynamic behavior of those interactions." The challenge is: how do we create a realistic Test Environment and conditions during System Verification that are representative of both the (1) external Operating Environment and (2) combinations of operating conditions within the system or product being verified.
- System Verification requires more than robotically testing one or a set of requirements to verify compliance with an SPS or EDS. System Verification should verify that all most likely or probable combinations of capabilities operate and interact flawlessly and seamlessly without mechanical, thermal, electrical, or other types of interference that degrade performance.

28.9 CHAPTER SUMMARY

Our discussion of SITE as one of the V&V practices explored the key activities of DT&E under controlled laboratory conditions. Work products and formal QRs such as test results and data provide *objective evidence* (Principles 13.8 and 19.1 and Heuristic 19.1) that demonstrates and proves compliance with the SPS or an EDS. Data collected during SITE enables SEs to:

1. Develop confidence in the integrity of the Developmental Configuration.
2. Support the FCA and PCA.
3. Answer the key verification question: did we build the SYSTEM or PRODUCT in compliance with the SPS or IDS?

28.10 CHAPTER EXERCISES

28.10.1 Level 1: Chapter Knowledge Exercises

1. What is SITE?

2. When is SITE conducted?
3. What is the objective of SITE?
4. The text refers to a System/Entity's "prescribed OPERATING ENVIRONMENT." Where would you find it documented?
5. What is the relationship of SITE to DT&E and OT&E?
6. What is SITE expected to accomplish?
7. What are the Quality Records (QRs) and work products of SITE?
8. What are the roles and responsibilities for SITE accountability?
9. What are the types of test plans?
10. What is a TEMP?
11. Who owns and prepares the TEMP?
12. What is an SIVP?
13. Who owns and prepares the SIVP?
14. Differentiate the contexts of the TEMP versus the SIVP.
15. What is a TEWG?
16. Why do you need test logs?
17. What is regression testing?
18. What is a DR?
19. When is a DR prepared?
20. How do you prioritize SITE activities and record SITE deficiencies and issues?
21. What are TRRs? Why do you need to conduct them?
22. What are the SITE phases and supporting tasks for an item?
23. What are SITE observations, defects, discrepancies, and anomalies?
24. How do you prepare TRs?
25. What is the importance of formally approving and archiving test data?
26. How do you certify test data?
27. What are some common SITE challenges and issues?
28. Why is it important to document the events leading up to a test failure and documenting them in a DR?

28.10.2 Level 2: Chapter Knowledge Application Exercises

Refer to www.wiley.com/go/systemengineeringanalysis2e

28.11 REFERENCES

DAU (2005), *Test and Evaluation Management Guide,* 5th Edition, Ft. Belvoir, VA: Defense Acquisition University (DAU) Press. Retrieved on 1/16/14 from http://www.dtic.mil/cgi-bin/GetTRDoc?AD=ADA436591

DAU (2012), *Glossary: Defense Acquisition Acronyms and Terms*, 15th ed., Ft. Belvoir, VA: Defense Acquisition University (DAU) Press. Retrieved on 6/1/15 from http://www.dau.mil/publications/publicationsDocs/Glossary_15th_ed.pdf

FAA (2012), *COTS Risk Mitigation Guide*, Washington, DC: Federal Aviation Administration (FAA).

IEEE Std. 829-2008 (2008), IEEE Standard for Software and System Test Documentation, New York: Institute of Electrical and Electronic Engineers (IEEE).

ISO/IEC/IEEE 24765:2010 (2010), Systems and software engineering-Vocabulary, Geneva: International Organization for Standardization (ISO).

MIL-HDBK-881C (2011), *Military Handbook: Work Break Down Structure,* Washington, DC: Department of Defense (DoD).

Reason, James (1990), *Human Error*, Cambridge, UK: Cambridge University Press.

SEVOCAB (2014), Software and Systems Engineering Vocabulary, New York, NY: IEEE Computer Society. Accessed on 5/19/14 from www.computer.org/sevocab

Warwick, Graham and Norris, Guy (2010), "Designs for Success: Calls Escalate for Revamp of Systems Engineering Process", Aviation Week, Nov. 1, 2010, Vol. 172, Issue 40, Washington, DC: McGraw-Hill.

29

SYSTEM DEPLOYMENT, OM&S, RETIREMENT, AND DISPOSAL

On completion of final acceptance by the System Acquirer/User, the next step is deployment and delivery of a system, product, or service to job site(s) designated by the User or to commercial markets for distribution. The deployment and delivery may be accomplished by the User or part of a System Development agreement. To most System Developer Engineers, the "Engineering" of the SYSTEM or PRODUCT is viewed as complete and usage begins—end of story. This is not true!

The System Development Phase is complete and the System or Product, for example, has been transitioned to the User's Enterprise or organization. However, Systems Engineering and Development (SE&D) continues with the User or their technical representative's SEs. This includes performance monitoring to ensure that the system, product, or service:

- Delivers the User required capabilities to conduct missions.
- Provides operational utility, suitability, availability, usability, effectiveness, and efficiency (Principle 3.11).
- Remains current and with no capability "gaps."

The System's Development was based on the User, System Acquirer, and System Developers establishing assumptions, objectives, and constraints—technical, technology, cost, schedule, and risk—about *how* the system, product, or service would perform its missions in a prescribed OPERATING ENVIRONMENT. On delivery, several questions emerge:

1. Does the system, product, or service perform as expected in meeting its performance-based outcomes and objectives?
2. Is the User satisfied with the SYSTEM's performance?
3. Does the System or Product contain residual *latent defects*—design flaws, errors, and deficiencies; defective materials and components; or workmanship problems?

This chapter addresses SYSTEM Deployment; Operations, Maintenance, and Sustainment (OM&S); Retirement; and Disposal from an SE perspective. The intent is to answer the three above-mentioned questions, especially for User SEs or their technical representative. You may ask: *what is the value in doing this*? There are four reasons:

1. If a system, product, or service contains latent defects, there could be significant risk implications if left uncorrected, especially in terms of mission accomplishment contributors such as security, health, safety, and environmental.
2. Systems and products have finite useful service lives (Figure 34.6), require on-going *preventive* and *corrective* maintenance actions, and experience performance degradation over time (Figure 4.25) that require informed awareness of trends as well as the need for recalibration and realignment.
3. Achievement of a system or product's useful service life requires Condition-Based Maintenance (CBM)

System Engineering Analysis, Design, and Development: Concepts, Principles, and Practices, Second Edition. Charles S. Wasson.

Companion Website: www.wiley.com/go/systemengineeringanalysis2e

(Chapter 34) through timely and disciplined monitoring and inspection.

4. The success of operator-based systems requires continual awareness and tracking of User—operator and maintainer—training, skills, proficiency, discipline, and performance.

Contrary to popular belief, SE and Analysis of a SYSTEM or a PRODUCT continues after System Development and throughout the Deployment and OM&S Phases of the System/Product Life Cycle. This will be the basis for our discussion in Chapter 29, especially for User SEs.

Author's Note 29.1

System Deployment Paradigms

The concept and term, *System Deployment*, is sometimes viewed as reserved for military organizations. The reality is Users *deploy* system, products, and services through their homes, business, and communities. For example:

- Fire trucks, Emergency Services (EMS) and utility vehicles are *dispatched—deployed*—to locations to provide emergency assistance.
- A business or corporation's Information Technology (IT) organization *deploys* desktop, laptop, and so forth computers and software downloads to offices via a Local Area or Wide Area Network (LAN/WAN).
- A hospital *deploys* medical devices to patient rooms and surgical suites to perform missions.

The term *deployment* is not unique to military organizations and is equally applicable to all Enterprises.

29.1 DEFINITIONS OF KEY TERMS

- **Analysis Paralysis**—Refer to the Chapter 27 Definitions of Key Terms.
- **Deactivation**—An authoritative decision made to phase-out/retire a specific series, version, or instance of a SYSTEM or PRODUCT active duty service for storage or disposition for removal from inventory. Some Enterprises use the term *decommission*.
- **Deployment**—The assignment, tasking, and physical relocation of a system, product, or service to a new location or staging area for storage or conducting Enterprise missions.
- **Deployment Facility**—A physical facility—building, structure, compound—that provides shelter, security, protection, or an environment for a system, product, or service during or between missions.
- **Deployment Site**—A geophysical location that represents where a system, product, or service is deployed and serves as its base of operations.
- **Disposal**—"… 2.) The act of getting rid of excess, surplus, scrap, or salvage property under proper authority. Disposal may be accomplished by, but not limited to, transfer, donation, sale, reclamation, demilitarization, abandonment, or destruction." (DAU, 2012, p. B-71)
- **Disposal**—"The act of getting rid of excess, surplus, scrap, or salvage property under proper authority. Disposal may be accomplished by, but not limited to, transfer, donation, sale, declaration, abandonment, or destruction." (DAU, 2012, p. B-80)
- **Disposal** (Lifecycle perspective)—"All activities associated with disposal management, dismantlement/demolition/removal, restoration, degaussing, or destruction of storage media and salvage of decommissioned equipment, systems, or sites." (FAA, 2006, Vol. 3, p. B-3)
- **Disposal (Waste)**—"The discharge, deposit, injection, dumping, spilling, leaking, or placing of any solid waste or HW into or on any land or water. The act is such that the solid waste or HW, or any constituent thereof, may enter the environment or be emitted into the air or discharged into any waters, including ground water (40 CFR 260.10)." (AR 200–1, 2007, p. 102)
- **Facility Interface Specification (FIS)**—A specification that specifies boundary envelope space, environmental conditions, capabilities, and performance requirements to ensure that all facility interfaces are capable, compatible, and interoperable with the new SYSTEM.
- **Failure Reporting and Corrective Action System (FRACAS)**—"A closed-loop system of data collection, analysis and dissemination to identify and improve design and maintenance procedures." (MIL-HDBK-470A, 1997, p. 1–3)
- **Maintenance—Depot Level**—"… includes any action performed on materiel or software in the conduct of inspection, repair, overhaul, or the modification or rebuild of end-items, assemblies, subassemblies, and parts. Depot-level maintenance generally requires extensive industrial facilities, specialized tools and equipment, or uniquely experienced and trained personnel that are not available in lower echelon-level maintenance activities." (DAU, 2012, p. B-131)
- **Maintenance—Intermediate Level**—"… includes assembly and disassembly beyond the capability of the organizational level …" (DAU, 2012, p. B-131)
- **Maintenance—Field Level**—"… is comprised of both (1) organizational maintenance, which includes inspections, servicing, handling, preventative and

corrective maintenance, and (2) Intermediate Maintenance, …" (Adapted from DAU, 2012, p. B-131.

- **On-Site Survey**—A planned, authorized, and coordinated tour of a potential deployment site to understand the physical context and terrain; natural, historical, political, and cultural environments; and issues related to developing it to accommodate a system.
- **Operational Site Activation**—"The real estate, construction, conversion, utilities, and equipment to provide all facilities required to house, service, and launch prime mission equipment at the organizational and intermediate level." (MIL-HDBK-881C, 2011, p. 229)
- **Packaging**—"The process and procedures used to protect materiel. It includes cleaning, drying, preserving, packing, and unitization." (DAU, 2012, p. B-160)
- **Packaging, Handling, Storage, and Transportation (PHS&T)**—The combination of resources, processes, procedures, design, considerations, and methods to ensure that all system, equipment, and support items are preserved, packaged, handled, and transported properly, including environmental considerations, equipment preservation for the short and long storage, and transportability. Some items require special environmentally controlled, shock isolated containers for transport to and from repair and storage facilities via all modes of transportation (land, rail, air, and sea). (Product Support Manager Guidebook) (DAU, 2012, p. B-160)
- **Problem Report (PR)**—A formal document of a problem with or failure of Mission System or Enabling System hardware such as the Equipment System Element.
- **Retrofitting**—The process of (1) improving System/Product capabilities through performance or technology enhancement upgrades or (2) correcting latent defects that were discovered after delivery.
- **Site Development**—The process of preparing a site—real estate—to host deployment of a System or Product and/or its facility for conducting System of Interest (SOI) Mission System(s) and Enabling System(s) OM&S. Includes the planning; licenses and permits; grading; installation of utilities, sewer, communications, landscaping, lighting, security perimeter boundaries; and inspections.
- **Site Installation and Checkout (I&CO)**—The process of unpacking, erecting, assembling, aligning, and calibrating, installing, checking out, and activating a System or Product to a state of readiness for mission service.
- **Site Selection**—The process of identifying candidate sites to serve as the location for a System Deployment and making the final selection that balances operational needs with environmental, historical, cultural, political, and religious constraints or customs.
- **Staging Area**—A rendezvous location, gathering point, warehouse for preparing to deploy a System or Product for a mission that may require assembly, integration, test, and verification.
- **Sustainment**—Refer to the Chapter 3 Definitions of Key Terms
- **System Retirement**—A decision made to decommission and phase-out: (1) all instances of a specific type of system from the Enterprise's inventory or (2) specific instances of a System/Product due to aging, obsolescence, maintenance costs, and so forth.

29.2 APPROACH TO THIS CHAPTER

Fielded System Performance Principle

Principle 29.1 Continuously monitor and Assess System or Product performance throughout the System Deployment, OM&S, Phase-Out, and Disposal Stages of the System/Product Life Cycle.

Due to the lack of Systems Engineering (SE) "up front" and throughout the System Development, there is a general *misperception* that SE activities end when an Engineered System or Product has been verified, validated, and accepted. However, SE activities continue into the System Deployment, OM&S, Retirement, and Disposal Phases of the System/Product Life Cycle. The difference is that SE, in general, shifts from System Development of the System or Product to assessing the achievement of the verified performance by the User. The scope of our discussions in Chapter 29 focuses on SE activities and considerations required to plan for System Deployment, OM&S, Retirement, and Disposal.

Due to the challenges and complexities of physical deployment of small to large systems such as Heating, Ventilation, and Air Conditioning (HVAC) systems, automobiles, heavy construction equipment, and military systems, Chapter 29 addresses these challenges. Smaller systems such as consumer products and software typically do not (1) require the Engineering of a large Enabling System to transport the items, (2) contain sensitive measurement equipment that must survive the transport, (3) have an impact on the environment or roadways, (4) require sites and facilities to locate the System. However, items such as *software* have their own set of challenges. For example, consider the deployment of new software loads to:

- A spacecraft traveling to the Moon or another planet and returning to Earth.

- Desktop computers in homes, offices, or vehicles, or portable devices in which the current configuration must be interrogated and the download or upload verified.

Our discussion begins with System Deployment followed by System OM&S. We conclude with a brief discussion of System Retirement and Disposal.

29.3 SYSTEM DEPLOYMENT OPERATIONS

Most systems, products, or services require *deployment* or *distribution* by their System Developer or Series Provider to a User's *designated field site* or *staging area*. During the deployment, the SYSTEM or PRODUCT may be subjected to numerous types of OPERATING ENVIRONMENT states and conditions such those listed earlier in Table 7.4.

System Deployment involves more than physically deploying the SYSTEM or PRODUCT. The SYSTEM being deployed, at a *minimum* and as applicable, may also require key activities shown in Figure 7.5:

- Storage in and/or interface with temporary, interim, or permanent warehouse or support facilities.
- Set-up, I&CO, and integration into the User's HIGHER-ORDER Level 0 or Tier 0 SYSTEM (Figure 8.4).
- Training for ENABLING SYSTEM operators and maintainers.
- Calibration and alignment.
- Transport.
- Teardown.
- Instrumentation to monitor NATURAL and INDUCED environmental conditions during deployment—temperature, humidity, shock, and vibration.
- Retrofitting upgrades since its deployment.
- Reverification.

To accommodate these challenges, SYSTEM or PRODUCT designs and components must be *sufficiently* robust to *survive* in these conditions, either in an *operational* or in a *non-operational* state. For a System Design Solution to accommodate these challenges, the System Performance Specification (SPS) must define and bound the required operational capabilities and performance to satisfy these conditions.

Accomplishing this task requires informed awareness of the OPERATING ENVIRONMENT through site visits and collaboration with the Stakeholders. In this context, the *scope* of Stakeholders goes beyond the SYSTEM or PRODUCT's Users and End Users. It includes the ENABLING SYSTEMS—transporters—and Stakeholders along the route of deployment that include the public; HUMAN SYSTEMS and NATURAL ENVIRONMENTS; and local, state, and federal governments.

29.3.1 Objectives of System Deployment

The objective of System Deployment is to *safely* and *securely* relocate or reposition a SYSTEM or PRODUCT from one field site or *staging area* to another. This requires using the most *efficient* and *effective* methods that offer an optimal balance of technical and operational performance; acceptable risk; and least cost and schedule impacts.

To accomplish a deployment for most mobile systems, we decompose this objective into several supporting objectives:

- Prepare the SYSTEM or PRODUCT for shipment, including disassembly, inventory, packaging of components, and crating.
- Coordinate the land, sea, air, or space-based mode of transportation.
- Transport the SYSTEM to the new location, job site, or staging area.
- Store the SYSTEM or PRODUCT in a safe and secure area or FACILITY.
- Install, erect, assemble, align, calibrate, checkout, and verify capabilities and performance at the deployment site.

29.3.2 System Deployment Contexts

System Deployment has three contexts:

- **First Article(s) Deployment** The relocation of *first article* systems to a test location range during the System Development Phase to support User Operational Test and Evaluation (OT&E) (Chapters 12) activities.
- **Production Distribution Deployment** The relocation of production systems via distribution systems to User sites or consumer accessible sites.
- **Physical System Redeployment** The relocation of a deployed SYSTEM during the System OM&S Phase to a new *field site* or *staging area*.

Let's address each of these contexts.

29.3.2.1 First Article Deployment Context First article(s) deployment such as Developmental Configuration Engineering Models (Chapter 12) of large, complex systems can be very risky. This is especially true due to the cost and length of time required to replace a SYSTEM due to long lead-time items—PARTS. Time and/or resources may *prohibit* building another system, especially if it is inadvertently destroyed or damaged beyond repair during deployment. Depending

on the SYSTEM, first article deployment may involve moving a *first article system* to test facilities or test ranges for completion of Developmental Test and Evaluation (DT&E) (Chapters 12 and 13) or initiation of OT&E.

First article deployment for commercial systems typically includes significant promotional fanfare and publicity. Referred to as a "rollout," this event represents a key milestone toward the physical realization of the end deliverable system, product, or service. Examples of commercial system, product, or service deployment include a limited number of test markets for the Developmental Configuration (Chapters 12 and 16) to assess customer satisfaction and elicit feedback.

29.3.2.2 Physical System Redeployment Context Some first article SYSTEMS or PRODUCTS may be deployed to the marketplace after completion of Developmental Configuration Verification Validation (V&V), *refurbishment,* and System Acceptance by the System Acquirer or User. Examples of Use Cases (UCs) for redeployment activities to be considered may be found in Table 6.3.

For road-based deployments of large Systems such as cranes, heavy construction equipment, it is important to know where the system is at all times for coordination of traffic slowdowns, rerouting of traffic. This may require consideration of specification requirements for vehicle escorts, Global Positioning System (GPS), hand-held radios, or other devices. For commercial product distribution, Packaging, Handling, Storage, and Transportation (PHS&T), Table 20.1 should be a key consideration.

29.3.2.3 Production Distribution System Deployment Context Small quantity or mass-produced Commercial SYSTEMS or PRODUCTS, in general, are packaged partially or fully assembled and deployed in boxes or containers for shipment to distribution centers as *staging areas* for shipment to retailers. A key issue is often tracking of deliveries—a *problem space*. The *solution space* requires informed knowledge of the shipment's geographic location via bar codes at key pick-up or transfer points, Radio Frequency Identification (RFID) chips in packages. Examples include parcel-shipping services such as UPS, FedEx, or DHL. This translates into specification requirements for the deployment system.

29.3.3 Types of System Deployment

System Deployment requires a diverse set of SE considerations that are dependent on the SYSTEM or PRODUCT. For example:

- *Consumer products* require distribution from manufacturers to retail outlets for purchase by consumers and self-delivery or delivery by the seller or third parties.
- Small to large commercial or military systems typically require transport via one or more modes of transportation—land, sea, air, or space—from a System Developer's facility to a User's designated site or staging area by either the Developer, User, or third parties.

Let's explore each of these types of considerations.

29.3.3.1 Consumer Products Deployment Table 29.1 provides examples of SE considerations for commercial product deployment.

29.3.3.2 Small to Large Commercial, Military, and so forth Systems Deployment System Deployment of moderate to large SYSTEMS or PRODUCTS typically requires various modes of transportation and planning.

A Word of Caution 29.1

Systems Thinking and Deployment Constraints

System Deployment is a key example of Systems Thinking (Chapter 1) in action. A common paradigm among Engineers and others is to develop a SYSTEM or PRODUCT, then figure out *how to* ship or transport it to the User *after* it is designed. The reality is: modes of transportation and statutory/regulatory requirements impose *constraints* on system, product, or service requirements. Therefore, learn to employ *Systems Thinking* before launching into Engineering actions that may have *adverse* or *negative* consequences.

System Deployment of moderate to large physical systems requires SE consideration in two areas:

TABLE 29.1 System Deployment—Commercial Product Engineering Considerations

System Deployment Considerations	Example Engineering Considerations
Consumer products	• Product packaging—antitheft • Product assembly, cautions, and warnings • Shipping container restrictions—length, width, and height • Shipping container lift points, tie downs, and weight restrictions • Shipping container markings—This Side Up, stacking, wetness • Bar code tracking and Point of Sale (POS) • Radio Frequency Identification (RFID) tracking devices • Fluid volatility • Battery removal and installation • And so forth

- Design of the SYSTEM or PRODUCT and its interfaces.
- Operational deployment to a User's site.

Let's investigate each of these areas.

29.3.3.2.1 System-Specific Design and Interface Considerations In general, deployment of these systems requires consideration in the following areas:

- System and interface constraints.
- Freight shipping container constraints.
- Transport vehicle constraints.
- Environmental constraints.

Table 29.2 provides a summary of examples of these areas that require SE consideration.

Since roads vary from riverbeds and streams, unimproved roads, to interstate highways, *shock* and *vibration* become a major factor. The same is true for rough handling of containers. Consider the following example:

Example 29.1

System Application versus Deployment Environments

As an SE, suppose you are specifying a commercial desktop computer for use in a benign office environment. That does not mean that it may not encounter *shock* and *vibration* due to rough handling or rough roads in-transit to the consumer. Employ Systems Thinking to anticipate the deployment environment, bound, and specify the OPERATING ENVIRONMENT requirements accordingly.

29.3.3.2.2 System Deployment Operational Considerations Once loaded onto a transport vehicle, the second aspect of moderate to large system deployment is operational considerations. The scope of operational considerations includes *how* the SYSTEM or PRODUCT and its freight shipping containers will be transported to the User's site. This may require sequences of transfers to different modes of transportation such as land → air → land → space → land. Table 29.3 provides a summary of examples of these areas that require SE consideration.

Some SYSTEMS or PRODUCTS may require shipping in specialized containers that are *environmentally controlled* for temperature, humidity, and protection from salt spray.

Heading 29.1

Now that we have selected a deployment site and investigated what is required to deploy or transport the SYSTEM or PRODUCT, the next stage is to *begin Operational Site Selection and Activation, our next topic.*

29.3.4 Operational Site Selection and Activation

Completion of the System Development Phase of the System/Product Life cycle occurs when the system, product, or service is *operationally ready* to conduct missions at a specific site or base of operations. From an SE perspective, this is a key performance-based outcome for Systems Thinking; specifically, the derivation of requirements for the System Development Phase.

This thought process leads to a chain of questions.

1. What is required for the SYSTEM to become *operationally ready* to conduct missions? It requires the following:
   a. An operational SOI.
   b. A FACILITY (if applicable).
   c. A deployment site.
   d. A mission.
   e. MISSION RESOURCES.
   f. PERSONNEL

   Using reverse logic, this expands into three other SE questions:
2. What is required achieve a SYSTEM *operational readiness*? It requires the following:
   a. Transport to the deployment site.
   b. I&CO.
   c. V&V of SYSTEM operation.
   d. *Certification* or *recertification* and operator licensing, as applicable.
3. What is required to develop a FACILITY to accommodate the SYSTEM? You need the following:
   a. A geographical site that is ready for the FACILITY.
   b. FACILITY Development plan, Facility Interface Specification (FIS), Environment Impact Statement (EIS) designs, funding, and permits that have been approved.
   c. A site that has been developed and ready for FACILITY construction.
4. What is required to develop a site for the FACILITY? You need to:
   a. Select the deployment site.
   b. Prepare the site.
   c. Inspect the site for FACILITY development.

This chain of questions provides the basis for our discussion. *How do we do this*?

This requires development and approval of an *Operational Site Activation Plan*.

The Operation Site Activation Plan describes the organization, roles, responsibilities and authorities, tasks, resources, and schedule required to develop or modify and *activate* a new or existing facility. One of the key objectives

TABLE 29.2 System Deployment Engineering Considerations

System Deployment Considerations	Example Engineering Considerations
SYSTEM and interface constraints	• Bar code tracking • Length, width, height, and weight restrictions • Retraction or removal of appendages for transport • Disassembly and reassembly • Tie down eyelets and sizes • Safety chains • Lift and jack points • CG markings • Cautions and warnings, Points of Contact (POCs). • Auxiliary power • Auxiliary equipment stowage—CSE and PSE • Heating/cooling • Engine inlet coverings • System security and protection • Pressurization or depressurization • Easy removal and replenishment of fuels and fluids • Easy removal and installation of sensitive components • And so forth
Freight shipping container constraints	• Type of container • Tie down eyelets • Lift points • Volumetric—cargo length, width, and height restrictions • Weight restrictions • Auxiliary power • Heating/cooling • System security and protection • Container markings—cautions and warnings, content placards and codes, and POCs • And so forth
Transport vehicle constraints	• Land, sea, rail, and space requirements • Vehicle certifications • Vehicle markings • Vehicle size constraints—height, width, length, and weight • Cargo size constraints—height, width, length, and weight • Tie down attachment points • Cargo—fluid, fuel, and air-pressure removal • Cargo manifests • Fire extinguishers • Emergency roadside markers and flashers • And so forth
Environmental constraints	• International (Road) Roughness Index (IRI) • ASTM E1926–08 (2008) • ASTM E1364–95 (2012) • Shock and vibration • Saltwater and spray • Sand and dust • Temperature and humidity control • Electrical fields and discharges • Lightning protection • RF transmission towers • Flying debris, hail, rain, sleet, and snow • Altitude and atmospheric pressure changes • Environmental Instrumentation • HAZMAT • And so forth

TABLE 29.3 System Deployment Operational Considerations

System Deployment Consideration	Example Operational Considerations
Transport vehicle operators	• Driver certifications, licensing, experience, and skills • Awareness of System Deployment objectives • Cargo awareness and sensitivity training • And so forth
Deployment routes	• Land—highway, rail, sea, air, and space • Inclines and down grades • Emergency Stopping • Narrow bridges and tunnels • Mock Dry Runs with Simulated Equipment • Bridge, highway, and street load width, height, length, and weight restrictions • Power line height restrictions • Traffic flow rerouting and detours • Law enforcement traffic directing • HAZMAT routing restrictions • And so forth
Government and regulatory	• Plan and route approvals • Licenses and permits • Route coordination • Emergency response teams—medical, fire, law enforcement, and so on • And so forth

of the plan is to describe how the SYSTEM will be assembled and installed, aligned, calibrated, and integrated, as applicable, into the User's HIGHER-ORDER Enterprise LEVEL 0 System (Figure 8.4). If the System is being integrated into an existing facility, a key objective is to perform the integration *without* disrupting normal operations. The plan evolves through a series of updates as more information becomes available.

Another key consideration for Operational Site Activation is how the SYSTEM OR PRODUCT will be *maintained* and *sustained*. This is actually the implementation of the Maintenance and Sustainment Concepts (Table 6.1). Specifically, *how* and *where* maintenance will be performed.

Maintenance of a SYSTEM and PRODUCT occurs in a number of different forms. For example, maintenance of government military systems occurs at two levels: (1) Field Level and (2) Depot—a *highly specialized* facility or the Original Equipment Manufacturer (OEM). Decisions on the levels of maintenance may impact the planning and design of a deployment site and/or facility based on the specific maintenance actions to be performed (Chapter 34).

Given this overview discussion, let's explore some of the topics further that serve as inputs to the Site Operational Site Activation Plan.

29.3.4.1 Identify and Specify FACILITY Requirements

Selection of a deployment site may require (1) *development* of new land and facility or staging area or (2) *modification* of an existing facility. In either case, requirements for a new or modified facility require SE consideration. This includes considerations for physical size, compartmentalization such as secure areas, laboratories; assembly areas with loading dock(s), lifts, tool cribs, overhead doors, and cranes; security systems; networks; and utilities.

FACILITIES as ENABLING SYSTEM Elements exist for a purpose … to support the deployed SYSTEM or PRODUCT via physical interfaces. During the development of the SYSTEM or PRODUCT, especially for large, complex systems, the FIS should be developed. The FIS specifies and bounds the boundary envelope conditions, capabilities, and performance requirements to ensure that all facility interfaces are capable, compatible, and interoperable with the new SYSTEM.

Author's Note 29.2

Facility Interface Specification (FIS) Current Usage

The FIS, which was once used by government and military organizations, explicitly communicates by title what the document contains. From an SE perspective, the document title is still *valid* and *beneficial*.

As its title implies, the FIS specifies FACILITY interface requirements for a SYSTEM or PRODUCT during *non-operational* storage or *operational* use. This includes: Facility layouts—anchor bolts, space; utilities—electrical power and grounding, water, sewer; environmental—HVAC;

data, phone, and radio communications. Although these are FACILITY-based interface descriptions, requirements from land, sea, air, or space transport constraints information can be obtained from the transport vehicle interface documentation without the need for a separate document.

29.3.4.2 Select Deployment Site Preparations for SYSTEM or PRODUCT deployment to a field site or staging area require that the location be selected, developed, and activated. On delivery of the SYSTEM, the site must be *operationally ready* and *available* to accept the SYSTEM or PRODUCT for System I&CO and integration into a HIGHER-ORDER Level 0 SYSTEM (Chapter 9).

Development of the deployment site to support the SYSTEM depends on the mission. Some systems may require *temporary storage* in a staging area until they are ready to be moved to a permanent location. Others require assembly, I&CO, and integration into a HIGHER-ORDER Level 0 SYSTEM *without* disruption to existing facility operations. Some facilities provide high bays with cranes to accommodate SYSTEM assembly, I&CO, and integration into HIGHER-ORDER Systems. Other facilities may require you to provide your own rental or leased equipment such as cranes and transport vehicles.

Whatever the plan is for the facility, SEs are tasked to select, develop, and activate the field site. These activities include site surveys, site selection, site requirements derivation, Facility Engineering or site planning, site preparation, and System Deployment safety and security.

Observe that this discussion focuses on Earth-based site selection. Now, consider an organization such as NASA planning space missions. During the Apollo Space Program, landing sites on the Moon had to be selected for a series of unmanned and manned missions. The same is true today in selecting landing sites on Mars as well as rover explorations to sites away from its mother craft.

29.3.4.2.1 Land Considerations Selection of a site(s) for deploying large or mobile systems may be designated by the System Acquirer and User or may require "open" selection within a region. For example, selection of a state and city/town for locating a new manufacturing plant from a set of viable candidate locations using an Analysis of Alternatives (AoA) (Chapter 32).

System Deployment to a geographical location requires two physical considerations: (1) real estate or land and (2) facility. Scenarios for each consideration include the following:

- *Undeveloped* or *unimproved* land-range from limited or no access to *improved land* at an industrial park with existing plots with utilities ready for facility development.
- The need to develop a new facility or a facility ranges from renovation and upgrades to a facility ready for occupancy.

Remember—System Deployment does not necessarily require development of facilities. Military systems, for example, may only require stable, rough-graded, and level plots for transportable or mobile shelters or trailers.

Site selection also requires more than land or FACILITY development. There may be constraints that influence those decisions. In our discussion of the OPERATING ENVIRONMENT architecture (Figure 9.3), we noted that external HUMAN SYSTEMS include historical, ethnic, and cultural systems that must be considered and preserved when deploying the SYSTEM. The same is true for NATURAL ENVIRONMENT ecosystems such as drinking water aquifers, wetlands, rivers, and habitat. Other examples include preservation of historical and cultural sites, location of nuclear power plants near rivers for cooling towers, hospital radiological suites to limit radiation, radio and TV towers in restricted areas.

For some systems, the act of deploying a system, product, or service to a site with ENABLING SYSTEM FACILITY capabilities such as utilities—power, phone, data, water, and sewer—does not mean that it can perform missions. *System sustainment* becomes a major consideration, especially for those systems requiring MISSION RESOURCES such as raw materials, parts, fuel, and natural resources—lumber, wood pulp, and rocks. Location of the SYSTEM near MISSION RESOURCES is a major driver in the site selection process.

This leads to the question: *how is a site selected for System Deployment*? Site selection requires informed knowledge from a number of sources. Examples include previous usage, the Internet, satellite photos, and topographic maps. Although this is beneficial information, the ultimate information comes from on-site surveys of candidate locations. This brings us to our next topic, Site Selection Decision Factors and Criteria.

29.3.4.2.2 Site Selection Decision Factors and Criteria Site selection often involves exploring various options that require conducting a trade study (AoA), especially if the User has not designated a site. This requires identifying and weighting *decision factors* and *criteria* based on the User values and priorities. Consider the following example:

Example 29.2

Decision Factors and Criteria (Chapter 32)

Site Selection:

- *Decision factor* examples might include access to customers, airports, waterways, major highways, level land, availability of labor force and skills,

taxes, operating costs, educational resources, and climate conditions.
- *Decision factor criteria* examples consist of further refinements as contributory elements of the decision factors.

Collaborate with the User via Acquirer contract protocol to establish site selection decision factors and criteria. Each criterion requires identifying *how* and from *whom*—Stakeholders—the data will be collected while on site or as follow-up data requests via the System Acquirer.

Decision factors and criteria include two types of data: *quantitative* and *qualitative*.

- **Quantitative** For example, the facility operates on 220 vac, 3-phase, 60-Hz power.
- **Qualitative** For example, what problems you encountered with the existing or legacy SYSTEM that Users want to avoid when installing, operating, and supporting the new SYSTEM.

Obviously, we prefer all data to be *quantitative*. However, *qualitative* data may provide insights into *how* the User truly feels about an existing or legacy system or the agony of installing one. Therefore, structure *open-ended* questions to encourage the User to *openly express* their thoughts about previous SYSTEM or PRODUCT deployments and experiences. These might include what they believe to be their key decision factors, criteria, and weights. Then, aggregate all the responses into a draft list for collaborative review by the set of Stakeholders, followed by User weighting of the decision factors and criteria (Figure 32.8).

Given an understanding of Stakeholder expectations for selection of a deployment site, we shift our attention to Site Surveys.

29.3.4.3 Perform Site Surveys On-site surveys provide a key opportunity for a Site Survey Team to *observe how* the User envisions operating, maintaining, storing, and sustaining a SYSTEM or PRODUCT. Some sites may not be developed or postured to accommodate a new SYSTEM or PRODUCT.

On-site surveys are more than surveying the landscape. Key factors include environmental, historical, and cultural heritage artifacts considerations. Site survey activities include developing a list of Critical Operational or Technical Issues (COIs/CTIs) to be investigated and resolved prior to the site visit. For existing facilities, the site surveys also provide insights concerning the physical state of the existing facility, as well as COIs/CTIs related to *modifying* the building or integrating the new SYSTEM while minimizing interruptions to the Enterprise's workflow.

On-site surveys are also a valuable means of investigating and assessing *environmental* and *operational* challenges. The Site Survey Team should explore various options for installation such as Day of the Week (DOW), Time of Day (TOD), holiday, and plant shutdown periods. Generally, the site surveys consist of a preparatory phase during which NATURAL ENVIRONMENT information about geographical, geologic, and regional life characteristics are collected and analyzed to properly understand the potential environmental issues.

If existing or legacy SYSTEMS exist that are comparable, they may provide an *invaluable* opportunity to explore physical challenges. These challenges may pertain to constrained spaces that limit hands-on access, height restrictions, crawl spaces, lighting, environmental control—HVAC, data, phone, and satellite telecommunications.

On-site surveys (Mini-Case Study 4.1) reveal significant information and insights about entrance, passage, and doorway sizes, blocked entrances, entrance corridors with hairpin switchbacks, considerations of 60 Hz versus 50 Hz versus 400 Hz electrical power, 110 vac versus 230 vac, that drive SPS requirements. Research or request facility documentation and carry it with you to review during each visit. Visually observe the facility and conduct measurements, if required, and *validate* the currency and accuracy of the documentation including operating policies and procedures through discussions with facility PERSONNEL.

Author's Note 29.3

Field Documentation Integrity and Maintenance

Site surveys are crucial for *validating* User documentation for decision making and identifying unexpected obstacles. Enterprise source documentation of fielded MISSION SYSTEMS and ENABLING SYSTEMS tends to be lax; drawings are often out-of-date and may not reflect current configurations of EQUIPMENT and FACILITIES.

Visit the deployment site, preferably *after* a thorough review of site documentation and interviews. If impractical, you may need to re-evaluate this business opportunity. Otherwise, you need an innovative, Cost Plus Fixed Fee (CPFF) contract that *shifts* financial, technical, and schedule *risk* to the Acquirer or User.

The context here is Earth-based site surveys. Now consider some System Deployment challenges of space-based missions to the Moon, Mars, or other destinations.

In the earlier days of space travel, NASA site surveys, for example, were dependent on:

- *Remote, off-site* observation-based knowledge gleaned over several centuries through astronomy and physics.

- *On-site visits* beginning with unmanned Surveyor moon-landers equipped with sensor and camera systems and culminating with the Apollo manned missions with experiments.

Site surveys are also applicable to harsh OPERATING ENVIRONMENT conditions following an event. Examples include the 1986 Chernobyl nuclear plant disaster or the 2011 Tōhoku earthquake and tsunami in Japan. In cases such as these, site surveys require remote platforms such as satellites and Unmanned Aerial Systems (UASs), robots with special sensors, and other types of devices.

Given this backdrop, let's address how site surveys are planned and conducted.

29.3.4.3.1 Define Site Survey Data Collection Requirements When identifying site data requirements, *prioritize* questions to accommodate time restrictions for site personnel interviews. There is a tendency to prepare site survey forms, send them out for responses, await their return, and then analyze the data. While this can be helpful in some cases, potential respondents today do not have time to fill out surveys. On-site collaboration based on specific data to be obtained is often the best method for gaining true insights.

One method for pre-site visit information may come from alternative sources such as satellite or aerial photos, assuming that they are current, and teleconferences with site personnel. When conducting teleconferences, ask *open-ended* questions that encourage the participants to answer freely rather than asking closed-ended questions that require *yes* or *no* answers. Clarify your understanding about the site facility, their capabilities and limitations.

Author's Note 29.4

Suggestion—Develop Draft Site Survey Report

Inevitably, site survey reports sometimes *overlook or fail to address* topics that are important to the System Deployment. One method for *minimizing* the risk of overlooked topics is to create a draft of the report prior to the site survey using mock data. Then, distribute to peers for review and comment. Typically, the review comments provide new insights for data to be collected or issues to be addressed before going on a site survey.

29.3.4.3.2 Coordinate the On-Site Survey(s) One of the most fundamental rules of site surveys is advance coordination with the System Acquirer via contract protocol. In preparation for the site visit, verify the need for special or security clearances and vehicle access and operating procedures *several days or weeks in advance*. Obtain *written* approval for use of cameras and recording devices—if permissible—for documenting meeting notes.

Author's Note 29.5

Site Visit Limitations

Always confer with site decision *authorities*, not general personnel, prior to the visit as to what media are permitted on site for data collection and any data approvals required before leaving. Some Enterprises require written notes and data to be relinquished at the end of the visit for internal review, approval, and delivery following the visit. You may be allowed only one visit—make sure you obtain the site data you need in that single visit.

29.3.4.3.3 Conduct the On-Site Survey(s)

Heuristic 29.1 Site Survey Access

Collect all the information you require on the first trip; you may be denied access for a second visit.

Heuristic 29.2 On-Site Survey Observations

Site survey visits are more than data collection activities. Equally important is what you *did not see that you expected to see.*

During the site visit, *observe* and *ask* about everything related to the SYSTEM or PRODUCT's deployment, installation, integration, operation, and sustainment including processes and procedures. *Leave nothing to chance*!

Before you leave, request some time to assemble your team in a conference area and reconcile notes. Think about what you *observed* and *did not* see that you *expected* or would have *expected* and *why not*! If necessary, follow up on these questions *before* you leave the site.

Raw data results from each site survey should be summarized and compiled in a formal report for reference and development of an AoA, especially if site trade-offs are required. Since observations of the site visits may differ among team members, if appropriate, ask the host to clarify or validate those perspectives. Remember—the Site Survey Report is more than just a perfunctory document. The report(s) establishes the foundation knowledgebase from which technical and design operational decisions will be made. For those who were unable to participate, the *accuracy and integrity* of this information in the Supply Chain shown earlier in Figure 4.1 are crucial!

29.3.4.4 Select Deployment Site (As Applicable) On the basis of data collected and documented in the site survey data reports, *collaborate* with the Users to jointly conduct an AoA (Chapter 32) to select the deployment site. When the site has been selected or designated and approved by the User, the System Deployment process will require development and approval of a Site Development Plan.

29.3.4.5 Create and Approve Site Development Plan (As Applicable) In general, the Site Development Plan should describe actions required to provide road access, land surveys and development, utilities—power, water, and sewer, communications—security, fire, parking, lighting. Investigate and document the types of permits, approvals, and inspections required by the local, state, and federal levels.

Approval of site development plans may require several weeks, months, or years. At issue are things such as statutory and regulatory compliance, environmental impact considerations, site of historical artifacts. Thoroughly perform your research and homework to achieve approval success; it may require more than you anticipate.

A Word of Caution 29.2

Leverage Qualified Consultant Services

Avoid the notion that all you have to do is simply write the plan and have it approved in a few days. When you prepare the plan, employ the services of a qualified, professional consultant, or Subject Matter Expert (SME) to make sure that all key tasks and activities are properly identified and comply with federal, state, and local statutes, regulations, and ordinances.

Identify the following:

- The decision-making chain, decision makers by name, and authorities.
- *What* types of documentation —plans, specifications, design drawings, permits—are required to obtain approval.
- *When* documentation—forms, permits, licenses, deviations, waivers —must be submitted for approval.
- *When, where, and to whom* documentation approval requests should be submitted.
- *How long* the typical documentation approval cycle is.

29.3.4.6 Prepare and Develop the Deployment Site Once the site has been selected and acquired, the next step is to prepare and develop the site. Site preparation includes those activities required to prepare land to accept the deployed System or a Facility to house the deployed System. This may include surveying and grading the land, building temporary bridges; installing utilities—power, water, sewer; lines; landscaping and drainage.

When the site preparation has been completed, the next step is to conduct *site inspections*. Site inspections require on-site compliance assessments:

- By the Stakeholders to ensure that the site and/or Facility is ready to accept the new System.
- By local and Acquirer representative authorities to verify compliance to statutory and regulatory constraints.

Author's Note 29.6

Site Inspections Versus Site Capabilities

Remember—site inspections by local, state, and federal authorities assess *compliance* to statutory or regulatory requirements. They *do not assess* whether the site has the requisite Mission Resources required to install, integrate, operate, maintain, and sustain the System or Product. That is the responsibility of the System Acquirer that may be fully or partially transferred to the System Developer SE.

29.3.4.7 Construct and/or Modify the Facility (As Applicable) During or after completion of the site development, if applicable, the next step is to construct a new facility or modify an existing Facility, wherever is applicable. The key here is to prepare the Facility for Assembly and I&CO of the System. In preparation for the System Deployment, Facility interfaces should be verified in accordance with the FIS. Completion of the Facility will require additional inspections and approvals.

29.3.4.8 Deploy the System to the Site Facility The deployment of a System or Product should occur Just-in-Time (JIT) with the completion of a Facility or to an interim storage Facility on-site or nearby until the permanent Facility is ready for System I&CO.

Author's Note 29.7

As a reminder, transport of a System to a deployment site may be performed by the System Developer, User, or a third party under contract to the User. The System Development contract should specify *who* is *accountable* for the deployment.

29.3.4.9 Assemble or Set-up and/or Install and Checkout the System Once the field site is prepared to accept the System, the next step is to Assemble or set-up, I&CO, and verify the System, assuming that its mission is at this Facility or staging area. System I&CO covers a sequence of activities, Enterprise roles and responsibilities, and tasks before the newly deployed System can be located at a specific job site. System requirements unique to on-site System I&CO must be identified by site surveys and analysis and incorporated into the SPS *prior to* the Contract Award.

29.3.4.9.1 Train User Operators and Maintainers When a new System is ready to be deployed, a key task is to train Personnel to deploy, install, and check out, operate, and maintain the System, as applicable. Generally, a System Development contract will require the System Developer to train User Personnel *prior to* disassembly at the System Developer's Facility or system integration at a deployment

FACILITY or *staging area*. The training sessions should prepare Users to *properly* and *safely* operate and to support OT&E (Chapters 12 and 13) during the final portions of the System Development Phase.

29.3.4.9.2 Validate SYSTEM *Performance* I&CO of new systems often requires integration into User's HIGHER-ORDER—LEVEL 0—SYSTEM. The integration may involve introduction of the new SYSTEM as an additional element or as a replacement for an existing or legacy SYSTEM. Whichever is the case, Level 0 System integration often involves CTIs, especially from the standpoint of *compatibility*, *interoperability,* and *security*. Thoroughly investigate these issues and mitigate.

Depending on the level of urgency to place the SYSTEM into *active service*, some Acquirers and Users may require a new SYSTEM to operate in a "shadow" mode to validate SYSTEM RESPONSES to external stimuli, excitations, or cues, while the existing SYSTEM remains in place as the primary operating element. This is very important, especially for financial or medical systems in which End User health, resources, and life may be at risk due to an *unproven* SYSTEM in a new facility.

On completion of the evaluation, assessment, and certification, the new SYSTEM may be brought "on line" as an Initial Operational Capability (IOC) (Figures 15.5 and 15.6) to replace the legacy SYSTEM. Incremental capabilities may be added via upgrades until Full Operational Capability (FOC) is achieved. To illustrate the *criticality* of this type of deployment and integration, consider the following example:

Example 29.3

System Results Validation as a Pre-Condition for System Integration

Financial Enterprises such as banks depend on highly integrated, audited, certified systems that *validate the integrity* of the overall SYSTEM. Consider the magnitude and significance of decisions related to integrating either new or replacement software systems to ensure *interoperability* without degrading SYSTEM performance or compromising User and End User confidence in its *integrity* or security such as transfer errors.

Lastly, as part of the System Deployment, User operators and maintainers may require training and certification, our next topic.

29.3.4.9.3 Certify/Recertify the FACILITY/SYSTEM*; License Operators (As Applicable)* Some types of systems, such as industrial sites, aircraft, power plants, may require certification and periodic recertification based on some calendar or operational use-based metric. Observe that the context here relates to *certification/recertification* of a site, FACILITY, and EQUIPMENT. However, these require PERSONNEL—operators and maintainers, which is a separate issue. The SOI's PERSONNEL Element may require both certification/recertification and licensing to operate the EQUIPMENT Element. Thoroughly investigate local, state, and federal requirements for each of these areas.

Heading 29.2

The preceding discussions provide a high-level overview of what is to be accomplished to deploy a system. On the basis of this information, we can develop an SE methodology for System Deployment considerations.

29.3.5 Deployment SE Methodology

From an SE perspective, System Deployment requires a strategic methodology that supports analysis of Pre-Deployment, Deployment, and Post-Deployment Operations (Figure 6.4). The following is one example of the methodology:

- Step 1—Collaborate with Deployment Stakeholders—Users and End Users
- Step 2—Identify Deployment Constraints
- Step 3—Evaluate and Select Deployment Mode(s) of Transportation and Sequences
- Step 4—Model Deployment Operations and Interactions
- Step 5—Bound and Specify System Deployment Interactions and Interfaces
- Step 6—Define Deployment Route and Required Modifications
- Step 7—Conduct a Mock Deployment (Optional)
- Step 8—Mitigate Deployment Risks

29.3.5.1 Step 1—Collaborate with Deployment Stakeholders—Users and End Users System Deployment often involves large numbers of geographically dispersed Stakeholders. Therefore, Stakeholders should be *actively* involved in the decision-making process from the early planning stage, but subject to contract type limitations. So, *what happens if you fail to include these Stakeholders?*

Depending on the situation, a Stakeholder—User or End User—could become a *showstopper* and significantly impact System Deployment schedules and costs. Do yourself and your Enterprise a favor. Understand the deployment, site selection and development, SYSTEM I&CO, and acceptance decision-making chain. This is key to ensuring success when the time comes to deploy the system. *Avoid* a "downstream" showstopper situation simply because you and your Enterprise chose to *ignore* some odd suggestions and views during the System Development Phase.

29.3.5.2 Step 2—Identify Deployment Constraints Environmental constraints can have a major impact on all facets of system analysis, design, and development as well as on all phases of the System/Product Life Cycle—specifically, local, state, and federal statutory and regulatory requirements.

29.3.5.2.1 Environment, Safety, & Occupational Health (ES&OH) Constraints ES&OH is a critical issue during System Development and Deployment. The objective is to *safely* and *securely* relocate a SYSTEM without impacting its capabilities and performance or endangering the health of the public, the NATURAL ENVIRONMENT, or the deployment team. *Always* investigate the requirements to ensure that the ES&OH concerns are properly addressed in the design of the MISSION SYSTEM and ENABLING SYSTEM EQUIPMENT as well as the mode of transportation—land, sea, air, or space. ISO 14000, for example, serves as the international standard used to assess and certify Enterprise environmental management processes and procedures.

29.3.5.2.2 Statutory and Regulatory Constraints Statutory and regulatory requirements concerning environmental protection and transportation of Hazardous Materials (HAZMAT) are mandated by local, state, federal, and international organizations. These regulations are intended to protect the cultural, historical, religious, and political environment and the public. SEs have significant challenges in ensuring that new systems and products are properly specified, developed, deployed, operated, sustained, and fully comply with statutory and regulatory requirements. Consider the following example:

Example 29.4

Environmental Constraints

The US National Environmental Policy Act (NEPA; U.S. Public Law 91–190, 1969) of 1969 and Environmental Protection Agency (EPA) establish requirements on SYSTEM deployment and OM&S, retirement, and disposal that impact the NATURAL ENVIRONMENT. In many cases, System Developers, Acquirers, and Users are required to submit advance documentation such as Environmental Impact Statements (EIS) and other types of documents for approval prior to implementation.

The US Occupational Safety and Health Administration OSHA 29 CFR 1910 (1971) establishes standards for occupational safety and health.

Author's Note 29.8

Research Contract and Enterprise

Always consult the provisions of your contract as well as with your Contracts; Legal; and ES&OH organizations for guidance in complying with the appropriate statutory and regulatory environmental requirements.

29.3.5.2.3 Deployment Environment Constraints Once the deployment mode of transportation and sequences is established, specify and bound the land, sea, air, or space transport environment conditions such as the examples provided in Table 29.2. Temperature, humidity, shock and vibration, dust and sand, and altitude are key examples of parameters SEs should consider. Bound operating constraints such as sea states, roadway conditions—International Roughness Index (IRI). Refer to ASTM E1926–08 (2008) and ASTM E1364–95 (2012).

29.3.5.2.4 Environmental Reclamation Constraints Environmental resources are extremely fragile. Today, great effort is being made to preserve the NATURAL ENVIRONMENT for future generations to enjoy. Therefore, during the transport of a SYSTEM to a new site, the risk of spills onto the ground and emissions into the atmosphere should be *minimized* and *mitigated* to a level acceptable by law. When the System is *relocated*, *retired*, or *disposed*, there may be requirements for environmental *reclamation* to re-establish the NATURAL ENVIRONMENT back to its natural or original state.

29.3.5.3 Step 3—Evaluate and Select Deployment Mode(s) of Transportation and Sequences

Principle 29.2

System Delivery Method Compatibility Principle

The design of every system, product, or service must be physically *compatible* with its method of delivery and constraints.

In general, most SYSTEMS or PRODUCTS are deployed by one of several approaches:

- **Approach #1**—Containerized shipping—commercial products, parcel packages, fruits and vegetables, modular containers on ships.
- **Approach #2**—Combinations of land, rail, sea, air, or space-based transport.
- **Approach #3**—Deployment/relocation of a SYSTEM under its own power such as aircraft, ships, spacecraft.

Transportation by land, sea, air, space or combinations of these options is the way most systems, products, and services get from Point A to Point B. Each mode of transportation should be investigated and evaluated in a trade study AoA that includes cost, schedule, efficiency, and timing considerations.

Author's Note 29.9

Avoid Specification of Modes of Transportation

As a reminder, unless there are *compelling* reasons to do otherwise, the SPS does not need to specify *how to deploy* the SYSTEM for

delivery or during the System OM&S Phase. Instead, the required operational interface capabilities and performance should be bounded to allow the System Developer the flexibility to select the *optimal mix* of deployment method(s) and modes of transportation. Remember—the SPS or any specification specifies *what* has to be accomplished and *how well*, not *how*. If the Acquirer or User is compelled to dictate the mode(s) of transportation in the SPS, that is a Statement of Work (SOW) topic, not a specification requirement.

Verify that a system, product, or service design is *compatible* and *interoperable*, if necessary, with the transport vehicle used to deploy the SYSTEM to its designated field site. When planning deployment of a SYSTEM or PRODUCT, include considerations by those Stakeholders such as municipalities and states that permit System Deployment through their jurisdictions. Key considerations include bridge clearance heights above roads and *maximum* load weight limits; barge, truck, aircraft payload restrictions; hazardous material passage through public areas, and aircraft landing constraints.

Modes of transportation require other considerations that go beyond electromechanical interfaces. Aircraft, for example, require depressurization of vehicle tires of vessels and removal of fuel and fluids due to altitude changes that may create *unsafe* operating conditions or a catastrophe.

Investigate Local, State, and Federal Transportation Regulations

Author's Note 29.10 When investigating modes of transportation, consult local, state, and federal regulations concerning transport requirements. Examples include the following:

- Government documents—land, sea, air, and space.
- Highway regulations such as the US Department of Transportation (DOT) Federal Highway Administration FHWA-HOP-04-022 (2004).

Consult regulations appropriate to a country and mode of transportation.

29.3.5.4 Step 4—Model Deployment Operations and Interactions PART I SYSTEM ENGINEERING AND ANALYSIS CONCEPTS provides the foundation for the application of SE to System Deployment. Key analytical methods include Model-Based Systems Engineering (MBSE) methods (Chapters 10 and 33) using SysML™[1] tools such as UCs and scenarios; Interaction Diagrams such as Sequence and Collaboration; Activity Diagrams; State Diagrams; and other diagrams. SEs employ these tools to analytically model Pre-Deployment, Deployment, and Post-Deployment Phases, modes, and states such as those as illustrated in Table 6.2. Information provided in these diagrams provides a framework the identification and derivation of specification requirements for MISSION SYSTEM and ENABLING SYSTEM capabilities, interfaces, operating constraints.

System Deployment modeling and analysis include approaches such as the System Operations Model discussed earlier in Figures 6.1 and 6.2. Perform operational and task analysis by sequencing the chain of events required to relocate the SYSTEM from Point A to Point B. This includes cost, performance, and risk trade-offs for competing land, sea, air and space modes of transportation and transport mechanisms such as truck, aircraft, ship, train, rocket.

29.3.5.5 Step 5—Bound and Specify System Deployment Interactions and Interfaces Models of the deployment operations should aid in identifying interactions between the MISSION SYSTEM being deployed and the ENABLING SYSTEM transporting it (Figure 7.4). This requires special SE considerations related to electromechanical interfaces and measurement devices between the transportation device and container. This includes lift and tie downs points, Center of Gravity (CG) and Center of Mass (COM) markings, and electronic temperature, humidity, shock, and vibration sensors to assess the health and status of the deployed system and record worst-case transients. Additional special considerations may include environmentally controlled transport containers to maintain or protect against cooling, heating, or humidity. Each of these types of interfaces represents requirements for the SPS.

29.3.5.6 Step 6—Select Deployment Route and Required Modifications Some System Deployment efforts require temporary modifications to roads, highways, and bridges; utility poles and power lines; signal and traffic light relocation; rerouting of traffic. Consider the following example:

Relocation of a House

Example 29.5 An existing house is to be physically relocated elsewhere within a town. In preparation for the move, planning and coordination are required to temporarily move utility lines and reroute traffic. In addition to the necessary permits and licenses, law enforcement officers will need to redirect traffic, and utility crews will need to lift and reinstate utility lines, including telephone, power, signal lights, and signage.

29.3.5.7 Step 7—Conduct a Mock Deployment (Optional) For large, complex systems that require special handling considerations, an optional mock deployment exercise may be appropriate, if practical and affordable. A mock deployment

[1] SysML™ is a registered trademarks or trademarks of Object Management Group, Inc. in the United States and/or other countries.

requires some form of surrogate SYSTEM such as a prototype or model that exhibits identical physical attributes and properties—size, weight, CG, and COM—to the SOI being transported to a deployment site. The exercise *debugs* the System Deployment Process and operations to facilitate identification of *unanticipated* scenarios and events for the processes, methods, and tasks of deployment. Table 29.2 provides a summary of example System Deployment Engineering considerations.

System Deployment transport not only includes deployment of the SOI but also includes its Enabling SYSTEM EQUIPMENT such as Common Support Equipment (CSE) and Peculiar Support Equipment (PSE) (Chapter 8). This requires consideration of how the SYSTEM or PRODUCT will be stowed and transported. A mock deployment can provide valuable insights for how CSE and PSE will be transported. This includes items such as the following:

- **CSE** such as hammers, screwdrivers, and other hand tools applicable to most systems.
- **PSE** such as specialty tools and devices unique to a specific system.

29.3.5.8 Step 8—Mitigate Deployment Risk When systems, products, or services are deployed, the natural tendency is to assume you have selected the best approach for deployment. However, political conditions in various states or countries can disrupt System Deployment and force you to reconsider alternative methods. Develop risk mitigation plans that accommodate all UCs and *most likely* or *probable* scenarios (Chapter 5) anticipated for System Deployment such as COIs and CTIs. Key considerations include safety and security risk, transportation environment risk, and NATURAL ENVIRONMENT risk. Let's explore each of these.

29.3.5.8.1 Investigate and Mitigate Route Safety and Security Issues The deployment of a new system from one location to another should be performed as expeditiously, efficiently, and effectively as practical. The intent is to *safely* and *securely* transport the system with *minimal* impact to it, the public, and the environment. Safety and security planning considerations include physical protection of and access to the SYSTEM to be deployed, the PERSONNEL who perform the deployment, and the ENABLING SYSTEM EQUIPMENT.

When developing a system, product, or service, at a minimum, factor in considerations to protect the SYSTEM during deployment operations and conditions to *minimize* the effects of physical harm such as appearance, form, fit, or stability. Examples include jet engine inlet covers or former NASA Space Shuttle Afterbody Fairing.

SYSTEM requirement considerations should include modularity of the EQUIPMENT for easy *removal*, separate transport, and *reinstallation* of sensitive components during deployment. This includes the removal of computer hard drives containing sensitive data that require special handling and protection. The same is true with HAZMAT items such as flammable liquids, toxic chemicals, explosives, munitions, and ordinances. In these cases, special equipment and tools—CSE and PSE (Chapter 8)—may be required to ensure their safe and secure transport by courier or by security teams. In addition, Material Safety Data Sheets (MSDS) concerning ES&OH should accompany the deployment.

Some systems require a purposeful deployment using *low-profile* or *visibility* methods to *minimize* publicity, depending on the *sensitivity* and *security* of the situation. This includes elimination of vehicle markings, time, day, or night deployments.

29.3.5.8.2 Monitor and Mitigate Transportation Environment Risk Professional systems and products such as precision instruments can be very sensitive to *shock* and *vibration*. Plan ahead for these critical design factors. Make sure that the SYSTEM while in a stand-alone mode is adequately designed to buffer any in-transit shock or vibration conditions that occur during deployment. This includes establishing appropriate design safety margins. Where appropriate, the SPS should specify transportation *shock* and *vibration* requirements.

29.3.5.8.3 Mitigate NATURAL ENVIRONMENT Risk Despite meticulous planning, environmental emergencies can and do occur during the deployment of a SYSTEM. Develop risk mitigation plans and coordinate resources along the transportation route to clean up and remediate any environmental spills or catastrophes. Transportation vehicles, systems, and shipping containers should fully comply with all applicable federal and state statutory regulatory laws for labeling and handling. Organizations such as the US Environmental Protection Agency (EPA) may require submittal of environmental documentation for certain types of projects. Investigate how the US NEPA, if applicable, and other legislation apply to your SYSTEM's development, deployment, OM&S, retirement, and disposal.

When transporting EQUIPMENT systems and products that contain various volatile, flammable, biological, or toxic fluids, HAZMAT spillage or explosions are always a major concern, particularly for wildlife estuaries, rivers, streams, and underground water aquifers. Perform insightful ES&OH planning and coordination to ensure that ENABLING SYSTEMS—PERSONNEL, PROCEDURAL DATA, and EQUIPMENT—are readily available to provide a rapid response to a hazardous event.

29.4 SYSTEM OPERATION, MAINTENANCE, & SUSTAINMENT (OM&S)

People often believe that SE is complete when the System Acquirer formally accepts delivery of a new System

or Product or its upgrade. In their minds the Engineering is complete. In terms of comparing the SE Level of Effort (LOE) during System Development, this is true; however, the SYSTEM performance assessment activities continue throughout the SYSTEM's *useful service life* (Figure 34.6) but with a smaller SE LOE, especially on large, complex systems. Thus, when the SYSTEM or PRODUCT is fielded, the System OM&S Phase begins with the shift from System Developer SE to User SEs. Depending on the type of system, product, or service, SE technical and analytical expertise may be required to monitor, track, and analyze SYSTEM performance based on *actual* field operations.

During the OM&S of a SYSTEM or PRODUCT, accountability for SE resides with the User Enterprise. This can be accomplished via Enterprise SE personnel, through another organization within the Enterprise, or via a Systems Engineering and Technical Assistance (SETA) contractor.

Assessment of the SOI's performance begins with the System Element Architecture template shown in Figure 9.2. The architecture forms the basis for characterizing *how* a system, product, or service is intended to operate internally and respond to stimuli, excitation, or cues in its OPERATING ENVIRONMENT.

Since the SOI may be composed of one or more MISSION SYSTEMS that are supported by one or more ENABLING SYSTEMS, we will approach the discussion from these two perspectives. This includes the following:

- Assessment of System *operational utility, suitability, availability*, *usability*, *effectiveness*, and *efficiency* (Principle 3.11)
- Elimination of *latent defects* such as design flaws, errors, deficiencies, as well as COIs/CTIs.
- Identification and removal of *defective* materials or components.
- Corrective action for *poor workmanship* practices.
- Optimization of SYSTEM performance.

These activities also represent the *beginning* of collecting requirements for:

- Procurement of new follow-on systems, products, or services.
- Upgrade or retrofit of capabilities to the existing or legacy SYSTEM.
- Refinement of current MISSION SYSTEM and ENABLING SYSTEM performance.

29.4.1 OM&S Objectives for System Engineering (SE)

Once an SOI is fielded, SE continues to be an integral part of the project. Specifically, System utility, suitability, usability, availability, efficiency, and effectiveness be *monitored* and *tracked* continuously. The actual project SE LOE when compared to System Development will generally be small and tend to be task driven. Depending on the size and complexity of the SYSTEM, this may require a full-time staff of one or more SEs or requests for support on specific tasks from external Enterprises with SE capabilities on an *As Requested*(AR) basis. This requires SE expertise to address the following objectives:

1. Monitor and analyze the *operational utility, suitability, availability, usability, effectiveness, and efficiency* (Principle 3.11) This includes system applications, capabilities, performance, and the *useful service life* of the newly deployed system including products and services—relative to its intended mission(s) within a prescribed OPERATING ENVIRONMENT.
2. Identify and correct any residual *latent defects* such as design flaws, errors, deficiencies, and faulty or defective components and workmanship.
3. Maintain awareness of the "gap" (Figure 4.3) in the *problem space* that evolves over time between this existing or legacy system, product, or service and comparable competitor and adversarial capabilities.
4. Accumulate and evolve requirements for a new SYSTEM, PRODUCT, or upgrade to the existing system to fill the *solution space(s)* and eliminate or alleviate the Enterprise *problem space*.
5. Propose interim operational solutions such as plans and tactics to fill the "gap" until a replacement capability is established.
6. Maintain SYSTEM Developmental or Production Configuration baselines (Chapter 16).
7. Maintain MISSION SYSTEM and ENABLING SYSTEM *compatibility* and *interoperability* (Principle 10.3).

Let's explore each of these objectives.

29.4.2 Monitor and Analyze SYSTEM/ELEMENT Performance

Once a system, product, or service is delivered to its Users for the OM&S Phase, their lingering question is: *did we acquire the right system to accomplish specific Enterprise missions*? The answer begins with *customer satisfaction*, which is influenced by the SYSTEM's *operational utility, suitability, availability, usability, effectiveness, and efficiency* (Principle 3.11).

From a contract perspective, formal completion of System V&V, in general, and Acquirer-User *acceptance* formally closes arguments concerning SYSTEM compliance to its Exceptions include System or Product warranties. At delivery, Stakeholder Users and End Users will ultimately have to live

with the SYSTEM and answer the Six Stakeholder Decisions presented earlier in Chapter 3 Section 3.9.

Accountability for answering these questions resides with SE. This leads to the question: *if SE is expected to answer these questions, what aspects of* SYSTEM *performance would you monitor to gather data to support your analysis to arrive at a result?*

The answer begins with the System Element Architecture template shown in Figure 9.2. Since the model represents the integrated set of capabilities required to achieve Enterprise mission objectives and, SYSTEM UCs, we allocate SPS requirements to each of the System Elements—EQUIPMENT, PERSONNEL, FACILITIES. Given this analytical framework, SEs and EDS

1. *How well* is each element of the System Element Architecture Model and their respective physical components performing relative to its current SPS or EDS requirements?
2. What is the operating condition of each System Element and physical components in terms of its *useful service life* (Figure 34.24)?

Let's explore each of these points.

29.4.2.1 Monitoring* SYSTEM *Level Operational Performance Real-time system performance monitoring should occur throughout the Pre-Mission, Mission, and Post-Mission Phases of operation (Figure 6.4) based on the Mission Event Timeline (MET) (Figure 5.5). Depending on the MISSION and SYSTEM application, the achievement of Enterprise mission and SYSTEM objectives, in combination with the MET, provides the basis for operational system performance evaluation.

In general, answering *how well* the SYSTEM is performing depends on the following:

- Who you interview—Stakeholders such as Users and End Users.
- Stakeholder roles, missions, and objectives.

Consider the following example from the perspectives of the System Owner and the System Developer:

Example 29.6

System Level Performance—System Owner—User Perspectives

From the System Owner's perspective, example questions might include the following:

1. Are we achieving mission objectives and performance?
2. Are we meeting our projected financial Total Cost of Ownership (TCO) targets?
3. Are the SYSTEM maintenance costs in line with projections?
4. Are there key *preventive* or *corrective* maintenance actions that can be reduced through Pre-Planned Product Improvements (P3I)?

From the SYSTEM User's perspective – operator, maintainer, or instructor, example questions might include the following:

1. Is the SYSTEM achieving its performance contribution thresholds established by the SPS?
2. Are there capability and performance areas that need to be improved?
3. Have SYSTEM upgrades or corrective actions enhanced system capabilities or degraded performance?
4. Is the SYSTEM responsive to our mission needs?
5. Does the SYSTEM exhibit any *instabilities, latent defects*, or *deficiencies* that require corrective action?

Author's Note 29.11

Observe the User's first item, SPS "performance contribution thresholds." Users often complain that a SYSTEM does not "live up to their expectations." Several key questions are as follows:

1. Were those "expectations" documented as explicit requirements in the SPS that served as the basis for System Development?
2. Did the System Acquirer, as the User's contract and technical representative, *verify* and *technically accept* the SYSTEM as meeting those SPS requirements?
3. Using System Acceptance as a point of reference, have User expectations changed?

Remember, aside from normal performance *degradation* from *use, misuse, misapplication, abuse*, lack of *proper maintenance*, and OPERATING ENVIRONMENT threats, the SYSTEM design, as an inanimate object, does not change: PERSONNEL—operators, maintainers, trainers—MISSION RESOURCES, and PROCEDURAL DATA evolve and change. So, if the SYSTEM *is not* meeting User expectations, what has changed: (1) the SYSTEM, (2) the operators, or (3) the Enterprise?

These are a few examples of the types of SYSTEM-level questions SEs need to ask. Once you have a good understanding of SYSTEM COI / CTI performance areas, the next question is: *what System Elements are primary and secondary performance effecters that drive these results?* Figure 5.8 illustrating an automobile's mileage Measure of Effectiveness (MOE) serves as an example.

29.4.2.2 Monitoring System Elements' Operational Performance On the basis of SYSTEM operational performance results, the key question for SEs to answer is: *what are the System Element contributions*? (Figure 10.12 and MOEs Figure 5.3) System Element performance areas include the following:

1. EQUIPMENT Element
2. PERSONNEL Element
3. MISSION RESOURCES Element
4. PROCEDURAL DATA Element
5. SYSTEM RESPONSES Element—behavior, products, by-products, and services
6. ENABLING SYSTEM elements such as the DoD's Integrated Logistics Product Support (ILPS) (DoD, 2011, p. 12–13) comprised of:
   a. Computer Resources.
   b. Design Interface.
   c. Facilities and Infrastructure.
   d. Maintenance Planning and Management.
   e. Manpower and Personnel.
   f. Packaging, Handling, Storage, and Transportation (PHS&T).
   g. Product Support Management.
   h. Supply Support.
   i. Support Equipment—for example, CSE and PSE (Chapter 8)
   j. Sustaining Engineering.
   k. Technical Data Management (Chapter 17).
   l. Training and Training Support.

At this juncture, we have established SE OM&S objectives. We now shift our focus to how SYSTEM performance monitoring is accomplished.

29.4.2.3 Why Analyze MISSION and SYSTEM Performance Data? You may ask: *why do SEs need to analyze system performance data*? If the SYSTEM works as specified, *what do you expect to gain from the exercise*? *What is the Return on Investment (ROI)* for the exercise? These are valid questions. Actually, there are several objectives that drive the need to analyze MISSION SYSTEM and ENABLING SYSTEM performance data. We partition these into two contexts: (1) current SYSTEM performance and (2) next generation systems.

- Context #1: Current SYSTEM performance.
  a. Objective #1: Benchmark nominal SYSTEM performance.
  b. Objective #2: Identify and track SYSTEM performance trends.
  c. Objective #3: Improve/remediate operator training, skills, and proficiency.
  d. Objective #4: Correlate mission events with SYSTEM performance.
- Context #2: Next Generation Systems
  a. Objective #5: Support Enterprise mission and SYSTEM capability *gap analysis* (Figures 4.2 and 4.3).
  b. Objective #6: Validate Models and Simulations (M&S) (Chapters 10 and 33).
  c. Objective #7: Evaluate and improve human performance.

Let's explore each of these objectives.

29.4.2.3.1 Objective #1: Benchmark Nominal System Performance Establish statistical performance benchmarks via baselines, where applicable, for what constitutes *actual nominal* system performance. For example, automobile owners are often curious about how their vehicle's mileage compares with the manufacturer's window sticker metric of 30 miles per gallon average over specified driving, fuel, and road conditions (Figure 5.8). So track and compare vehicle fuel efficiency, then analyze what may be degrading performance.

29.4.2.3.2 Objective #2: Identify and Track System Performance Trends Use the nominal performance baselines as a benchmark comparison for SYSTEM *performance degradation* and *trends* (Figure 34.25) over its *useful service life* to ensure *preventive* and *corrective* maintenance actions occur at the proper time and are performed as prescribed.

Remember, the original SPS establishes a set of requirements based on human analysis, observations, M&S, prototyping, and informed estimates of achieving required performance. Verification simply *proved* that the deliverable SYSTEM or PRODUCT performed within specified boundary limits and conditions. Every HUMAN SYSTEM has its own unique idiosyncrasies that require monitoring and understanding whether it is *stable* or *drifting out* of specification over time, the *rate of drift*, and the *level of urgency* for corrective action. Consider the following example:

Example 29.7 **Benchmarking Nominal System Performance**

If a hypothetical performance requirement is 100 ± 10 units, you need to know and correlate System X's nominal performance of 90 with System Y's *nominal* performance of 100. Sometimes, this is important; sometimes it is not. The borderline "90" system could stay at that level due to variations in components throughout its useful service life. In contrast, the perfect

"100" system could drift to "115" beyond specification and require continual maintenance.

Returning to automobile example in Objective #1, assume that a commuter vehicle is only used to go back and forth to work over the same roads every day. At 20,000 miles, it averages 26 MPG. Now, at 30,000 miles, it only averages 24 MPG. *If this is a trend, should actions be taken to investigate the probable cause(s)*? *What has caused the trend*? SEs, System Analysts, and Engineers should be able to answer these questions.

29.4.2.3.3 Objective #3: Improve Operator Training, Skills, and Proficiency Investigate to determine if the System is being *unnecessarily* stressed, misused, abused, or misapplied by operators or maintainers. If so, characterize the conditions and sequences for further corrective action analysis. Likewise, determine if there are ways of improving the System to reduce *operator stress* from unnecessary actions (Figure 24.6).

29.4.2.3.4 Objective #4: Correlate Mission Events with System Performance Correlate System events with Mission events and operator observations in terms of System Responses and performance data. Ask yourself: *are we observing a problem area or a symptom of a problem that has a root case traceable to latent defects, human error, or efficiency? (Figure 24.1)*

29.4.2.3.5 Objective #5: Support Mission and System Capability Analysis Collect *objective evidence* of existing System capabilities and performance to support "gap" analysis (Figures 4.2 and 4.3) between the current System and projected competitive or adversarial system performance. *Is the System or product becoming obsolete due to outdated technology? Is it time for a technology upgrade?*

29.4.2.3.6 Objective #6: Validate M&S *Validate* laboratory System M&S against actual System performance data to support future mission planning or assess proposed capability or performance upgrades (Chapter 32).

29.4.2.3.7 Objective #7: Evaluate and Improve Human Performance Personnel such as infantry, pilots, and NASA astronauts are subjected to operating environments that can *overstress* human performance (Figure 24.6). Thus, operator performance within the context of the overall Mission System performance must be well understood to ensure that operator and maintainer training *corrects* or *enhances* their performance for future missions. In addition, investigate ways of improving User operator or maintainer proficiency either through procedural changes or through System upgrades.

29.4.2.4 Performance Monitoring Methods System Element performance monitoring presents several challenges.

- Firstly, the User—operator or maintainer—SEs are *accountable* for questions about system performance and trends. If they do not have SEs on staff, support contractors may be tasked to collect data and make recommendations.
- Secondly, System Developer Enterprises proposing next-generation systems or upgrades to existing or legacy systems have to *apply Systems Thinking* in System performance areas as well as COIs and CTIs in a very short period of time. As a general rule, proposal Offerors should track and demonstrate performance in these areas over several years before they qualify themselves as competent suppliers. The competitive advantage resides with the incumbent contractor unless the User decides to change. Often, your chances of success are *diminished* if you wait to start answering these questions when the Draft Request for Proposal (RFP) solicitation is released; it is simply impractical.

So, for those Enterprises that prepare for and posture themselves for success well in advance of system acquisition, *how do they obtain the data*? Example data collection methods include the following, if authorized and accessible:

1. Personal interviews with Stakeholders—User operator and maintainer personnel and End Users.
2. Post-Mission data analysis of System PRs, mission debriefings, and after-action reports.
3. Visual inspections such as on-site surveys and checklists.
4. Analysis of *preventive* and *corrective* maintenance action records such as a FRACAS (Chapter 34), if available.
5. Observation of System Equipment and Personnel in action.

Although these methods may appear impressive on paper, they are only as "good" as the "corporate memories" of the User and maintainer Enterprises. *Data retention* following a mission drops significantly over several hours and days. This is why After-Action Reports are due immediately following a mission. Reality tends to become embellished over time. So, every event during the Pre-Mission, Mission, and Post-Mission Phases of Operation (Figure 6.4) becomes a critical *staging point* for after action and follow-up reporting. This requires three actions as follows:

1. Establishing record-keeping systems such as mission logs, FRACAS.

2. Ingrain professional discipline in PERSONNEL to record mission or maintenance event data.
3. Thoroughly document the sequence of actions leading up, during, and after a MISSION or SYSTEM maintenance event.

Depending on the consequences of the event, failure investigation boards may be convened that investigate the *who*, *what*, *when*, *where*, *why*, and *why not* of emergency and catastrophic events (Figure 24.1).

SYSTEM Users often lack proper training in reporting malfunctions or events. People, in general, dislike documenting events. So-called event reporting tools may not be *user friendly* and perform poorly. For these reasons, Enterprises should train PERSONNEL and assess their performance from Day #1 of employment concerning:

1. *What* Quality Records (QRs)—data—are to be maintained and in what form or media.
2. *Why* the QRs are required.
3. *When* QRs are to be collected.
4. *Who* the Users—SysML™ Actors.
5. *How* the User(s) employ the data for improving MISSION SYSTEM or ENABLING SYSTEM performance.

29.4.2.5 Reality Check The preceding discussions illustrate *Systems Thinking* (Chapter 1) related to SYSTEM performance monitoring. As a reality check, SEs need to ask themselves the question: *if we simply asked Users to identify and prioritize three to five* SYSTEM AREAS *for improvement, would we glean as much knowledge and intelligence as analyzing warehouses full of data?* The answer depends. If you have to have objective evidence to *rationalize* a decision, the answer is *yes*. Alternatively, establishing database reporting systems that can be queried provide an alternative. If you choose to take the shortcut and only identify the three to five areas for improvement, you may filter out what appears to be a *miniscule* topic that may become tomorrow's headlines. Choose the method *wisely*!

29.4.3 SE Focus Areas During OM&S

Every system, product, or service has a *useful service life*. (Chapter 34) From both an Enterprise and a project perspective, there are two contexts:

- **Context #1**—Sustaining and improving current system performance.
- **Context #2**—Planning for next generation systems or upgrades.

These contexts drive the need for SEs to assess SYSTEM performance in several focus areas:

- **Focus Area 1:** Correct *latent defects* such as design flaws, errors, and deficiencies; poor workmanship practices; and materials.
- **Focus Area 2:** Improve Human–System Integration (HSI) (Chapter 24) performance.
- **Focus Area 3:** Maintain MISSION SYSTEM training device concurrency such as Figure 33.5.
- **Focus Area 4:** Maintain the Developmental Configuration or Production baselines (Chapter 16).
- **Focus Area 5:** Perform and maintain a SYSTEM capabilities "gap" analysis (Figures 4.2 and 4.3).
- **Focus Area 6:** Bound and partition the MISSION-Level and SYSTEM-Level *problem* and *solution spaces* (Figures 4.7).
- **Focus Area 7:** Formulate and develop new capability requirements. (Figure 4.5)

29.4.3.1 SE Focus Area 1: Correct Latent Defects Systems, products, and services have *degrees of satisfaction* as viewed by the User and System Developer. System Developers employ design V&V practices to discover any latent defects, such as design errors, flaws, or deficiencies early in the System Development Phase when corrective actions are less costly (Figure 13.1). Despite the best of human attempts to perfect systems, latent defects inevitably go *undiscovered* until someone identifies the problem (Figure 13.2) during the System OM&S Phase—hopefully *without adverse effects or catastrophic consequences*.

New systems, especially large complex systems, inevitably have residual *latent defects*. Sometimes, these are *minor*; other times they are *major*. Any latent defect that has the potential to impact mission completion, life, or health of the Users—operators, maintainers, or trainer; the public; or the environment can become a *major risk item*.

From a System Acquirer, User, and System Developer perspective, the COI to be resolved is to ensure that the delivered EQUIPMENT and the User's operators and maintainers are able to achieve their mission objectives. This must be accomplished without subjecting themselves to injury, damage, or threats that might jeopardize the mission, public, or environment. Software-intensive systems are especially prone to *latent defects* that escape detection during formal System V&V and acceptance.

Latent defects that escape the System V&V process may go *undiscovered* after a SYSTEM is fielded until it encounters a unique set of OPERATING ENVIRONMENT conditions (Figure 24.1). Sometimes, they are highly obvious, and sometimes they are only detected over a period of time (Figure 13.2). The discovery may occur directly or indirectly during analysis of large amounts of data. Therefore, monitor and analyze SYSTEM Hardware PRs closely to determine if there are latent defects that need to be corrected and, if so, the *degree of urgency* in correcting them.

29.4.3.2 SE Focus Area 2: Improve HSI Performance

Our discussions up to this point focused on improving EQUIPMENT Element performance. However, EQUIPMENT is just one of several System Element performance effecters that contribute to overall SYSTEM performance such as the Ishikawa or Fishbone Diagram shown in Figure 10.12. Measurable SYSTEM performance may also be achieved by improving the PERSONNEL Element performance aspects (Figure 24.1) without having to procure new EQUIPMENT Element solutions. *How do we do this*?

EQUIPMENT Element performance is often limited by SYSTEM operator and maintainer skills, proficiency, and performance. SYSTEM operators and maintainers such as aircraft pilots may require *on-going* training and assessment to improve their knowledge, skills, and proficiency in understanding the limitations of the EQUIPMENT and how to properly apply it.

Human performance improvement requires education and training in several skills areas. These include areas such as basic training, remedial/refresher training, and advanced training.

- **Basic Training** Fundamental instruction complemented with hands-on experience to achieve a level of competence in basic system capabilities and levels of performance.
- **Remedial/Refresher Training** Retraining operator skills, proficiencies, and disciplines that may be deficient.
- **Advance Training** Specialized training in mission scenario environments that challenge the limitations of the human and machine and ensure a level of proficiency in achieving mission objectives.

The mechanisms for improving human performance include operator and maintainer selection, classroom training, field experience operating the system, and sometimes *luck*. Experienced SYSTEM operators and maintainers serving as Instructors are often responsible for training new students. The training, however, is dependent on the availability of training aids and devices that provide the students the look, feel, and decision-making environment that enables them to become proficient.

The question for SEs is: *how do we specify system capabilities that enable instructors to train and evaluate student performance on the equipment?* Training sessions need to employ devices that *immerse* the student in the types of operational and decision-making environments they will actually confront in the Operating Environment.

29.4.3.3 SE Focus Area 3: Maintain Mission System-Training Device Concurrency

Training devices such as models, simulations, and simulators must remain in synchronized lock step with the MISSION SYSTEM they simulate such as ground vehicles, aircraft, ships, nuclear reactors, robotics, surgical trainers, and so forth. For example, to avoid *negative training*, an aircraft and its simulator must perform *identically*. Otherwise, you may produce a *negative training* result.

By virtue of expertise and capabilities, the MISSION SYSTEM Developer is often different from the training device(s) Developer. Where *concurrency* of a simulator and the SYSTEM it represents is a COI/CTI, Users and the Acquirers must ensure that contracts are synchronized to support simulator device *concurrency* requirements to promote communications among the Enterprises.

29.4.3.4 SE Focus Area 4: Maintain SYSTEM or PRODUCT Baselines

Principle 29.3 **Fielded Systems Baseline Principle**

The As-Maintained Developmental Configuration baseline should *always* be maintained to identically match the physical configuration of the fielded SYSTEM or PRODUCT.

One of the challenges of fielded systems is failure to keep the "As-Maintained" Developmental Configuration baseline (Table 16.2) current. This commonly occurs when budgets are reduced or priorities are focused on other activities. Unfortunately, some Enterprises have a view that if an activity such as maintaining documentation does not help the Enterprise financial bottom line or to accomplish a mission, it is not worthy of investing resources. So, MISSION SYSTEM Users and System Developers ask: *do we invest money in the actual system to get more capability or in maintaining system baselines in lock step?* Generally, the capability argument prevails over SYSTEM documentation. As a result, there may be a discrepancy "gap" between the physical MISSION SYSTEM and its latest configuration baseline.

Once a SYSTEM or PRODUCT is developed, delivered, and accepted, a key question is: *who maintains the product baseline for fielded systems?* Systems often go through a series of major upgrades or Service Life Extensions (Figure 34.10) over their useful service life. Each improvement or upgrade may be performed internally or externally by a new System Developer contract. The challenge for the User and the Acquirer to ultimately answer is: *how do we assure the developer that the baseline configuration documentation identically matches the improved or upgraded, As-Maintained system?* Current, approved documentation, in general, is absolutely necessary if you expect to upgrade and retrofit an existing SYSTEM. As a User SE, you will be expected to answer this question and authenticate the integrity of the As-Maintained Product Baseline.

Maintaining the currency of the Developmental configuration Product Baseline (Chapter 16) is important not only for the existing fielded systems but also for future production runs. Some systems are fielded in small quantities for

test markets to assess consumer feedback in the marketplace. After the initial trials, production contracts may be released for large quantities. Some markets may saturate quickly or, if your Enterprise is fortunate to be first and have a lot of Rogers' (2003, p. 281) Adoption Curve Early Adopters and Early Majority, enjoy the success of SYSTEM production over several years. These production runs may involve a single production contractor or multiple "build-to-print" contractors; the baseline maintenance and integrity challenges, however, remain the same.

29.4.3.5 SE Focus Area 5: Perform and Maintain a System Capabilities "Gap" Analysis As MISSION SYSTEM and ENABLING SYSTEM performance data are collected and analyzed, a repository of knowledge is established. So, *what is the value of this data and its analysis?* Despite its title, the analytical results could be a single page for management with a brief rationale depicting the current gap and the rate of expansion over time. Figure 4.3 provides an illustration of the "capability gap" to be resolved.

The OPERATING ENVIRONMENT domain of most Enterprises and SYSTEMS is often highly competitive; conditions may range from *benign* to *adversarial* to *hostile*. Depending on the overall Enterprise mission, competitors and adversaries will continually improve and upgrade their system capabilities. Regardless of your Enterprise's operating domain, the marketplace, military threat environment, and consumer marketplace environments are dynamic and continually change in response to MISSION SYSTEM changes.

For some systems, *survival* means changing, either by necessity due to *product obsolescence*, cost of maintenance, or in response to marketplace trends and demands. Thus, operational capability and performance gaps emerge (Figure 4.3). This, in turn, forces investment to improve existing or legacy SYSTEM performance via upgrades or develop new systems, products, and services.

29.4.3.6 SE Focus Area 6: Bound and Partition the "Gap" Problem Space into Solution Spaces Enterprise "gap analysis" provides an assessment of current SYSTEM capabilities and performance relative to projected organizational needs as well as competitor and adversarial capability projections. Analysis of the "gaps" may reveal one or more potential *problem spaces* (Figure 4.3), each with its own *degree of urgency* for fulfillment. Each *problem space*, in turn, must be dynamically partitioned into one or more *solution spaces*. When a commitment is made to develop a new system, product, or service, the respective *Solution Space* becomes the basis for bounding and specifying the acquisition of the next-generation system.

29.4.3.7 SE Focus Area 7: Formulate and Develop New Capability Requirements Over time, the capability "gap" between the existing or legacy SYSTEM and projected needs widens as illustrated by Figure 4.3. At some point in time, a determination will be made to initiate actions to improve or upgrade existing system performance or to develop a new system or upgrade (Figure 4.5). Thus, the evolving *problem* and *solution space* boundaries specified by the Requirements Domain—SPS—will have to be captured in terms of required operational capabilities and costs. As an SE, you may be assigned responsibility to collect, derive, and quantify these requirements.

29.5 SYSTEM RETIREMENT (PHASE-OUT) OPERATIONS

Systems, products, or services inevitably reach a point in their *useful service life* in which they lack: (1) missions to perform or (2) the capabilities required to support projected organizational missions. The cost differential to (1) upgrade capabilities or (2) to operate, maintain, and sustain it as an Enterprise asset becomes *cost prohibitive*.

When systems fail to support Enterprise missions, their *utility* to the User is *diminished* and become an unnecessary maintenance cost. As a result, new system(s) may have to be acquired (Figure 4.5) to fulfill the planned missions. This requires (1) insightful planning, orchestration, and phase-out of existing systems and (2) introduction of new systems (Figures 4.3 and 4.5).

A decision is made to initiate these activities marks the beginning of the System Retirement Phase of the System/Product Life Cycle for existing systems, product, or services that are currently in *active duty service*. Figure 4.5 illustrates the transition.

29.5.1 System Storage Requirements and Design Considerations

Although SYSTEM storage (Figure 7.5) may appear to be an operational issue, the activity requires SE consideration, especially from a requirements and design perspective. EQUIPMENT Element SYSTEMS and PRODUCTS experience deterioration, rust, corrosion, residual fuel has a shelf life, lubricants and surfaces dry out, seals crack and leak, spontaneous combustion, hoses burst, tires flatten and crack, if left *unmaintained* in various types of environmental conditions. Therefore, SPS requirements must specify requirements for ensuring the SYSTEM or PRODUCT in a *serviceable condition* prior to reintroduction to *active duty service*. Consider the following example:

Example 29.8

Ancillary Specification Requirements and Design Considerations

Example ancillary specification requirements and design considerations such as:

- Easy removal of flammable liquids and fuels.
- Jacks to lift vehicle to relieve weight off tires.
- Engine block freeze plugs and heater connections for coolant and oil.
- Shelter and storage facilities, environmental control.
- Rodent damage prevention and inspection of coolant and hydraulic lines.
- Protective covers to prevent leaks.
- Coverings/attachment points for engine inlet/outlet coverings, propeller movement constraints, notifications such as "Remove Before Flight" flags and wheel chocks.

Author's Note 29.12

Specification Requirements versus Design Options

In today's world, composite materials provide a high-strength alternative to metal structures and parts that are prone to corrode over time. Remember—specifications specify *what* has to be accomplished and *how well*, not *how to* design the system. Corrosion, as a *problem space*, should be addressed as an SPS or EDS requirement. In contrast, usage of composite materials is a design Analysis of Alternatives Trade Study option (Chapter 32), not a specification requirement.

The above-mentioned example represents the technical aspects of System storage. Equally important is the need to accommodate these features and capabilities at the least operating maintenance cost.

29.6 SYSTEM DISPOSAL OPERATIONS

System Disposal has a number of different contexts as discussed in Chapter 3. Disposal of a System or Product could mean sale, disassembly, total destruction, parts salvage and partial destruction, burning, burial, and other approved methods. From an SE perspective, specification requirements and design considerations examples include the following:

- Easy removal of high value, sensitive, or toxic components and technologies such as heavy metals, fluids HAZMAT, and materials.
- Storage containers for salvaged devices and hazardous and toxic materials.

29.7 CHAPTER SUMMARY

Chapter 29 addressed key deployment, OM&S, retirement, and disposal of a system, product, or service. Key points of discussion included the following:

- System Deployment of a system, product, or service interface requirements for transport via an Enabling System mode of transportation such as land, sea, air, or space in accordance with local, state, deferral, and international statutes and regulations.
- Recognition of the potential need to develop sites—real estate—and facilities for *basing* or *staging* the new system, product, or service, site selection, site planning, operational site activation planning, System Assembly, I&CO, integration into High-Order Level 0 User systems (Figure 8.4).
- SE consideration for transport of a system, product, or upgrade to the deployment site including route modifications, coordination, environmental, licenses and permits.
- SE objectives during System OM&S include the need to baseline nominal System performance, monitor, and track trends as performance indicators of maintenance needs or issues. This includes both tracking and assessment of Mission System and Enabling System Elements such as Personnel, Equipment, Mission Resources, Procedural Data, System Responses, and Facilities performance.
- Information from the performance monitoring may reveal COIs and CTIs that represent the need for changes in Mission Resources or Procedural Data, upgrades for corrective action of latent defects, or new system development.
- Best practice examples for deploying a system, product, or service to a new site.
- Completeness of requirements and designs to accommodate System Disposal through ease of removal of components, hazardous or toxic materials (HAZMAT), and the need for containers for those until they can be properly dispositioned and disposed.

29.8 CHAPTER EXERCISES

29.8.1 Level 1: Chapter Knowledge Exercises

1. What is the objective of System Deployment?
2. What are some common System Deployment issues that require consideration?
3. What is site development? When does it start and end?
4. What is a site survey?
5. Who conducts site surveys?
6. How should you approach conducting a site survey?
7. How should a site survey be conducted?
8. What types of considerations go into selecting and developing a deployment site?

9. What is operational site activation and what is its scope?
10. Compare and contrast System Deployment and commercial SYSTEM or PRODUCT distribution.
11. What are some considerations that need to go into specifying Systems Deployment capabilities?
12. What is SYSTEM I&CO?
13. What are some of the considerations for integrating a SYSTEM or PRODUCT that has completed I&CO being integrated into a HIGHER ORDER Level 0 User's System? Use a specific example SYSTEM or PRODUCT as the basis for your answer.
14. What are some methods to mitigate risk during System Deployment?
15. Why is ES&OH a critical issue during System Deployment?
16. What are the primary objectives for System Operation, Support, and Sustainment (OM&S)?
17. What are key areas for monitoring and analyzing SYSTEM-Level performance?
18. What are the four key questions for assessing SYSTEM performance?
19. What are common methods for assessing SYSTEM-Level and Element performance?
20. What are the key SE focus areas for current SYSTEM performance?
21. What are the key SE focus areas for planning next generation systems?

29.8.2 Level 2: Chapter Knowledge Application Exercises

Refer to www.wiley.com/go/systemengineeringanalysis2e

29.9 REFERENCES

AR 200–1 (2007), Army Regulation: *Environmental Protection and Enhancement*, Washington, DC: Department of Defense (DoD).

ASTM E1926–08 (2008), *Standard Practice for Computing International Roughness Index of Roads from Longitudinal Profile Measurements*, West Conshohocken, PA: ASTM International.

ASTM E1364–95 (2012), *Standard Test Method for Measuring Road Roughness by Static Level Method*, West Conshohocken, PA: ASTM International.

DAU (2012), *Glossary: Defense Acquisition Acronyms and Terms*, 15th ed. Ft. Belvoir, VA: Defense Acquisition University (DAU) Press. Retrieved on 6/1/15 from: http://www.dau.mil/publications/publicationsDocs/Glossary_15th_ed.pdf

DoD (2011), *Logistics Assessment Guidebook*, Washington, DC: U.S. Department of Defense. Retrieved on 8/15/13 from http://www.dau.mil/publications/publicationsDocs/Logistics%20Assessment%20(LA)%20Guidebook%20(Final)%20July%202011.pdf

FAA (2012), *COTS Risk Mitigation Guide*, Washington, DC: Federal Aviation Administration (FAA).

FHWA-HOP-04-022 (2004), *Federal Size Regulations for Commercial Motor Vehicles*, U.S. Department of Transportation (DOT), Retrieved on 8/14/13 from http://ops.fhwa.dot.gov/freight/publications/size_regs_final_rpt/size_regs_final_rpt.pdf

ISO 14000 Series, *Environmental Management*, Geneva: International Organization for Standardization (ISO).

MIL-HDBK-470A (1997), *DoD Handbook: Designing and Developing Maintainable Systems and Products*, Vol. 1, Washington, DC: Department of Defense (DoD).

MIL-HDBK-881C (2011), *Military Handbook: Work Breakdown Structure*, Washington, DC: Department of Defense (DoD).

OSHA 29 CFR 1910 (1971), *Occupational Safety and Health Standards*, Occupational Safety and Health Administration (OSHA), Washington, DC: Government Printing Office.

Rogers, Everett M. (2003), *Diffusion of Innovation*, New York: Simon & Schuster – Free Press.

U.S. Public Law 91–190 (1969) *National Environmental Policy Act (NEPA)*. Washington, DC: Government Printing Office.

PART III

ANALYTICAL DECISION SUPPORT PRACTICES

30

INTRODUCTION TO ANALYTICAL DECISION SUPPORT

Chapter 1 introduced the Accreditation Board of Engineering and Technology (ABET) definition of "Engineering" as the " … application of *mathematical* and *scientific* principles … " We then expanded the scope of this definition for SE to include the application of *analytical* principles. You can debate the implicit relationship of analytical principles to Traditional Engineering mathematical and scientific principles. The reality is: you need to establish an analytical framework based on System or Product Life Cycle concepts - Deployment; Operations, Maintenance, and Sustainment (OM&S); Retirement; and Disposal - … before … you start applying the mathematical and scientific principles, which are all interdependent. SEs must be capable of analytically bounding and specifying:

- The User's OPERATING ENVIRONMENT, its constituent *opportunity, problem, and solution* spaces, and relevance to specific SYSTEM missions and applications.
- A System of Interest (SOI), its Use Cases (UCs), scenarios, capabilities, and performance.
- Interactions between the SOI and its prescribed OPERATING ENVIRONMENT to ensure (1) mission and system *compatibility* and *interoperability* with *authorized* external Users and systems and (2) security and protection from threat sources.

Chapter 30 introduces analytical decision support practices. Our discussions provide insights into the analytical decision-making environment, factors affecting the decision-making process, technical reporting of analytical results, and its challenges and issues.

30.1 DEFINITIONS OF KEY TERMS

- **Analysis** "A logical examination or study of a system to determine the nature, relationships, and interaction of its parts and environment." (FAA SEM, 2006, p. 4.1)
- **Analysis Paralysis** Refer to Chapter 27 Definitions of Key Terms.
- **Conclusion** A reasoned opinion derived from a preponderance of fact-based findings and other objective evidence.
- **Circular Error Probability (CEP)** The Gaussian (Normal) Distribution Probability Density Function (PDF) referenced to a central point with concentric rings representing the standard deviations of data dispersion.
- **Cumulative Error** A measure of the total cumulative errors that may be time-dependent and inherent within and created by a SYSTEM or PRODUCT when processing statistically variant inputs to produce a standard output or outcome.
- **Finding** A commonsense observation supported by in-depth analysis and distillation of facts and other objective data. One or more findings culminate in a conclusion.
- **Gaussian (Normal) Distribution** A graphical plot depicting the symmetrical dispersion and frequency

System Engineering Analysis, Design, and Development: Concepts, Principles, and Practices, Second Edition. Charles S. Wasson.

Companion Website: www.wiley.com/go/systemengineeringanalysis2e

of independent data occurrences about a central mean (Figure 34.2).

- **Hazard** "Refer to Chapter 24's *Definitions of Key Terms*."
- **Integrity of Analyses** "A disciplined process applied throughout a program to ensure that analyses provide the required levels of fidelity, accuracy, and confirmed results in a timely manner." (FAA SEM, 2006, Vol. 3, p. 4.1-13)
- **Recommendation** A logically reasoned plan or *course of action* to achieve a specific outcome or results based on a set of conclusions.
- **Standard Deviation** "The square root of the variance. It is a measure of spread of data points about the mean." (FAA SEM, 2006, Vol. 3, p. 4.1-13)
- **Suboptimization** The preferential emphasis on the performance of a lower level entity at the expense of overall system performance (Principle 14.3).
- **System Optimization** The act of balancing the contributory performance of individual SYSTEM Elements to achieve a maximum level of integrated SYSTEM performance for a given set of boundary conditions and constraints. *Optimization* decision factors may include combinations of factors such as technical performance, cost, or technology.
- **System Performance Analysis and Evaluation** The investigation, study, and operational analysis of actual or predicted SYSTEM performance relative to planned or required performance as documented in System Performance Specification (SPS) or Entity Development Specifications (EDS).
- **Variance (Statistical)** "A measure of the degree of spread among a set of values; a measure of the tendency of individual values to vary from the mean value. It is computed by subtracting the mean value from each value, squaring each of these differences, summing these results, and dividing this sum by the number of values in order to obtain the arithmetic mean of these squares." (DAU, 2012, p. B-238)

30.2 WHAT IS ANALYTICAL DECISION SUPPORT?

Decision support is a technical services response to a contract or task commitment to gather, analyze, clarify, investigate, recommend, and present objective evidence of fact-based evaluations and work products, findings, conclusions, and recommendations. This enables decision makers to select a proper (best) course of action from a set of viable alternatives bounded by specific constraints such as cost, schedule, technical, technology, support, and *acceptable level of risk*.

30.2.1 Analytical Decision Support Objective

The primary objective of analytical decision support is to respond to tasking or the need for technical analysis, demonstration, and data collection recommendations to support informed SE Process Model (Figure 14.1) decision-making.

30.2.2 Expected Outcome of Analytical Decision Support

Decision support results should be documented as work products identified in task objectives. *Work products* and Quality Records (QRs) include analyses, Trade Study Reports (TSRs), and performance data. In support of these work products, decision support develops operational prototypes and Proof of Concept, Proof of Technology, and Proof of Principle demonstrations, models and simulations, and mock-ups to provide data for supporting the analysis.

From a technical decision-making perspective, decisions are substantiated by the facts of the formal work products such as analyses and TSRs provided to the decision maker. The reality is that the decision may have subconsciously been made by the decision maker long before the delivery of the formal work products for approval. This brings us to our next topic, Attributes of a Technical Decisions.

30.3 ATTRIBUTES OF TECHNICAL DECISIONS

Every decision has several attributes you need to understand to be able to properly respond to the task. The attributes you should understand are the following:

- What is the central issue or problem to be addressed?
- What is the scope of the task to be performed?
- What are the *boundary constraints* for the solution set and *degree of flexibility*?
- Is the timing of the decision crucial?
- What *decision factors and criteria* are to be used in making the decision?
- What level of *accuracy* and precision is required for the decision?
- How is the decision to be documented and delivered?

30.3.1 Problem or Issue to be Resolved

Decisions represent approval of solutions intended to lead to *actionable* tasks that will resolve a Critical Operational or Technical Issue (COI/CTI). The System Analyst begins with *understanding* what problem, issue, or question the User is trying to solve. Therefore, begin with a *clear, concise,* and *succinct* Problem/Opportunity Statement (Chapter 4) from the decision maker or one they approve as characterizing the problem/opportunity.

If you are tasked to solve a technical problem and are not provided a documented tasking statement, discuss it with the decision authority. *Active listening*—verbal feedback to the source—enables analysts to *verify* their understanding of the tasking and *validate* the problem to be solved. Add corrections based on the discussion and return a courtesy copy to the decision maker. Then, when briefing the status of the task, always include a restatement of the task, so all reviewers have a clear understanding of the analysis you were tasked to perform.

30.3.2 Scope of the Task to be Performed and Success Criteria

Problem solving requires having a reasonable amount of time and resources such as expertise, technology, cost, and schedule available to successfully complete the task. Therefore, properly scope the task to be performed including outcome-based objectives and performance. Document the task and these attributes including decision maker signature of agreement *before* starting the task. Recognize and appreciate the difference between *problem solving* versus *symptom solving* (Principle 4.13).

30.3.3 Document the Decision-Making Completion and Success Criteria

Once the Problem Statement is documented and boundary constraints for the decision are established, identify the Decision Factors (Chapter 32) that will be used to assess the success of the decision results. Obtain Stakeholder concurrence with the Decision Factors.

Make corrections as necessary to provide scoping definitions of Decision Factors to *avoid misinterpretation* when the decision is presented for approval. If the Decision Factors are *undocumented* by the decision maker "up front," you may be subjected to the changing whims of the decision maker to determine "when" or "if" the task is complete.

Some analytical results may be impractical to achieve or less than optimal. Therefore, establish what constitutes success to avoid being labeled as having failed to achieve results.

30.3.4 Decision Boundary Condition Constraints, Assumptions, and Flexibility

Technical decisions are bounded by cost, schedule, technical, technology, and support constraints. Often, certain *assumptions* are required that should be documented as *constraints*. In turn, the *constraints* must be reconciled with an *acceptable* level of risk. Constraints sometimes are also flexible. Talk with the decision maker and assess the amount of flexibility in the constraint. Document the constraints and acceptable level of risk as part of the task statement.

30.3.5 Criticality of Decision Timing

Author's Note 30.1

Reminder of Principle 17.1 Task Expectation Principle

On a professional level, a 1-hour analysis represents high-level findings with limited detail. An 8-hour analysis should produce similar results with more supporting detail.

Just-in-Time (JIT) delivery of analytical decisions is crucial, especially from the perspective of the decision maker, as well as their time to read or listen to a Trade Study Report (TSR) presentation. Be *sensitive* to the decision authority's schedule when the recommendations are presented.

30.3.6 Understand How the Decision Will Be Used and by Whom

Decisions often require approvals by multiple levels of Enterprise and customer stakeholder decision makers. *Avoid* wasted effort trying to solve a *symptom space* rather than the actual *problem space* (Principle 4.13). Tactfully *validate* the decision problem statement.

30.3.7 Identify the Accuracy and Precision of the Analysis

Principle 30.1

Data Precision Principle

Data with two-digit precision that require multiplication *do not* yield four-digits of precision; the best you can achieve is the source's two-digits of precision.

Every technical decision involves data that have a level of *accuracy* and *precision*. Determine "up front" what *accuracy* and *precision* will be required to support analytical results and make sure that these are clearly communicated and understood by everyone participating. One of the worst things analysts can do is discover "after the fact" that they need four-digit decimal data precision when they only measured and recorded two-digit data. Some data collection exercises may not be repeatable or practical. Apply *System Thinking* (Chapter 1) and *Plan Ahead*. Similar rules should be established for rounding data digits.

30.3.8 Identify How the Decision Is to Be Delivered

Decisions need a point of closure or delivery. Identify in what format and media the decision is to be delivered—as a document, presentation, hardcopy or electronics, color or Black & White (B&W). In any case, make sure that your response is documented for the record via a cover letter or e-mail.

30.4 TYPES OF ENGINEERING ANALYSES

Engineering analyses cover a spectrum of disciplinary and specialty skills. The challenge for SEs is to understand the following:

- What types of analyses may be required?
- At what level of detail?
- What tools are best suited for various analytical applications?
- What level of formality is required for documenting the results?

To illustrate a few of the many analyses that might be conducted, here's an example list. Analyses marked with an "*" are described in the INCOSE (2011) *Systems Engineering Handbook* pp. 314–331.

- Mission Operations and Task Analysis
- Interoperability Analysis
- Usability Analysis*
- Human Systems Integration Analysis*
- Environmental Impact Analysis (EIA)*
- Fault Tree Analysis (FTA)
- Finite Element Analysis (FEA)
- Mass Properties Engineering (MPE) Analysis*
- Stress Analysis
- Electromagnetic Interference (EMI) Analysis
- Electromagnetic Compatibility (EMC) Analysis*
- Optical Analysis
- Thermal Analysis
- Timing Analysis
- System Latency Analysis
- Failure Modes and Effects Analysis (FMEA)
- Failure Modes and Effects Criticality Analysis (FMECA)*
- Level of Repair (LoR) Analysis*
- Logistic Support Analysis (LSA)*
- Reliability, Availability, and Maintainability (RAM) Analysis
- Reliability-Centered Maintenance (RCM) Analysis *
- System Safety Analysis*
- System Hazard Analysis*
- Vulnerability Analysis
- System Security Analysis*
- Survivability Analysis*
- Sustainment Engineering Analysis*
- Training Needs Analysis*
- Life Cycle Cost Analysis*
- Cost Effectiveness Analysis
- Manufacturing and Producibility Analysis*

Heading 30.1 The application of various types of Engineering analyses should focus on providing objective, fact-based data that support informed technical decision-making. These results at all levels aggregate into overall system performance that forms the basis of our next topic, System Performance Analysis and Evaluation.

30.5 SYSTEM PERFORMANCE ANALYSIS AND EVALUATION

System performance analysis and evaluation is the investigation, study, and operational analysis of actual or predicted system performance relative to planned or required performance as documented in SPS or EDS. The analysis process requires the planning, methodology, data collection, and post-data analysis to thoroughly understand a SYSTEM's performance.

30.5.1 System Performance Analysis Tools and Methods

System performance analysis and evaluation employs a number of decision tools and methods to collect data to support the analysis. These include models, simulations, prototypes, interviews, surveys, and test markets.

30.5.2 Optimizing System Performance

SYSTEM components at every *level of abstraction* inherently have statistical variations such as physical characteristics, reliability, performance, manufacturing processes, or workmanship. Systems that involve humans involve statistical variability in knowledge and skill levels and thus involve an element of uncertainty. The challenge question for SEs to consider is: *what combination of system configurations, conditions,* PERSONNEL-EQUIPMENT *tasks (Figure 24.14), and associated levels of performance optimizes overall system performance?*

SYSTEM *optimization* is subject to the Stakeholder Observer's Frame of Reference. *Optimization* decision factors and criteria (Figure 14.8) should reflect a consensus of the Stakeholder community based on a balance of cost, schedule, technical, technology, and support performance, or combinations thereof.

30.5.3 Suboptimization

Suboptimization is a condition that exists when one element of a SYSTEM at the PRODUCT, SUBSYSTEM, ASSEMBLY, SUBASSEMBLY, or PART Levels is optimized at the expense of overall system performance. During System Integration, Test, and Evaluation (SITE), SYSTEM entities at each *level of abstraction* may be optimized. Theoretically, if the entity is designed *correctly*, *optimal* performance occurs at the

planned nominal or midpoint of any adjustment ranges subject to variations in material properties or performance.

The underlying design philosophy here is that if the SYSTEM is properly designed and component statistical variations are accounted for in the analysis, only *minor* adjustments may be required for an output to be *centered* about some hypothetical nominal or mean value (Figure 30.1). If the variations have not been taken into account or design modifications have been made, the output may be "off-set" from the mean value at its midrange setting but within its operating range when "optimized." Thus, at higher levels of integration, this *off-nominal* condition may impact overall system performance, especially if further control beyond the component's adjustment range is required. Bottom line: select components with the required performance and tolerances. Then, screen the devices during the Enterprise's Receiving Inspection Process.

30.5.4 The Danger of Analysis Paralysis

Analyses serve as a powerful tool for understanding, predicting, and communicating system performance estimates. Analyses, however, cost money and consume valuable resources. The challenge question for SEs to consider is: *How "good is good enough"*? At what level or point in time does an analysis meet *minimal sufficiency criteria* to be considered valid for decision-making? Since Engineers, by nature, tend to become enamored with the elegance of analysis, we sometimes suffer from a condition referred to as "analysis paralysis." So, *what is analysis paralysis*?

Analysis paralysis is a condition in which an analyst becomes *preoccupied or immersed* in the details of an analysis while failing to recognize the marginal utility and diminishing returns of continual investigation. So, *how do SEs deal with this condition*?

- Firstly, you must learn to recognize the signs of this condition in yourself as well as others. Although the condition varies from one person to another, some are more prone than others.
- Secondly, aside from personality characteristics, the condition may be a response mechanism to the work environment, especially from paranoid, overbearing managers who micro-manage tasks and who suffer from the condition themselves.

30.5.5 Engineering Analysis Reports

As a discipline requiring *integrity* in analytical, mathematical, and scientific data and computations to support downstream or lower level decision making, Engineering documentation is sometimes poor at best or simply nonexistent, due to either a lack of discipline or impractical work environments. One of the hallmarks of a professional discipline is an expectation to document recommendations and decisions supported by factual, objective evidence derived empirically by observation, lessons learned, or best practices.

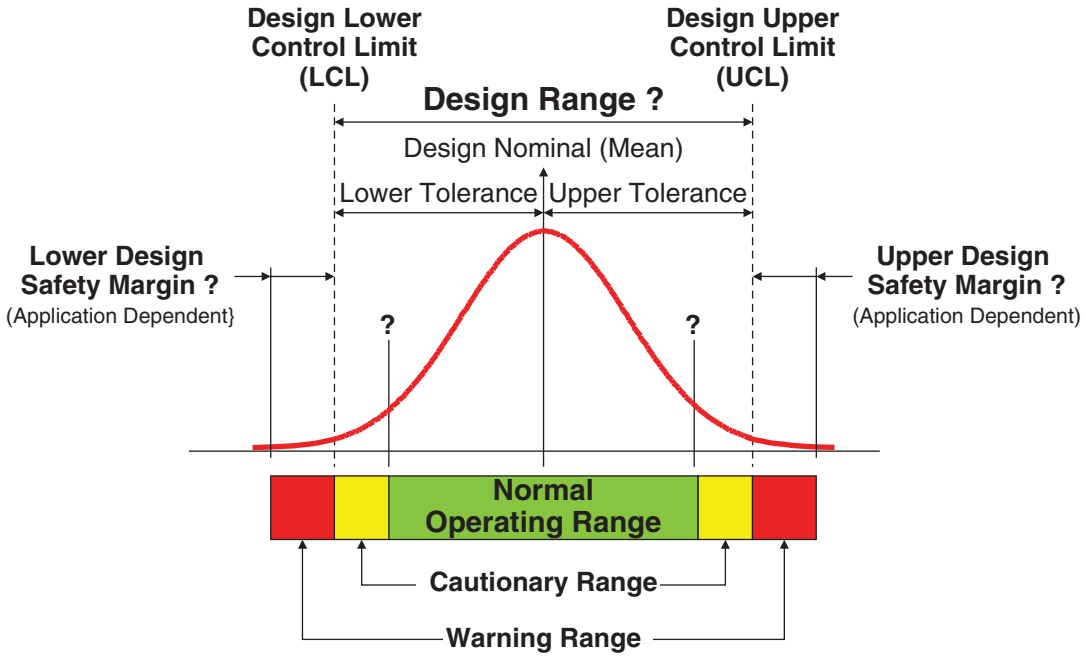


Figure 30.1 Application Dependent Gaussian (Normal) Distribution Illustrating the Concept of Acceptable Operating Control Limits

Data that contribute to *informed* SE decisions are often dependent on the assumptions, boundary conditions, and constraints surrounding the data collection. While most Engineers competently consider relevant factors affecting a decision, the tendency is to *avoid* documenting the results; they view paperwork as *unnecessary, bureaucratic* documentation that does not add value directly to the deliverable product. As a result, a professional, high-value analysis ends in *mediocrity* due to the analyst lacking personal initiative and discipline to perform the task correctly.

To better appreciate the professional discipline required to document analyses properly, consider a hypothetical visit to a physician:

Example 30.1

You visit a physician for a condition that requires several treatment appointments at 3-month intervals over a year. The physician performs a high-value diagnosis and prescribes treatments and/or prescription medication but fails to record the medication and actions performed at each treatment event. At each subsequent visit, you and the physician have to *reconstruct* to the best of everyone's knowledge the assumptions, prescriptions and dosages, and actions performed. Aside from the medical and legal implications, can you imagine the frustration, foggy memories, and "guesstimates" associated with these interactions. Engineering, as a professional discipline, is no different. Subsequent decision-making is highly dependent and builds on the documented facts, conditions, assumptions, and constraints of previous decisions.

The difference between *mediocrity* and *high-quality* professional results may be only a few minutes to simply document critical considerations that yielded the analytical result and recommendations presented. For SEs, decision QRs should be recorded in a personal Engineering notebook preferably on-line in a network-based journal which is preferable.

30.5.5.1 Engineering Report Format Where practical and appropriate, Engineering analyses should be documented in formal technical reports. Contract or Enterprise command media -policies and procedures - sometimes specify the format of these reports. If you are expected to formally report the results of an analysis and do not have specific format requirements, consider the example outline shown in Table 30.1. Here is a brief overview of key sections:

Section 1.0. INTRODUCTION

The introduction establishes the context and basis for the analysis. Opening statements identify the document; its context, relevance, and usage in the project; and references to decision-maker tasking that authorized the analysis.

TABLE 30.1 Example Engineering Report Outline

Section 1.0	**INTRODUCTION**
1.1	Scope
1.2	Charter including Problem Statement
1.3	Objectives
1.4	Analyst/Team Members
1.5	Acronyms and Abbreviations (Optional)
1.6	Definitions of Key Terms (Optional)
Section 2.0	**REFERENCED DOCUMENTS**
2.1	System Acquirer/User Documents
2.2	Project Documents
2.3	Vendor Documents
2.4	Standards and Specifications
Section 3.0	**EXECUTIVE SUMMARY**
3.1	Summary of Findings
3.2	Summary of Observations
3.3	Summary of Conclusions
3.4	Summary of Recommendations
Section 4.0	**METHODOLOGY**
4.1	Background
4.2	Assumptions and Constraints
4.3	Methodology
4.4	Data Collection
4.5	Analytical Tools and Methods—Versions and Configurations
4.6	Statistical Analysis (if applicable)
4.7	Analysis Results
Section 5.0	**FINDINGS, OBSERVATIONS, AND CONCLUSIONS**
5.1	Findings
5.2	Observations
5.3	Conclusions
5.4	Dissenting Opinions
Section 6.0	**RECOMMENDATIONS**
6.1	Priority #1
6.2	Priority #2
…	
6.n	Priority #n
APPENDICES	
A	Vendor Sources
B	Supporting Analytics and Data

Author's Note 30.2

Some Enterprises and individuals prefer to place the Acronyms, Abbreviations, and Definitions in an appendix. Some prefer placement of Definitions "up front" to introduce the reader to terms that have significant or contextual meanings in the sections that follow. Either approach is fine.

Section 2.0. REFERENCED DOCUMENTS

This section lists only the documents cited in other sections of the document (Principle 23.3). Note the

operative title "Referenced Documents" as opposed to "Applicable Documents." We will address this topic later.

Section 3.0. EXECUTIVE SUMMARY

Summarize the results of the analysis such as findings, observations, conclusions, and recommendations: tell them the bottom line "up front." Then, if the reader chooses to read about the details of *how* you arrived at those results, they can do so in subsequent sections.

Section 4.0. METHODOLOGY

Informed decision-making is heavily dependent on a valid methodology for conducting the research and analysis. As such, the methodology used to perform the analysis must be established as a means of providing credibility for the results.

Section 5.0. FINDINGS, OBSERVATIONS, AND CONCLUSIONS

As with any scientific study, it is important for the analyst to communicate: (1) what they found, (2) what they observed, and (3) what conclusions they derived from the findings and observations.

Section 6.0. RECOMMENDATIONS

On the basis of the analyst's findings, observations, and conclusions, Section 6.0 provides a set of prioritized recommendations to decision makers concerning Table 30.1 Section 1.3 objectives. You may ask: *why document recommendations … the results are what they are*? Decision makers require inputs to make informed decisions. They do not have time to interpret results. After all, the System Analyst is supposed to be the expert. That is *why* they were tasked to perform the analysis. The decision maker expects you to summarize insightful nuggets of technical wisdom and propose a prioritized list of recommendations from which the decision maker can make an informed decision that may involve other considerations such as cost, risk.

APPENDICES

Appendices provide areas to exhibit supporting documentation collected during the analysis or support the author(s) findings, conclusions, and recommendations. The *credibility* and *integrity* of an analysis often depend on *who* collected and analyzed the data. Analysis report appendices provide a means of organizing and preserving any supporting vendor, test, simulation, or other data used by the analyst(s) to support the results. This is particularly important if, at a later date, *conditions* that served as the basis for the initial analysis task change, thereby creating a need to revisit the original analysis. Because of the changing conditions, some data may have to be regenerated; some may not. For those data that have not changed, the appendices *minimize* work on the new task analysis by *avoiding* the need to *recollect* or *regenerate* the data.

30.5.5.2 Decision Documentation Formality

There are numerous ways to address the need to *balance* document decision making with time, resource, and formality constraints. Approaches to documenting critical decisions formalities range from a single page of *informal,* handwritten notes to *highly* formal documents. Establish disciplinary standards for yourself and your Enterprise related to documenting decisions. Then, scale the documentation formality according to task constraints. Regardless of the approach used, the documentation should capture the key attributes of a decision in sufficient detail to enable "downstream" understanding of the *factors* and criteria that resulted in the decision.

Author's Note 30.3

The reality is that we do not always have the time to formally document a decision or results even if we exhibit personal initiative and professional discipline. When time does not permit, document the results on the "back of napkin" with a date and key factors and file in a folder until you have time to properly document the results. At least the decision is documented, even informally.

30.5.6 Analysis Lessons Learned

Once the performance analysis tasking and boundary conditions are established, the next step is to conduct the analysis. Let's explore some lessons learned that you should consider in preparing to conduct the analysis.

30.5.6.1 Establish a Decision Development Methodology

Decision paths tend to veer off-course midway through the decision development process. Establish a valid decision-making methodology "up front" to serve as a roadmap for keeping the effort on track. When you establish the methodology "up front," you have the visibility of *clear, unbiased* thinking *unencumbered* by the adventures along the decision path. If you and your team are convinced you have a sound, proven methodology, that plan will serve as a compass heading. This is not to say that some conditions may warrant a change in methodology. *Avoid changes unless there is a compelling reason with justifiable rationale to change*.

Technical and scientific methodologies should always be professionally and ethically driven by an objective evaluation *unmodified* to fit political decisions. Remember, when the decision is later *proven faulty* or results in *system failures*, those same politicians who influenced the original methodology and outcome will distance themselves and not hesitate to chastise you for (1) "not following proven, best practices" and (2) taking appropriate professional and disciplinary action.

30.5.6.2 Acquire Analysis Resources As with any task, success is partially driven by simply having the *right* resources and LOE available when they are required. This includes:

- Subject Matter Experts (SMEs).
- Analytical tools.
- Access to personnel who may have relevant information concerning the analysis area.

30.5.6.3 Document Assumptions and Caveats Every decision involves some level of *assumptions* and/or *caveats*. Document the assumptions in a clear, concise manner. Verify that the *caveats* are properly documented on the same page as the decision including assumptions, data sources, and footnotes. If the decision recommendations are copied, the caveats will always be on the page. Otherwise, people may *intentionally or unintentionally* apply the decision or recommendations out of context.

30.5.6.4 Date the Decision Documentation Every decision document page header should be marked indicating the document title, revision level, date, page number, and classification level, if applicable. Using this approach, the reader can always determine if the version they possess is current. In addition, if a single page is copied, the source document is readily identifiable. Most people fail to perform this simple task. When multiple versions of a report or draft are distributed without dates, the *de facto* version, which may or may not be correct, is determined by the document closest to the top of a stack on someone's desk.

30.5.6.5 State the Facts as Objective Evidence Technical reports must be based on the most current, factual information from *credible* and *reliable* sources. Conjecture, hearsay, and personal opinions should be *avoided*.

30.5.6.6 Cite Only Credible and Reliable Sources Technical decisions often leverage and expand on existing knowledge and research, published or verbal. If you use this information to support findings and conclusions, *explicitly* cite the source(s) with attribution. *Avoid* vague references such as "read the [author's] report" documented in an obscure publication published 10 years ago that may be inaccessible or only available to the author(s). If these sources are *unavailable,* contact the owner or publisher for permission to quote.

30.5.6.7 Reference Documents versus Applicable Documents Due to a lack of proper education and training, experiential Engineering education builds on mimicking what others have done in the past. Unfortunately, Engineers create documents with a Section 2.0 titled "Applicable Documents" rather than what they actually are—"Referenced Documents." Applicable documents are simply one person's perspective of the universe of material on the topic that may or may not be relevant. Recognize the difference (Principle 23.3)!.

30.5.6.8 Cite Referenced Documents When citing referenced documents, include the document ID, title, source, date, and version containing data that serve as inputs to the decision. People often believe that if they reference a document by title they have satisfied analysis criteria. Technical decision-making is only as good as the *credibility and integrity* of its sources of objective, fact-based information. Source documents may be revised over time. Do yourself and your team a favor: make sure that you *clearly* and *concisely* document the critical attributes of source documentation.

Today, the tendency is to cite Internet documents that are often *uncontrolled* and *may or may not* be valid sources prepared by qualified experts. If you feel compelled to quote an Internet website, corroborate the information from several reliable sources and include the following information:

- Website URL—for example, www.XXXX.com
- Author
- "Date Retrieved."

30.5.6.9 Conduct SME Peer Reviews Technical decisions are sometimes Dead-on-Arrival (DOA) due to poor assumptions, flawed decision criteria, methodology, and incomplete research. Plan for success by conducting an informal peer review with trusted and qualified colleagues—the SMEs—of the evolving decision document. Listen to their *challenges* and *concerns*. *Are they highlighting Critical Operational and Technical Issues (COIs/CTIs) that remain to be resolved or overlooked variables and solutions that are obscured by the analysis or research*?

30.5.6.10 Prepare Findings, Conclusions, and Recommendations

Principle 30.2

Analysis Validity Principle

Analysis results are only as valid as their underlying assumptions, models, and methodology. Validate and preserve their integrity.

There are a number of reasons *why* an analysis is conducted. The technical decision maker may not possess current technical expertise or the ability to *internalize* and *assimilate* data for a complex problem. So, they seek out those who do posses this capability such as competent, qualified consultants, or Enterprises. In general, the decision maker wants to know what the recognized SMEs who are closest to the problems, issues, and technology propose as recommendations regarding the decision. Therefore, analyses should include findings, conclusions, and recommendations.

On the basis of the results of the analysis, the decision maker can choose to:

- Ponder, place "on hold," return for more analysis, or reject the findings and conclusions from their own perspective.
- Accept the recommendations as a means of arriving at an informed decision.

In any case, they need to know what the SMEs have to offer regarding the decision.

30.6 STATISTICAL INFLUENCES ON SYSTEM DESIGN

For many Engineers, System Design evolves around abstract phrases such as "bound environmental data" and "receive data." The challenge is: *how do you quantify and bound the conditions for a specific parameter? How does an SE determine performance measures such as:*

- Acceptable signal and noise (S/N) ratios?
- Computational errors in processing the data?
- Time variations required to process system data?

The reality is that the hypothetical boundary condition problems Engineers studied in college aren't so ideal (Figure 2.4). In addition, when a SYSTEM or PRODUCT is developed, multiple copies may produce varying degrees of responses to a set of controlled inputs. *So, how do SEs deal with the challenges of these uncertainties*?

SYSTEMS and PRODUCTS have varying degrees of stability, performance, and uncertainty that are influenced by their unique *form, fit, and function* performance characteristics. Depending on the price the User *wants, needs, can afford, and is willing to pay* (Figure 21.4), we can improve and match material characteristics and processes used to produce SYSTEMS and PRODUCTS. If we analyze a SYSTEM's or PRODUCT's performance characteristics over a controlled range of inputs and conditions, we can *statistically* state the variance in terms of its standard deviation.

This section provides an introductory overview of how statistical methods can be applied to system design to improve capability performance. As a prerequisite to this discussion, you should have basic familiarity with statistical methods and their applications.

30.6.1 Understanding the Variability of Engineering Data

In an *ideal*, theoretical world, Engineering data *identically match* predicted values with *zero error margins*. In the *real* world, however, variations in mass properties and characteristics; attenuation, propagation, and transmission delays; and human responses are among the *uncertainties* that must be accounted for in engineering analysis calculations. In general, the data are *dispersed* about the *mean* of the *frequency distribution*.

30.6.1.1 Normal and Logarithmic PDFs

Statistically, we characterize the range dispersions about a central mean in terms of Normal (Gaussian) and Logarithmic (Poisson) frequency distributions as shown in Figure 34.3.

Normal and logarithmic frequency distributions can be used to mathematically characterize and bound Engineering performance data related to Statistical Process Control (SPC); queuing or waiting line theory for customer service and message traffic; production lines; reliability and maintainability data, pressure containment; temperature/humidity ranges.

30.6.1.2 Applying Statistical Distributions to Systems

Chapter 3 characterized a system as having *acceptable* and *unacceptable* inputs and producing *acceptable* outputs (Figure 3.2). A SYSTEM or PRODUCT can also produce *unacceptable* outputs such as electromagnetic, optical, chemical, thermal, or mechanical outputs that make the system vulnerable to adversaries or create self-induced feedback that diminishes or degrades system performance. The challenge for SEs is bounding:

- The range of *acceptable inputs* and conditions from *undesirable or unacceptable* inputs.
- The range of *acceptable outputs* and conditions from unacceptable outputs.

30.6.1.2.1 Design Input/Output (I/O) Range Acceptability

Statistically, we can bound and characterize the range of *acceptable* inputs and outputs using the frequency distributions. Figure 30.1 illustrates an example of a Gaussian (Normal) Distribution that we can employ to characterize I/O variability.

In this illustration, we employ a Gaussian (Normal) Distribution with a central mean. Depending on the boundary conditions imposed by the SYSTEM application, SEs determine the *acceptable* design range that includes *upper* and *lower* control (UCL/LCL) limits relative to the mean.

30.6.1.2.2 Range of Acceptable System Performance

During normal system operations, SYSTEM or PRODUCT capabilities perform within an *acceptable* (Normal) Design Range. The challenge for SEs is determining what the thresholds are for alerting User operators and maintainers when SYSTEM performance is *off-nominal* and begins to pose a risk or threat. To better understand this point, refer to Figure 30.1.

Observe that we have a Normal Distribution about a *central mean* and characterized by four types of operating ranges:

- **Design Range** The acceptable range of Engineering parameter values for a specific capability and conditions that bound the *acceptable* and *unacceptable* Upper/Lower Control Limits (UCL/LCL) introduced in Figure 30.1.
- **Normal Operating Range** The range of *acceptable* Engineering parameter values for a specific capability that clearly indicates nominal performance for a given set of conditions that does not necessarily pose a risk or threat to the PERSONNEL, EQUIPMENT, general public, or the environment.
- **Cautionary Range** The range of Engineering parameter values for a specific capability that clearly indicates reasonable performance measures beyond or outside the Normal Operating Range that have the potential to injure or harm MISSION SYSTEM or ENABLING System PERSONNEL, EQUIPMENT, the general public, or environment.
- **Warning Range** The range of Engineering parameter values for a specific capability and conditions that clearly poses a *clear* and *present danger* to the PERSONNEL, EQUIPMENT, the general public, or environment with catastrophic consequences.

This presents several decision-making challenges for SEs as illustrated in Figure 30.1:

- What is the *acceptable* Design Range that includes Upper and Lower Caution Ranges?
- What are the Upper and Lower Control Limits and conditions of the *acceptable* Normal Operating Range?
- What are the threshold conditions for the Warning Range?
- What Upper and Lower Design Safety Margins and conditions must be established for the SYSTEM relative to the Caution Range and Warning Range?

These questions, which are application dependent, are typically difficult to answer. Also, recognize that this graphic reflects a single Measure of Performance (MOP) for one SYSTEM or ENTITY at a specific level of abstraction. The significance of this decision is exacerbated by the need to allocate the Design Range to lower level entities, which also have comparable performance distributions, ranges, and safety margins. Obviously, this poses a number of risks. For large, complex systems, *how do we deal with this challenge*?

There are several approaches for supporting the design thresholds and conditions.

- First, you can Model and Simulate (M&S) the SYSTEM and employ Monte Carlo techniques to assess the *most likely* or *probable* outcomes for a given set of Use Cases (UCs) and scenarios.
- Second, you can leverage *validated* M&S results and develop a prototype of the SYSTEM for further analysis and evaluation.
- Third, you can employ Spiral Development (Figure 15.4) to evolve a set of requirements over a set of sequential prototypes.

Given this overview, let's shift our focus to understanding how statistical methods apply to System Engineering Development (SE&D).

30.6.2 Statistical Method Applications to System Development

Statistical methods are employed throughout the System Development Phase (Figure 12.2) by various disciplines. For SEs, statistical challenges occur in two key areas:

- Bounding and specifying specification requirements performance limits.
- Verifying specification requirements compliance.

30.6.2.1 Statistical Challenges in Writing Specification Requirements During specification requirements development, Acquirer SEs are challenged to specify the *acceptable and unacceptable* ranges of inputs and outputs (Figure 3.2) for performance-based specifications (Figure 20.4). Consider the following example:

Example 30.2

When exposed to (specified environmental operating conditions), the Sensor System **shall** have a probability of detection of 0.XX over a range of (minimum performance limit) to (maximum performance limit).

When the contract is awarded, System Developer SEs are challenged to determine, allocate, and flow down system performance budgets and safety margins requirements (Chapter 31) derived from higher-level requirements. The challenge is analyzing Example 30.2 to derive requirements for PRODUCT, SUBSYSTEM, ASSEMBLY, and other levels. Consider Example 30.3 below.

Example 30.3

The XYZ output shall have a $\pm 3\sigma$ worst-case error of 0.XX vdc for Input Parameter A distributions that vary between 0.000 vdc to 10.000 vdc.

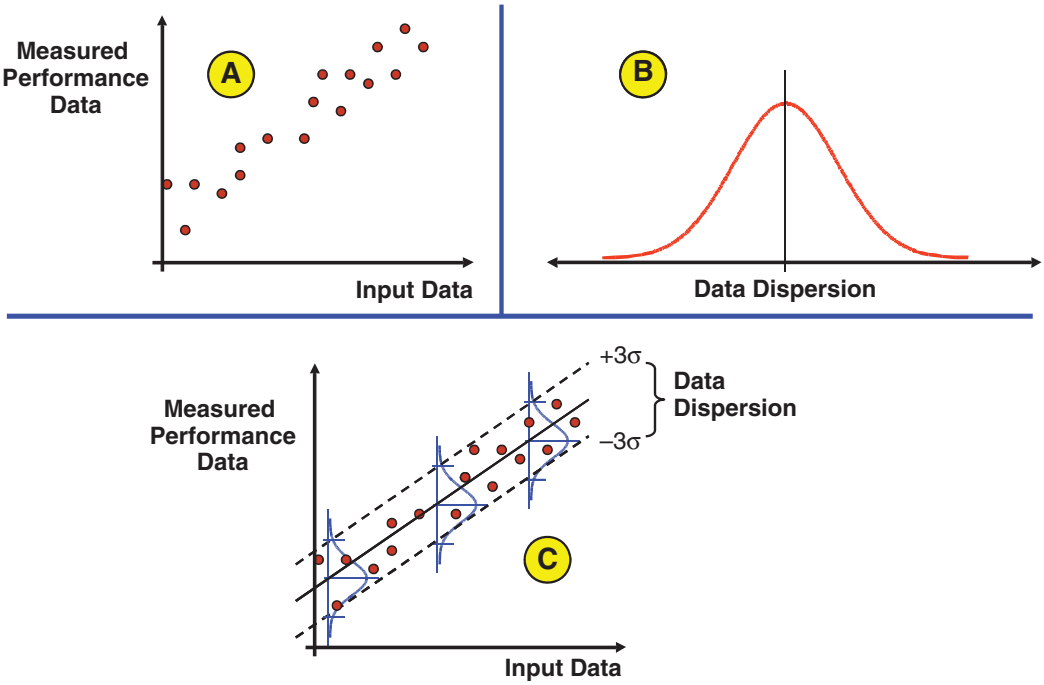


Figure 30.2 Engineering Data Dispersion Concept

30.6.2.2 Statistical Challenges in Verifying Specification Requirements Suppose that an SE's mission is to verify the requirement stated in Example 30.3. For simplicity, let's assume that the sampled *end points* for Input Parameter A data are 0.000 vdc and +10.000 vdc with a couple of points in between. We collect data measurements as a function of Input Data and plot them. Panel A of Figure 30.2 might be representative of this data.

Applying statistical methods, we determine the central mean *trend line* and $\pm 3\sigma$ boundary conditions based on the requirement. Next, we superimpose the central mean trend line and $\pm 3\sigma$ boundary limits to verify that the dispersion of system performance data is within the *specified* requirement control limits indicating the SYSTEM *passed* (Panel C).

Let's explore how SEs and System Analysts apply statistical test data to SE design decision making or verifying requirements specified in an SPS or EDS have been achieved.

30.6.3 Understanding Test Data Dispersion

Suppose that we conduct a test to measure SYSTEM or ENTITY performance over a range of input data as shown in Panel A of Figure 30.2. As illustrated, we have a number of data points that have a *positive* slope. This graphic has two important aspects:

- Upward sloping central mean trend line of data
- A dispersion of data along the trend line.

In this example, if we performed a Least Squares mathematical fit of the data, we could establish the slope and intercepts of the trend line using a simple $y = mx + b$ construct.

Using the *central mean* trend line for the data set as a function of Input Data (X-axis), we find that the corresponding Y data points are dispersed about the mean as illustrated in Panel C. On the basis of the standard deviation of the data set, we could say that there is 0.9973 probability that a given data point lies within the $\pm 3\sigma$ boundaries about the mean. Thus, Panel D depicts the results of projecting the $\pm 3\sigma$ lines along the trend line.

30.6.4 Cumulative System Performance Effects

Our discussions to this point focus on statistical distributions relative to a specific capability parameter. The question is: *how do these errors propagate throughout the system?* There are several factors that contribute to the error propagation:

1. OPERATING ENVIRONMENT influences on PART-Level properties.
2. Timing variations.
3. Computational precision and accuracy.
4. Drift or *aliasing* errors as a function of time.

From a total SYSTEM perspective, we refer to this concept as *Cumulative System Performance Error*. Figure 30.3 provides an example.

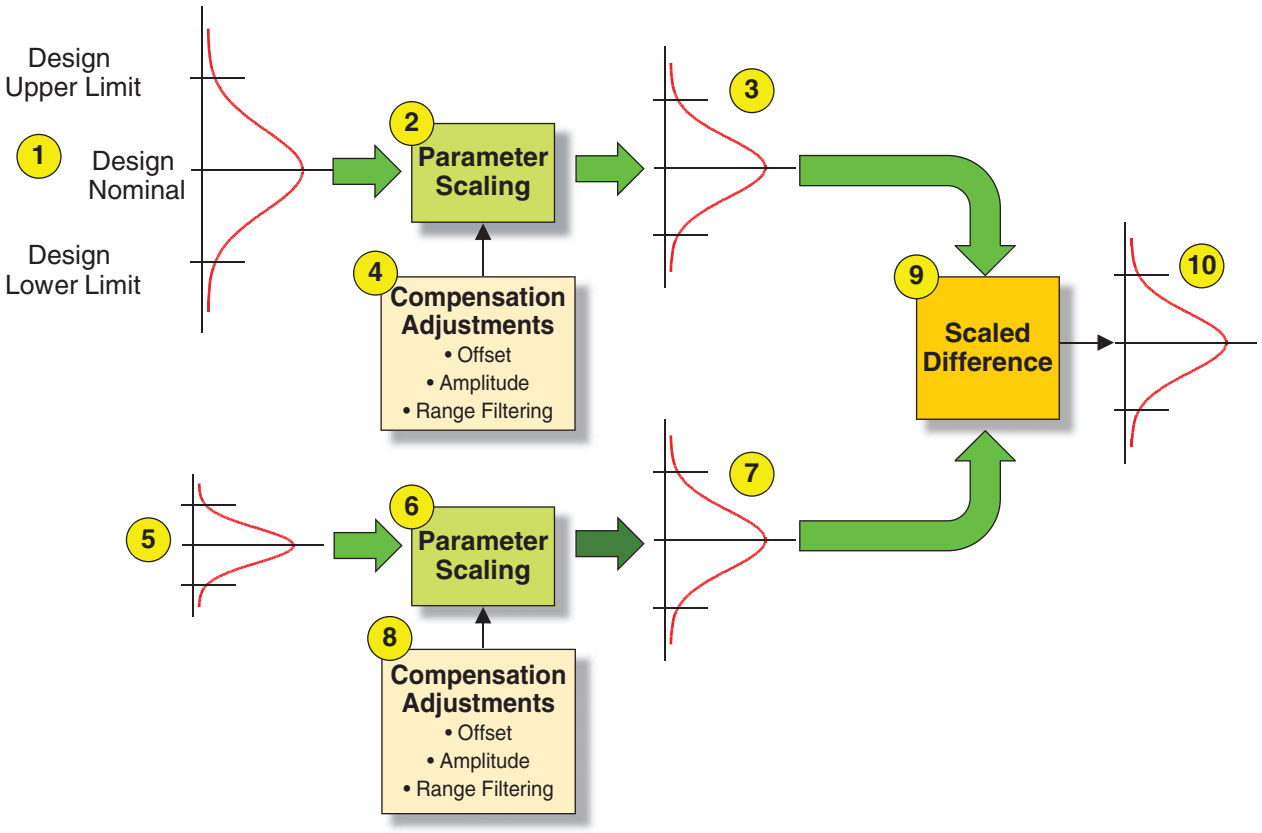


Figure 30.3 Understanding Cumulative Error Statistics in SYSTEM or PRODUCT Performance

Assume that we have a simple SYSTEM that computes the difference between two Parameters A and B. If we examine the characteristics of Parameters A and B, we find that each parameter has different data dispersions about a predicted performance mean and tolerance limits.

Ultimately, if we intend to compute the difference between Parameter A and Parameter B, both parameters have to be *scaled* relative to some normalized value. Otherwise, we get an "apples and oranges" comparison. So, we scale each input and make any correctional offset adjustments. This simply solves the functional aspect of the computation. Now, *what about errors originating from the source values about a nominal mean plus all intervening scaling operations?* The answer is: SEs have to account for the cumulative error distributions related to errors and dispersions. Once the SYSTEM is developed, integrated, and tested, SYSTEM-Level *optimization* (Figure 14.8) may require adjustment in SUBSYSTEM, ASSEMBLY, AND SUBASSEMBLY performance to correct for cumulative errors and dispersions.

30.6.5 Circular Error Probability (CEP)

The preceding discussion focused on analyzing and allocating system performance within the system. The ultimate test for SE decision-making comes from the actual field results. The question is: how do cumulative error probabilities impact overall operational and system effectiveness? Perhaps, the best way to answer this question is a "bull's-eye target" analogy.

Our discussions up to this point have focused on the *dispersion* of data along slope of the central mean. There are system applications whereby data are dispersed about a central point such as the "bull's eye" illustrated in Figure 30.4. In these cases, the $\pm 1\sigma$, $\pm 2\sigma$, and $\pm 3\sigma$ points lie on *concentric* circles aligned about a *central mean* located at the bull's eye. Applications of this type are generally target based such as munitions, firearms, and financial plans. Consider Example 30.4:

Example 30.4

Let's assume you are tasked to conduct an evaluation of two competing rifle systems, Rifle System A and Rifle System B. We will assume that sampling methods are employed to determine a statistically valid sample size. Specification requirements state that 95% of the shots must be contained within a circle with a diameter of X inches centered at the bull's eye.

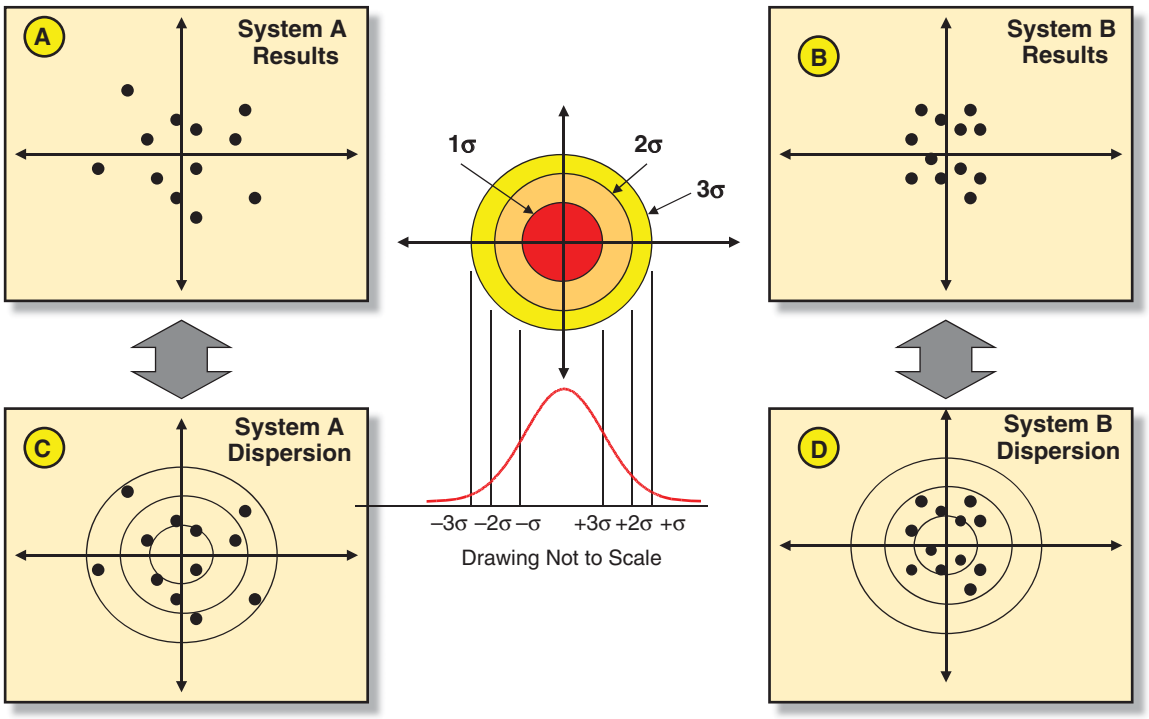


Figure 30.4 Circular Error Probability (CEP) Example

Each Rifle System is mounted in a test fixture and bore sighted for distance. When environmental conditions are acceptable, expert marksmen "live fire" the required number of rounds from each rifle. Marksmen are unaware of the manufacturer of each rifle. Miss distance firing results are shown in Panels A and B.

Using the theoretical bullseye as the origin, you superimpose the concentric lines about the bull's eye representing the $\pm 1\sigma$, $\pm 2\sigma$, and $\pm 3\sigma$ limits as illustrated in the center of the graphic. Panels C and D depict the results with miss distance as the deciding factor; System B is superior.

In this simple, ideal example, we focused exclusively on System *effectiveness*, not on *cost effectiveness*, which includes *system effectiveness*. The challenge is: things are not always ideal. Rifles, ammunition, reliability, and maintenance are not identical in cost. *What do you do*? A solution lies in the Cost as an Independent Variable (CAIV) (Figure 32.4) and Utility Functions introduced later (Figure 32.9). Given the test results, what are the Users' pre-defined Utility Function Figures of Merit (FOMs) concerning the accuracy relative to cost and other factors?

If Rifle System A costs one-half as much as Rifle System B, does the increased performance of Rifle System B substantiate the cost differential? You may decide that the $\pm 3\sigma$ point is the *minimum* threshold requirement for System acceptability. Thus, from a CAIV perspective, System A meets the specification threshold requirement and costs one-half as much, yielding the best value.

You can continue this analysis further by evaluating the utility of hitting the target on the first shot for a given set of time constraints, .

30.6.6 Data Correlation

Engineering often requires developing mathematical algorithms that model best-fit approximations to real-world data set characterizations. Data are collected to *validate* that a System produces high-quality data within predictable values. We refer to the degree of "fit" of the actual data to the standard or approximation as *data correlation*.

Data correlation is a measure of the degree to which actual data dispersion regress toward a central mean of predicted values. When *actual* values match *predicted* values, data correlation is 1.0. Thus, as data set variances diverge away from the central mean, the degree of correlation represented by r, the correlation coefficient, progresses toward zero. To illustrate the concept of data correlation and convergence, Figure 30.5 provides examples.

30.6.6.1 Positive and Negative Correlation Data correlation is characterized as *positive* or *negative* depending on the slope of the line representing the central mean of the data set over a range of input values. Panel A of Figure 30.5 represents a *positive* (slope) correlation; Panel B represents a

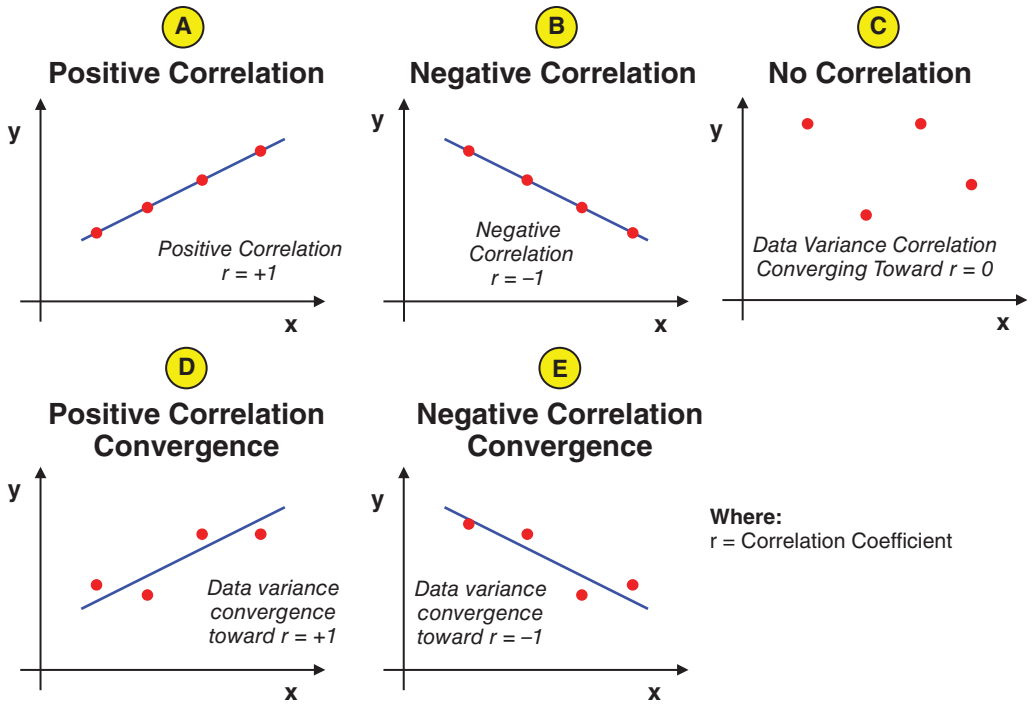


Figure 30.5 Data Correlation Concept

negative (slope) correlation. This brings us to our next point, convergence or regression toward the mean.

30.6.6.2 Regression Toward the Central Mean Since Engineering data reflect variations in physical characteristics, *actual* data do not always perfectly match the predicted values. In an ideal situation, we could state that the data *correlate* over a bounded range *if* all of the values of the data set are perfectly aligned along the central mean line as illustrated in Panels A and B of Figure 30.5.

In reality, data are typically dispersed along the central mean trend line. Thus, we refer to the *convergence* or *data variance* toward the mean as the *degree of correlation*. As data sets regress toward a central mean, the data variance or correlation *increases* toward $r = +1$ or -1 as illustrated in Figure 30.5, Panels D and E. Data variances that *decrease* toward $r = 0$ indicate *decreasing convergence* or *low correlation*. Therefore, we characterize the relationship between data parameters as *positive* or *negative* data variance *convergence* or *correlation*.

30.6.7 Statistical Influences Summary

Our discussions of statistical influences on System Design practices were predicated on a basic understanding of statistical methods and provided a high-level overview of key statistical concepts that influence SE design decisions.

We highlighted the importance of using statistical methods to define *acceptable* or *desirable* design ranges for input and output data. We addressed the importance of establishing boundary conditions for Normal operating ranges, *cautionary* ranges, *warning* ranges, as well as establishing safety margins. Using the basic concepts as a foundation, we addressed the concept of cumulative errors, Circular Error Probabilities (CEP), and data correlation. We also addressed the need to bound *acceptable* and *unacceptable* system inputs and outputs that include products, by-products, and services.

Statistical data variances have significant influence on SE technical decisions such as system performance, budgets, and safety margins and operational and system effectiveness. What is important is that SEs:

- Learn to recognize and appreciate Engineering I/O data variances.
- Know *when* and *how* to apply statistical methods to understand SYSTEM interactions with its OPERATING ENVIRONMENT.

30.7 CHAPTER SUMMARY

Our discussion of analytical decision support provided data and recommendations that support the Wasson SE Process Model (Figure 14.1) decision-making at all levels of abstraction. As an introductory discussion, analytical decision support employs various tools addressed in these chapters:

- Chapter 31 System Performance Analysis, Budgets, and Safety Margins
- Chapter 32 Trade Study: Analysis of Alternatives (AoA)
- Chapter 33 System Modeling and Simulation (M&S)
- Chapter 34 System Reliability, Maintainability, and Availability (RMA)

Remember—As a professional, you have an obligation to yourself, Enterprise, and Users to:

- Employ best practices to ensure the *integrity* of the results that reflect on your reputation.
- Produce decisions based on an *unbiased* preponderance of the objective, fact-based evidence that will withstand professional scrutiny and challenges.

30.8 GENERAL EXERCISES

30.8.1 Level 1: Chapter Knowledge Exercises

1. What is analytical decision support?
2. What are the attributes of a technical decision?
3. What is the SEs role in technical decision-making?
4. What is system performance analysis and evaluation?
5. What types of Engineering analyses are performed? Provide 10 primary types.
6. How are Engineering analyses documented?
7. What format should be used for documenting Engineering analyses?
8. What are SE design and development rules for analytical decision making?
9. How do you characterize random variations in system inputs sufficiently to bound the range of acceptable values in terms of Normal Distribution with standard deviations?
10. How do SEs establish criteria for *acceptable* and *unacceptable* system inputs and outputs?
11. What is a design range?
12. How are upper and lower tolerance limits established for a design range?
13. How do SEs establish criteria for caution and warning indicators?
14. What development methods can be employed to improve our understanding of the variability of Engineering input data?
15. What is CEP? How is it used?
16. What is meant by the degree of correlation?
17. What are two types of correlation?
18. Explain the differences between Technical Report Recommendations, Findings, Conclusions, and their sequences of presentation in the document.
19. Explain why it is important to have a documented Problem Statement (Chapter 4) from the decision-maker authorizing and tasking an analysis.
20. Explain why it is important to state the decision-maker's Problem Statement in an Analysis Technical Report.
21. Why is it important to establish a methodology for an analysis before its start?

30.8.2 Level 2: Chapter Knowledge Application Exercises

Refer to www.wiley.com/go/systemengineeringanalysis2e

30.9 REFERENCES

FAA SEM (2006), *System Engineering Manual*, Version 3.1, Vol. 3, Washington, DC: Federal Aviation Administration (FAA), National Airspace System (NAS).

DAU (2012), *Glossary: Defense Acquisition Acronyms and Terms*, 15th ed. Ft. Belvoir, VA: Defense Acquisition University (DAU) Press. Retrieved on 4/12/15 from: http://www.dau.mil/publications/publicationsDocs/Glossary_15th_ed.pdf

INCOSE (2011), *System Engineering Handbook, TP-2003-002-03.2.2*, Seattle, WA: International Council on System Engineering (INCOSE).

INCOSE (2015), *Systems Engineering Handbook: A Guide for System Life Cycle Process and Activities* (4th ed.). D. D. Walden, G. J. Roedler, K. J. Forsberg, R. D. Hamelin, and, T. M. Shortell (Eds.). San Diego, CA: International Council on Systems Engineering.

MIL-STD-882E (2012), *System Safety*, Washington, DC: Department of Defense (DoD).

31

SYSTEM PERFORMANCE ANALYSIS, BUDGETS, AND SAFETY MARGINS

System performance manifests itself via the cumulative performance results of the integrated set of System Elements (Figures 5.3 and 10.12) at a specific instance in time. That performance ultimately determines success in achieving MISSION SYSTEM and ENABLING SYSTEM objectives—in some cases, survival.

When SEs allocate and flow down SYSTEM performance requirements, there is a tendency to think of those requirements as *static* parameters. For example, "shall be +12.3 ± 0.10 vdc." Aside from User switch settings or configuration parameters, *seldom* are parameters static or steady state. For example, from a weight perspective, fuel in an aircraft is measurable only at a specific instant in time. Fuel weight is dynamic and diminishes over time as a function of its consumption.

From an SE perspective, we allocate and flow down System Performance Specification (SPS) requirements via the System Architecture's hierarchical framework. As an example, consider static weight. SEs have a budget of 100 pounds to allocate equally to three components. Invariant static parameters such as weight make the SE requirements allocation task a lot easier. However, this is not the case for many System Performance Specification (SPS) and Entity Development Specifications (EDS) requirements. *How do we establish values for acceptable SYSTEM inputs (Figure 3.2) that are subject to variations such as environmental conditions, Time of Day (TOD), Time of Year (TOY), signal properties, human error, and other variables*?

System requirement parameters are often characterized by statistical value distributions such as Gaussian (Normal), Binomial, and LogNormal (Poisson) (Figure 34.3) with frequency distributions that vary independently about a *central mean*. Using our aforementioned static requirements example, we can state that the input voltage must be constrained to a range of +12.20 vdc (-3σ) to +12.40 vdc ($+3\sigma$) with a *mean* of +12.30 vdc for a *prescribed* set of operating conditions.

On inspection, this sounds very simple and straightforward. The challenge is: *how did SEs decide* that the:

1. Central mean value needed to be +12.30 vdc?
2. Variations could not exceed ± 0.10 vdc?

This simple example illustrates one of the most *challenging and perplexing aspects* of System Engineering—*allocating* performance requirement tolerances. It also illustrates the fallacy of focusing on Functional Analysis, which is the easy part; derivation and allocation of performance values such as those above is the difficult part. This is why Functional Analysis is relevant but outdated and Capability Analysis is the appropriate action to perform.

Many times, SEs simply do not have any precedent data. For example, consider human attempts to build bridges, develop and fly an aircraft, launch rockets and missiles, and land on the Moon and Mars. Analysis supported by a lot of *trial and error* data collection and observation may be all you have to establish initial estimates of these parameters.

There are a number of ways one can determine these values. Examples include the following:

- Unscientific educated guesses or *guesstimates* based on seasoned experience.

System Engineering Analysis, Design, and Development: Concepts, Principles, and Practices, Second Edition. Charles S. Wasson.

Companion Website: www.wiley.com/go/systemengineeringanalysis2e

- *Theoretical* and *empirical* trial and error analysis.
- *Validated* data modeling and simulation with increasing fidelity.
- Prototyping demonstrations such as "live fire" tests.

The challenge is being able to identify a reliable, low-risk, level of confidence method of determining values for statistically variant parameters.

Chapter 31 describes how we allocate SPS or EDS requirement performance values to lower levels. We explore how capability—*functional* and *non-functional*—performance is analyzed and allocated. This requires building on previous practices such as statistical influences concerning System Design introduced in Chapter 30. We introduce the concepts of partitioning - decomposing - cycle-time-based performances into queue, process, and transport times. Finally, we conclude with a discussion of System Performance Optimization and System Analysis Reporting.

31.1 DEFINITIONS OF KEY TERMS

- "**Design-to**" Measure of Performance (MOP) A targeted mean value bounded by *minimum* and/or *maximum* threshold values levied on a SYSTEM capability performance parameter to constrain decision making.
- **Design Safety Margin** An *additive* or *multiplicative* factor that is applied to a Measure of Performance (MOP) for a capability or physical characteristic to accommodate variations and uncertainty in component materials, performance, and environmental conditions.
- **Performance Budget Allocation** A *minimum, maximum, or min–max* constraint that represents the absolute thresholds that bound a capability or performance characteristic.
- **Processing Time** The statistical mean time and tolerance that characterizes the time interval between an input stimulus or cue event and the completion of processing of the input(s).
- **Queue Time** The statistical mean time and tolerance that characterizes the time interval between the *arrival* of an input for processing and the point when processing begins.
- **System Latency** The time interval between a stimulus, excitation, or cue event and a system response event. Some refer to *latency* as the Input/Output (I/O) *throughput* or *response time* of the system.
- **Transport Time** The statistical mean time and tolerance that characterizes the time interval between transmission from an Entity and its receipt at the next Entity for processing.

31.2 PERFORMANCE "DESIGN-TO" BUDGETS AND SAFETY MARGINS

Performance Budgets and Margins Principle

Principle 31.1 Every SYSTEM/ENTITY design requires performance budgets that bound capabilities and provide a *margin of safety* to accommodate *variability* and *uncertainty* in component and operational performance for a prescribed set of OPERATING ENVIRONMENT conditions.

Every capability or physical characteristic of a SYSTEM or ENTITY must be bounded by performance constraints. This is very important in top-down/bottom-up/horizontal design, whereby SYSTEM capabilities' performance is partitioned, allocated, and flowed down into multiple levels of design detail. Therefore, every specification should have supporting performance allocations budgets and margins that are managed by the System/Product Development Team (SDT/PDT) accountable for implementing the specification.

31.2.1 Achieving MOPs

MOP Risk Principle

Principle 31.2 Every MOP has an element of development risk. Mitigate the risk by establishing one or more boundary condition thresholds to trigger risk mitigation actions to reduce the risk to acceptable levels (Figure 21.12).

The mechanism of partitioning and allocating SYSTEM performance into subsequent levels of detail is referred to as *performance budgets* and *design safety margins*. In general, performance budgets and design safety margins allow SEs to impose performance constraints on capabilities that include a margin of safety. Philosophically, if overall SYSTEM performance must be controlled, so should the contributing entities at lower levels of abstraction.

Performance constraints are further partitioned into: (1) "Design-to" MOPs and (2) performance safety margins.

Derivation of MOPs Context

Author's Note 31.1 Our discussion here as has two contexts concerning derivation of MOPs:

- Context #1—Proposal Phase – Derivation of System Level MOPs for developing an SPS to submit as part of a proposal response.
- Context #2—System Development Phase – SPS requirements derivation, allocation, and flow down to lower level EDS' as Design-to MOPs.

31.2.2 Design-to MOPs

Design-to-MOPs serve as the key mechanism for allocating, flowing down, and communicating performance constraints to lower-levels ENTITIES (Figure 14.3). The actual allocation process is accomplished by a number of methods. Grady (2006, pp. 59–60) as an example identifies three types of requirements allocation and flow down methods: (1) *equivalence*, (2) *apportionment*, and (3) *synthesis*. Let's explore each of these further.

31.2.2.1 Requirement Allocations by Equivalence

Principle 31.3

Equivalence Allocation Principle

Apply the *equivalence* allocation method to allocate and flow down *non-functional* requirement constraints to PRODUCTS, SUBSYSTEMS, and ASSEMBLIES.

Non-functional specification requirements such as color, safety, or security are allocated and flowed down directly to lower level ENTITIES with the same units. Observe that *non-functional* requirements represent constraints documented in specification Section 3.6 Design and Construction Constraints (Table 20.1). To illustrate *allocation by equivalence*, consider the following example:

Example 31.1

SPS Requirement Statement

3.2.1 The System **shall** be painted XYZ color in accordance with ABC Standard, para. XXX (title).

Allocation of this requirement via *equivalence* to SUBSYSTEMS results in the following:

- Subsystem #1 **shall** be painted XYZ color in accordance with ABC Standard, para. XXX (title).
- …
- Subsystem #*n* **shall** be painted XYZ color in accordance with ABC Standard, para. XXX (title).

Observe how the SUBSYSTEM (child) requirements (Figure 21.2) are flowed down directly from the SYSTEM (parent) requirement with no change in wording other than the *subject* of the requirement—SYSTEM versus SUBSYSTEM.

You may observe that other *non-functional* constraints such as *size* and *weight* are not included as candidates for allocation by *equivalence*. The reason is that multi-level System Architecture components vary in size and weight unless they are replicated at the same level of abstraction. As such, their allocations must be *apportioned* due to a number of factors that drive overall SYSTEM performance. This brings us to our next topic, Requirements Allocations by Apportionment.

31.2.2.2 Requirement Allocations by Apportionment

Principle 31.4

Apportionment Allocation Principle

Apply the apportionment method to allocate and flow down quantifiable, MOPs such as electrical power, timing, weight, or size based on informed decisions, component or design history, boundary conditions, technology constraints, feasibility, and professional experience.

Since higher-level SYSTEM capability requirements must produce performance-based behavioral outputs that result in the achievement of a mission outcome, multi-discipline PDTS must allocate the performance to lower level PRODUCTS, SUBSYSTEMS, ASSEMBLIES, and SUBASSEMBLIES (Figure 8.4). On inspection, it may appear that we can simply employ an *arbitrary* or *discretionary* approach to allocating the performance values based on *seasoned* experience. In some cases, this may be true. However, other factors such as complexity may lead to the need to *apportion* the values among SUBSYSTEM, ASSEMBLIES, or SUBASSEMBLIES. Some performance allocations may require Modeling and Simulation (M&S) validated by test data.

Observe that *quantifiable* MOPs is an operative phrase in Principle 31.4. We know that:

1. Performance is the result of the integrated chain of performance contributions—SUBASSEMBLIES → ASSEMBLIES → SUBSYSTEMS → PRODUCTS → SYSTEM-Level outputs and outcomes.
2. The SYSTEM's Input/Output (I/O) *transfer function* of that integrated chain can be represented mathematically.

Therefore, we can state that *apportionment* allocations must be based on valid mathematical derivations and contributions to achieving the overall SYSTEM-Level performance requirement. To illustrate *allocation by apportionment*, consider the following example:

Example 31.2

SPS Requirement Statement

3.6.8.X The System's average power consumption shall not exceed 20.0 W.

Allocation of this requirement via *apportionment* to SUBSYSTEMS results in the following:

1. SUBSYSTEM #1's average power consumption shall not exceed 5.0 W(watts).
2. SUBSYSTEM #2's average power consumption shall not exceed 12.0 W.
3. SUBSYSTEM #3's average power consumption shall not exceed 3.0 W.

31.2.2.3 Requirement Allocations by Synthesis

Synthesis Allocation Principle

Principle 31.5 Apply the *synthesis* allocation method to flow down *complex*, *quantifiable* SYSTEM-Level performance requirements values such as Measures of Effectiveness (MOEs) or Measures of Suitability (MOSs) to PRODUCTS, SUBSYSTEMS, ASSEMBLIES using analytical models.

Some SPS or EDS capability requirements require analytical or simulation models to derive *quantifiable nominal performance and tolerance values*. Examples include parameters related to outcomes such as range and accuracy due to the complexities of SYSTEM usage, OPERATING ENVIRONMENT conditions. The analytical models should be based on *validated* mathematical models of the physical SYSTEM. As an example, consider the complexities of allocating an MOE such as a vehicle's *fuel economy* as shown in Figure 5.8 to engine efficiency, vehicle aerodynamic drag, fuel octane and quality, electronic timing, road roughness and friction, and Environmental Conditions—temperature, humidity.

31.2.3 Performance Safety Margins

Simply allocating performance values to PRODUCTS, SUBSYSTEMS, and ASSEMBLIES does not mean that the SYSTEM or PRODUCT will achieve the expected result. Although components can be performance tested, *inevitably* there are *latent defects* such as design errors, flaws, and so on, as well as *variances* in their compositional properties. As a result, we need to incorporate performance *safety margins* into their analysis to accommodate these factors.

Performance safety margins enable us to accomplish two objectives. They provide a:

1. Means to accommodate variations in tolerances, accuracies, and latencies in SYSTEM RESPONSES plus human judgment – slips, lapses, and errors (Chapter 24) – as well as effectiveness and efficiency.
2. Safety margin "reserve" for decision makers to trade off previously allocated performance values to specific entities to *optimize* overall SYSTEM or PRODUCT performance.

Performance safety margins serve as contingency reserves to compensate for component variations or to accommodate *worst-case* scenarios that:

1. May have been *underestimated*.
2. Potentially create safety risks and hazards.
3. Result from human errors (Chapter 24) in computational precision and accuracy.
4. Are due to physical variations in component and material *mass properties*.
5. Are due to poor workmanship.
6. Result from the "Unknown-Unknowns."

Every Engineering discipline employs rules of thumb and guidelines for incorporating safety margins. Typically, performance safety margins might vary from 2X (200%) to 3X (300%) the nominal SPS or EDS value, depending on the application. Examples include (1) Mechanical Engineering designs for structural, tension, and compression loads and (2) Electrical Engineering power loads.

There are limitations to the practicality of safety margins in terms of (1) cost—benefits, (2) *probability* or *likelihood of occurrence*, (3) alternative actions, and (4) reasonable measures, among other things. In some cases, the implicit cost of *increasing* safety margin MOPs above a practical level can be offset by taking appropriate SYSTEM or PRODUCT safety precautions, safeguards, markings, and procedures that reduce the probability of occurrence.

Safety Margins

Warning 31.1 Always seek guidance from your project and technical management, disciplinary standards and practices, or local Enterprise Engineering command media to establish a consensus about *safety margins* for your project. When these are established prior to the Contract Award, document the authoritative basis for the selection, or incorporate into project command media, project memoranda, or plans, and disseminate to all personnel.

Safety margins, as the name implies, involve technical decision making to prevent potential *safety hazards* from occurring (Figure 24.1). Any potential safety hazards carry safety, financial, and legal liabilities. Establish safety margins that safeguard the system and its operators, general public, property, and the environment.

31.2.4 Applying Design-To MOPs and Performance Safety Margins

Figure 31.1 illustrates how Design-To MOPs and *performance safety margins* are established. In this figure, the MOPs for SYSTEM capabilities and physical characteristics are given in generic terms or units. Note that the units can represent MOPs such as time, electrical power, or mass properties.

Author's Note 31.2 The example in Figure 31.1 shows the basic method of allocating performance budgets and safety margins. In reality, this *highly iterative*, time-consuming process often

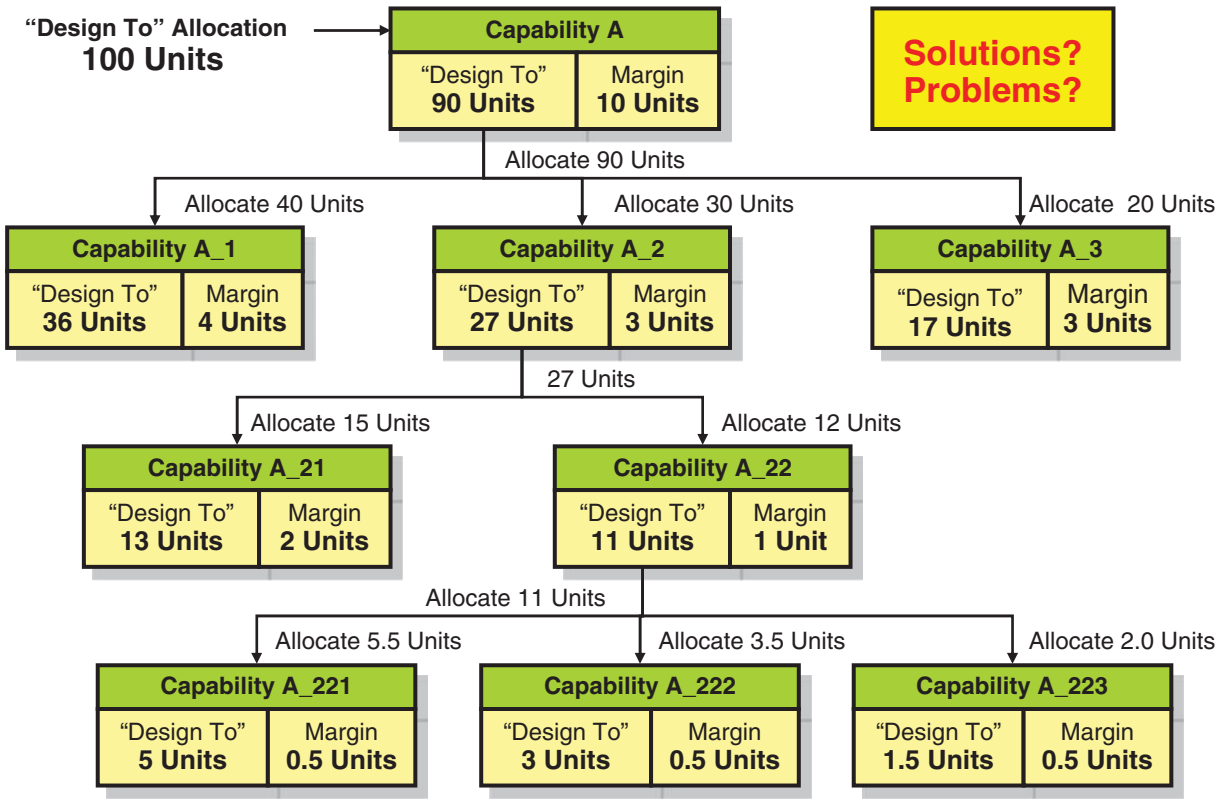


Figure 31.1 Performance Budget and Design Margin Allocations - Solutions or Problems?

requires analyses, trade studies, modeling, simulation, prototyping, and negotiations to balance and optimize overall SYSTEM performance.

Let's assume the SPS or EDS specifies that Capability A have a performance constraint of not-to-exceed 100 units (Figure 31.1). System designers decide to establish a 10% safety margin at all levels of the design. Therefore, they begin with a Design-To MOP of 90 units and a safety margin of 10 units. The Design-To MOP of 90 units is allocated as follows: Capability A1 is allocated an MOP of 40 units, Capability A2 is allocated an MOP of 30 units, and Capability A3 is allocated an MOP of 20 units.

Focusing on Capability A2, the MOP of 30 units is partitioned into a Design-To MOP of 27 units (90%) and a safety margin MOP of 3 units (10%). The resultant Design-To MOP of 27 units is then allocated to Capability A21 and Capability A22 in a similar manner.

Some SEs will rightfully argue that once you establish the initial 10 (10%) unit safety margin MOP at the Capability A level, there is no need to establish design safety margins at lower levels of the capability—(Capability A3 safety margin, etc.). Observe how the second level has reserved an additional 10 units of margin to the Capability A-1, A-2, and A-3 budgets above and beyond the 10 units at the Capability A level. They emphasize that as long as all subordinate level capabilities meet their Design-To MOP performance budgets, the 10-unit MOP safety margin adequately covers situations where lower level performance exceeds allocated budgets.

A key question arises concerning the approach shown in Figure 31.1. Some SEs make the case that imposing *performance safety margins* at lower levels *unnecessarily* constrains critical resources and *increases* SYSTEM cost due to the need for higher performance EQUIPMENT. For some non-safety critical, non–real-time performance applications, this may be true; every SYSTEM and application is unique.

One approach to this situation is to establish performance safety margins at all levels of abstraction except for the lowest level. Consider the following example:

Example 31.3 **Performance Safety Margins**

Using Figure 31.1 as a reference, performance safety margins would be restricted to Capability A22 and 1 Unit would be held in reserve in case Capabilities A221, A222, or A223 required more flexibility in their allocated performance of 5.5 units, 3.5 units and 2.0 units, respectively.

If you pursue the approach in Example 31.3, you may have unnecessarily reserved valuable Safety Margin at lower

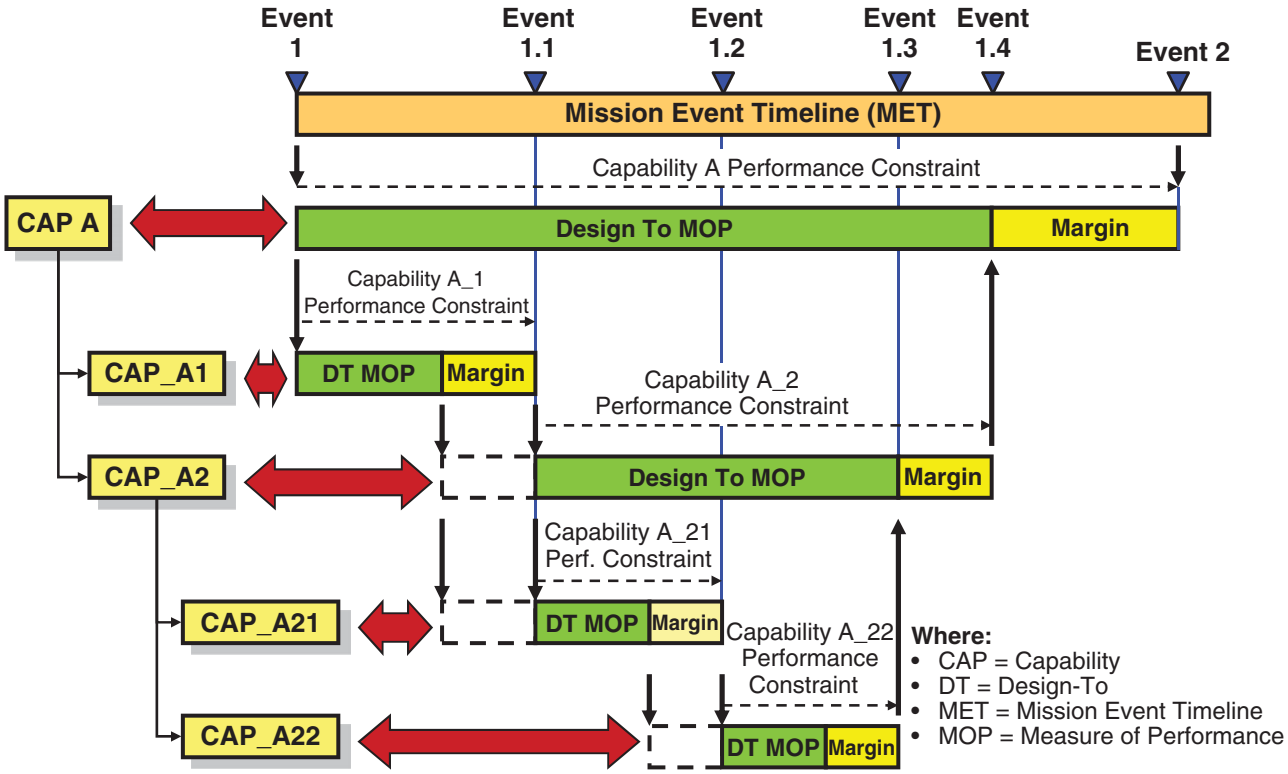


Figure 31.2 Time-based Performance Budgets and Safety Margins Application.

levels and driven up cost as a consequence - just to reserve a 10% Safety Margin at all levels of abstraction. Consider the implications:

- At System delivery, Capability A is required to have an MOP of 90 Units with a Safety Margin of 10 Units.
- Instead, those 90 units have been reduced by an additional 14.5 Units at lower levels due to the need for all capabilities to have a 10% Safety Margin.

In summary, think smartly about these decisions and avoid unnecessary performance constraints that may increase technical, cost, and schedule risk. Keep it simple! Maintain a reserve at the higher levels for those occasions when the reserve may be required to resolve performance due to issues at lower levels.

If you encounter *unreasonable* difficulties meeting lower level performance allocation constraints, you should weigh options, benefits, costs, and risks. Since SYSTEM/ENTITY performance *inevitably* requires fine-tuning for SYSTEM *optimization*, always establish design performance margins at higher levels to ensure flexibility in the integrated performance in achieving SPS requirements. Once all levels of design are defined, rebalance the hierarchical structure of performance budgets and design safety margins as needed.

To illustrate how we might implement time-based performance budgets and safety margins allocations, let's explore another example.

Example 31.4

Mission Event Timeline (MET) Allocations

To illustrate this concept, refer to Figure 31.2. The left side of the graphic portrays a hierarchical partitioning - decomposition - of a Capability A. The SPS or EDS requires that Capability A completes processing within a specified period of time such as 200 ms (milliseconds). System designers designate the initiation of Capability A as Event 1 and its completion as Event 2. System Designers partition the Capability A time interval constraint into (1) a Design-To MOP and (2) a safety margin MOP. The Design-To MOP constraint is designated as Event 1.4.

Author's Note 31.3

Event Identification Labeling

Initially, Mission Event Timeline (MET) Event 1.4 may not have this label. We have simply applied the Event 1.4 label to provide a degree of sequential consistency across the MET (Events 1.1, 1.2, 1.3, etc.). Events 1.2 and 1.3 are established based on lower level allocations for Capabilities A1 and A2.

Referring to Figure 31.2, Capabilities CAP_A1 and CAP_A2 are derived from Capability A. Let's assume that CAP_A1 and CAP_A2 are sequential processes. Given that Capability A has a Design-To MOP constraint, the challenge is: *how do we allocate the performance constraint across CAP_A1 and CAP_A2*? We employ analyses, models and simulations, prototypes, and so forth to make a decision as to how to appropriately allocate the performance. Then, for CAP_A1 and CAP_A2 spilt their respective MOP performance constraints into a Design-To MOP and a Safety Margin. We continue the process to successively lower levels of abstraction.

31.2.4.1 Reconciling Performance Budget Allocations and Safety Margins As the SDT and PDTs apply Design-To MOP allocations, *what happens if there are critical performance issues with the initial allocations?* Let's assume that Capability A22 in Figure 31.2 was initially allocated 12 units. An initial analysis of Capability A22 indicates that 13 units *minimum* are required. *What should an SE do*?

Capability A22 owner confers with the higher level Capability A2 and peer level Capability A21 owners. During the discussions, the Capability A21 owner indicates that Capability A21 was allocated 15 units but only requires 14 units, which includes a safety margin. The group reaches a consensus to *reallocate* an MOP of 14 units to Capability A21 and an MOP of 13 units to Capability A22.

31.2.4.2 Final Thoughts about Performance Budgets and Margins As part of the Wasson SE Process Model (Figure 14.1), the process of allocating performance budgets and safety margins is *highly iterative* and *recursive*—top-down/bottom-up/left-right/right-left negotiation process. The intent is to reconcile the inequities as a means of achieving and optimizing overall SYSTEM performance. Without negotiation and reconciliation, you get a condition referred to as *suboptimization* of a single item, thereby degrading overall SYSTEM performance.

31.2.5 Performance Budget and Safety Margin Ownership and Control

A key question is *who owns performance budgets and safety margins?* In general, the multi-discipline SDT or PDT accountable for implementing the specification controls *derivation* and *allocation* of the performance budget MOPs and safety margins within that ENTITY (Figure 21.9). Remember – MOPs originate from specifications owned and controlled by the SDT/PDT that controls the architecture that includes the specified Entity (Figure 27.1). A PDT cannot change the allocated and flowed down MOPs unless they request a change from the specification owner. Therefore, within the boundaries of their Entity, they are free to allocate MOPs into Design-To and Safety Margin budgets assuming that they do not impact their interface performance requirements.

31.2.6 How are Performance Budgets and Margins Documented?

Performance budgets and safety margins are documented in a number of ways, depending on the size, resources, and tools.

- First, requirements allocations should be documented in a decision database or spreadsheet controlled by the Lead SE or SDT. Requirements management tools based on relational databases provide a convenient mechanism to record the allocations.
- Second, as performance allocations, they should be formally documented and controlled as specific requirements flowed down to lower level specifications.

A relational database Requirements Management Tool (TOOL) allows you to:

- Document the allocation in a Performance Allocations Document that can be linked to an SPS or EDS via the RMT.
- Flow the allocation down to lower level PRODUCT, SUBSYSTEM, and other EDSs with traceability linkages back to the higher-level *parent* performance constraint (Figure 21.2).
- Trace the allocated performance up to higher levels.

31.3 ANALYZING SYSTEM PERFORMANCE

The preceding discussion introduced the basic concepts of performance budgets and design safety margins. Implementations of these Design-To MOPs are discussed in Engineering textbooks such as electronics engineering and mechanical engineering. However, from an SE perspective, integrated electrical, mechanical, optical, or chemical systems have performance variations interfacing with similar EQUIPMENT and PERSONNEL within larger structural systems. The interactions among these SYSTEMS and levels of abstractions require in-depth, multi-discipline, SE analysis to determine *acceptable limits* for performance variability (Figure 30.1).

31.3.1 Understanding System Performance and Tasking

Overall SYSTEM performance represents the integrated performance of the System Elements (Chapter 8) such as EQUIPMENT, PERSONNEL, and MISSION RESOURCES that provide SYSTEM capabilities, operations, and processes. As integrated elements, if the performance of any of these

mission critical items such as PRODUCTS or SUBSYSTEMS is *degraded*, so is the overall SYSTEM performance (Figure 10.12), depending on the robustness of the System Design Solution. Robust designs often employ redundant EQUIPMENT - HARDWARE and/or SOFTWARE - design implementations to *minimize* the effects of SYSTEM performance degradation on achieving the mission and its objectives (Chapters 26 and 34).

From an SE perspective, SYSTEM capabilities, operations, and executable processes are response mechanisms to "tasking" assigned and initiated by HIGHER ORDER Systems with authority. Thus, SYSTEM "tasking" requires the integrated set of sequential and concurrent capabilities to accomplish a desired performance-based outcome or result.

To illustrate the tasking of capabilities, consider the simple graphic shown in Figure 31.3. Note the chain of sequential tasks, Tasks A through "n." Each task has a finite duration bounded by a set of MET performance parameters. The time period marked by the Start of one task until the Start of another is referred to as the *throughput* or *cycle time*. The cycle time parameter brings up an interesting point, especially in establishing a convention for decision-making.

31.3.2 Establishing Cycle Time Conventions

When you establish cycle times, you need to define a convention that will be used *consistently* throughout your analyses. There are a couple of conventions as shown in Figure 31.4. Convention A defines *cycle time* as beginning with the Start of Task A and ending with the Start of Task B. In contrast, Convention B defines *cycle time* as starting at the End of Task A and completing at the start of Task B. Either method is *acceptable.*

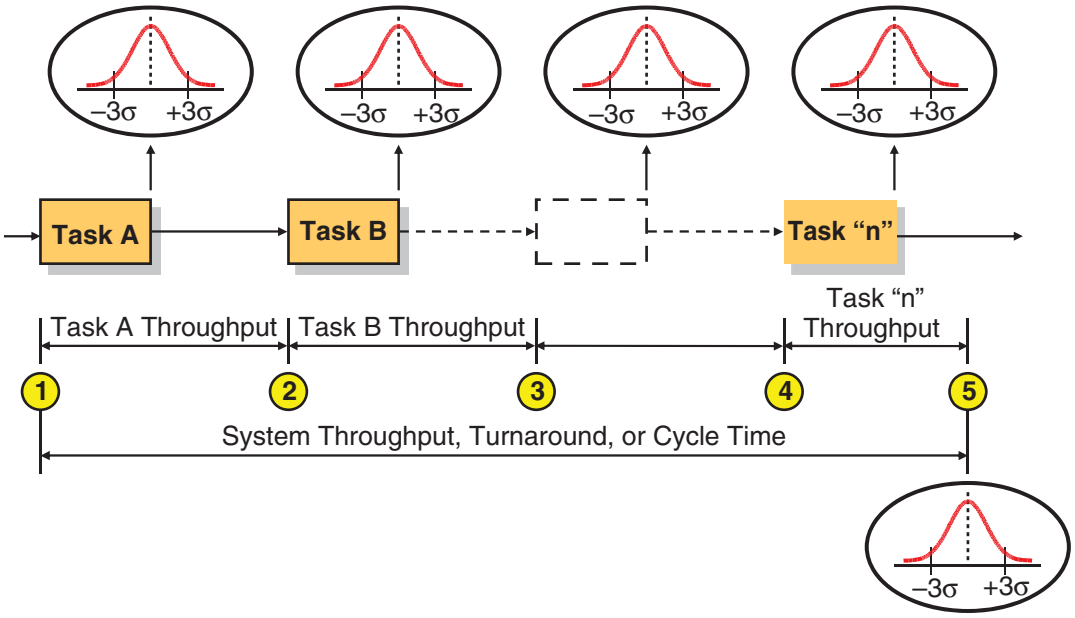


Figure 31.3 Mission Event Timeline (MET) Analysis

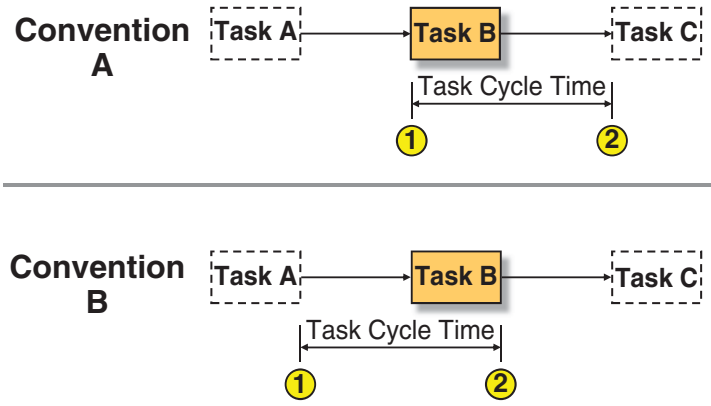


Figure 31.4 Task Cycle Time Conventions

31.3.3 Operational Tasking of a System Capability

Principle 31.6

Processing Segments Principle

When planning processing tasks, at a minimum, include considerations for *queue* time, *processing* time, and *transport* time in all computations.

Most tasks, whether performed by PERSONNEL or EQUIPMENT (Figure 24.14), involve three phases as follows:

- Pre-Mission Phase preparation, configuration, or reconfiguration.
- Mission Phase performance of the task.
- Post-Mission Phase delivery of the required results and any residual Post-Capability Housekeeping Operations (Figure 10.17) in preparation for the next tasks.

Communications intensive systems such as humans and computers have a similar pattern that typically includes *queues* or waiting line theory. We illustrate this pattern in Figure 31.5.

In the illustration, a typical task provides three capabilities: (1) queue new arrivals, (2) perform processing, and (3) output results. Since new arrivals may overwhelm the processing function, new arrivals reside in a buffer or holding area based on a First In–First Out (FIFO) processing protocol or some other priority processing algorithm.

Each of these capabilities is marked by its own cycle time where:

- T_{Queue} = Queue Time.
- $t_{\text{Procesing}}$ = Processing Time.
- $t_{\text{Transport}}$ = Transport Time.

Figure 31.6 illustrates how we might partition -decompose- each of these capabilities into lower level time constraints for establishing budgets and performance safety margins.

Heading 31.1

At this juncture we have identified and partitioned task performance into three phases: Queue Time, Processing Time, and Transport Time. Philosophically, this partitioning enables us to partition allocated task performance times into smaller, more manageable entities. Beyond this point, however, tasks analysis becomes more complicated due to timing uncertainties. This brings us to our next topic, Understanding Statistical Characteristics of Tasking.

31.3.3.1 Understanding Statistical Characteristics of Tasking All SYSTEM tasks involve some level of performance *variability* and *uncertainty*. The level of *uncertainty* is created by the *inherent reliability* (Chapter 34) of the System Elements—PERSONNEL, MISSION RESOURCES, and EQUIPMENT. In general, task performance and its uncertainty can be characterized with statistics using Normal, Binomial, and LogNormal (Poisson) frequency distributions (Figure 34.2). On the basis of measured performance over a large number of samples, we can assign a probability that task or capability performance will complete processing within a most probable *minimum* amount of time and *not-to-exceed* a specified amount of time. To see how this relates to SYSTEM performance and allocated performance budgets and margins, refer to Figure 31.7.

Let's suppose that a task is initiated at Event 1 and must be completed no later than by Event 2. We refer to this time as $t_{\text{Allocation}}$. To ensure a *safety margin* for contingency and for growth purposes, we establish a performance safety margin, t_{Margin}. This leaves a remaining time, t_{Exec}, as the *maximum budgeted* performance time or Late Finish Time—that is, Queue Time + Processing Time + Transport Time.

Using the lower portion of the graphic, suppose that we perform an analysis and determine that the Task Execution Performance, represented by the gray rectangular box, is expected to have a mean finish time, t_{Mean}. We also determine with a level of probability that task completion may vary about the mean by $\pm 3\sigma$ points designated as *early finish* and *late finish*. Therefore, we equate the *latest finish* to the *maximum* budgeted performance, t_{Budget}. This means that once initiated at Event 1, the task must complete and deliver the output or results to the next task no later than t_{Budget}.

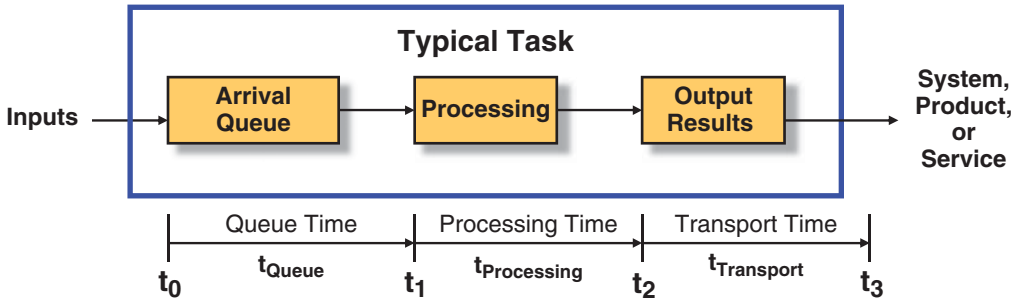


Figure 31.5 Anatomy of Typical Task Structure Performance

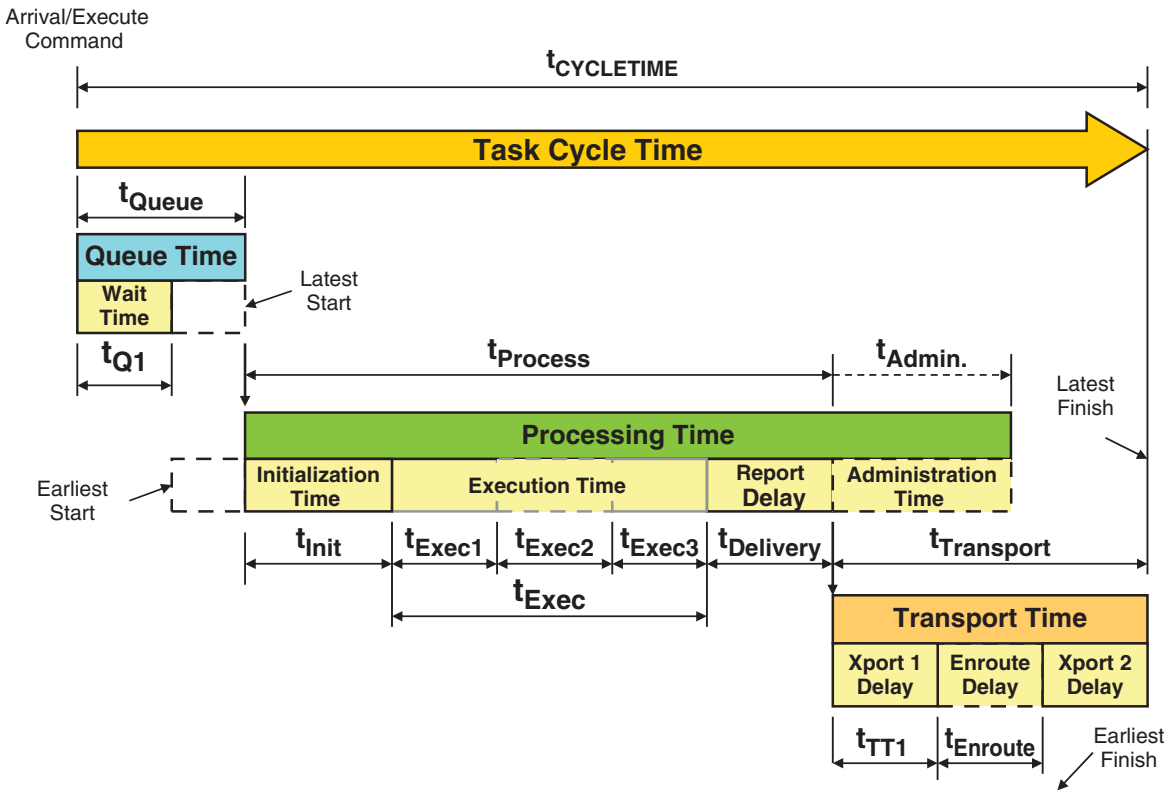


Figure 31.6 Task Throughput / Cycle Time Analysis

On the basis of the projected distribution, we also expect the task to be completed no sooner than the -3σ point, the Early Finish time.

If we translate this analysis into the Wasson SE Process Model (Figure 14.1) Requirements Domain Solution, we prepare a requirements statement that captures the capability and its associated performance allocation. Let's suppose that Capability B requires that Capability A complete processing and transmit data within 250 ms when Event 1 occurs. Consider the following example.

Example 31.5

"On receipt of a command (Event 1), Capability A **shall** process the incoming data within 250 + 0 ms."

Now let's suppose that Capability B requires receipt of the data within a window of time between 240 and 260 ms. The requirement might read as follows:

Example 31.6

"On receipt of a command (Event 1), Capability A **shall** transmit the results to Capability B within 250 ± 10 ms."

Heading 31.2

Our discussion emphasizes how overall SYSTEM task performance can vary. To understand how this variation occurs, let's take it one step further and discuss it relative to the key task phases.

31.3.3.2 Applying Statistics to Multi-task System Performance Our previous discussion focused on the performance of a single capability. If the statistical variations for a single capability are aggregated for a multi-level system, we can easily see how this impacts overall SYSTEM performance. We can illustrate this by the example shown in Figure 31.8.

Let's assume that we have an overall capability labeled Perform Task A. The purpose of Task A is to perform a computation using independent variable inputs, I_1 and I_2, and produce a computed result as an output. The key point of our discussion here is to illustrate time-based statistical variances to complete processing.

Let's assume that Perform Task A is composed of two subtasks, Subtask A1 and Subtask A2. Subtask A1 enters inputs I_1 and I_2. Each input, I_1, and I_2, has values that vary over a Gaussian (Normal) Distribution range with a central mean.

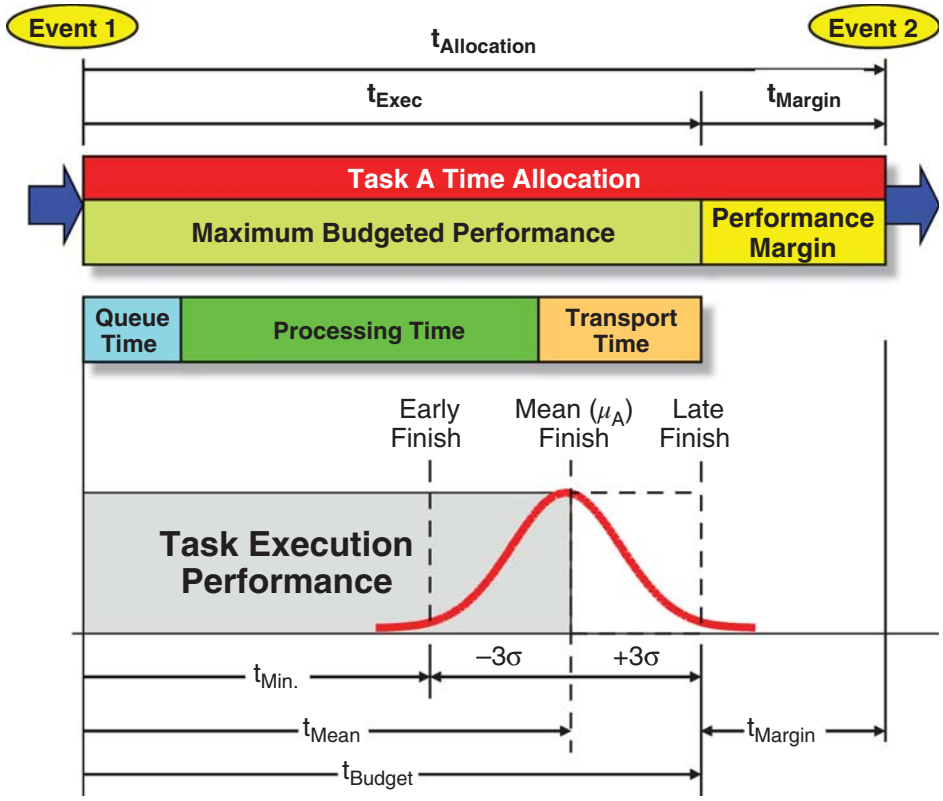


Figure 31.7 Task Timeline Elements and Their Statistical Variability

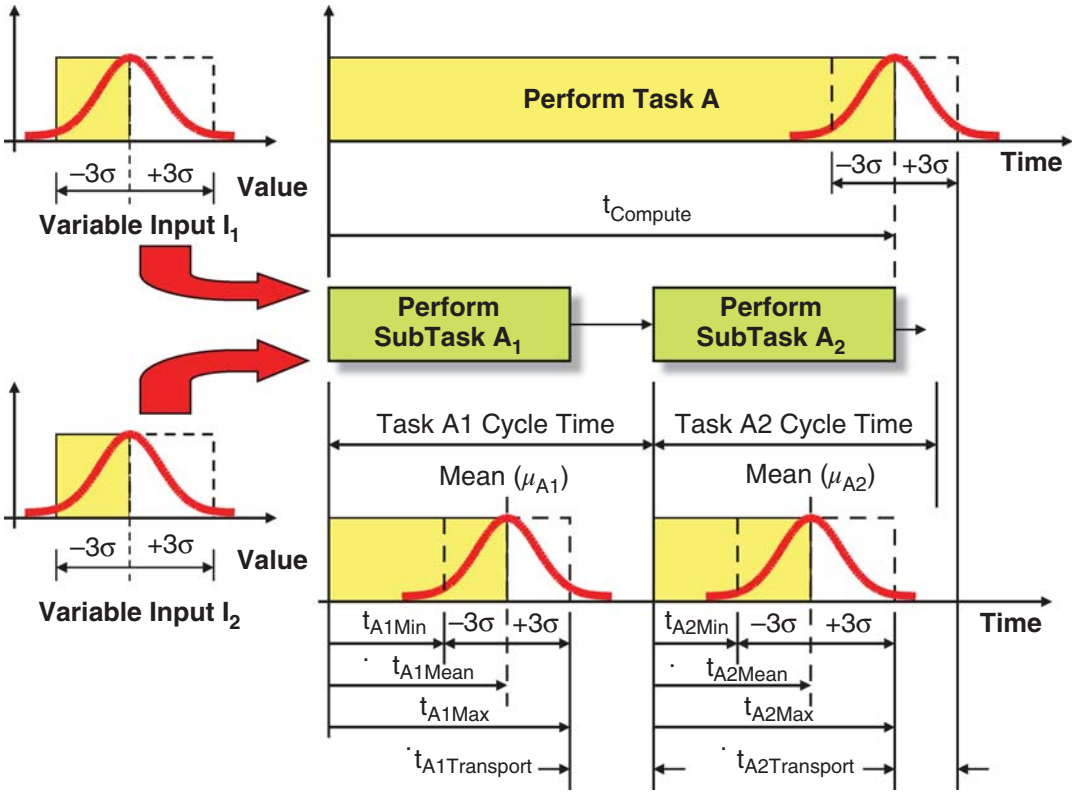


Figure 31.8 Statistical Analysis of Input and Series Task Performance Variability

When Subtask A1 is initiated, it produces a response within t_{A1MEAN} that varies from t_{A1MIN} to t_{A1MAX} .The output of Subtask A1 serves as an input to Subtask A2. Subtask A2 produces a response within t_{A2MEAN} that may occur as early as t_{A2MIN} or as late as t_{A2MAX}.

If we investigate the overall performance of Task A, we find that Task A is computed within $t_{Compute}$, as indicated by the central mean. The overall Task A performance is dependent on the statistical variance of the sequential series summation of Subtask A1 and Subtask A2 processing times as shown in Figure 31.8.

Heading 31.3

We have seen how statistical variations in inputs and processing impact SYSTEM performance from a timing perspective. Similar methods are applied to the statistical variations of Inputs I_1 and I_2 as independent variables over a range of values. The point of our discussion is to heighten your awareness of these variances. *Think* about and *consider* statistical variability when allocating performance budgets and design safety margins as well as analyzing data produced by the SYSTEM to determine compliance with requirements.

Given this understanding, let's return to a previous discussion about applying statistical variations to task phases of operation.

31.3.3.3 Applying Statistical Variations to Intra-Task Phases In our earlier discussion of task phases consisting of Queue time, Processing Time, and Transport Time (Figure 31.6), statistical variations in each of those phases impact the overall task performance outcome. Figure 31.9 serves as a guide for our discussion.

The central part of the figure represents an overall task and its respective Queue Time, Processing Time, and Transport Time phases. Below each phase is a statistical representation of the execution time. The top portion of the chart illustrates the aggregate performance of the overall task execution.

How does this relate to SE? If a given capability or task is required to be completed within the allocated cycle time and you are designing the SYSTEM with queues, computational devices, and transmission lines, you need to factor in these times as performance budgets and safety margin requirements to flow down to lower levels.

31.3.4 Applying Statistics to Overall System Performance

Our discussions to this point focused on the task and multi-tasking level. The ultimate challenge for SEs is: *how will the overall system perform?* Figure 31.10 illustrates the effects of statistical variability of the System Element Architecture (Chapter 8) and its OPERATING ENVIRONMENT.

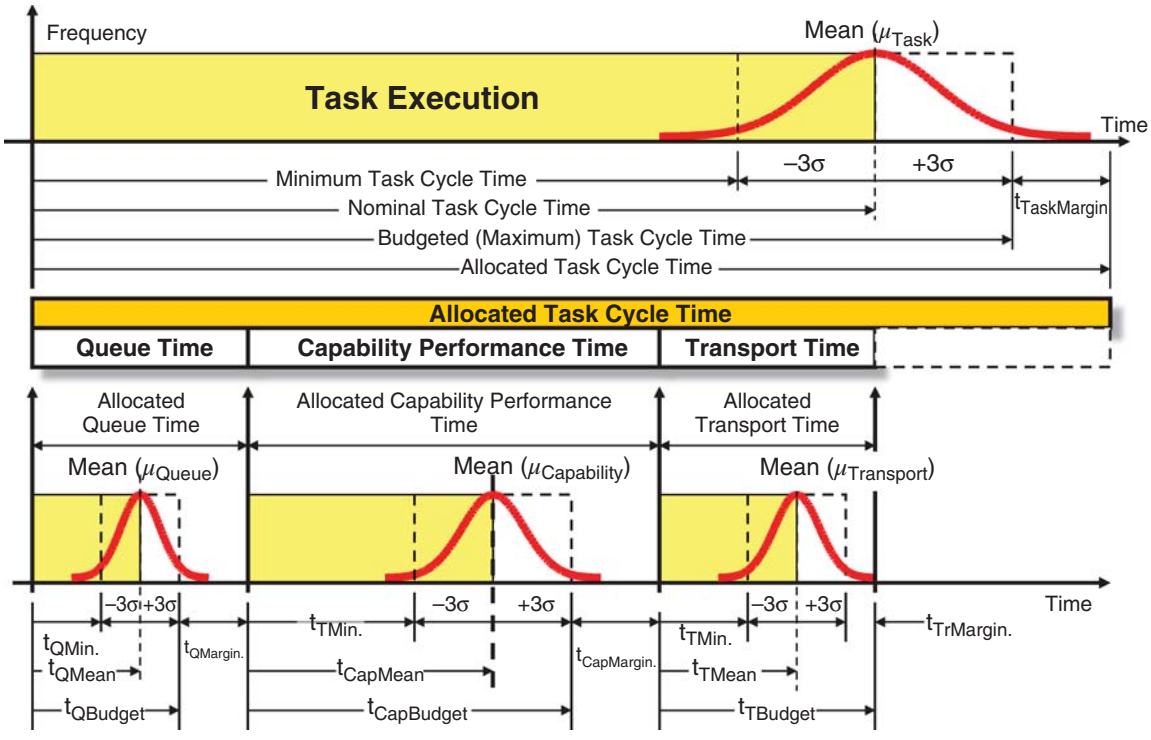


Figure 31.9 Statistical Analysis Illustrating the Variability of Queue time, Capability Performance Time, and Transport time.

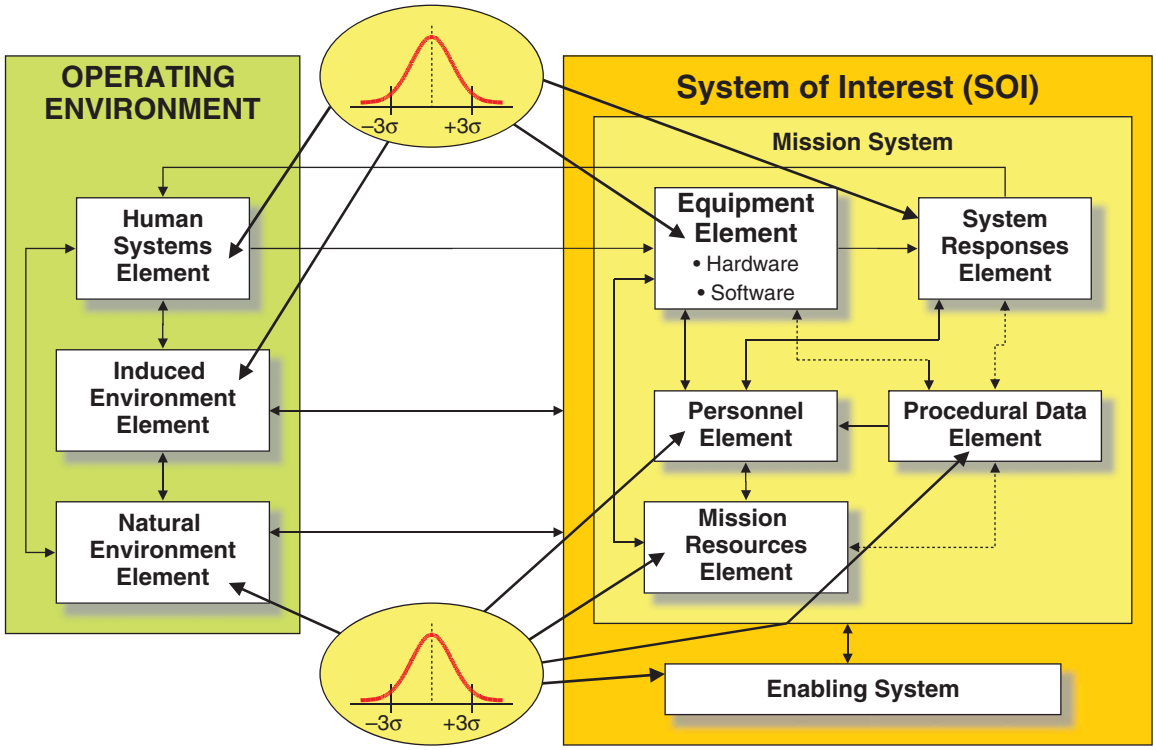


Figure 31.10 Statistical Analysis of Variations in SYSTEM Element Performance.

From a contributory performance perspective, Figure 10.12 represents the results using an Ishikawa or Fishbone Diagram.

31.3.4.1 Mathematical Approximation Alternative Our conceptual discussions of statistical performance analysis were intended to highlight key considerations for establishing and allocating performance budgets and design safety margins and analyzing data for balancing overall system performance tuning. Most people do not have the time or skills to perform the statistical analyses. For some applications, this may be acceptable and you should use the method appropriate for your application. There is an alternative method you might want to consider using, however.

Scheduling techniques such as the Program Evaluation and Review Technique (PERT) employ approximations that serve as analogs to a Gaussian (Normal) Distribution (Figure 34.3). The formula stated as follows is used:

$$\text{Expected or mean time} = \frac{t_A + 4t_B + t_C}{6} \qquad (31.1)$$

where:

t_A = Optimistic Time
t_B = Most Likely Time
t_C = Pessimistic Time

A Word of Caution 31.1

Recognize that Eq. 31.1 simply represents a "quick look" estimate as a *reasonableness* or *sanity check* and is not intended as a design practice.

31.3.4.2 Do SEs Actually Perform This Type of Analysis? Our discussion here highlights the theoretical viewpoint of task analysis. A key question is: *do Engineers and Analysts actually go to this level of detail?* In general, the answer is yes, especially in manufacturing and scheduling environments. In those environments, Statistical Process Control (SPC) is used to *minimize* process and material *variations* in the production of parts, and this translates into cost reduction. Figure 30.1 is an illustrative example.

31.3.5 Modeling and Simulation

If you develop a model of a SYSTEM or PRODUCT whereby each of the capabilities, operations, processes, and tasks is represented by a series of *sequential* and *concurrent* elements and dependencies or feedback loops, you can apply statistics to the processing time associated with each of those elements. By analyzing how each of the input varies over value ranges bounded by the $\pm 3\sigma$ points, you can determine the overall SYSTEM performance relative to a central mean.

Heading 31.4

Our discussions highlighted some basic task-oriented methods that support a variety of Systems Engineering activities. These methods can be applied to METs, system capabilities, and performance as a means of determining overall SYSTEM performance (Figure 10.12). Through derivation, allocation, and flow down of SPS or EDS requirements, developers can establish the appropriate performance budgets and design safety margins for lower level ENTITIES.

31.4 REAL-TIME CONTROL AND FRAME-BASED SYSTEMS

Some systems operate as real-time, closed loop, feedback systems. Others are multi-tasking whereby they have to serve multiple processing tasks on a priority basis. Let's explore each of these types.

31.4.1 Real-Time, Closed Loop Feedback Systems

Electronic, mechanical, and electro mechanical systems include real-time, closed loop, feedback control systems that condition or process input data and produce an output, which is sampled and summed with the input as *negative* feedback. Figure 10.1 is a generalized example. Otherwise, the SYSTEM might *overcompensate* and go *unstable* while attempting to *regain control*. The challenge for SEs is to determine and allocate performance for the optimal feedback responses to ensure overall SYSTEM stability Principle 3.9.

31.4.2 Frame-Based System Performance

Electronic systems often employ software to accomplish cyclical data processing tasks using combinations of *open* and *closed* loop cycles. Systems of these types are referred to as *frame-based systems*.

Frame-based systems perform multitask processing via time-based blocks of time such as 30 or 60 Hz. Within each block, processing of multiple concurrent tasks is accomplished by allocating a portion of each frame to a specific task, depending on priorities. For these cases, apply performance analysis to determine the appropriate mix of concurrent task processing times.

Author's Note 31.4

One approach to frame-based system task scheduling is Rate Monotonic Analysis (RMA). Research this topic further if frame-based systems apply to your business domain.

31.5 SYSTEM PERFORMANCE OPTIMIZATION

System performance analysis serves as a valuable tool for modeling and predicting the intended interactions of the SYSTEM with its OPERATING ENVIRONMENT. The *validity* of *underlying assumptions* concerning SYSTEM optimization becomes reality when the SYSTEM or PRODUCT is powered up and operated. The challenge for SEs is to *optimize* overall SYSTEM performance (Figure 14.8) to compensate for the variability of the embedded PRODUCTS, SUBSYSTEMS, ASSEMBLIES, SUBASSEMBLIES, and PARTS and their workmanship, material composition and integrity.

31.5.1 Minimum Conditions for System Optimization

Principle 31.7

System Optimization Principle

Theoretically, a SYSTEM can only be *optimized* within its specified performance requirements and boundary conditions when most, if not all, *latent defects* such as capability deficiencies, design errors, design flaws, and weak components have been eliminated.

When the SYSTEM enters the System Integration, Test, and Evaluation (SITE) Phase, *latent defects* such as system deficiencies, design flaws, and design errors often consume a significant amount of the SE's time. The challenge is getting the SYSTEM to a *state of equilibrium and stability* that can best be described as compliant with the SPS.

Once the SYSTEM is in a state of Normal Operation with no outstanding *latent defects* or deficiencies, there may be a need to tweak performance to achieve *optimum* performance. Let's reiterate the last point: you must correct all major deficiencies *before* you can attempt to *optimize* SYSTEM performance in a specific area. Exceptions include minor items that are not SYSTEM performance drivers. Consider the following example:

Example 31.7

You have been assigned a task to develop a fuel-efficient automobile and are attempting to optimize road performance using a test track. If the fuel flow has a deficiency, *can you optimize overall system performance?* Absolutely not! Does having a taillight out impact fuel efficiency performance? No, it's not a contributory performance driver to SYSTEM performance. It may, however, impact driver and passenger safety, especially in inclement weather or low-light driving conditions such as dawn, dusk, or night.

31.5.2 Pareto-Driven Performance Improvements

Principle 31.8

Pareto 80–20 Principle

On average, 80% of SYSTEM/ENTITY performance issues are attributable to 20% of the SYSTEM tasks.

There are a number of ways to *optimize* system performance, some of which can be very time-consuming. For many systems, however, there is very little time to *optimize* system performance prior to delivery, simply because the System Development schedule has been *consumed* with correcting system deficiencies and latent defects.

When you investigate options for where to focus System performance *optimization* efforts, one approach employs the Pareto 80/20 rule originally conceived by Vilfredo Pareto (1896), economist and scientific sociology theorist, based on observations of humans, nature, and so forth. Later, Dr. Joseph Juran's research in 1941 lead to a rediscovery of Pareto's work. Juran conducted independent research and in recognition of Pareto's work, the concept later become known as Pareto Principle (Wikipedia, 2015). For SE, example variants include the following:

- 20% of a System's defects cause 80% of the problems.
- 20% of System tasks consume 80% of the processing time.
- 20% of Project Management tasks consume 80% of the available resources, and so forth.

If you accept this premise, the key is to identify which processing tasks represent the 20% and focus performance analysis efforts on *maximizing* or *minimizing* their impact. So, employ *diagnostic* tools to understand how each item is performing as well as interface latencies between Subsystems, Assemblies, HWCIs, and CSCIs (Chapter 16).

Today, electronic instrumentation devices and diagnostic software are available to identify and track processing tasks that consume System resources and performance. Plan from the beginning of System Development how these devices and software can be employed to identify and *prioritize* System processing task performance and *optimize* it.

System performance *optimization* begins on Day #1 After Contract Award (ACA) via:

- System performance requirements allocations.
- Plans for "test points" to monitor performance after the System has been fully integrated.
- Focused attention on reducing System *latent defects* such as capability deficiencies, design flaws, and design errors.

Author's Note 31.5

Increasing Cost-to-Correct Defects by Development Phase

As a reminder, the cost-to-correct defects increases almost exponentially (Table 13.1) as you progress from Contract Award through System Development.

31.6 SYSTEM ANALYSIS REPORTING

As a disciplined professional, document the results of each an analysis. For Engineering tasks that involve simple assessments, always document the results, even informally, in an electronic Engineering Notebook on-line or portable device. For tasks that require a more formal, structured approach, you may be expected to deliver a formal report. A common question many SEs have is: *how do you structure an analytical report*? You should consult your contract or task for specific requirements for analytical or technical reports.

- If the contract does not require a specific a structure, consult your Enterprise command media.
- If your Enterprise command media does not provide guidance, consider using the Engineering Report outline or tailoring a version of the example provided in Table 30.1.

31.7 CHAPTER SUMMARY

Our discussions of system performance analysis, budgets, and margins practices provided an overview of key SE design considerations. We described the basic process and methods for:

- Allocating MOPs to lower levels.
- Understanding the impact of statistical variability on task-based processing.
- Publishing analysis results in a report using a recommended outline.

We also offered an approach for estimating task processing time durations and introduced the concept of RMA for frame-based processing.

Finally, from an SE perspective, we discussed the performance variability of System Elements (Chapter 8) that must be factored into performance allocations.

- Each SPS or EDS MOP should be partitioned into a "Design-To" MOP and a performance *safety margin* MOP.
- Each project must provide guidance for establishing *safety margins* for all System Elements.

31.8 CHAPTER EXERCISES

31.8.1 Level 1: Chapter Knowledge Exercises

1. What is a cycle time?
2. What is a queue time?
3. What is a transport time?

4. What is a processing time?
5. What is a performance budget?
6. If you are confronted with analyzing SYSTEM throughput performance across a chain of Processing Tasks, what are the key performance attributes associated with each task and between tasks that you should consider?
7. How do you establish and track performance budgets at the SPS and EDS levels?
8. What is a design safety margin?
9. What are the challenges of establishing performance budgets and margins and how should you document and control them?
10. Assume you have established SYSTEM performance budgets and margins. Describe how would you allocate them to PRODUCTS, SUBSYSTEMS, ASSEMBLIES, and so on, verify traceability, and maintain intellectual control (Principle 1.3) and integrity of the technical program to manage their status?

31.8.2 Level 2: Chapter Knowledge Application Exercises

Refer to www.wiley.com/go/systemengineeringanalysis2e.

31.9 REFERENCES

Grady, Jeffrey O. (2006), *System Requirements Analysis*, Academic Press: Burlington, MA.

Pareto, Vilfredo (1896), *"Cours d'économie politique,"*, Lausanne, Switzerland: University of Lausanne.

Wikipedia (2015), *Joseph M. Juran webpage*, San Francisco, CA: Wikimedia Foundation, Inc. Retrieved on 6/20/15 from: https://en.wikipedia.org/wiki/Joseph_M._Juran

32

TRADE STUDY ANALYSIS OF ALTERNATIVES (AoA)

The Engineering and development of systems require SEs to identify and work through a broad range of Critical Operational and Technical Issue (COI/CTI) decisions. These issues range from the miniscule to the complex, requiring in-depth analyses supported by models, simulations, and prototypes. Adding to the complexity, many of these decisions are interdependent. *How can SEs effectively work through these issues and keep the program on schedule*?

The following section answers this question with a discussion of Trade Study Analysis of Alternatives (AoA) (Chapter 32). We:

- Explore what a trade study is and how it relates to a constrained trade space. Define how to charter a trade study or Trade Study Team.
- Provide a methodology for conducting a trade study.
- Define a reference format for a Trade Study Report (TSR).
- Suggest recommendations for presenting trade study results.
- Investigate technical challenges, issues, and risks related to performing trade studies.

We conclude with a discussion of Enterprise trade study issues that SEs should be prepared to address.

32.1 DEFINITIONS OF KEY TERMS

- **Conclusion** Refer to Chapter 30 Definition of Key Terms.
- **Decision Authority** An internal or external individual or team from project, functional, or executive management that has the authority to initiate an activity, provide the required resources, and implement decisions.
- **Decision Criteria** Attributes of a decision factor. For example, if a key decision factor is maintainability, decision criteria might include component modularity, interchangeability, accessibility, test points, and so forth.
- **Decision Factor** A key attribute of a system, as viewed by Stakeholders—Users and End Users—that has a major influence on or contribution to a requirement, capability, COI or CTI being evaluated. Decision factor examples include elements of technical performance, cost, schedule, technology, and support.
- **Finding** Refer to Chapter 30 Definition of Key Terms.
- **Figure of Merit (FOM)** A *unitless* quantity that represents the results of a *multi-variant* evaluation based on a comparison scoring of the subject's characteristics relative to predefined set weighted decision factors.
- **Recommendation** Refer to Chapter 30 Definition of Key Terms.
- **Sensitivity Analysis** A numerical validation approach that *reduces* the weighted value of discrete Decision Factor by an arbitrary amount such as 10% to test the *robustness* of a trade study decision to remain unchanged
- **Trade Off** "Selection among alternatives with the intent of obtaining the optimal, achievable system

System Engineering Analysis, Design, and Development: Concepts, Principles, and Practices, Second Edition. Charles S. Wasson.

Companion Website: www.wiley.com/go/systemengineeringanalysis2e

configuration. Often a decision is made to opt for less of one parameter in order to achieve a more favorable overall system result." (DAU, 2012, p. B-232)

- **Trade Space** An area of evaluation or interest bounded by a *prescribed* set of boundary constraints that serve to scope the set of acceptable candidate alternatives, options, or choices for further trade study investigation and analysis.
- **Trade Study** Analysis conducted to methodically evaluate a series of design alternatives and recommend the preferred feasible solution(s) that enhance the value and performance of the overall system and/or functions. Each assessment is taken to an appropriate level of detail that allows differentiation between alternatives. (FAA, 2006, Vol. 3, p. B-13)
- **Trade Study Charter** A document issued by a decision authority to conduct of a trade study. A well-defined charter should include COI/CTI Problem Statement; trade study objectives, deliverable(s), and schedule; Stakeholders, as appropriate; the trade study individual or team; and resources.
- **Trade Study Report (TSR)** A document prepared by an individual or team that captures and presents key considerations such as objectives, candidate options, and methodology used to recommend a prioritized set of options or course of action to resolve a critical operational or technical issue.
- **Utility Function** A *linear* or *nonlinear* characteristic profile or value scale that represents the level of importance different stakeholders place on a SYSTEM or ENTITY attribute or capability relative to constraints established by a specification.
- **Utility Space** An area of interest bounded by *minimum* and/or *maximum* performance criteria established by a specification or analysis and a *degree of utility* within the performance range.
- **Viable Alternative** A candidate approach that is qualified for consideration based on its technical, cost, schedule, support, and risk level merits relative to a set of specification and boundary condition requirements.

32.2 INTRODUCTION TO MULTIVARIATE ANALYSIS OF ALTERNATIVES (AoA)

Principle 32.1

Analyses Versus Trade Studies Principle

Analyses investigate, analyze, and document a specific condition, state, circumstances, or "cause-and effect" relationships; Trade studies formulate, analyze and evaluate viable candidate alternatives and propose prioritized recommendations for selection.

Marketers in an attempt to acquire business often express a variety of terms to Acquirers and Users that communicate lofty goals to achieve. Terms include *best solution, optimal solution, preferred solution, solution of choice, ideal solution*. This leads to two key questions:

- *How do we structure a course of action to know when we have achieved a "best solution"*?
- *What is a "preferred" solution? Preferred by whom?*

These questions emphasize the importance of structuring a course of action that enables us to establish at a consensus of what these terms mean to a Stakeholder - Users and End Users - community. Recognize that stakeholders have differing priorities, motivations, and opinions concerning what a "best" or "preferred" solution is; at most a consensus can only be *optimal*. With budgetary, schedule, and technology constraints, you can only arrive at one solution. Even that *solution may not be optimal*.

Author's Note 32.1

Understanding What a Consensus Represents

Remember—a consensus represents a decision everyone agrees is one they can support. It does not mean that they like it; however, as a collective decision, they can "live with it" and will *proactively* support its implementation. The most important aspect is the opportunity to have "due process" to openly and objectively explore, debate, and investigate the merits of viable candidate solutions. The intent is to filter out, prioritize the top two or three candidates, and recommend a solution that has the highest Figure of Merit (FOM) scoring.

A low-cost mechanism that provides objective evidence for selecting an *optimal* solution is an AoA or *multi-variate analysis* resulting in a TSR. The trade study may require development or use of tools such as User surveys, test markets, models and simulations, rapid prototypes, brass boards, demonstrations, tests, and so on. Where budgets permit, risk is a driver, and schedule is a constraint; an Enterprise may decide to: (1) pursue development of the top candidate solution as their primary strategy and (2) conduct concurrent development of the next highest candidate up to a point where risks of the top candidate have been resolved.

To better understand how trade studies establish a course of action to achieve lofty goals, let's begin by establishing the objectives of a trade study.

32.2.1 Trade Study Objectives

The objectives of a trade study are to:

- Investigate a COI or CTI.

- Formulate *viable candidate solutions* for evaluation, scoring, and selection.
- Explore the fact-based merits of candidate solutions relative to Decision Factors and Criteria derived from Stakeholder requirements via the contract, Statement of Objectives (SOO), specification requirements, Stakeholder—User and End User—interviews, cost, or schedules.
- Prioritize solution recommendations for decision authorities.

32.2.2 Typical Trade Study Decision Areas

The hierarchical decomposition of a system into entities at multiple levels of abstraction, bounding of performance requirements, and selection of physical components requires a multitude of technical and programmatic decisions. Many of these decisions are driven by the Specification Assessment Checklist (Chapter 23 Section 23.3) and resource constraints.

If we analyze the sequences of many technical decisions, categories of trade study areas emerge across numerous system, product, or service domains. Although every system, product, or service is unique and has to be evaluated on its own merits, most system decisions can be characterized using Figure 32.1. Specifically, the Typical Trade Decision vertical box in the center of the graphic depicts the top-down chain of decisions common to most entities regardless of system level of abstraction.

Beginning at the top of the center box, the decision sequences include the following:

- Architecture trades.
- Interface trades including human–machine interfaces.
- EQUIPMENT - HARDWARE/SOFTWARE (HW/SW) trades.
- Commercial Off-the-Shelf (COTS) versus Non-Developmental Item (NDI) versus new development trades.
- HW/SW component composition trades.
- HW/SW process and methods trades.
- HW/SW integration and verification trades.

This chain of decisions applies to entities at higher levels of abstraction from SYSTEM to PRODUCT to SUBSYSTEM and

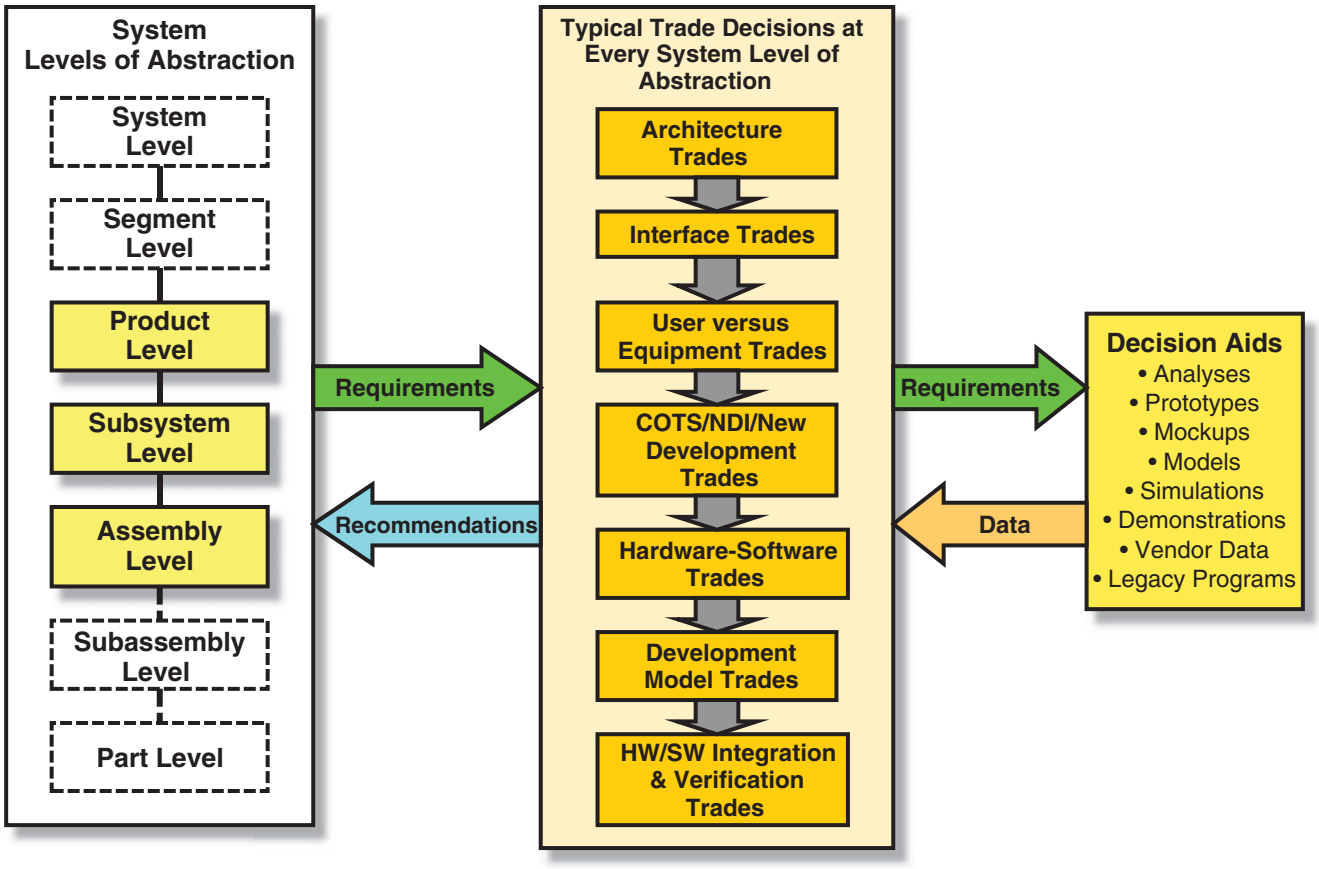


Figure 32.1 Typical Trade Study Decision Sequences Applicable to Every System Level of Abstraction

so forth, as illustrated Figure 32.1. SEs employ decision aids to support these decisions, such as analyses, prototypes, mock-ups, models, simulations, technology demonstrations, vendor data, and their own experience, as illustrated by the box shown at the right-hand side. The question is: *how are the sequences of decisions accomplished*?

32.2.3 Trade Studies Address COIs/CTIs

The sequence of trade study decisions represents a basic "line of questioning" intended to facilitate the SE design solution of each entity. What System Acquirers want to know and see articulated in technical proposals is: *does the System Developer have a valid, least cost strategy concerning how they propose developing a System Development solution*? Let's explore an example of such a technical decision strategy.

Figure 32.2 is an example of a decision sequence. Tailor it to best fit the needs of your Enterprise or organization.

- What type of SYSTEM or ENTITY architecture enables the User to best leverage the required System, Product, or Service capabilities and levels of performance?
- Given an architecture decision, what is the best approach to establish low-risk, *interoperable* interfaces, or interfaces to minimize susceptibility or vulnerability to external system threats?
- How should we implement the architecture, interfaces, capabilities, and levels of performance? As EQUIPMENT tasks? PERSONNEL tasks? Or a combination of these (Figures 10.15, 10.16, and 24.14)?
- What development approach represents a solution that minimizes cost, schedule, and technical risk? Commercial-Off-the-Shelf (COTS)? NDI? Acquirer Furnished Equipment (AFE)? New development? A combination of COTS, NDI, Acquirer Furnished Property (AFE), and new development?
- Given the development approach, what is the appropriate mixture of HARDWARE and SOFTWARE, as applicable?
- For HARDWARE and SOFTWARE, what Development Model(s) (Chapter 15) should be employed to design and develop the entity?
- Once the HARDWARE and SOFTWARE components are developed, how should they be integrated and verified to demonstrate full compliance?

We answer these questions through a series of technical decisions. A trade study, as an AoA, provides the basis for comparative evaluation of viable candidate options based on a predefined set of decision factors and criteria to make an informed decision. Observe that if the trade-offs at any level reach a point in which requirements - technical, cost,

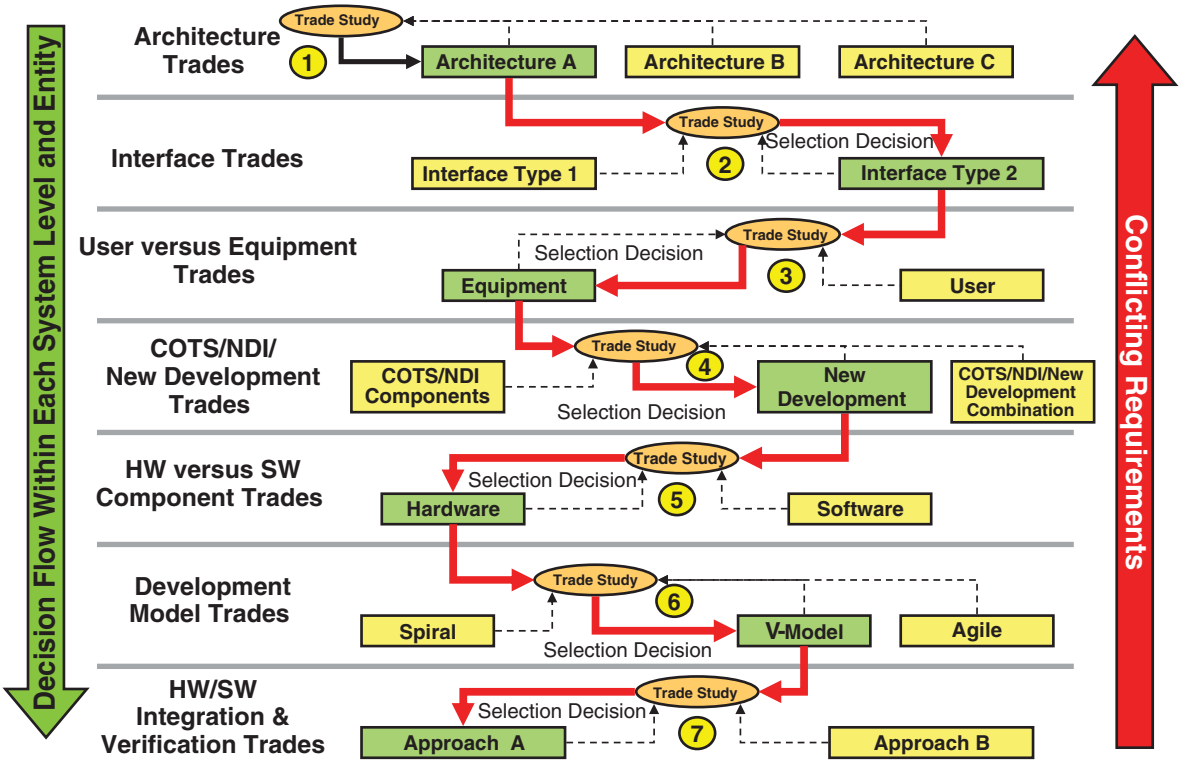


Figure 32.2 Example Trade Study Decision Tree - Key Technical Decisions and Sequences.

schedule, technology, or risk - conflict, the decision requires elevation to higher levels for resolution as illustrated in Figure 15.3.

32.2.4 Trade Study Decision Dependencies and Sequences

Principle 32.2

Key Decisions Tree Principle

Always establish, communicate, and implement the decision tree that expresses the sequences and options for key technical decisions and synchronize it with the project schedule milestones.

Technical programs usually have a number of COIs/CTIs that must be resolved to enable progression to the next decision in the chain of decisions. If we analyze the sequences of these issues, we discover that the process of decision-making resembles a tree structure over time. Thus, the branches of the structure represent decision dependencies as illustrated in Figure 32.2.

During the proposal phase of a program, the Proposal Team conducts preliminary trade studies to rough out key design decisions and issues that require more detailed attention After Contract Award (ACA). These studies enable us to understand the COI or CTI to be resolved ACA. In addition, in-depth studies of *viable* candidates provide a level of confidence in the cost estimate, schedule, and risk leading to an understanding of the problem and solution spaces.

Author's Note 32.2

Trade Study Decision Tree

Depending on the type of project/contract, a trade study tree such as the one shown in Figure 32.2 is often helpful to demonstrate to a customer that you have a logical decision path toward a timely System Design Solution.

32.2.5 System Architectural Element Trade Studies

As noted earlier, System Acquirers expect offerors to identify the COI/CTI decisions that need to be made, their sequences, and dependencies. This requires insightful knowledge about the system, product, or service they plan to develop. One method is to establish a framework of decision areas for trade studies. Figure 32.3 provides an example of trade study areas for consideration concerning a generic Vehicle System.

We establish a simple hierarchy diagram consisting of the vehicle's subsystems. This enables us to identify CTI/COI *areas of concern*. As the proposed System Architecture development expands, lower levels can be evaluated for more refinement in areas of concern. Each of these elements involves a series of technical decisions that form the basis for subsequent, lower level decisions. In addition, decisions made in one element as part of the SE process may have an impact on one or more other elements. Consider the following example:

Example 32.1

Constraint Drivers for Decision Factors and Criteria

Vehicle cargo/payload constraints influence decision factors and criteria used in the

- Propulsion System trades involving technology and power.
- Energy Transfer System involving torque, heat, reliability, and so on.

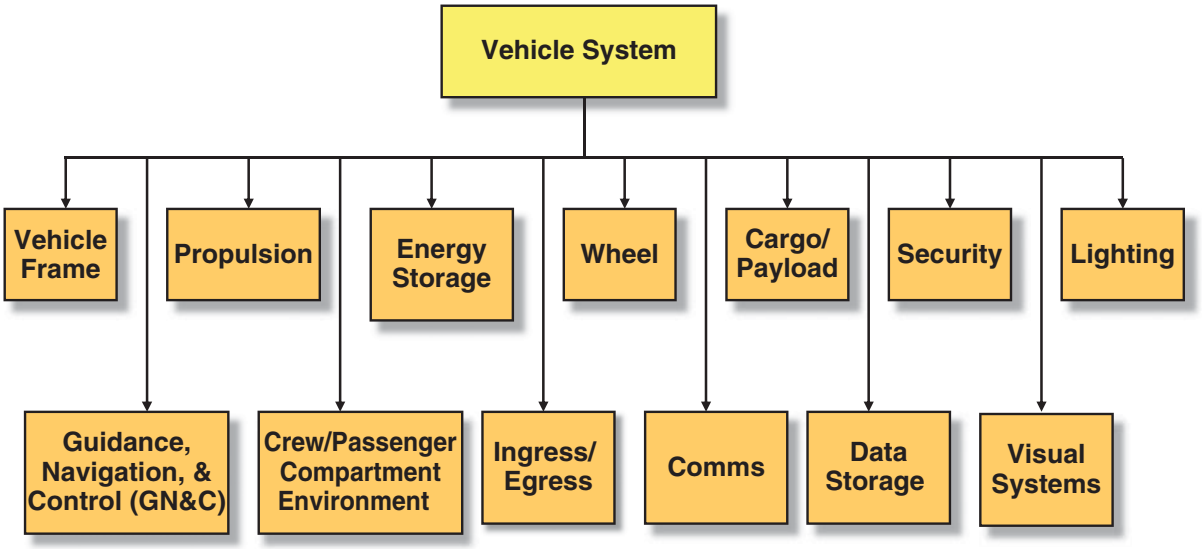


Figure 32.3 Trade Study Areas Example - Mobile Vehicle

- Vehicle Frame trades involving size, strength, materials, and durability.
- Wheel System trades involving type and braking.

32.2.6 Understanding the Prescribed Trade Space

Despite the appearance that trade study efforts have the freedom to explore and evaluate options, there are often limiting constraints. These constraints bound the area of study, investigation, or interest. In effect, the bounded area scopes what is referred to as the *trade space*.

32.2.6.1 The Trade Space We illustrate the basic trade space shown in Figure 32.4. Let's assume that the System Performance Specification (SPS) identifies specific Measures of Performance (MOPs) that can be aggregated into a *minimum* acceptable level of performance represented by a FOM as noted by the Minimal Acceptable Performance—vertical line. Marketing analyses or the Acquirer's proposal requirements indicate there is a Cost-Per-Unit Ceiling as illustrated by the horizontal line. A Trade Space requiring further study is bounded by the intersection of the Minimum Acceptable Performance, Cost-Per-Unit Ceiling, and Cost–Performance Curve.

Now, suppose that we identify viable Candidate Solutions 1, 2, 3, and 4 for our trade study. We construct the Cost–Performance Curve. To ensure a level of objectivity, we normalize the Cost-Per-Unit Ceiling to the Acquirer *maximum* requirement. We plot cost and relative performance of each of the Candidate Solutions 1, 2, 3, and 4 on the Cost–Performance Curve.

By *inspection* and *comparison* of plotted cost and technical performance relative to required performance, we make the following decisions:

- Solution 1 is *cost compliant*—below the Cost-Per-Unit Ceiling—but *technically non-compliant*—below the minimum technical performance threshold.
- Solution 4 is *technically compliant*—above the minimum technical performance threshold—but cost *non-compliant*—exceeds the Cost-Per-Unit Ceiling.

When this occurs, the TSR documents that Solutions 1 and 4 were evaluated, determined by analysis to be *noncompliant* with the trade space decision criteria, and were subsequently eliminated from consideration.

Following elimination of Solutions 1 and 4, Solutions 2 and 3 undergo further analysis to thoroughly evaluate and score other considerations such as organizational risk.

32.2.6.2 Optimal Solution Selection The previous discussion illustrates the basic concept of a two-dimensional trade space. A trade space, however, is multi-variant—that is, multi-dimensional. For this reason, it is more aptly described as a *multi-variate* trade volume that encompasses technical, technology, life cycle cost, schedule, support, and risk decision factors.

We can illustrate the trade volume using the graphic shown in Figure 32.5. To keep the diagram simple, we constrain our discussion to a three-dimensional model representing the convolution of technical, cost, and schedule factors. Let's explore each dimension represented by the trade volume.

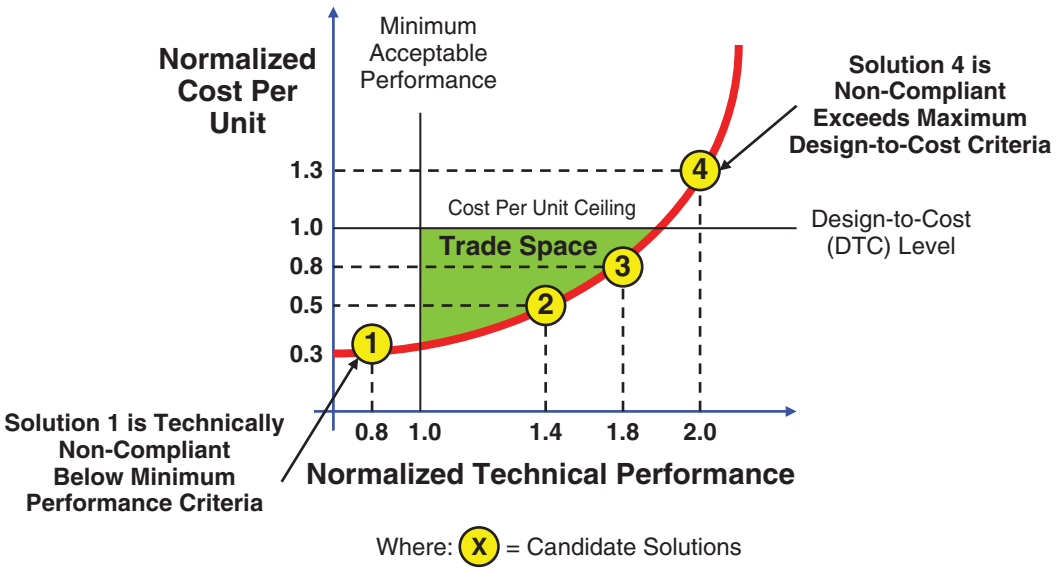


Figure 32.4 Example Illustrating a Trade Space with Viable Candidates and Constraint Boundaries

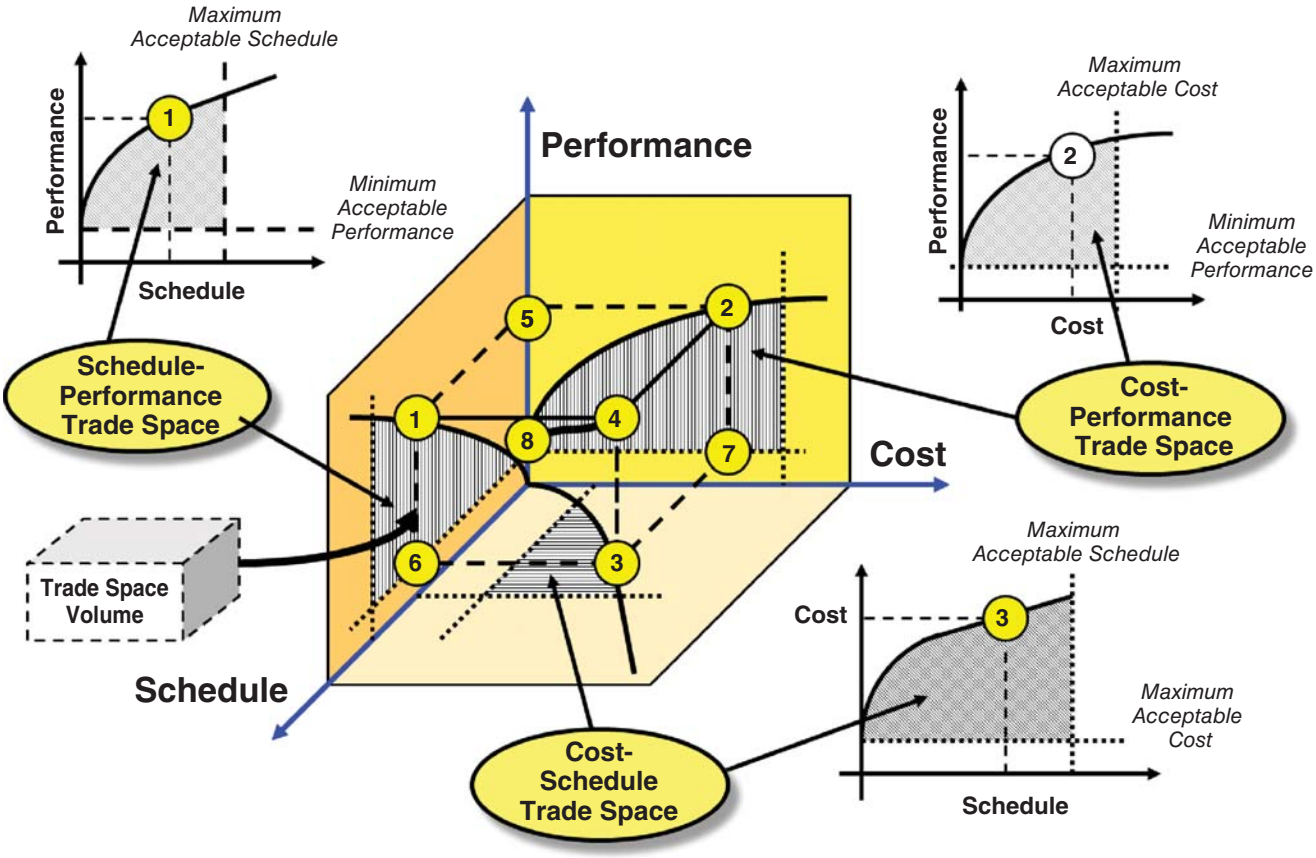


Figure 32.5 Example - 3-D Trade Space Illustration

- **Performance–Schedule Trade Space** The graphic in the upper left-hand corner of the diagram represents the Performance *versus* Schedule trade space. Identifier (1) marks the location of the selected Performance versus Schedule solution.
- **Performance–Cost Trade Space** The graphic in the upper right-hand corner represents the Performance versus Cost Trade Space. Identifier (2) marks the location of the selected Performance versus Cost solution.
- **Cost–Schedule Trade Space** The graphic in the lower right-hand corner of the diagram represents the Cost–Schedule trade space. Identifier (3) marks the location of the selected Cost Versus Schedule solution.

If we convolve these trade spaces and their boundary constraints into a three-dimensional model, the cube in the center of the diagram results.

The *optimal solution* (4) is represented by the intersection of orthogonal lines in their respective planes. Conceptually, the optimal solution would lie on a curve that represents the convolution of the Performance–Schedule, Performance–Cost, and Cost–Schedule curves. Since each plane includes a restricted trade space, the integration of these planes into a three-dimensional model results in a trade space volume.

Given this introduction, we are now ready to proceed to how trade studies are conducted.

Heading 32.1

32.3 CHARTERING A TRADE STUDY

Trade Studies can be conducted *informally* by Key Stakeholders such as Engineers and Analysts as part of their normal jobs or *formally* chartered by a decision authority such as project, functional, or executive management. Since: (1) trade studies have an impact on the project and (2) all Stakeholders want to ensure that trade study results will be beneficial and not a waste of time, let's pursue the formal chartering approach to reduce technical cost and schedule risk.

Trade studies consist of *highly iterative* steps to analyze a COI/CTI area of concern and respond with a set of *prioritized* recommendations for selection by the decision authority.

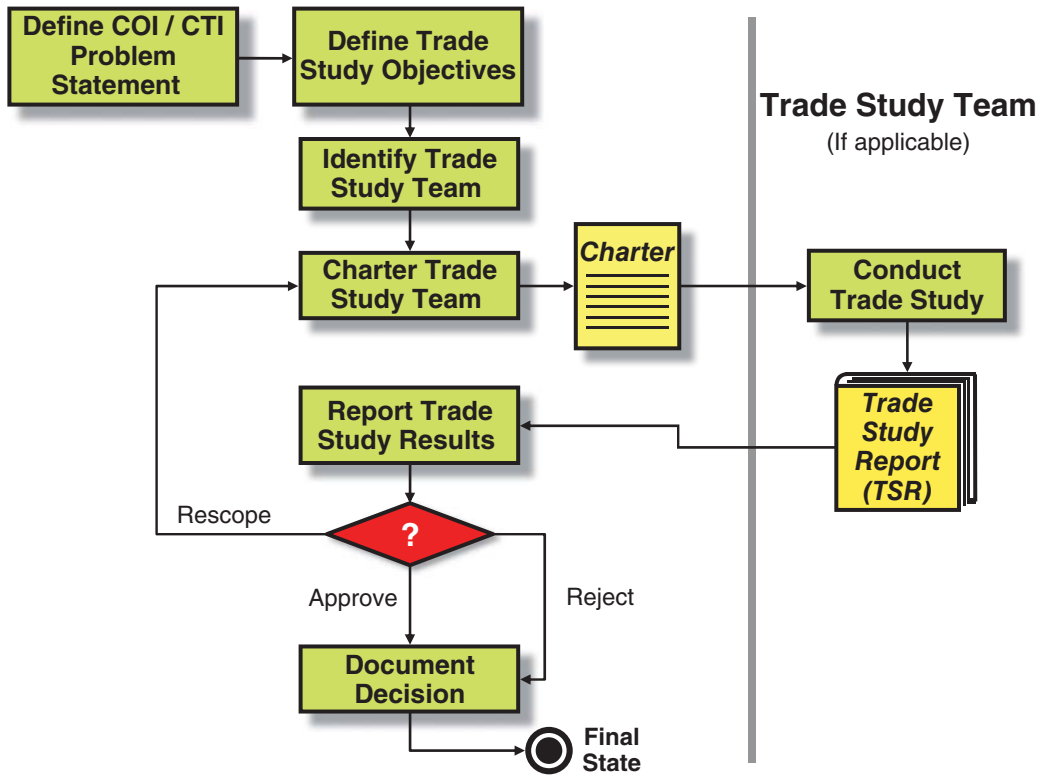


Figure 32.6 Example - Process Workflow for Chartering a Trade Study Team

Note that the decision authority may be an individual or a team that has the authority and the resources to implement the trade study recommendations. Figure 32.6 provides an example process for chartering a team or initiating a trade study by an individual.

Please note that a well-defined charter should document the COI/CTI Problem Statement (Chapter 4), trade study objectives, deliverable, and schedule, stakeholders as appropriate, trade study individual or team; and resources.

Heading 32.2

Our preceding discussion defined the trade study chartering or initiation process. Now let's focus our attention on understanding how an individual or team establishes the methodology used to guide and conduct the trade study.

32.4 ESTABLISHING THE TRADE STUDY METHODOLOGY

Principle 32.3

Trade Study Teams Principle

A trade study team without an approved charter and methodology is prone to wander aimlessly.

Objective technical and scientific investigations require a methodology for making decisions. The methodology facilitates the development of strategy, course of action, or "roadmap" of the planned technical approach to investigate, analyze, and evaluate the viable candidate solution approaches or options. Methodologies, especially *proven* ones that have been *verified* and *validated*, keep the study effort on track and *prevent unnecessary* excursions that consume resources and yield no productive results.

There are numerous ways of establishing the trade study methodology. Figure 32.7 provides an illustrative example based on the process that follows. The process includes terms such as weighted Decision Factors and Criteria and utility functions that are introduced next.

- **Step 1: Understand Problem Statement**, its objectives and constraints concerning the COI/CTI to be resolved.
- **Step 2: Identify Users** - Identify the key Stakeholder decision makers – Users and End Users (Chapter 3).
- **Step 3: Identify & Weight Decision Factors** - Collaborate with Stakeholder Users to identify the primary Decision Factors and allocate relative weights that total of 100 points, for example. If six Decision Factors are identified and the top four drive 90% of the selection,

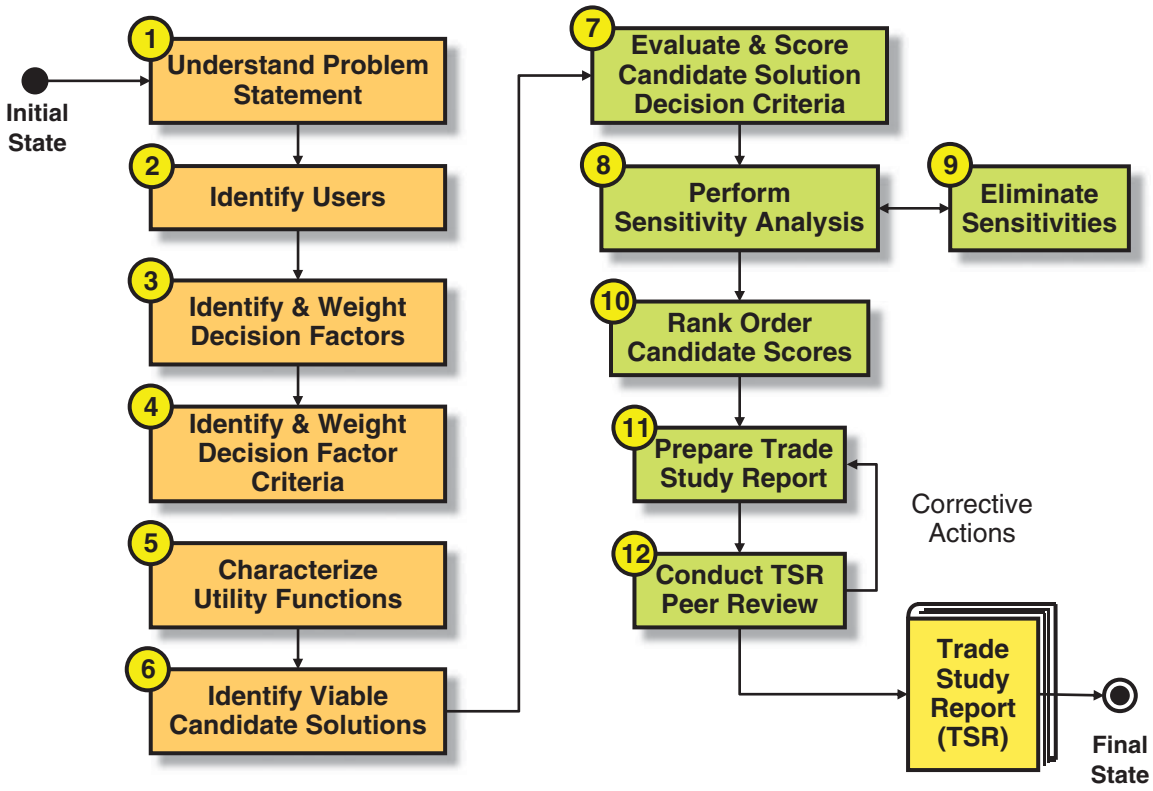


Figure 32.7 Example - Trade Study Methodology Workflow.

place the lower two scores in reserve for Sensitivity Analysis (Step 9), if required.

- **Step 4: Identify & Weight Decision Criteria** - Collaborate with Stakeholder Users to identify Decision Criteria for each Decision Factor; allocate each Decision Factor's weight - points - to its Decision Criteria.
- **Step 5: Characterize Utility Functions** - Collaborate with the Stakeholder Users to develop and characterize Utility Function profiles for Decision Criteria, if appropriate. (Figure 32.9)
- **Step 6: Identify Viable Candidate Solutions** - Identify 2 - 6 viable candidate solutions or alternatives for evaluation. To conserve valuable resources, eliminate *non-compliant* candidates (Figure 32.4). Select 2 - 4 candidates for in-depth evaluation.
- **Step 7: Evaluate & Score Candidate Solution Decision Criteria** - Analyze, evaluate, and assign Figure of Merit (FOM) scores for each candidate's Decision Criteria; total Decision Criteria FOMs into a final score for each candidate.
- **Step 8: Perform Sensitivity Analysis** to assess the *degree* of *influence* each Decision Factor has on the candidate solution FOMs. For example, individually reduce each Decision Factor's weight to assess how it impacts each candidate's final score, especially if the scores are clustered together.
- **Step 9: Eliminate Sensitivities,** if necessary, by factoring in additional Decision Factors (Step 3); reevaluate score for each new Decision Factor.
- **Step 10: Rank Order Candidate Scores** and identify the highest scoring candidate solution.
- **Step 11: Prepare Trade Study Report** using Table 32.1; identify dissenting views and rationale, is necessary.
- **Step 12: Conduct TSR Peer Review** with peers/ Subject Matter Experts (SMEs); correct deficiencies as necessary.

In general, the above-mentioned steps are straightforward. However, the selection of the Decision Factors, Criteria, and Weighting requires some additional explanation.

32.5 TRADE STUDY QUANTITATIVE APPROACHES

32.5.1 Case Study Example - Normalized Trade Study Method

Since the intent of conducting a trade study is to objectively evaluate viable candidate solutions based on the User's

TABLE 32.1 Example Trade Study Outline

Section	Section Title	Sub-Para.	Subsection Title
1.0	Introduction		
		1.1	Scope
		1.2	Authority (Team Charter)
		1.3	Trade Study Team Members
		1.4	Acronyms and Abbreviations
		1.5	Definitions of Key Terms
2.0	Referenced Documents		
		2.1	System Acquirer Documents
		2.2	User Documents
		2.3	System Developer (role) Documents
		2.4	Vendor Documents
		2.5	Specifications and Standards
		2.6	Other Documents
3.0	Executive Summary		
		3.1	Trade Study Objective(s)
		3.2	Trade Space Boundaries
		3.3	Findings
		3.4	Conclusions
		3.5	Recommendations
		3.6	Other Viewpoints
		3.7	Selection Risks and Impacts
4.0	Trace Study Methodology		
		4.1	Step 1
		4.2	Step 2
		4._	
		4.*Z*	Step z
5.0	Decision Factors, Criteria, and Weights		
		5.1	Selection of Decision Factors and Criteria
		5.2	Selection of Weights for Decision Factors or Criteria
		5.3	Selection of Criteria Utility Functions
6.0	Evaluation and AoA		
		6.1	Option A
		6.2	Option B
		6.3	Option x
7.0	Recommendations		
Appendix			
	Appendix A		Data Item 1
	Appendix B		Appendix B Data Item 2
	Et al		

operational needs and priorities, we need to establish a *scoring approach* and *method* that will enable us to:

1. Evaluate the set of candidates using a common set of decision factors and criteria.
2. Evaluate each candidate independently.
3. Compare the final results.

To solve this challenge, SEs establish *decision factors*, *decision factor criteria*, and weights for each. Let's explore each of these. Figure 32.8 provides an oversimplified graphical approach to support our discussion. The example presented has *pros* and *cons*. As such, it will provide a backdrop for a better understanding of a better approach.

For this case study, let's assume the objective is to evaluate and select from a set of viable alternatives a ground-based vehicle for carrying cargo. The trade study will serve as a basis to perform an AoA. Let's apply the 10-step methodology.

32.5.2 Identify Key Decision Factors

Any type of technical, managerial, or other decision is driven by a set of Stakeholder—User and End User—priorities that may be externally driven, internally derived, or both. On inspection, these priorities can best be categorized as Decision Factors that have a relative importance to the Stakeholders.

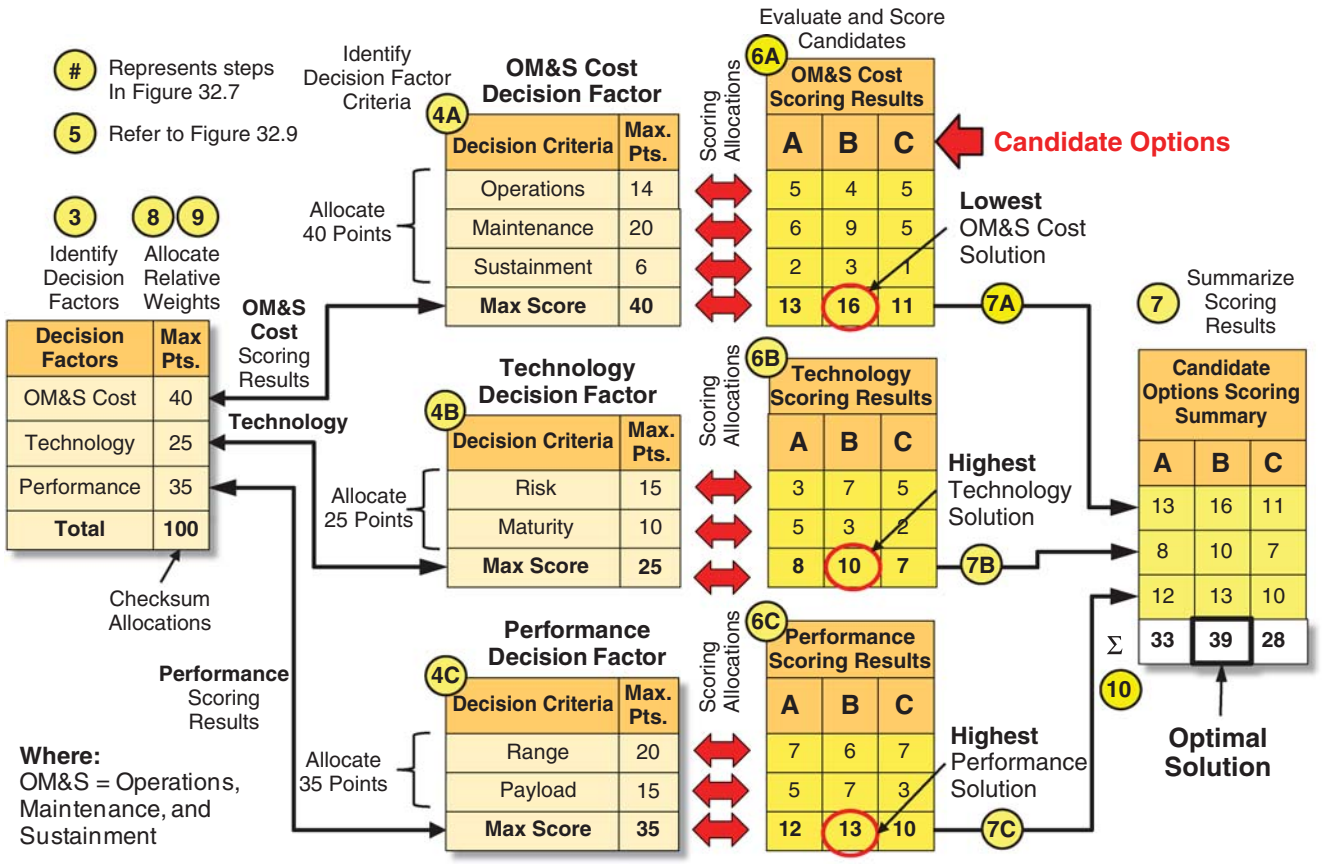


Figure 32.8 Trade Study Example—Normalized Decision Factors and Criteria Approach

Referring to Figure 32.8, identify the key Decision Factors that drive the decision. Objectively, the Decision Factors – OM&S cost, Technology, and Performance - should be derived from collaborative interviews with the Stakeholders—Users and End Users—and their weighted priorities for each Decision Factor.

For the Normalized Approach, we establish a *maximum* weighting of 100 points and allocate the total number of points across each of the Decision factors based on User weight priorities. In the example, a consensus of Stakeholders reveals that the key Decision Factors and priorities are as follows:

- Priority #1—Operations, Maintenance, and Sustainment (OM&S) Cost – 40 points
- Priority #2—Technology – 25 points
- Priority #3—Performance – 35 points

32.5.2.1 Pairwise Testing Comparison Method Often, SEs will ask Stakeholders to sort Decision Factors top-down in terms of priority or level of importance. Stakeholders sometimes have difficulty sorting a large list of four to six factors, especially if they have comparable importance. If this occurs, one solution may be to break the problem down into smaller pieces using a Pairwise Comparison Method.

The Pairwise Comparison Method simply asks the evaluator to compare two options with each other—A to B, A to C, B to C, and so forth. For each comparison, pick one. Then, rank order the results.

32.5.2.2 Brainstorming Method Another approach, Brainstorming, convenes Stakeholders in a meeting to identify Decision Factors using a two-pass approach.

- First Pass—Stakeholders brainstorm a list of factors in an open forum without critique.
- Second Pass—A facilitator helps the Stakeholders reconcile the list in terms of Primary Factors versus Subordinate Factors and achieve a consensus.

Then, using a Nominal Grouping Technique (NGT), each Stakeholder is asked to rank order the Decision factors or Criteria via a balloting process. The results are then summarized and presented for final discussion and a consensus decision.

32.5.3 Allocate Weights to Decision Factors

Once the key Decision Factors are identified, we allocate weights to each one based on their relative importance to the Stakeholders. For the Normalized Method, we collaborate with the Stakeholders and allocate 100 points across each of the Decision factors based on relative importance (Figure 32.8).

32.5.4 Checksum Decision Factor Relative Weights

We follow-up with a check-sum of the weight allocations as verification.

32.5.5 Identify Decision Criteria for Each Decision Factor

Once the Decision Factors are established, collaborate with the Stakeholders to identify Decision Criteria that are key contributors to each Decision Factor. For example, assume collaboration with the Stakeholders identifies the following Decision Factor and Criteria:

- OM&S Cost Decision Factor Criteria (40 points)
  - Operations
  - Maintenance
  - Sustainment
- Technology Decision Factor Criteria (25 points)
  - Maturity
  - Risk
- Performance Decision Factor Criteria (35 points)
  - Range
  - Capacity

32.5.6 Allocate Decision Factor Points Across Decision Criteria

Decision Factors often have an abstract context that requires further refinement into lower level criteria. On the basis of identification of the Decision Criteria, collaborate with the Stakeholders to establish relative weights for each criterion. Using the Decision Factor weight allocations, we develop separate tables for each of the Decision Factors.

32.5.7 Evaluate and Score Candidates

Evaluate and score each of the candidates based on their respective Decision Factors and Criteria.

Author's Note 32.3

Establishing Performance Value Scales

As we address the evaluation and scoring of each candidate, observe the reduced scoring results. For example, one would naturally expect a vehicle with a higher level of performance to be allocated more points than candidates with lesser performance. However, *what about Cost or Risk*? *Does increased Cost or Risk equate to more points*? Absolutely not! Instead, they score fewer points.

32.5.7.1 Candidate Scoring for OM&S Cost Each vehicle candidate is evaluated and scored based on its OM&S Cost merits as shown in Figure 32.8. Several key points:

- The scores determined for each candidate are sometimes referred to as *raw scores*, particularly if they *have not been* scaled or normalized, or FOMs.
- Observe that Candidate B receives only 4 points for Operations Costs. Candidate B has higher costs than its peers. Therefore, it receives a lower score.
- Also observe that Candidate C has a score of 1 point for Sustainment Costs. *What if the score were actually 2 points?* This represents a shortcoming of this method that will be addressed in the Auto-Normalized Trade Study Approach introduced later.

Author's Note 32.4

Candidate Evaluation Approach

Figure 32.8 represents scoring of Candidates A–C at one time. Some choose instead to evaluate Candidate A for all Decision factors and Criteria, and then proceed to Candidate B, and finally to Candidate C. There are *pros* and *cons* to both approaches.

Pro—The *advantage* of evaluating each Candidate's Decision Factors and Criteria all at once is that it provides an objective assessment without being influenced by features of the other candidates.

Con—The disadvantage of this approach is that Candidate scores may and probably will not sum to the points allocation for the Criteria without adjusting the evaluation scores. At this point the evaluation has shifted from *objective* to *subjective*.

32.5.7.2 Candidate Scoring for Technology On completion of the OM&S Cost scoring, we proceed to the Technology scoring as shown in Figure 32.8. As a reminder, observe that *increased* Risk, which is *not desirable*, results in lower scores; *increased* Maturity results in *higher* scores.

32.5.7.3 Candidate Scoring for Performance Finally, we evaluate and score each candidate for the Performance Decision factor. Scores are shown in Figure 32.8.

32.5.8 Summarize All Scoring Results

Summarize all of the scoring results for each of the candidate. For a given Decision Factor (row), verify that the checksum of Decision Factor scores equals those allocated. Figure 32.8 provides a summary of FOMs for each candidate in this case study.

32.5.9 Sum Each Candidate's Scores

Sum each candidate's scores for all Decision Factors as verification.

Author's Note 32.5

Verification and Validation (V&V) of Trade Study Scoring Results

Summarizing and verifying the scores may appear to be a case of stating the obvious and fundamental. However, it is not uncommon for teams to go through intensive trade study exercises and make scoring errors. As an example, a report by the Inspector General revealed errors in NASA's scoring decision matrix for placement of the NASA's Space Shuttle Orbiters after retirement (NASA, 2011, pp. 2, 9, 13, and 18). After all, a simple math error could potentially result in selection of the wrong solution as further evidence of the need for V&V.

32.5.10 Assess and Select Optimal Solution Results

Despite the best of intentions to objectively differentiate the Candidate Options A, B, and C via scoring methods and arrive at a clear winner, the results may *cluster closely together*. Such is the case shown in Figure 32.8 in which Candidates A, B, and C scored 33 points, 39 points, and 28 points, respectively.

32.5.11 Summary—100 Points Trade Study Scoring Approach

The preceding Normalized Trade Study Method example establishes the basic concept for a trade study methodology. The problem with this approach is that it forces the trade study individual or team and Stakeholders to spend valuable time allocating and "adjusting" point spreads. For example:

- Should OM&S Cost receive 40 points, 35 points, 45 points, and so forth?
- Within the OM&S Cost Decision Factor, should the Maintenance Decision Criterion receive 20 of those 40 points or should it be 15 or 25?
- Should a Candidate score be 6 points of the 20 points? 5 points? 7 points?

The point here is that the Normalized Trade Study Method becomes a distraction that shifts an SE's focus on the evaluation to "pushing points around" within a boundary to ensure they sum to the allocated total.

Another point … most trade study approaches *normalize* scoring to 100 points as a comparative *benchmark*. *Why 100 points*? Whether it traces back to our educational systems or numbering systems, humans instinctively normalized scoring to 1.0 or 100 points and percentages within each. So, if OM&S Costs are allocated 40 points versus 25 points for Technology versus 35 points for Performance, the fact that all are benchmarked relative to 100 points provides insights into the levels of importance at least to that SYSTEM application and its Stakeholders.

There is a better approach to avoid "playing points games" that refocuses SEs on the objectivity of the evaluation … an Auto-Normalized Trade Study Approach, our next topic.

32.5.12 Auto-Normalized Trade Study Method

To avoid the challenges of the Normalized Trade Study Method, we exploit the powers of a spreadsheet application. Instead, of "playing points games," we simply reorient the trade study by simply asking the Stakeholders to focus on a specific Decision Factor or Decision Criteria.

- Ask each Stakeholder … "On a scale from 1 (lowest) to 10 (highest), what *level of importance* do you as a Stakeholder place on this Factor or Criteria?"
- In each instance, determine what that value's fractional proportion is relative to the sum of values for each Decision Factor or Criterion.
- Multiply the fractional portion times the Decision Factor's point allocation.

We will defer elaboration of the solution to the exercises at the end of the chapter.

The advantage of this approach is that Stakeholders:

1. Focus on one Decision Factor or Criterion at a time.
2. Apply a consistent method (1–10) that simply asks "*What is the relative importance to you*" rather than allocating X points for this and Y points for something else.

Heading 32.3

At this point, we have established a basic trade study methodology. On inspection, the methodology is straightforward. However, *how do we evaluate alternatives that have linear and nonlinear degrees of utility to the stakeholder*? This brings us to a special topic, Trade Study Utility Functions.

32.6 TRADE STUDY UTILITY OR SCORING FUNCTIONS

When scoring some Decision Factors and Criteria, the natural tendency is to do so on a linear scale such as 1–5 or 1–10. This method assumes that the User has a linear *value scale*; in many cases, it is *nonlinear*. In fact, some candidate options data have *levels of utility*. One way of addressing this issue is to employ *Utility or Scoring Functions* which may also be referred to as *Utility Curves*. The reality is that the curves represent mathematical models of Stakeholder value scales. Although one can employ math models to compute value scores, perhaps a better representation are the *Utility* or *Scoring Function Curves* as shown in Figure 32.9. The curves illustrated here are simply examples.

Observe that Utility Value scales are *normalized* to 1.0 and range from 0 to 1.0 in 0.1 increments. An alternative would be to assess the values a scale from 0 to 10 in 1.0 increments.

You may ask: *how do these curves facilitate evaluating candidates*? Consider the following example:

Example 32.2

Utility Functions and Value Scales

We noted in the previous case study that some scores are the inverse of their magnitudes. Consider the Cost versus Risk trade space. Increased Costs and Risk equate to lower scores. As a result, SEs *intuitively* and *subjectively* assign scores. However, on a scientific basis, we want to remove *subjectivity* as much as possible.

One way of removing *subjectivity* is to establish a Utility Function Curve that enables us to determine the Utility of a Criterion within specified limits. We determine the appropriate Utility Function profile (Figure 32.9 Panels 1 – 6). Then, establish decision *boundary constraints* such as Total Cost of Ownership (TCO), cost per unit, technology, or risk for Utility = 0.0 and 1.0 (Figure 32.4).

Let's assume that Candidate C in Figure 32.8 Table 6A has the *lowest* OM&S Operations Criterion cost of the three candidates (11 points); Candidate B has the *highest* cost (16 points). Based on the selected Utility Curve (Panel 1), we score the three candidates as follows:

- Candidate A – Utility Score = 0.8
- Candidate B – Utility Score = 0.4
- Candidate C – Utility Score = 0.9

We would then apply the Utility Scores to the computation in Figure 32.8 Table 6A.

Heading 32.4

The preceding discussions addressed various methods for conducting and developing trade studies. Ideally, the numerical results will yield a very distinct winner that stands above all the other candidates.

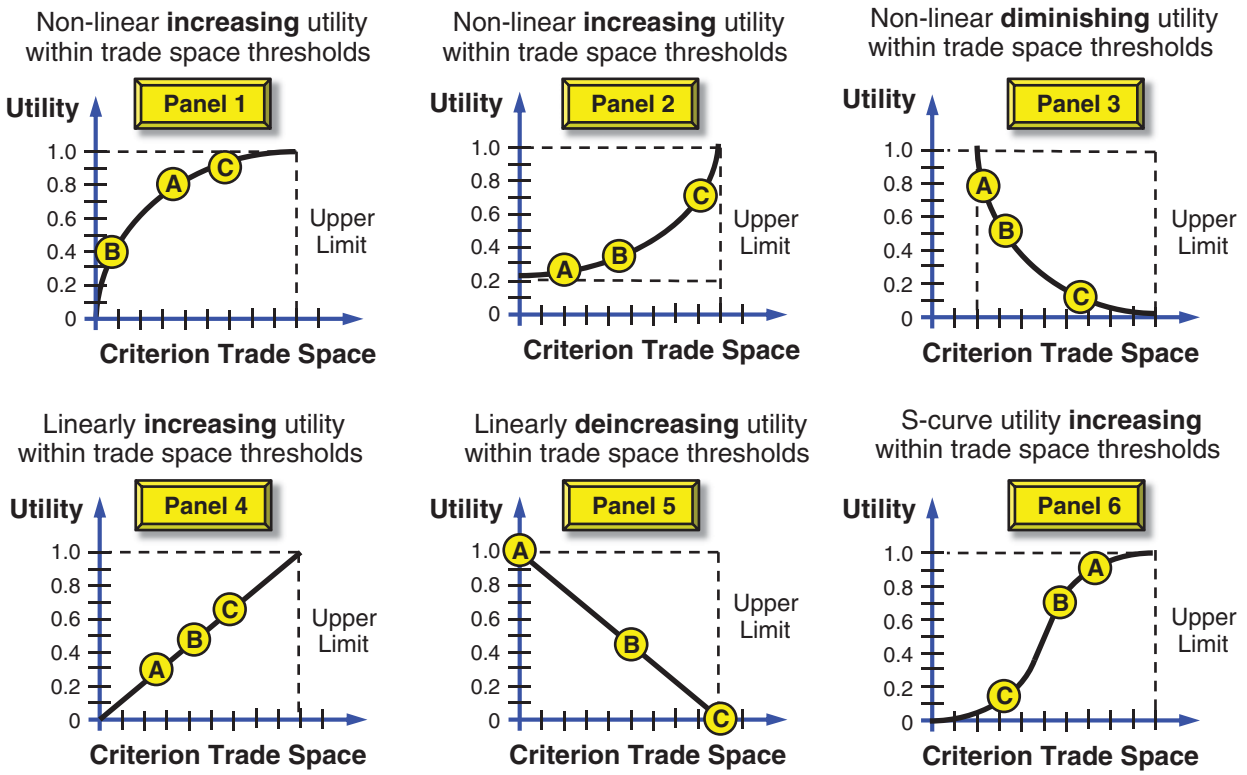


Figure 32.9 Examples of Utility or Scoring Function Curves

However, this is not always the case, and clustering among the candidate scores occurs. So, how do SEs deal with these situations. This brings us to our next topic, Sensitivity analysis.

32.7 SENSITIVITY ANALYSIS

Informed Engineering Judgment Principle

Principle 32.4 Trade studies provide reasoned evaluation substantiated by seasoned experience and facts. As such, they serve as inputs to support *informed* decision making and should be evaluated with sound Engineering judgment.

If we return to the Normalized Trade Study Method (Figure 32.8), observe that the final scores of Candidates A, B, and C were 33, 39, and 28 respectively with Candidate B as the *optimal* selection with the highest overall score of 39 points. Observe the 6-point spread between Candidates A and B, an 11-point spread between Candidates B and C, and a 5-point spread between Candidates A and C.

Do we declare a winner based on the numerical scores? *How do SEs deal with these situations*? In this case, Candidate B's 6-point spread from its nearest competitor makes it a clear selection. However, this is not always the case. So, in answer to the question, a technique called Sensitivity Analysis provides a solution.

In general, Sensitivity Analysis allows us to perform "what if" games scenarios by *changing* the point allocations of each Decision Factor by an arbitrary number such as 10% to test the *robustness* of the apparent winner to stand as verification that the end result will remain unchanged. As a brief example, suppose that we reduce the Cost point allocation from 40 points to 36 points. We then reallocate and normalize the Technology and Performance Decision Factor point allocations. Then, investigate to assess the "cause and effect" relationship on the final selection Candidate B (Figure 32.8).

What happens if the end result changes? In general, *nothing* assuming an objective trade study was conducted. It simply gives *pause to validate* the original selection and reflect on the result. Remember—trade studies, similarly to any type of human tool such as calculators, provide inputs for *informed decision making*. In the end, sound Engineering judgment should always prevail!.

32.7.1 Alternative Sensitivity Analysis Approach

A better approach to differentiating *clustered* trade study data resides in selection of the Decision Factors and Criteria. When we initially identified Decision Factors or Criteria with the Stakeholders, chances are that there was a board range of factors. To keep things *simple*, assume we arbitrarily selected six criteria from a set of 10 provided by Stakeholders and weighted each criterion. As a result, the competing solution FOMs became clustered *eliminating* the possibility of a clear-cut winner.

The next logical step is to factor in the $n + 1$ Decision Factor and *renormalize* the weights based on earlier ranking. Then, continue to factor in additional Decision Factor until the data decluster. However, recognize that if the $n + 1$ has a relative weight of 1%, it may not significantly influence the results. This leaves two options:

- Option A: Make a judgmental decision and pick a solution—not preferable.
- Option B: Establish a ground rule that the initial selection of Decision Factors should not constitute more than 90 or 95 of the total points and scale the list based on 100 points. This effectively leaves 5–10 points for the remaining Decision Factors or Criteria that are not included in the primary list that could have a *level of significance* on the outcome. If clustering continues, based on a Pareto rank ordering of Decision Factors, include the next Decision Factor or Criterion in steps and rescale the weights relative to their initial weights within the total set.

Rhetorically, one can argue that this approach, by virtue of the smaller point values, will have diminishing utility. So, apply common sense. If clustering continues, it may mean that the candidates solutions have similar FOMs, and you may have to make a decision based on *subjective factors* that were not part of the decision. Before you do this, make sure that factors such as life cycle Total Cost of Ownership (TCO) and so forth that are of major concern to the Stakeholders were properly considered, not just weight, cost, power, and so on.

Heading 32.5 When the trade study is complete, the next step is to document the results in a meaningful way that will explicitly provide definitive evidence that the study has been conducted objectively and has integrity. The mechanism for documenting the trade study results is the TSR, our next topic.

32.8 TRADE STUDY REPORTS (TSRs)

On completion of the trade study analysis, the next challenge is being able to articulate the *salient points and results* in the TSR. The TSR serves as an ISO 9000 *Quality Record (QR)* that documents accomplishment of the chartered or assigned task.

32.8.1 Why Document the Trade Study?

Undocumented Trade Studies Principle

An undocumented trade study is nothing more than personal opinion.

Principle 32.5

A common question is: *why document a trade study*? There are several reasons:

- First, the Acquirer's Contract Data Requirements List (CDRL) (Chapter 17) or your Enterprise's command media may require that you document trade studies.
- Second, trade studies *formally* document key decision criteria, assumptions, and constraints of the trade study environment for posterity. Since SE, as a *highly iterative* process, requires decision making based on prescribed conditions and constraints, those same conditions and constraints can *change quickly* or over time. Therefore, you or others may have to revisit previous trade study decisions to investigate how the changing conditions or constraints have impacted the decision or selected course of action such that corrective actions should be initiated.
- Third, as a professional, document key decisions and rationale as a matter of disciplinary practice to preserve the integrity of the decision.

32.8.2 Trade Study Documentation Formality

Trade studies are documented with various *levels of formality*. The level of formality ranges from simply recording the considerations and deliberations in an Engineering Notebook—preferably electronic—to formally approved and published reports intended for wide distribution.

A Word of Caution 32.1

Always check your contract, local Enterprise command media, and/or program's Technical Management Plan (TMP) for explicit direction concerning the *level of formality* required. At a *minimum*, document the key facts of the trade study in an electronic Engineering Notebook.

32.8.3 Preparing the TSR

There are numerous ways of preparing the TSR. First and foremost, *always* consult your contract or Enterprise command media for guidance. If there are no specific outline requirements, consider using or tailoring the outline provided in Table 32.1:

Proprietary and Copyrighted Information

Warning 32.1 Most vendor literature is copyrighted or deemed proprietary. *Avoid* reproducing and/or posting any copyrighted information unless you have the expressed, written permission from the owner/vendor to reproduce and distribute the material. *Always* establish proprietary data exchange agreements before accepting any proprietary vendor data.

Export Controlled Information

Warning 32.2 As stated previously, some vendor data may be subject to Export Control restrictions and the U.S. International Traffic and Arms Regulations (ITAR). *Always* consult with your Enterprise's Project, Contracts, legal, and Export Control organizations before disseminating technical information that may be subject to this constraint.

32.8.4 Proof Check the TSR Prior to Delivery

Some Trade Study Teams develop powerful and compelling trade studies only to have the effort falter due to poor writing and communications skills. When the TSR is prepared, thoroughly edit it to ensure *completeness* and *consistency*. Perform a spell check on the document. Then, peer review the document to see that it is *self-explanatory, consistent*, and *does not contain errors*. Make sure the deliverable TSR reflects the professionalism, research, and quality of effort contributed to the trade study. Finally, *valid* TSR results stand the *test of time* and *professional scrutiny*.

32.9 TRADE STUDY DECISION

Delivery of a trade study initiates a series of steps toward a decision by the decision authority that chartered the study. These steps range from a simple delivery and discussion of the results to a formal meeting with the decision authority and their invitees that may include a presentation and discussion.

32.9.1 Reporting TSR Results

TSR Recommendations Principle

If you charter a trade study, be prepared to implement its recommendations unless there is a compelling reason to avoid doing so.

Principle 32.6

Trade Study Distribution Protocol Principle

Decision authorities disseminate or delegate dissemination of TSR copies. To avoid

Principle 32.7

"rumor mill" damage control, obtain advance approval prior to discussion, dissemination, or presentation to anyone. Remember—the decision authority chartered your work, not third parties.

The Trade Study decision authority such as the Technical Director, Project Engineer, or SDT may request an advance review of the TSR by Stakeholders prior to the TSR presentation and discussion. If a decision is expected at the meeting, advance review of the TSR enables the Stakeholders to come prepared to:

- Address any *open* questions or concerns.
- Make a decision concerning the recommendations.

Advanced Reviews of Preliminary TSR Findings

Author's Note 32.6 Please note that the intent here is to brief progress and status as well as clarify any issues or concerns such as missing expertise, schedule, and resources. Any attempt by the decision authority to purposely steer the directional heading of the trade study to a specific solution is *inappropriate* and a *violation* of professional ethics. Conversely, if new requirements or constraints have arisen that change the *problem or constraints* to be addressed, assuming they are valid, the decision authority needs to issue new direction and an update to the team's charter.

Decision-makers, in general do not like surprises. Once the TSR is completed, it may be appropriate to *pre-brief* the decision authority concerning the results and recommendations. This will avoid their being blind-sided in an open meeting by those who have other objectives.

32.9.2 TSR Submittal

Cover Letter Principle

Principle 32.8 Every work product should be delivered to a decision authority via a cover letter that outlines what has been accomplished; any outstanding issues, risks, or concerns; and task charter compliance statement.

For review, approval, and implementation, deliver the TSR directly to the chartering decision authority that commissioned the study. The TSR should *always* include a cover letter prepared by the Trade Study Team lead and reviewed for concurrence by the team. Additionally, a cover letter establishes an official delivery date for the record.

TSRs can be delivered via mail or by personal contact. It is advisable, however, that the Trade Study Team lead or team, if applicable, personally deliver the TSR to the decision authority. This provides an opportunity to discuss the contents and recommendations.

During the meeting, the decision authority may solicit team recommendations for disseminating the TSR to other Stakeholders such as the Acquirer, Users, or End Users. If a meeting forum is selected to present a TSR briefing, the date, time, and location should be coordinated through notification to the stakeholders and Trade Study Team members.

32.9.3 Presenting the Trade Study Results

TSR Principle

Principle 32.9 Trade Study Reports (TSRs) present fact-based results and prioritized recommendations, not decisions; *decision authorities* make decisions based on the recommendations, their validity, and confidence in the objectivity of the TSR.

Occasionally, a Trade Study Team is unable to reach a consensus; it may be appropriate to include professionally constructive dissenting opinions and their supporting rationale. Once team members approve the TSR, they present the trade study results. There are a number of approaches for presenting TSR results. The approaches generally include delivery of the TSR as a document, briefings, or combination.

32.9.4 TSR Briefings

TSR briefings to Stakeholders can be helpful or a hindrance. They are helpful if additional clarification is required. Conversely, if the presenter does a poor job with presentation, the level of confidence in the TSR may be questioned. Therefore, *go prepared* and deliver a high-quality presentation that establishes confidence in the trade study and its recommendations.

Figure 32.6 Process Step 9 "Make Decision" provides the basis for approval of the trade study. The chartering decision authority has several options as follows:

- Option 1—*Accept* and *adopt* TSR recommendations.
- Option 2—*Place* TSR recommendations *on-hold*.
- Option 3—*Reject* TSR recommendations.
- Option 4—*Return* TSR recommendations for additional action(s).
- Option 5—*Do nothing*.

Option 5 is unacceptable and reflects poorly on the decision authority's professional reputation. If a decision authority makes a decision to charter to a trade study and selected a competent individual or team, they should be *committed* to act on the trade study recommendations (Principle 32.6). If not, why did they waste valuable time and resources on the effort!

Heading 32.6

The preceding discussions provided insights concerning the process of chartering, conducting, and implementing trade study results. We now shift our attention to understanding risk areas that can occur during a trade study.

32.10 TRADE STUDY RISK AREAS

Trade studies, similarly to most decisions, have a number of risk areas: let's explore a few examples.

32.10.1 Risk Area 1: Dependencies and Timing of Trade Studies

Sometimes, a trade study effort is dependent on the results of other on-going trade studies. Even worse, one of those trade studies may be dependent on this trade study resulting in a "circular reference." In addition, there may be a new technology emerging that has not been announced that may impact the results of a trade study. As a decision authority, you should apply *Systems Thinking* to these *interdependencies* and *timing* before you charter a trade study (Principle 32.6).

32.10.2 Risk Area 2: Unproven or Invalid Methodology

Principle 32.10

Trade Study Methodology Principle

Trade studies are only as *valid* as their task constraints, underlying assumptions, methodology, and data collection. Document and preserve the *integrity* of each.

Trade study success begins with a strong, robust strategy and methodology that will withstand professional scrutiny (Principle 32.3). Flaws in the methodology influence and reduce the *integrity and effectiveness* of the trade study. Solicit peer reviews by trusted colleagues and SMEs to ensure that the trade study begins on the right track and yields results that will *withstand professional scrutiny* by the Enterprise, Acquirer, User, and professional community, as applicable.

32.10.3 Risk Area 3: Decision Factor and Criteria Relevancy

Occasionally, selection criteria that have little or no contribution to Decision Factors or Criteria are selected. *Scrutinize* the validity of Decision Factors and Criteria. Document the supporting rationale.

32.10.4 Risk Area 4: Overlooked Decision Factor and Criteria

Sometimes there are COI or CTI attributes that do not make the Selection Criteria List. Selection criteria checks and balances should include verification of traceability of selection criteria to the COIs/CTIs to be resolved for including or excluding Decision Factors and Criteria.

32.10.5 Risk Area 5: Invalid or Incorrect Assumptions

Formulation of candidate solutions often requires a set of dependencies and assumptions such as availability of funding or technology. Stakeholders often challenge trade study results, especially when the Trade Study Team makes *invalid* or *incorrect* assumptions. Where appropriate and necessary, discuss and validate assumptions with the decision authority and Stakeholders to preclude consuming resources developing a decision that may be flawed due to *invalid* or *incorrect* assumptions.

32.10.6 Risk Area 6: Data Validity

Warning 32.3

Today's world is dependent on Internet-based research. *Beware*—materials posted on the Internet may not be current or trustworthy for communications or decision-making. *Always* authenticate, corroborate, and validate sources.

Technical decision-making must be accomplished with the latest most complete, accurate, and precise data available. *Authenticate* the currency, accuracy, and precision of all data as well as vendor commitment to stand behind the integrity of the data.

32.10.7 Risk Area 7: Scaling the Trade Study Task Activities to Resource Constraints

As typical with most SE tasks, you may not always have an adequate amount of time to perform trade studies). Yet, the results are expected to be professionally and competently

accomplished (Principle 23.2) Wasson's Task Significance Principle.

Whatever time frame you have available, assuming it is reasonable, the key results and findings, in general, need to be comparable whether you have 1 day or 1 week. If you have 1 day, the decision authority gets a 1-day trade study and data; 1 month gets a month's level of analysis and data. So, *how do you deal with the time constraints?* The depth of the research, analysis, and reporting may have to be *scaled* to the available time.

32.10.8 Risk Area 8: Test Article Data Collection Failures

When test articles are "on loan" for technical evaluation, failures may occur and *preclude* completion of data collection within the *allowable* time frame. Because of the limited time for the trade study, replacement of the test article(s) may not be practical or feasible. Plan for contingencies and mitigate their risks! Agreements should explicitly define who is accountable for paying for repairs and timing of the repairs.

32.10.9 Risk Area 9: Failure to Obtain User "Buy In"

Contrary to popular opinion, User acceptance does not begin with delivery of System, Product, or Service. The process begins at CA. Keep the User *informed and involved* as much as practical in the technical decision-making process to provide the foundation for *positive* delivery acceptance satisfaction. When high-level trade studies are performed that have an impact on SYSTEM capabilities, interfaces, and performance, solicit Stakeholder—User and End User—validation of Decision Factors and Criteria and their respective weights. Give the Stakeholders some level of *ownership* of the SYSTEM/PRODUCT, beginning with CA or initiation of the project.

32.10.10 Risk Area 10: Failure to Achieve an Optimal Solution

The perception of clear-cut options available for selection is sometimes deceiving. The reality is that the Candidates may be *unacceptable*. There may even be instances where the trade study may lead to yet another option that is based on *combinations* of the options considered or options *not considered*. Remember, the trade study process is *not* intended to answer: *is it A, B, or C*? The *objective* is to evaluate a range of viable candidate alternatives to evaluate and determine the *optimal* solution for a given a set of prescribed decision criteria. That includes other options that may not have been identified prior to the start of the trade study.

32.10.11 Risk Area 11: AoA Decisions are Commitments

Principle 32.11

Resource Commitments Principle

System Design Solution decisions represent commitments of resources long before the resources are actually expended. Beware of those who whimsically believe those decisions are *volatile* up until the expenditure of funds.

There is a subtlety in Figures 32.1 and 32.2 that may not be readily apparent and represents a shortcoming in higher-level decision maker mindsets. When technical decisions are made such as *acceptance* of trade study recommendations, the decision represents a *commitment* to a design solution that has "downstream" resource, cost, and schedule consequences.

Technical decisions including TSR decisions such as Figure 32.3 are not trivial, perfunctory exercises. They have a significant impact of Life Cycle Cost (LCC).

- For military systems, the DAU (2015) based on earlier work by Fabrycky (1994, Figure 3, p. 2) and more recently Blanchard and Fabrycky (2011, Figure 2.12, p. 49) estimates that approximately 90% of LCC are committed by the completion of Engineering & Manufacturing Development – System Development Phase (Figure 12.2). Yet, only approximately 10% of LCC have actually been expended.
- Fabrycky (1994) also observes that when those Cost-Commitment decisions are made, the level of "System-Specific Knowledge" maturity required to make those decisions does not occur until later in time.

As a result, technical decision risk becomes a key factor for the System being developed. Exacerbating the risk is a misperception sometimes by Executives and Project Managers (PMs) … "if the funds have not been expended, the System Design Solution is always changeable to accommodate ad hoc User requirements changes without commensurate contract – technical, cost, and schedule – modifications." This is factually incorrect! Engineering decisions commit project resources long before project resources are actually expended! Recognize and appreciate the difference!

There may be a significant period of time such as weeks, months, or years between trade study *acceptance* as a technical design solution *commitment*, incorporation of the decision into a design solution, and the actual commitment of financial resources to procure components. During these

intervening months, the project may get into cost or schedule overruns due to other decisions or conditions.

From a technical perspective, PMs and executives often have the *misperception* that they can arbitrarily upstage the original trade study decision simply because *they failed* to properly allocate and reserve resources to procure the resulting components when planned. Depending on how far downstream the project may be could result in significant redesign efforts. Recognize that acceptance of trade study recommendations represents an immediate *commitment of project resources* long before the actual expenditures occur. Project Managers (PMs) sometimes fail to understand this commitment when the decision is made.

32.11 TRADE STUDY LESSONS LEARNED

Although trade studies are intended to resolve COIs/CTIs, one of their ironies is they sometimes create their own set of issues related to scope, context, conduct, and reporting. Here are a few lessons learned to consider based on issues common to many trade studies:

- Suggestion 1: Select the right methodology.
- Suggestion 2: Adhere to the methodology without deviation unless it is flawed.
- Suggestion 3: Select the right Decision Factors and Criteria, and their respective weights.
- Suggestion 4: Avoid pre-determined trade study decisions.
- Suggestion 5: Establish acceptable data collection methods.
- Suggestion 6: Ensure data source *integrity* and *credibility*, especially Internet sources.
- Suggestion 7: Reconcile COI/CTI interdependencies.
- Suggestion 8: Accept/reject TSR recommendations.
- Suggestion 9: Document the trade study decision.
- Suggestion 10: Create TSR *credibility* and *integrity*.
- Suggestion 11: Respect trade study dissenting opinions.
- Suggestion 12: Maintain TSRs as conditions evolve.

Bohlman and Bahill (2010) provide additional insights concerning an analysis of 110 TSRs submitted over two decades by students and practicing engineers at the University of Arizona.

32.12 CHAPTER SUMMARY

During our discussion of trade study practices, we defined what a trade study is, discussed why trade studies are important and should be documented, and outlined the basic trade study process, methodology, and alternative methods. We also addressed methods for documenting and presenting the TSR results as a means of building confidence in the Trade Study results.

32.13 CHAPTER EXERCISES

32.13.1 Level 1: Chapter Knowledge Exercises

1. What is a trade study?
2. What is the work product of a trade study?
3. What are the attributes of a trade study?
4. How are trade studies conducted?
5. Who is responsible for conducting trade studies?
6. When are trade studies conducted?
7. Why do you need to perform trade studies?
8. What is a trade space?
9. What methodology is used to perform a trade study?
10. How do you select trade study decision factors, criteria, and weights?
11. What is a utility function?
12. What is a sensitivity analysis?
13. How do you document, report, and present trade study results?
14. What are some of the Enterprise issues, technical issues, and risks in conducting a trade study?

32.13.2 Level 2: Chapter Knowledge Application Exercises

Refer to www.wiley.com/go/systemengineeringanalysis2e.

32.14 REFERENCES

Blanchard, Benjamin S, and Fabrycky, Wolter J. (2011), *Systems Engineering and Analysis*, 5th ed. Upper Saddle River, NJ: Pearson Education, Inc.

Bohlman, James, and Bahill, A. Terry (2010), "Mental Mistakes Made by Systems Engineers While Creating Tradeoff Studies", University of Arizona, *Proceedings of the 20th Annual International Symposium of the International Council of Systems Engineering*, San Diego, CA: INCOSE. Accessed on 5/16/13 https://www.incose.org/ipub/10/0722.PDF.

DAU (2012), *Glossary: Defense Acquisition Acronyms and Terms*, 15th ed. Ft. Belvoir, VA: Defense Acquisition University (DAU) Press. Retrieved on 6/1/15 from http://www.dau.mil/publications/publicationsDocs/Glossary_15th_ed.pdf.

DAU (2015), *Lesson 6.2 "Cost Estimating", LOG200 Intermediate Acquisition Logistics, Slide 7 of 21*, Ft. Belvoir, VA: Defense Acquisition University (DAU).

Fabrycky, Wolter J. (1994), *"Modeling and Indirect Experimentation in System Design Evaluation," The Journal of INCOSE, Vol. 1, Number 1, July – September, 1994*, San Diego, CA: The International Council on Systems Engineering.

NASA (2011), *Review of NASA's Selection of Display Locations for Space Shuttle Orbiters*, Office of the Inspector General, Washington, DC: NASA. Accessed on 01/20/14 from http://oig.nasa.gov/audits/reports/FY11/Review_NASAs_Selection_Display_Locations.pdf.

33

SYSTEM MODELING AND SIMULATION (M&S)

Analytically, System Engineering requires several types of technical decision-making activities:

- **Problem-Solution Space Analysis**—Understanding the User and End User's *problem space* to identify and bound one or more Solution Spaces that provides *operational utility*, *availability, suitability*, *usability, effectiveness, and efficiency* (Principle 3.11) in accomplishing Enterprise missions and objectives.
- **Mission Analysis**—Understanding the types of missions the User expects to perform, their objectives, and performance-based outcomes.
- **Use Case and Scenario Analysis**—Understanding how the User intends to deploy, operate, maintain, sustain, retire, and dispose of the system.
- **Architecture Development**—Hierarchical organization, decomposition, and bounding of operational problem space complexity into manageable levels of lower risk *solutions spaces*, (Principle 4.17) each with a bounded set of requirements.
- **System Performance Analysis**—Model the Physical System to understand "what if" *cause and effect* relationships, predict SYSTEM performance for a given set of conditions, in support of specification analysis and development.
- **Analysis of Alternatives (AoA)**—Formulation, evaluation, and selection of an *optimal* solution from a range of viable alternatives to solve one or more Critical Operational or Technical Issues (COI/CTI).
- **Requirements Allocation**—*Informed* appropriation and assignment of capabilities and *quantifiable* performance to each lower risk *solution spaces*.
- **System Optimization**—Refer to Chapter 30 Definitions of Key Terms.
- **System Failure Replication**—For a given set of conditions, replicate a SYSTEM failure to understand the root cause and formulate a mitigating operational or design action (Figure 24.1).
- **Failure Modes and Effects Analysis (FMEA)**—Model the Physical SYSTEM to evaluate and assess potential FMEA (Figure 34.17).
- **Test Data Analysis**—Analyze SYSTEM performance test data to assess compliance to specification requirements, observe trends, and predict failures.

Depending on the size and complexity of the SYSTEM OF INTEREST (SOI), most of the points above require tools and decision aids to facilitate the decision-making. Because of the complex interacting parameters of the System of Interest (SOI) and its OPERATING ENVIRONMENT, humans are often unable to internalize solutions on a personal level and require assistance. For this reason, Engineers as a group tend to employ and exploit decision aids such as Modeling and Simulation (M&S) to gain insights into the SYSTEM interactions for a prescribed set of operating scenarios and conditions. Assimilation and synthesis of this knowledge and interdependencies via models and simulations enable SEs to collectively make these decisions.

System Engineering Analysis, Design, and Development: Concepts, Principles, and Practices, Second Edition. Charles S. Wasson.

Companion Website: www.wiley.com/go/systemengineeringanalysis2e

This chapter provides an introductory overview of how SEs employ models and simulations to support SE Process Model decision-making (Figure 14.1). Our discussions are not intended to instruct you in *how to* develop models or simulations; numerous textbooks are available on this topic. Instead, we focus on the *application* of Modeling and Simulation (M&S) as a decision aid to facilitate SE decision-making.

We begin our discussion with an introduction to the fundamentals of M&S. We identify various types of models, define model fidelity, address the need to certify models, and describe the integration of models into a simulation. Then, we explore how SEs employ models and simulations to support technical decisions involving architecture evaluations, performance requirement allocations, "what if" exercises, conflict resolution, to drive out COIs/CTIs.

33.1 DEFINITIONS OF KEY TERMS

- **Accreditation** "The formal certification that a model or simulation is acceptable for use for a specific purpose. Accreditation is conferred by the organization best positioned to make the judgment that the model or simulation in question is acceptable. That organization may be an operational user, the program office, or a contractor, depending upon the purposes intended." (DAU, 2001, p. 120)
- **Accreditation** "The official certification that a model or simulation is acceptable for use for a specific purpose … " (DoD 5000.59-M, 1998, p. 187)
- **Certified Model** A formal designation by an officially recognized decision authority for validating the products and performance of a model.
- **Deterministic** "Pertaining to a process, model, simulation or variable whose outcome, result, or value does not depend upon chance. Contrast with: stochastic." (DoD 5000.59-M, 1998, p. 108)
- **Deterministic Model** "A model in which the results are determined through known relationships among the states and events, and in which a given input will always produce the same output; for example, a model depicting a known chemical reaction. Contrast with: stochastic model." (DoD 5000.59-M, 1998, p. 108)
- **Discrete Model.** "A mathematical or computational model whose output variables take on only discrete values; that is, in changing from one value to another, they do not take on the intermediate values; for example, a model that predicts an organization's inventory levels based on varying shipments and receipts." (DoD 5000.59-M, 1998, p. 108)
- **Emulate** "To represent a system by a model that accepts the same inputs and produces the same outputs as the system represented. For example, to emulate an 8-bit computer with a 32-bit computer." (DoD 5000.59-M, 1998, p. 108)
- **Event** "A change of object attribute value, an interaction between objects, an instantiation of a new object, or a deletion of an existing object that is associated with a particular point on the federation time axis. Each event contains a time stamp indicating when it is said to occur." (DoD 5000.59-M, 1998, pp. 113–114)
- **Fidelity** "The accuracy of the representation when compared to the real world." (DoD 5000.59-M, 1998, p. 119) For example, in aircraft simulation, are actual working switches, knobs, or display required or can a touch-screen display overlaying a photograph or graphic of an actual cockpit be used?
- **Capability (Functional Model)** A model composed of functions configured to transform one or more sets of inputs into one or more outputs as a behavioral stimulus response for a given set of operating conditions and constraints without regard to physical implementation.
- **Initial Condition** "The values assumed by the variables in a system, model, or simulation at the beginning of some specified duration of time. Contrast with: boundary condition; final condition." (DoD 5000.59-M, 1998, p. 128)
- **Initial State** "The values assumed by the state variables of a system, component, or simulation at the beginning of some specified duration of time. Contrast with: final state." (DoD 5000.59-M, 1998, p. 128)
- **Model** "A physical, mathematical, or otherwise logical representation of a system, entity, phenomenon, or process." (DoD 5000.59-M, 1998, p. 138)
- **Model-Test-Model (MTM)** "An integrated approach to using models and simulations in support of pre-test analysis and planning; conducting the actual test and collecting data; and post-test analysis of test results along with further validation of the models using the test data." (DoD 5000.59-M, 1998, p. 139)
- **Modeling** "Application of a standard, rigorous, structured methodology to create and validate a physical, mathematical, or otherwise logical representation of a system, entity, phenomenon, or process." (DoD 5000.59-M, 1998, p. 138)
- **Modeling and Simulation (M&S)** "The use of models, including emulators, prototypes, simulators, and stimulators, either statically or over time, to develop data as a basis for making managerial or technical decisions. The terms 'modeling' and 'simulation' are often used interchangeably." (DoD 5000.59-M, 1998, p. 138)

- **Monte Carlo Algorithm** "A statistical procedure that determines the occurrence of probabilistic events or values of probabilistic variables for deterministic models; e.g., making a random draw." (DoD 5000.59-M, 1998, p. 140)
- **Monte Carlo Method** "In modeling and simulation, any method that employs Monte Carlo simulation to determine estimates for unknown values in a deterministic problem." (DoD 5000.59-M, 1998, p. 140)
- **Simulation Time** "A simulation's internal representation of time. Simulation time may accumulate faster, slower, or at the same pace as sidereal time." (DoD 5000.59-M, 1998, p. 159)
- **Stimulate** "To provide input to a system in order to observe or evaluate the system's response." (DoD 5000.59-M, 1998, p. 161)
- **Simulation**—A method for implementing a model. It is the process of conducting experiments with a model for understanding the behavior of the system modeled under selected conditions or of evaluating various strategies for the operation of the system within the limits imposed by developmental or operational criteria. Simulation may include the use of analog or digital devices, laboratory models, or "testbed" sites. (DAU, 2012, p. B-203)
- **Stimulation** "The use of simulations to provide an external stimulus to a system or sub system." (DoD 5000.59-M, 1998, p. 161)
- **Stochastic** "Pertaining to a process, model, or variable whose outcome, result, or value depends on chance. Contrast with: deterministic." (DoD 5000.59-M, 1998, p. 162)
- **Stochastic Process** "Any process dealing with events that develop in time or cannot be described precisely, except in terms of probability theory." (DoD 5000.59-M, 1998, p. 162)
- **Stochastic Model** "A model in which the results are determined by using one or more random variables to represent uncertainty about a process or in which a given input will produce an output according to some statistical distribution; for example, a model that estimates the total dollars spent at each of the checkout stations in a supermarket, based on probable number of customers and probable purchase amount of each customer. Syn: probabilistic model. See also: Markov-chain model. Contrast with: deterministic model." (DoD 5000.59-M, 1998, p. 162)
- **Validated Model** An analytical model whose outputs and performance characteristics identically or closely match the products and performance of the physical system or component.
- **Validation (M&S)** "The process of determining the degree to which a model or simulation is an accurate representation of the real-world from the perspective of the intended uses of the model or simulation." (DoD 5000.59-M, 1998, p. 170)
- **Virtual Reality** "The effect created by generating an environment that does not exist in the real world. Usually, a stereoscopic display and computer-generated three-dimensional environment giving the immersion effect. The environment is interactive, allowing the participant to look and navigate about the environment, enhancing the immersion effect. Virtual environment and virtual world are synonyms for virtual reality." (DoD 5000.59-M, 1998, p. 171)

33.2 TECHNICAL DECISION-MAKING AIDS

SE decision making related to SYSTEM performance allocations, performance budgets, and design safety margins requires Decision Support to ensure that informed, fact-based recommendations are made. Ideally, we would prefer to have an exact representation of a system, product, or service we are analyzing. In reality, *exact representations* do not exist until the System or Product is designed, developed, verified, and validated.

There are, however, some decision aids that SE can employ to provide degrees of representations of a system or product to facilitate technical decision-making. These include models, prototypes, and mock-ups. The purpose of the decision aids is to create a representation of a SYSTEM on a small, low-cost scale that can provide empirical data to support design decision-making on a much larger scale.

33.3 SIMULATION-BASED MODELS

Models generally are of two varieties: *deterministic* and *stochastic*.

33.3.1 Deterministic Models

Deterministic models are based on precise relationships such as mathematical models of *transfer functions* that produce *predictable*, *repeatable* results. Consider the following example:

Example 33.1 **Deterministic Model Example**

Each work period an employee receives a paycheck based on a formula that computes his/her gross salary—hours worked times hourly rate less any distributions such as insurance, taxes, and charitable contributions. The computation is *deterministic* due to the precise mathematical relationships established by accounting standards and federal/state/city tax regulations.

33.3.2 Stochastic Models

Although deterministic models are based on precise relationships, *stochastic models* are structured using *probability theory* to process a set of random event occurrences or variances in physical components or workmanship processes or practices. In general, stochastic models are constructed using data from statistically valid samples of a population that enable us to *infer* or *estimate* results about the population as a whole. Examples include environmental conditions and events, human reactions to publicity, and pharmaceutical drug medications. Consider the following Mini-Case Study:

Mini-Case Study 33.1

Retail Store Work Shift Scheduling Stochastic Model

A retail store chain decides to construct an analytical business model of customer activity for use in predicting work force needs for any date or range of dates during the calendar year based on historical performance, business projections, and other factors. The intent is to predict or estimate the *quantity and mix* of personnel such as shelf stockers, security, checkout associates, custodial services, customer services, required to support retail operations as a function of Time-of-Day (TOD), Day-of-the-Week (DOW), Day-of-the-Month (DOM), Day-of-the-Year (DOY), advertising, holidays, and weather predictions.

Random, independent inputs include variables such as (1) number of TOD customers, average shopping duration in store, number of TOD purchases, and checkout wait times; (2) weather; (3) stocked goods, inventory, and planned deliveries; (4) sales; (5) geographic region; and (6) customer satisfaction. Historical performance data reflect wide ranges of *uncertainty* across identical attributes for different years due to weather, economic conditions, and geography.

Due to the random *variability* of inputs and *uncertainty* from other factors, the Enterprise decides to construct a *stochastic* model of business operations. Performance models are constructed for seasonal weather, inventory, customer flow, and queuing lines and integrated into an overall simulation that can be employed at the store, region, national, and international levels. Monte Carlo methods are employed to simulate the *variance and uncertainties* of inputs and conditions that drive each of the underlying models to study the effects on overall business operations and projections.

Mini-Case Study 33.1 illustrates theoretically how *stochastic* models can be employed. Engineers often use a *worst-case analysis* in lieu of developing a model. For some applications, this may be acceptable. However, *suppose a worst-case analysis results in too much overstaffing or excess inventories resulting in increased labor and inventory costs that ultimately drive down profitability?* From a SYSTEM optimization perspective*, what is the optimal mix of the System Elements* – PERSONNEL, EQUIPMENT, MISSION RESOURCES, PROCEDURAL DATA, SYSTEM RESPONSES, *and* FACILITIES – *to achieve customer satisfaction goals and profitability?* In addition, as consumer buying habits and the marketplace evolve over time, decisions must be made concerning the *relevance* and *validity* of historical performance data to future projections. *One-year old data*? *Two-year old data*? *Five-year old data*?

In summary, *stochastic models* enable us to *estimate* or draw *inferences* about SYSTEM performance for highly complex situations and conditions. These situations involve random, independent, uncontrollable events or inputs that have a frequency of occurrence under *prescribed* conditions. On the basis of the frequency distributions of sampled data (Figure 34.3), we can apply statistical methods that *infer* a most probable outcome for a specific set of conditions.

33.3.3 Model Development

Analytical models require a *frame of reference* to represent the characteristics of a SYSTEM or ENTITY. For most systems, a model is created using an *Observer's Frame of Reference*, such as the Right-Handed Cartesian Coordinate System (Figure 25.1).

Analytically, Model Development is similar to System Development. SE model developers should fully understand the *problem space* and the *solution space(s)* the model is intended to satisfy. On the basis of this understanding, a design methodology requires that we first survey or research the marketplace to see if the model(s) we require has already been developed and is available. If available, we need to determine if the model has the *necessary and sufficient* capabilities and technical detail to support our SYSTEM or ENTITY application. *Conventional* design wisdom (Principle 16.7) says that new models should only be developed as a last resort *after* you have exhausted all other alternatives to locate an existing model that meets their needs.

33.3.4 Model Validation

Principle 33.1

Model Fidelity Principle

Explicitly define and delineate *levels of fidelity* using established industry standards. Model fidelity resides in the User's mind. *High* fidelity to one person may be *medium* fidelity to a second person and *low* fidelity to a third person.

Models are only as *valid* as the *quality* of its behavioral and physical performance characteristics used to represent the real-world SYSTEM or ENTITY. We refer to the *quality* of a model in terms of its *fidelity*—meaning its degree of realism. So, the challenge for SEs is: if we develop a model of our SYSTEM or PRODUCT, *how do we gain a*

level of confidence that the model is valid and accurately and precisely represents the physical instance of an item and its interactions with a simulated real-world operating environment?

In general, when developing models, we attempt to *represent* physical reality with *simulated* or *scaled* reality. Our goal is to try to achieve *convergence* between the performance results of physical reality and simulated reality within the practicality or resource constraints. So, *how do we achieve convergence*? We do this by collecting empirical data from actual physical systems, prototypes, field tests, or manufacturer's data, preferably certified. Then, we *validate* the model by comparing the *actual* field data with the *simulated* behavioral and physical performance characteristics. Finally, we continue to refine and validate the model until its results identically or closely match those of the actual system. This leads to the question: *how do we acquire field data to validate a model for a system or product we are developing?*

There are a number of ways to obtain field data. We can:

- Collect data using controlled laboratory experiments subjected to controlled OPERATING ENVIRONMENT conditions and scenarios that represent what this fielded system may experience.
- Install a similar component on an existing system and collect measurement data for OPERATING ENVIRONMENT conditions and scenarios.
- Instrument a field platform such as an aircraft with transducers and sensors to collect operating environment data.

Regardless of the method used, a model is *calibrated, adjusted, and refined* until it is validated as an *accurate* and *precise* representation of the physical system or device. The model is then placed under formal Configuration Management (CM) control (Chapter 16). We may also decide to have an independent decision authority or Subject Matter Expert (SME) certify the model, which brings us to our next topic, Model Certification.

33.3.5 Model Certification

Creating a model is the first step; creating a validated ED model and getting it *certified* requires additional steps. Remember, SE technical decision-making must be founded in objective, fact-based data that *accurately* and *precisely* represent real-world OPERATING ENVIRONMENT situations and conditions. The same applies with models. So, *what is model certification*?

SEVOCAB (2014, p. 41) defines *certification* as:

- "(1) a written guarantee that a system or component complies with its specified requirements and is acceptable for operational use (SEVOCAB, 2014, p. 41)(Source: ISO/IEC/IEEE 24765:2011. Copyright © 2012 ISO/IEC/IEEE. Used by permission.)

For many applications, an independent decision authority validates a model by *authenticating* that model results identically match those obtained from measurements of an actual system operating under a prescribed set of OPERATING ENVIRONMENT conditions.

In general, one SE can demonstrate to a colleague, their manager, or a Quality Assurance (QA) representative that the data match. Certification comes later when an independent decision authority recognized within industry or governmental organization reviews the data validation results and officially issues a Letter of Certification declaring the model to be certified for use in specific applications and conditions.

Do you need certified models? This depends on your project's needs. Certification:

- Is expensive to establish and maintain.
- Has an intrinsic value to the creator and marketplace.

Some models are used one time; others are used repeatedly and refined over several years. Since Engineering decisions must be based on the integrity of data, models are generally *validated internally* but not necessarily certified.

33.3.6 Understanding Model Characteristics

Models are generally developed to satisfy specific needs of the SE, Engineer, or System Analyst. Although models may appear to satisfy two different analysis needs, they may not satisfy the requirements of both. Consider the following example:

Example 33.2 **Functional Versus Physical Models**

Analyst A requires a sensor model to investigate a CTI. To meet this need, the analyst develops a *functional* model of Sensor XYZ to facilitate their understanding the behavioral responses to external stimuli.

Later, Analyst B in another Enterprise researches the marketplace for a physical model for a comparable sensor. During their research, they discover that Analyst A has already developed a sensor model that may be available. However, Analyst B soon learns that the model describes the *functional* behavior of Sensor XYZ, not the Physical Model for Sensor XYZ. As a result, Analyst B has to make a choice: either develop their own Physical Model of Sensor XYZ or translate Analyst A's Functional Model representation into a Physical Model representation, assuming that it is available.

33.3.7 Understanding Model Fidelity

One of the challenges of M&S is determining the type of model you need. Examples include *functional* and *physical* model representations. Ideally, a model should accurately and precisely exhibit stimulus–response behavioral characteristics that enable its User to (1) model SYSTEM performance and (2) conduct "what if" exercises with statistically random inputs and operating conditions.

Due to the complexities and practicalities of modeling, which include cost and schedule constraints, models as representations, estimates, or approximations of reality are characterized in terms of their *levels of fidelity*. For example, *is a first-order approximation sufficient*? *Second-order*? *Third-order*? The question we have to answer is: *what minimum level of fidelity do we need for a specific area of investigation?* Consider the following examples:

Example 33.3

Math Models of Physical Components

Hypothetically, a mechanical gear train has Input–Output (I/O) *transfer function* that can be described mathematically as:

$$y = 0.1x, \text{ where } x = \text{input and } y = \text{output.}$$

You may decide that a simple analytical math model may be *sufficient* for some applications. In other applications, the area of analytical investigation might require a Physical Model of each gear within the gear train, including the thermal expansion characteristics due to the *frictional* and *loading effects* on axle bearings with a specific type, quality, and level of lubrication.

Example 33.4

Aircraft Simulator Fidelity Example

Let's assume you are developing an aircraft simulator. The key questions are as follows:

1. Are computer-generated graphic displays of cockpit instruments with simulated moving needle instruments and touch screen switches *sufficient*, or do you need the actual working hardware used in the real cockpit to conduct training?
2. What level of fidelity in the instruments do you need to provide pilot trainees with the *look and feel* of flying the actual aircraft?
3. Is a static cockpit platform *sufficient* for training, or do you need a three-axis motion simulator to provide the *level of fidelity* in realistic training?

The point of these examples is: SEs, in collaboration with System Analysts, the Acquirer, and Users, must be able to determine what *levels of fidelity* are required and then be able to specify it. In the case of simulator training systems, various levels of fidelity may be acceptable depending on the training task. Where this is the case, create a matrix to specify the level of fidelity required for each physical item and include scoping definitions of each level of fidelity. To illustrate this point, consider the following example:

Example 33.5

The level of fidelity required for some switches may indicate computer-generated graphical images or a photographic.jpg image may be *sufficient* as a background. Touch screen displays that enable switch activation by touch may be *acceptable* to graphically simulate flipping the switch position from UP to DOWN and vice versa by simply changing the switch position image.

An aircraft simulator cockpit may require real-world fidelity with actual working switches, which are expensive. A lesser expense and fidelity approach might be to employ touch screen panels that display separate but registered two-dimensional photographs of switch positions such as ON or OFF with three-dimensional (3-D) lighting effects.

Example 33.6

In other instances, hand controls, brake pedals, and other mechanisms may require actual working devices that provide the tactile "look and feel" of devices of the actual system.

33.3.8 Specifying Model Fidelity

Understanding model fidelity is often a challenge in being able to realistically model the real world. In the case of training simulators that require visual representations of the environment inside and outside the simulated vehicle, what *level of fidelity* in the Out-the-Window (OTW) displays the landmass – terrain and trees – and cultural features such as roads, bridges, and buildings is *necessary and sufficient* for training purposes?

- Are computer-generated images with a synthetic texture *sufficient f*or landscapes?
- Do you require photographic images with computer-generated texture?

The answer to these questions depends on trade-offs between resources available and the *positive* or *negative* impacts to training. Increasing the level of fidelity typically requires significantly more resources such as data storage or computer processing performance. Concepts such as Cost as an Independent Variable (CAIV) enable Acquirer decision makers to assess what level of capability can be achieved at what cost Figure 32.4.

33.3.9 System Operations and Performance Simulations

Models serve as "building block" *representations or approximations* of physical reality. When we integrate these models into an executable framework that enables us to stimulate interactions and behavioral responses under controlled conditions, we create a simulation of an SOI.

As analytical models, simulations enable us to conduct *what if* exercises with each model or system. In this context, the intent is for SEs to understand the functional or physical behavior

Heading 33.1

The preceding discussions provide the foundation for understanding models and simulations. We now shift our focus to understanding how SEs employ M&S to support analytical decision making as well as create deliverable products for Users.

33.4 APPLICATION EXAMPLES OF M&S

M&S are applied in a variety of ways by SEs to support technical decision-making. SEs employ M&S for several types of applications:

- Application 1: Simulation-based architecture selection
- Application 2: Simulation-based architectural performance allocations
- Application 3: Simulation-Based Acquisition (SBA)
- Application 4: Test environment stimuli
- Application 5: Simulation-based failure investigations
- Application 6: Simulation-based training
- Application 7: Test bed environments for technical decision support
- Application 8: Understanding System Performance Characteristics.

To better understand how SEs employ models and simulations, let's describe each type of application.

33.4.1 Application 1: Simulation-Based Architecture Selection

When you Engineer systems, you should have a range of viable alternatives available to support *informed* selection of the best candidate to meet a set of prescribed Operating Environment scenarios and conditions. In practical terms, you cannot afford to develop and model every candidate architecture just to study it for purposes of selecting the best one. We can, however, construct models and simulations that represent *capability* or *physical* architectural configurations. To illustrate, consider the following example.

Example 33.7

Let's suppose that we have identified several promising Candidate Architectures 1 through "*n*" as illustrated on the left side of Figure 33.1. We conduct a trade study (Chapter 32) and determine that the complexities of selecting the right architecture for a given system application requires employment of M&S. Thus, we create Simulation 1 through Simulation "n" to provide the analytical basis for evaluating and selecting the preferred architectural configuration.

We exercise the simulations over a variety of Operating Environment scenarios and conditions. Results are analyzed, compiled, and documented in an Architecture Trade Study, which rank orders the results as part of its recommendations. On the basis of a review of the Architecture Trade Study, SEs select an *optimal* architecture. When the architecture is selected, the simulation serves as the framework for evaluation and refining each simulated architectural entity at lower levels of abstraction.

33.4.2 Application 2: Simulation-Based Architectural Performance Allocations

M&S are also employed to perform simulation-based performance allocations for specification requirements as illustrated in Figure 33.2. Consider the following example:

Example 33.8

Using M&S to Support Specification Performance Requirements Allocations

Assume specification Requirement A specifies "parent" Capability A. Our initial analysis derives three lower level "child" capabilities (Figure 21.2). A1 through A3 that are translated into System Performance Specification (SPS) Requirements A1 through A3. The challenge is: *how can models and simulations support SE requirements allocations of Capability A's performance to Capabilities A1 through A3?*

Let's assume that basic analysis provides us with an initial set of performance allocations that is "in the ballpark." However, the *interactions* among entities are complex and require M&S to support performance allocation decision-making. We construct a model of the Capability A's architecture to investigate the performance relationships and interactions of Entities A1 through A3.

Next, we construct the Capability A simulation consisting of Models, A1 through A3, representing subordinate Capabilities A1 through A3 as Input-Output (I/O) stimulus-response models. Each supporting capability, A1 through A3, is modeled using the System/Entity Capability Construct shown in Figure 10.17. The simulation is exercised for a variety of stimuli, cues, or excitations using Monte Carlo methods to understand the behavior of the interactions over a range of Operating Environment scenarios and

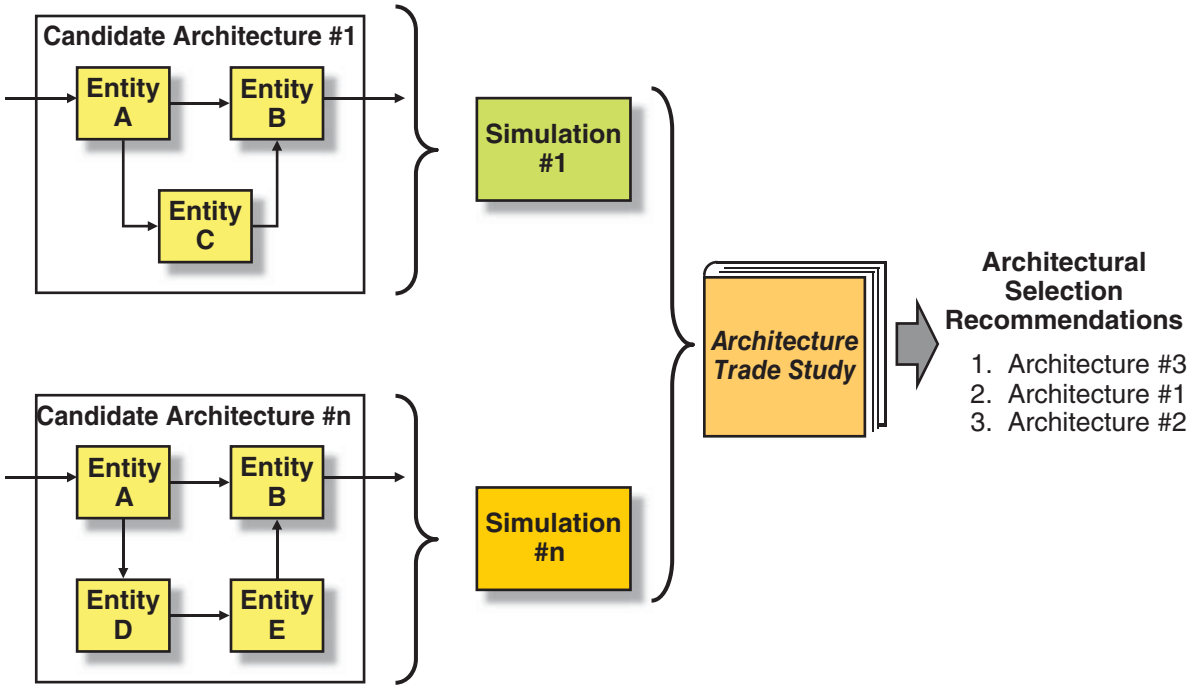


Figure 33.1 Simulation-Based Architecture Evaluation and Selection

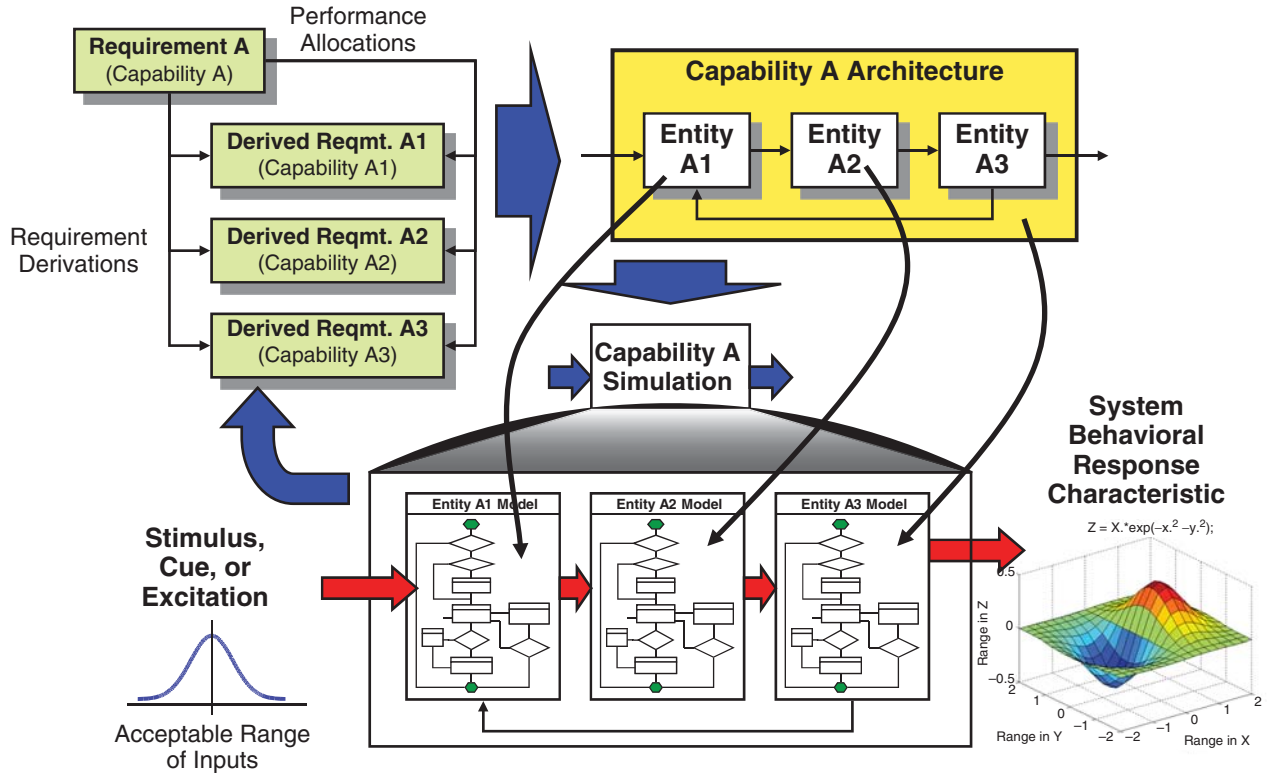


Figure 33.2 Simulation-Based Performance Requirement Allocations

conditions. The results of the interactions are captured in the system behavioral response characteristics.

After several iterations to optimize the interactions, SEs arrive at a final set of *performance allocations* that become the basis for specifying the SPS Capability A requirement. *Is this perfect*? No! Remember, this is a human approximation or estimate that may require independent corroboration. Due to variations in physical components and the OPERATING ENVIRONMENT, the final simulations may still have to be calibrated, aligned, and tweaked for field operations based on actual field data. However, we initiated this process to reduce the complexity of the *solution space* into manageable, lower risk solutions. Thus, we arrived at a very close approximation to support requirements allocations without having to go to the expense of developing the actual working hardware and software.

33.4.3 Application 3: Simulation-Based Acquisition (SBA)

Traditionally, when an Acquirer acquired a SYSTEM or PRODUCT, they had to wait until the Developer delivered the final system for Operational Test and Evaluation (OT&E) (Chapters 12 and 13) or final acceptance. During OT&E, the User or an Independent Test Agency (ITA) conducted field exercises to evaluate SYSTEM or PRODUCT performance under *actual* OPERATING ENVIRONMENT conditions. Theoretically, there should be no surprises. *Why*?

- The SPS described, bounded, and specified the well-defined *solution space*.
- The System Developer created the *ideal* physical solution that perfectly complies with the SPS.

In reality, every System Design Solution has compromises due to the constraints imposed. Acquirers and User(s) of a system need a level of confidence "up front" that it will perform as intended. *Why*? The cost of developing large complex systems, for example, and ensuring that they meet User *validated* operational needs is challenging.

One method for improving the chances of delivery success is SBA. *What is SBA*? In general, when the Acquirer releases a formal Request for Proposal (RFP) solicitation for a SYSTEM or PRODUCT, a requirement is included for each Offeror to deliver a working simulation-based model along with their technical proposal. The RFP stipulates criteria for meeting a prescribed set of functionality, interface, and performance requirements. To illustrate how SBA is applied, refer to Figure 33.3.

Example 33.9 **SBA Example**

Let's suppose a System Developer has an existing aircraft system and decides there is a need to upgrade the Propulsion SUBSYSTEM. In addition, the User has an existing Aircraft System Simulation that is presently used to investigate SYSTEM performance issues.

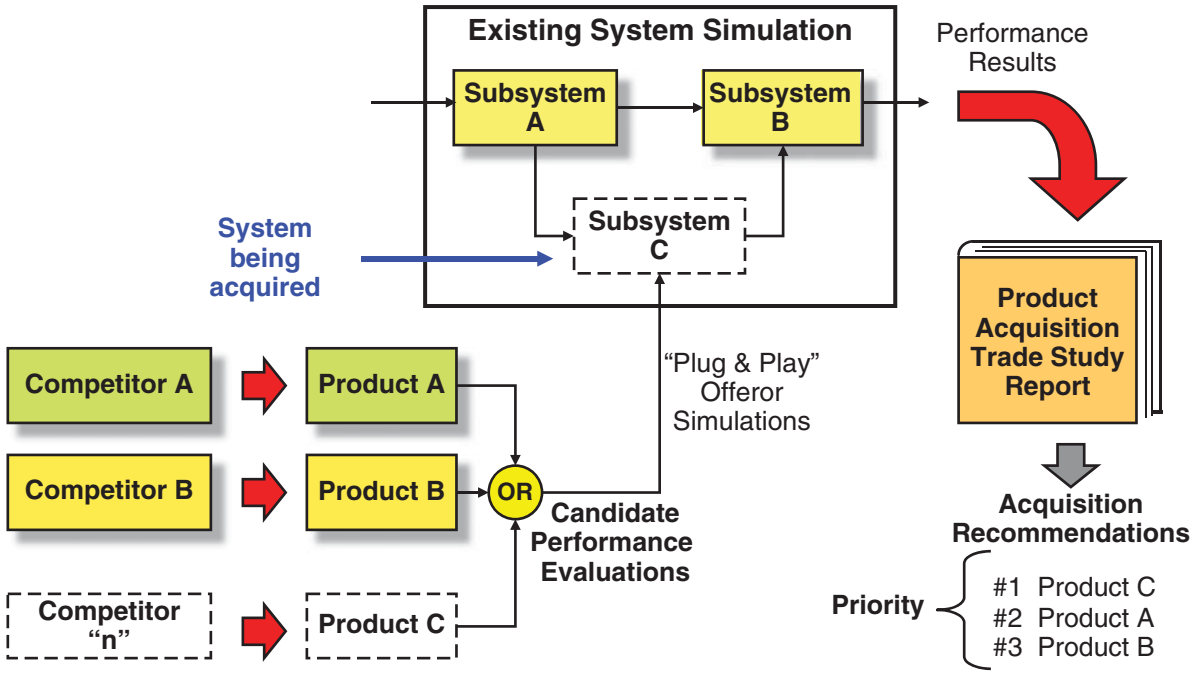


Figure 33.3 Simulation-Based Acquisition (SBA)

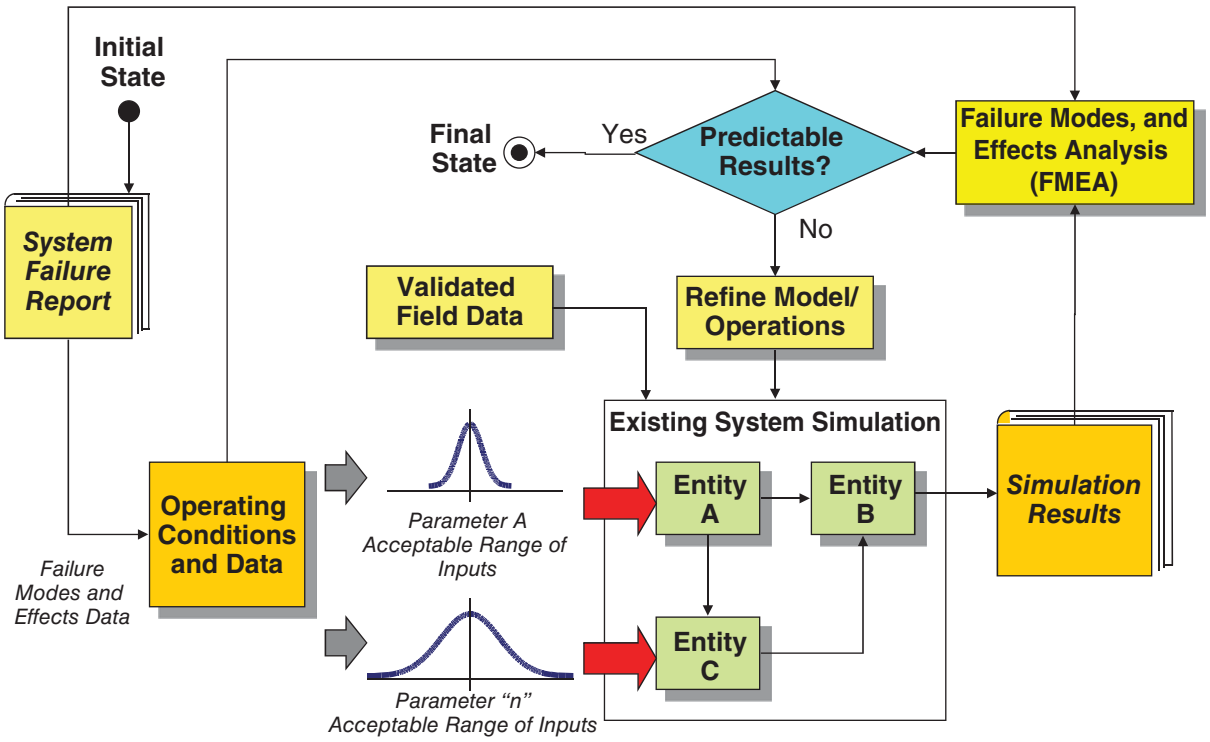


Figure 33.4 Simulation-Based Failure Mode Investigations

The User selects an Acquirer to procure the Propulsion SUBSYSTEM replacement. The Acquirer releases an RFP to a qualified set of Offerors, Competitors A through n. In response to RFP requirements, each Offeror delivers a *plug and play* simulation of their proposed Propulsion SUBSYSTEM to support the evaluation of their technical proposal.

On delivery, the Acquirer's Source Selection Evaluation Team (SSET) evaluates each technical proposal using predefined proposal evaluation criteria. The SSET also integrates each Propulsion SUBSYSTEM Simulation into the Existing System Simulation for further technical evaluation.

During source selection, the SSET evaluates Offeror technical proposals and simulations. Results of the evaluations are documented in a Product Acquisition Trade Study report. The TSR provides a set of Acquisition Recommendations to the SST, which in turn makes Acquisition Recommendations to a Source Selection Decision Authority (SSDA).

33.4.4 Application 4: Test Environment Stimuli

System Integration, Test, and Evaluation (SITE) can be a very expensive activity in System Development, not only from its labor intensiveness, but also from the creation of the Controlled Test Environment interfaces (Figure 28.1) to a Unit-Under-Test (UUT). There are several approaches SEs can employ to test a UUT. The usual SITE options include: (1) *stimulation*, (2) *emulation*, and (3) *simulation* (Figure 28.3). The test environment simulations in this context are designed to reproduce external system interfaces to the (UUT).

33.4.5 Application 5: Simulation-Based Failure Investigations

Large, complex systems often require simulations that enable decision makers to investigate different aspects of performance in employing the SYSTEM or PRODUCT in a *prescribed* OPERATING ENVIRONMENT. Occasionally, these systems encounter an *unanticipated failure mode* or anomaly that requires in-depth investigation. The question for SEs is: *what set of* SYSTEM *User – operator or maintainer actions or conditions and use case scenarios contributed to the failure?* Was the root cause due to (1) *latent defects*, design flaws, or errors, (2) workmanship and reliability of components, (3) operational fatigue, (4) lack of proper maintenance, (5) misuse, abuse, or misapplication of the SYSTEM from its intended or unintended application, or (6) an anomaly?

Due to safety and other concerns, it may be advantageous to investigate the *root* and *contributory* causes (Figures 10.12 and 24.1) of the failure using the existing simulation. The challenge for SEs is being able to:

- Reconstruct the *chain of events* leading to the failure.
- Reliably *replicate* the problem on a predictable basis as validation of the *root, contributory, or probable cause* (Chapter 24).

A decision could be made to use the *simulation* to explore the *most probable or likely root cause* of the failure mode. Figure 33.4 illustrates how you might investigate the cause of failure.

Let's assume that a System Failure Report (SFR) documents the OPERATING ENVIRONMENT scenarios and conditions that lead to a failure event. The SFR should include a maintenance history record among the supporting documents. Members of the Failure Analysis Team extract the OPERATING ENVIRONMENT Conditions and Data from the report and incorporate *actual data* into the Existing System Simulation. SEs perform analyses using Validated Field Data from recorded instrument data such as telemetry and a metallurgical analysis of components/residues. Then, derive additional inputs and make *valid* assumptions as necessary.

Root or Probable Cause Methodology Principle

Principle 33.2 Root, contributory, or probable cause investigations employ a *process of elimination* methodology to rule out failure causes based on objective assessment of the facts.

The Failure Analysis Team explores all possible actions and rules out probable causes using Monte Carlo simulations and other methods. As with any failure mode investigation, the approach is based on the premise that all scenarios and conditions are "suspect" until they are ruled out by a process of *fact-based elimination*. Simulation Results serve as inputs to an FMEA that compares the results the scenarios and conditions identified in the SFR. If the results are not predictable, the SEs continue to Refine the Model/Operations until they are successful in *replicating* the root cause on a predictable basis.

33.4.6 Application 6: Simulation-Based Training

Although simulations are used as analytical tools for technical decision-making, they are also used to train SYSTEM operators. Simulators are commonly used for air and ground vehicle training. Figure 33.5 provides an illustrative example of an aircraft simulator.

For these applications, simulators are developed as deliverable instructional training devices to provide the "look and feel" of actual systems such as aircraft. As instructional training devices, these systems support all phases of mission training including (1) briefing, (2) mission training, and (3) post-mission debriefing. From an SE perspective, these

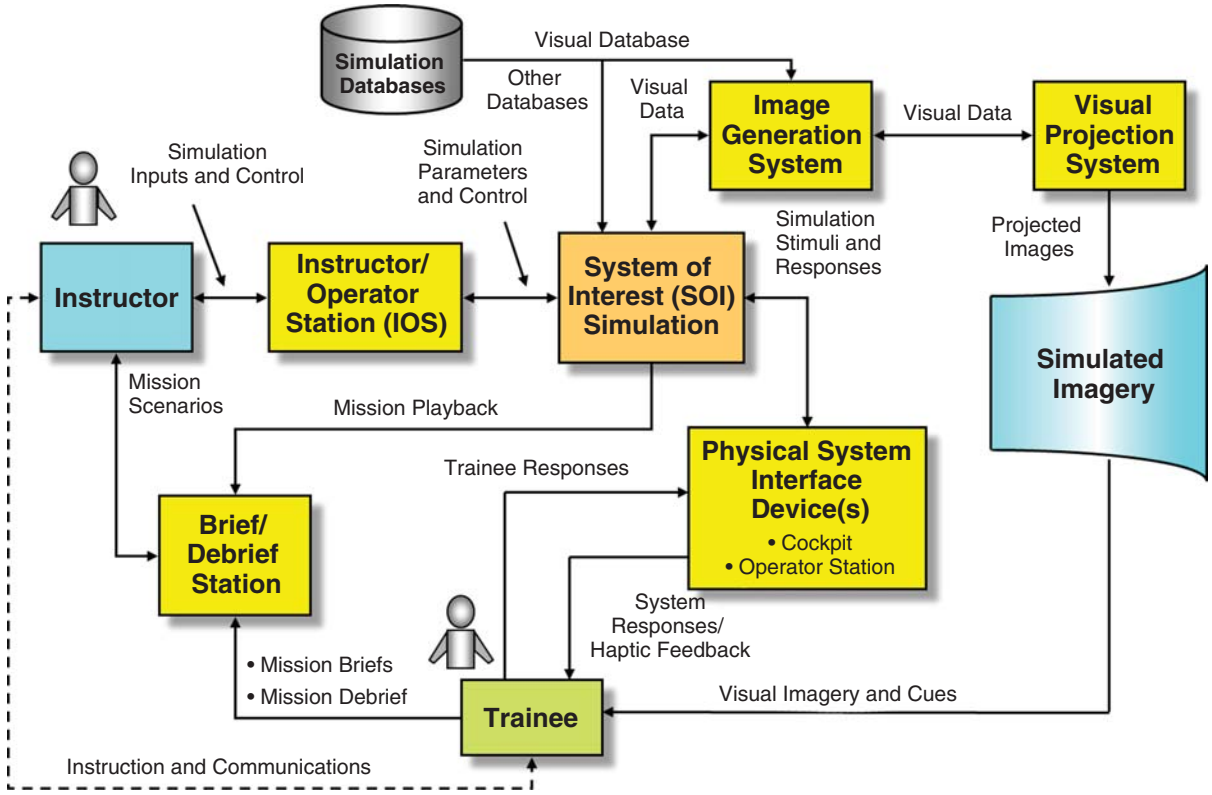


Figure 33.5 Example – Aircraft Simulator System

systems provide a Human-in-the Loop (HITL) training environment that includes the following:

- Brief/Debrief Stations support trainee briefs concerning the planned missions and mission scenarios and debriefing the results of the simulated training flight exercises.
- An Instructor/Operator Station (IOS) controls the training scenario and environment.
- SOI Simulation simulates the physical system the trainee is being trained to operate.
- Image Generation Systems (IGS) generate and display simulated operating environments.
- Simulation Databases such as landmass and cultural features support Visual System environments.
- Debrief Stations provide an instructional replay of the training mission and results.

Principle 33.3 **Negative Training Principle**

In Simulation and Training (S&T), simulated capabilities presented to the User must identically match the operating condition – normal, degraded, or failed – of the actual System and its physical components it represents with no discrepancies. Otherwise, *negative training* occurs.

One of the challenges of personnel training is to *avoid* a condition referred to as *negative training*. Remember—the objective of simulated training is to equip the trainee with the *essential* skills to be able to operate a SYSTEM or PRODUCT that results in a *positive* experience. Simulations that physically deviate or detract from the real world are considered *negative training*. Consider this example.

Example 33.10 Aircraft simulators should *identically mimic* the capabilities, displays, and procedures of a specific tail number aircraft dimensionally, functionally, visually, and audibly. If the actual aircraft has a capability that is *inoperable* such as a light, display, and switch, so should the simulator capability. Otherwise, continuation is considered *negative training*.

In general, there are several types of training simulators:

- **Fixed Platform Simulators** Provide static implementation and use only *visual* system motion and cues to represent *dynamic* motion of the trainee.
- **Motion System Simulators** Employ one-, two-, or three-axis Six-Degree-of-Freedom (6DOF) motion platforms (Chapter 25) to provide an enhanced realism to a simulated training session.

One of the challenges of training simulation development is the cost related to EQUIPMENT - HARDWARE and SOFTWARE. Technology advancements sometimes outpace the time required to develop and deliver new systems. For example, complex M&S may require 2 – 4 years to develop. During the System Development, the initial computers used to develop the SYSTEM will probably become outdated – Moore's Law - by the time it is ready for verification, acceptance, and delivery. This is especially challenging if the System is to be delivered with current state of the practice technology.

Human attempts to create an *immersive* training environment that *transcends* the synthetic and physical worlds are both challenging and expensive. The challenge comes in being able to transform the trainees' *perception of reality* into *believing* they are in a *realistic environment* filled with *physical sensations* such as sight, sound, and motion. One approach to these challenges is to develop a virtual reality simulation.

- **Virtual Reality Simulation** The employment of physical elements such as helmet visors and sensory gloves to psychologically immerse a subject in an audio, visual, and haptic feedback environment that creates the perception and sensation of physical reality.

You may ask: *why do we need to go to this level training and simulations*? The reality is that humans exhibit *erratic and unpredictable behavior* when confronted with *stressful* and potentially *life-threatening situations*. However, we also know that when we subject them to *repetitive* training in various types of simulated scenarios, humans will *instinctively default* to the *procedural imprints* of their training when subjected to such situations. Pilot and astronaut training, for example, employ a dictum that says "Train like you fly and fly like you train." NASA procedures, for example, required that a Shuttle Commander complete 1000 Shuttle Training Aircraft (STA) landings or "dives" in a Gulfstream II business jet modified to simulate Shuttle landings (NASA, 2005).

33.4.7 Application 7: Test Bed Environments for Technical Decision Support

When we develop systems, we need early feedback concerning the potential downstream impacts of technical decisions such as technology updates and configuration changes. While methods such as breadboards, brass boards, rapid prototyping, and technical demonstrations enable us to reduce technical and design risk; the reality is that the effects of these decisions may be unknown until the SITE Phase. Even worse, the cost to build actual HARDWARE and SOFTWARE plus cost-to-correct any latent defects (Table 13.1 and Figure 13.2) – design flaws, errors, or deficiencies–in these decisions or physical implementations *increases significantly* as a function of time After Contract Award (ACA).

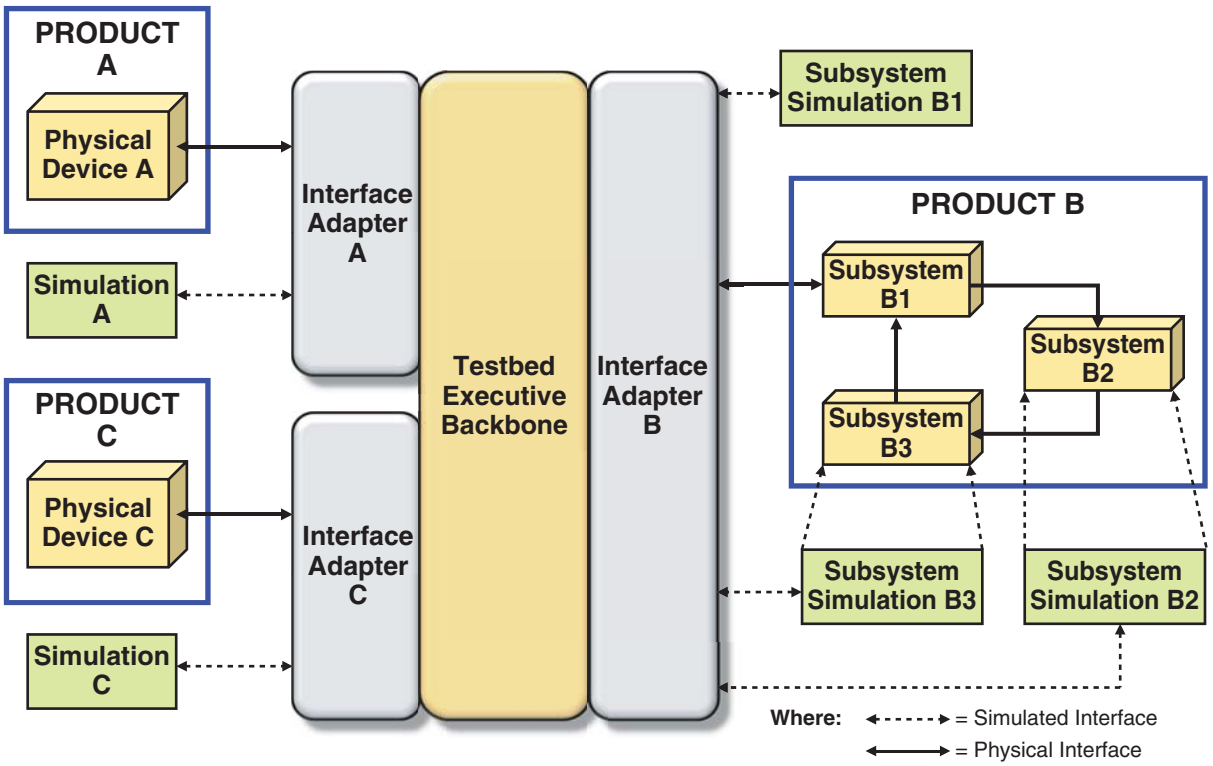


Figure 33.6 Simulation Test bed Approach to System Development

From an Engineering perspective, it would be desirable to evolve and mature models, or prototypes of a laboratory "working system" directly into the deliverable SYSTEM. An approach such as this provides continuity of:

- The *evolving* and *maturing* System Design Solution and its internal and external interfaces.
- Verification of those elements.

The question is: *how can we implement this approach?*

One method is to create a test bed. So, *what is a test bed and why do you need one?*

33.4.7.1 Test Bed Development Environments A test bed is a laboratory-based architectural framework and environment that allows simulated, emulated, or stimulated physical components to be integrated as "working" representations of a Physical SYSTEM or Entity and be replaced by actual components (Figure 28.3) as they become available. SEVOCAB (2014, p. 323) defines a test bed as "(1) an environment containing the hardware, instrumentation, simulators, software tools, and other support elements needed to conduct a test (SEVOCAB, 2014, p. 323) (Source: ISO/IEC/IEEE 24765:2011. Copyright © 2012 ISO/IEC/IEEE. Used by permission.).

Test beds may reside in environmentally controlled laboratories and facilities or they may be implemented on mobile platforms such as aircraft, ships, and ground vehicles. In general, a test bed serves as a framework-based model that enables the virtual world of M&S to transition to the physical world over time.

A test bed is implemented with a central framework that integrates the System Elements (Figure 9.2) and controls the interactions as illustrated in Figure 33.6. Here, we have a Test Bed Executive Backbone framework that consists of Interface Adapters A–C that serve as interfaces to simulated or actual physical elements - PRODUCTS A through C.

During the early stages of System Development, PRODUCTS A, B, and C are modeled and integrated as Simulation A; Simulations B1, B2, B3; and Simulation C into the test bed. The objective is to investigate COIs/CTIs and facilitate technical decision-making. These initial simulations may be of *low* to *medium* fidelity. As the SYSTEM Developmental Configuration (Chapters 12 and 16) evolves, *higher* fidelity models may be developed to replace the lower fidelity models, depending on specific requirements.

When PRODUCTS A, B, and C or their SUBSYSTEMS become available as prototypes, breadboards, or brass boards, the Simulations A through C are replaced with the actual "plug and play" physical entities. Consider the following example:

Example 33.11 **Plug and Play Simulations**

During the development of PRODUCT B, SUBSYSTEMS B1 through B3 may be implemented as Simulations B1, B2, and B3. At some point in time, SUBSYSTEM B2 is physically prototyped in the laboratory. Once the SUBSYSTEM B2 physical prototype reaches an acceptable *level of maturity*, Simulation B2 is removed and replaced by the SUBSYSTEM B2 prototype. Later, when the SUBSYSTEM B2 developer delivers the verified physical item, the SUBSYSTEM B2 prototype is replaced with the actual SUBSYSTEM B.

In summary, a test bed provides a Controlled Operating Environment framework (Figure 28.1) with interface "stubs" that enable developers to integrate *plug-and-play* functional models, simulations, or emulations. As Hardware Configuration Items (HWCIs) and Computer Software Configuration Items (CSCIs) (Chapter 16) are verified, they replace the models, simulations, or emulations. Thus, over time, the test bed evolves from an initial set of Functional and Physical Models and simulation representations to a fully integrated and verified SYSTEM.

33.4.7.2 Reasons That Drive the Need for a Test Bed

Principle 33.4 **Simulation Validity**

Models and simulations are only as good as the human attempts to algorithmically represent and validate the SYSTEM and its interactions with its OPERATING ENVIRONMENT based on actual field data.

Throughout the System Development and the Operation, Maintenance, and Sustainment (OM&S) Phases of the System/Product Life Cycle, SEs are confronted with several challenges that drive the need for using a test bed. Throughout this decision-making process, a mechanism is required that enables SEs to incrementally build a *level of confidence* in the *evolving* and *maturing* System Architecture and System Design Solution as well as to support field upgrades after deployment.

The Human translation process from requirements to design is often prone to *latent defects*; however, integrated tool environments *minimize* the translation errors but often suffer from *format compatibility* and *interoperability* problems. Due to discontinuities in the design and component development workflow, the success of these decisions and implementation may not be realized until the SITE Phase when the actual entities are physically integrated.

Under conventional System Development, breadboards, brass boards, rapid prototypes, and technology demonstrations are used to investigate COIs/CTIs. Data collected from these decision aids are translated by humans into design requirements such as mechanical drawings, electrical assembly drawings and schematics, and software designs and inherently contain defects. So, *how can a test bed overcome these problems*? There are several reasons why test beds can facilitate System Development.

- **Reason 1: Performance-allocation-based decision-making.** When we Engineer and develop systems, *iterative and recursive* application of the SE Process Model (Figure 14.1) requires informed, fact-based decision making at each level of abstraction using the most current data available. Models and simulations provide a means to investigate and analyze performance and system responses to OPERATING ENVIRONMENT scenarios for a given set of "what if" assumptions. The challenge is that M&S are only as "good" as the algorithmic representations used and validated based on actual field data measurements.
- **Reason 2: Prototype development expense.** Working prototypes Proof of Principle, Proof of Concept, and Proof of Technology demonstrations provide mechanisms to investigate a SYSTEM's behavior and performance. However, full prototypes for some SYSTEMS may be too risky due to the maturity of the technology involved and expense, schedule, and security issues. The question is: *should you incur the expense of creating a prototype of an entire* SYSTEM *just to study a part of it?* For example, to study an aerodynamic problem, you do not need to physically build an entire aircraft. Model a "piece" of the problem for a given set of boundary conditions. In the case of a sensor, this might include installation on the wing of a comparable aircraft and performing in-flight performance and test data collection.
- **Reason 3: System component delivery problems.** Despite insightful planning, projects often encounter late vendor deliveries. When this occurs, SITE activities may *severely* impact contract schedules unless you have a good risk mitigation plan in place. SITE activities may become "bottlenecked" until a critical component is delivered. As a workaround, M&S risk mitigation activities might employ some form of representation such as simulation, emulation, or stimulation of the *missing* component to enable SITE to continue to avoid interrupting the overall program schedule.
- **Reason 4: New technologies.** Technology drives many decisions. The challenge SEs must answer is as follows:
  - Is a technology as *mature* as its vendor literature claims?
  - Is this the *right* technology for this User's application and longer term needs?

- Can the technology be seamlessly integrated with the other SYSTEM components with *minimal* schedule impact?

So a test bed enables the integration, analysis, and evaluation of new technologies *without exposing* an existing system to *unnecessary* risk - for example, new engines for aircraft.

- **Reason 5**: **Post-deployment field support.** Some contracts require field support for a specific time frame following system, product, or service delivery during the System OM&S Phase. If the Users are planning a series of upgrades via builds, they have a choice:
  - Incur the cost of operating and maintaining a physical laboratory test article(s) of a fielded system for evaluating incremental upgrades to a fielded configuration.
  - Maintain a test bed that allows the evaluation of configuration upgrades. Depending on the type of system and its complexity, test beds can provide a lower cost solution.

33.4.7.3 Synthesizing the Challenges In general, a test bed provides for "plug-and-play" simulations of CIs or implementation of the actual physical component. Test beds are also useful for workarounds because they help *minimize* SITE schedule problems. They can be used to:

1. Integrate early versions of an architectural configuration that is populated with simulated model representations (functional, physical) of CIs.
2. Establish a "plug-and-play" working test environment with prototype SYSTEM components before an entire system is developed.
3. Evaluate SYSTEMS or ENTITIES to be represented by simulated or emulated models that can be replaced by *higher fidelity* models and ultimately by the actual CI.
4. Apply various technologies and alternative architectural and design solutions for CIs.
5. Assess incremental capability and performance upgrades to system field configurations.

33.4.7.4 Evolution of the Test Bed Test beds evolve in a number of different ways. Test beds may be operated and maintained until the final deliverable SYSTEM or PRODUCT completes SITE. At that point, actual systems serve as the basis for *incremental* or *evolutionary* development. Every system is different. So, assess the cost benefits of maintaining the test bed. All or portions of the test bed may be dismantled, depending on the development needs as well as the utility and expense of maintenance.

For some large complex systems, it may be impractical to conduct "what if" experiments on the *actual* systems in enclosed facilities due to:

1. Physical space requirements.
2. Environmental considerations.
3. Geographically dispersed development organizations.
4. Safety Factors

In these cases, it may be practical to keep a test bed intact. This, in combination with the capabilities of high-speed Internet access, may allow geographically dispersed development Enterprises to conduct work with a test bed without having to be physically co-located with the actual system. For example, an aircraft manufacturer might integrate an external developer's simulation of an engine into a simulation of the aircraft via an Internet or other high-speed interface.

33.4.8 Application 8: Modeling System Performance Characteristics

Engineers often think of System Integration, Test, and Evaluation (SITE) as an activity performed *after* components of the System or Products have been designed, fabricated, assembled, and coded. The shortcoming here is the lack of recognition that SITE occurs throughout the System Design Phase.

Fortunately, with today's high performance computers and models, we can model and simulate the integrated SYSTEM in an OPERATING ENVIRONMENT. This enables us to understand not only its performance characteristics but also its *cause and effect* reactions to dynamic conditions. Figure 33.7 provides an example.

- The left image "This image illustrates the OVERFLOW solution of the Space Shuttle Launch Vehicle flow-field at a Mach number of 1.25. The vehicle surface is colored by the pressure coefficient, and the color contours in the flow-field and plumes represent the local Mach number." (NASAFacts, 2014, p. 2)
- The right image illustrates a Computational Fluid Dynamics (CFD) simulation of high-velocity airflow around the Space Shuttle during re-entry. Advancements in computers and CFD are enabling Engineers to replace the end for wind tunnel testing in some cases. (NASA, 2014)

33.5 M&S CHALLENGES AND ISSUES

Although M&S offer great opportunities for SEs to exploit technology to understand the *problem* and *solution* spaces, there are also a number of challenges and issues. Let's explore some of these.

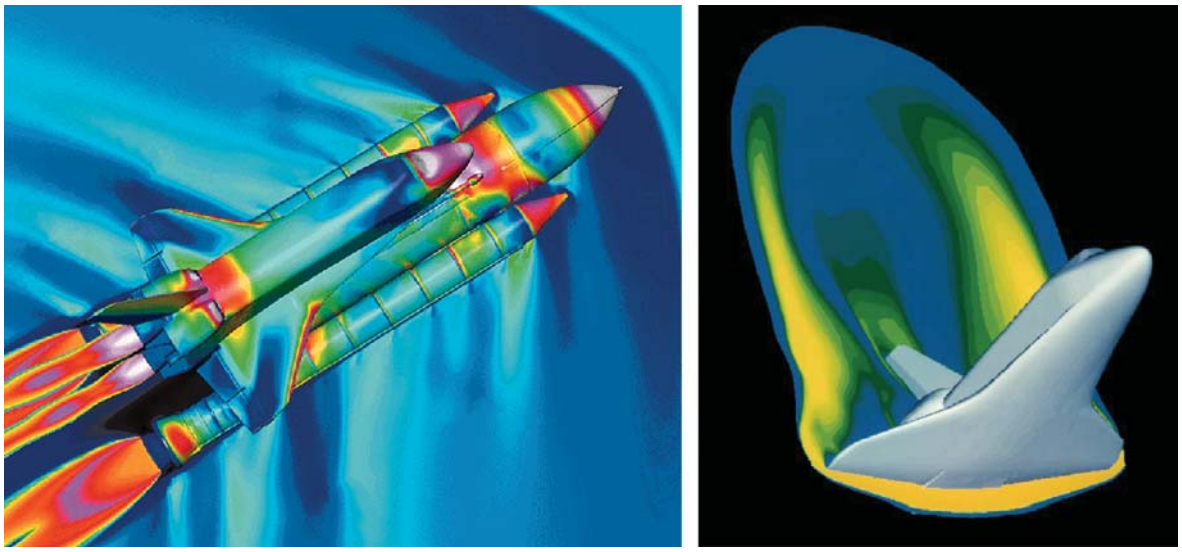


Figure 33.7 Examples - Space Shuttle Modeling and Simulation (M&S) (Sources: (Left image): NASAFacts (2014); (Right image): NASA (2014)).

33.5.1 Challenge 1: Failure to Record Assumptions and Scenarios

M&S require establishing a base set of M&S configurations, assumptions, scenarios, and operating conditions. Reporting M&S results without recording and noting this information in technical reports and briefings *diminishes* the *integrity* and *credibility* of the results. In addition, the version, date, and individual that performed the simulation should be documented.

33.5.2 Challenge 2: Improper Application of the Model

Before applying a model to a specific type of decision support task, the *intended application* of the model should be verified. There may be instances where models do not exist for the application. You may even be confronted with a model that has only a *degree of relevance* to the application. If this happens, you should take the relevancy into account and apply the results *cautiously*. The best approach may be to *adapt* the current model to your application, if feasible.

33.5.3 Challenge 3: Poor Understanding of Model Deficiencies and Flaws

Model Validity Principle

"All models are wrong but some are useful." (Box, 1979, p. 202)

Principle 33.5

Models and simulations generally evolve because an Enterprise has an operational need to satisfy or issue to resolve. When the need to resolve COIs/CTIs is immediate, the investigator may only model a segment of an application or "piece of the problem." Other Users with different needs may want to modify the model to satisfy their own needs. Before long, the model will evolve through a series of *undocumented patches by various recipients* and then documentation *accuracy* and configuration control become critical issues.

To a potential user, such a model may have risks due to potential deficiencies or shortcomings relative to the User's application. In addition, *undiscovered latent defects* such as design flaws, errors, and deficiencies may exist because parts of the model have not been exercised. *Beware of this problem*. Thoroughly investigate the model before selecting it for usage. Locate the originator of the model, assuming that they can be located or are available. *Ask* the developers what you should know about the model's performance, deficiencies, flaws, and documentation. Also inquire about *how well* the model has been tested, User community, specifications.

33.5.4 Challenge 4: Model Portability

Models tend to get passed around, patched, and adapted. As a result, configuration and version control becomes a critical issue. Maintenance and configuration management of models and simulations and their associated documentation is very expensive. Unless an Enterprise has a need to use a model for the long term, it may go onto a shelf. While the physics and logic of the model may remain constant over time, the execution of the model on newer computer platforms may be questionable. This often necessitates *migration* of the model to new computer platforms at additional cost. Therefore, model portability may be a key consideration.

33.5.5 Challenge 5: Poor Model and Simulation Documentation

Models tend to be developed for *specific* rather than *general* applications. Since M&S are often *non-deliverable* items, documentation tends to get *low priority* and is often *inadequate*. Management decision-making often follows a "do we put $1.00 into improving the M&S or do we place $1.00 into documenting the model. Unless the simulation is a deliverable, the view is that it is only for internal use and so *minimal* documentation is the strategy. What is *acceptable* documentation to the model's developer based on prior knowledge may be *unacceptable* to other Users.

33.5.6 Challenge 6: Failure to Understand Model Fidelity

Every model and simulation has a *level of fidelity* that characterizes its performance and quality. Understand what *level of fidelity* you need, investigate the level of fidelity of each candidate model, and determine the utility of the model to meet your needs.

33.5.7 Challenge 7: Undocumented Features

Models or simulations developed as laboratory tools typically are not documented with the *level of discipline* and *scrutiny* of formal deliverables. For this reason, a model or simulation may include *undocumented* "features" that the developer forgot to record because of the available time, budgets cuts, and the like. Therefore, you may think that you can *easily reuse* the model but discover that it contains *problem areas*. A worst-case scenario is planning to reuse a model only to discover deficiencies when you are "too far down the stream" to pursue an alternative course of action.

33.6 CHAPTER SUMMARY

Our discussion of M&S practices identified, defined, and addressed various types of models and simulations. We also addressed the implementation of test beds as evolutionary "bridges" that enable the virtual world of M&S to evolve to the physical world. Key points to remember include the following:

- Models and Simulations (M&S) are only as good as developer attempts to construct *validated* representations.
- Models have *levels of fidelity* with increasing cost for each level. Determine the *least cost level of fidelity* required for specific components for simulating your SYSTEM.
- *Test beds* provide a mechanism to evolve the Developmental Configuration (Chapters 12 and 16) from analytical simulations for replacement with actual working PRODUCTS, SUBSYSTEMS, ASSEMBLIES, OR SUBASSEMBLIES.
- *Deterministic models* are represented by a precise mathematical relationship; stochastic models employ random variables as inputs that are independent and may exhibit various levels of *uncertainty and probability*.
- When searching for existing models, thoroughly investigate the type of model, origin, history, current deficiencies, and currency-based on the requirements of your SYSTEM.
- When acquiring new components for an existing system, Simulation-Based Acquisition (SBA) may provide a mechanism for analyzing and evaluating the performance of the offeror proposed components that have been integrated into a validated simulation of that SYSTEM.
- To avoid a *negative training* result, simulators used for training should identically mimic the SYSTEM they represent including capabilities that may be inoperable.
- *Validated* simulations provide a mechanism to replicate failures as a means of improving multi-level SYSTEM designs.

33.7 CHAPTER EXERCISES

33.7.1 Level 1: Chapter Knowledge Exercises

1. What is an analytical model?

2. What are the various types of analytical models?

3. How are models employed in SE decision-making?

4. How are models validated?

5. Box (1979, p. 202) states that "All models are wrong but some are useful."

a. Provide an explanation of the quote.

b. How can the "usefulness" of models be improved?

6. What is a simulation?

7. What are 3 approaches for representing an OPERATING ENVIRONMENT to a model or simulation?

8. What is a mock-up?

9. How do SEs employ mock-ups? Provide examples and explain what types of knowledge SEs would expect to gain from them?

10. What is a test bed? Provide 3 example systems that might be candidates for a testbed?

11. How is a test bed employed in SE decision making?

12. What is meant by the "plug and play" capability of a testbed?

13. Testbeds can be used to minimize the effects of SITE schedule problems. Identify 5 reasons addressed in the chapter concerning how they can help alleviate the problems.

14. M&S has a number of challenges and issues SEs must be prepared to address. Identify the 7 challenges addressed in the chapter.

33.7.2 Level 2: Chapter Knowledge Application Exercises

Refer to www.wiley.com/go/systemengineeringanalysis2e

33.8 REFERENCES

Box, George E.P. (1979), "Robustness is the Strategy of Scientific Model Building" in *Robustness in Statistics*. eds. R.L. Launer and G.N. Wilkinson, New York, NY: Academic Press.

DAU (2001), *Systems Engineering Fundamentals*, Ft Belvoir, VA: Defense Acquisition University Press. Retrieved on 1/16/14 from http://www.dau.mil/publications/publicationsDocs/SEFGuide%2001-01.pdf.

DoD 5000.59-M (1998), *DoD Modeling and Simulation (M&S) Glossary*, Washington, DC: Department of Defense (DoD). Retrieved on 8/1/13 from http://www.dtic.mil/whs/directives/corres/pdf/500059m.pdf.

ISO/IEC/IEEE 24765:2010 (2010), Systems and software engineering—Vocabulary, International Organization for Standardization (ISO), Geneva: ISO Central Secretariat.

NASA (2005), *Test Drive: Shuttle Training Aircraft Preps Astronauts for Landing*. Accessed on 5/27/13 from http://www.nasa.gov/vision/space/preparingtravel/rtf_week5_sta.html.

NASA (2014), *Space Shuttle CFD Computer Simulation*, Washington, DC: NASA. Retrieved on 1/20/14 from http://www.nasa.gov/images/content/136479main_image_feature_431_ys_full.jpg.

NASAFacts (2014), *The Impact of High-End Computing on the Space Shuttle Program*, Mountain View, CA: NASA Ames Research Center. Retrieved on 01/20/14 from http://www.nasa.gov/centers/ames/pdf/153087main_fs_shuttle_nas.pdf.

SEVOCAB (2014), *Systems and software engineering—Vocabulary*, New York: Institute of Electrical and Electronic Engineers (IEEE). Retrieved on 4/14/15 from http://pascal.computer.org/sev_display/index.action.

34

SYSTEM RELIABILITY, MAINTAINABILITY, AND AVAILABILITY (RMA)

Chapters 1 – 33 introduced SE concepts, principles, and practices for define; bound and specify; architect; monitor, command, and control (MC2); design for Users; model and simulate; test, Verify & Validate (V&V) systems, products, and services. You and your team can create the most elegant System Design Solution that satisfies Principle 3.11's six User benefits:

1. Operational Utility.
2. Operational Suitability.
3. Operational Availability.
4. Operational Usability.
5. Operational Effectiveness.
6. Operational Efficiency.

However, mission and system success are ultimately dependent on:

- Operational Availability "on-demand" to conduct missions.
- Operational Effectiveness, which is a function of System Effectiveness. In turn, System Effectiveness is a function of the system, product, or service's Reliability and Maintainability (R&M).

These three topical areas - System Reliability, Maintainability, and Availability (RMA) - are designated as Specialty Engineering and serve as the focal point of our discussions in Chapter 34.

This leads to a key question: What is SE's role and relationship in identifying, developing, and achieving Reliability, Maintainability, and Availability (RMA) requirements?

As an SE, you are expected to:

- Lead and orchestrate efforts to specify (Acquirer role), analyze, and allocate (System Developer) RMA requirements for the System Performance Specification (SPS) and Entity Development Specifications (EDSs).
- Integrate Specialty Engineering disciplines (Chapter 1) into the SPS/EDS RMA requirements specification and analysis decision-making process beginning with Contract Award.
- Ensure those Engineering specialties are fully integrated and engaged into the multi-level System and Entity SE Design Solution and development activities.

RMA is one of the most vital SE Decision Support activities. However, it is often relegated to a mere "number crunching exercise." Barnard (2008) argues, "many companies think they are performing reliability engineering, while they are actually performing reliability accounting. They are not even close to complying with one of the fundamental aspects of reliability engineering, which is 'paying attention to detail' (MIL-HDBK-338B, 1998, p. 7–1)" Barnard (2008, p. 4).

Barnard (2008, p. 9) referencing Billinton and Allan (1996) also notes "The single most important factor that differentiates between effective and ineffective implemen-

System Engineering Analysis, Design, and Development: Concepts, Principles, and Practices, Second Edition. Charles S. Wasson.

Companion Website: www.wiley.com/go/systemengineeringanalysis2e

tation of a reliability program is timing of the reliability effort. The reliability activity must proceed as an integral part of the development project. If not, it becomes either a purely academic function or historical documentation process. Reliability engineering often becomes a numbers game after the real game is over. Reliability cannot be economically added after the system or product has been conceived, designed, manufactured and placed in operation" (Billinton and Allan, 1996).

Textbooks often treat RMA as separate topics using catchy titles such as Design for Reliability, Design for Maintainability, and Design for Availability. Such treatment infers these are independent "stovepipe" concepts. The reality is you must Design for RMA. A common paradigm is: "Once the design and R&M (Reliability and Maintainability) Engineers are finished, it's up to the maintainers to do the rest." The reality is RMA are not discrete stand-alone topics. Instead, they are integrally linked and form what is referred to as a Reliability-Centered Maintenance (RCM) and Condition-Based Maintenance (CBM) strategy.

If you ask most engineers, managers, and executives what the purpose of Reliability and Maintainability (R&M) is, a typical response is to "prevent the equipment from failing." This is not the purpose of R&M and is a *misinformed* paradigm. The reality is: R&M should ensure *continuity of capabilities* to ensure that a system, product, or service's mission is accomplished without failure (Principle 34.2). That does not mean that R&M is about designing ENTITIES or PARTS that *will not fail*. As a result, R&M should be an integral part of System Architecting (Chapter 26) configurations based on R&M principles coupled with knowledge of ENTITY or PART physical characteristics to ensure that a SYSTEM or PRODUCT will accomplish its mission.

Unfortunately, Reliability is treated as an afterthought in many Enterprises. Barnard (2008, p. 5) articulates Enterprise logic and its erroneous reasoning as follows:

- Reliability is a discipline 'about failures,' and since …
- Operations are responsible for handling failures, therefore …
- Reliability is delegated to the maintenance or logistics departments!"

Chapter 34 provides a foundation in understanding of RMA and how it is applied to overcome these paradigms.

34.1 DEFINITIONS OF KEY TERMS

Chapters 1-33 each included Definitions of Key Terms in a central location at the beginning of each chapter. RMA consists of numerous terms that have *contextual* usage meanings requiring explanation. As a result, many of the terms normally provided here are located in the appropriate sections of Chapter 34's text.

- **Condition-Based Maintenance (CBM)** Refer to the definition provided in Section 34.7.2.
- **Corrective Maintenance** Refer to the definition provided in Section 34.4.6.2.
- **End Effect** "The consequence(s) a failure mode has on the operation, function, or status of the highest indenture level" (MIL-STD-1629A, p. 4).
- **Fail-Safe System** A device or feature, which, in the event of failure, responds in a way that will cause no harm or at least a minimum of harm to machinery or personnel. (MIL-STD-3034, 2011, p. 3).
- **Failure** Refer to Chapter 24 *Definition of Key Terms*.
- **Failure Cause** "The physical or chemical processes, design defects, quality defects, part misapplication, or other processes which are the basic reason for failure or which initiate the physical process by which deterioration proceeds to failure" (MIL-STD-1629A, 1980, p. 4).
- **Fatigue** "A physical weakening of material because of age, stress, or vibration" (DAU, 2012, p. B-83).
- **Fault** Refer to Chapter 26 Definition of Key Terms.
- **Gaussian (Normal) Distribution** Refer to Chapter 30's Definitions of Key Terms.
- **Latent Defects** Refer to Chapter 3 Definition of Key Terms.
- **Life Units** "A measure of use duration applicable to the item (such as operating hours, cycles, distance, rounds fired, and attempts to operate)" (DAU, 2012, p. B-126).
- **Line Replaceable Unit (LRU)** Refer to Chapter 16 Definition of Key Terms.
- **Logarithmic (LogNormal) Distribution** An asymmetrical, graphical plot of the Poisson PDF depicting the dispersion and frequency of independent data occurrences about a mean of the data distribution (Figure 34.2).
- **Maintainability** Refer to definition provided in Section 34.4.1.
- **Mean Time to Repair (MTTR)** Refer to definition provided in Section 4.4.6.2.2.
- **Mission Maintainability** "The measure of the ability of an item to be retained in or restored to specified condition when maintenance is performed during the course of a specified mission profile"(MIL-HDBK-470A, p. G-12).
- **Mission Critical System** Refer to Chapter 5 Definitions of Key Terms.
- **Preventive Maintenance (PM)** Refer to Chapter 6 Definitions of Key Terms.
- **Reliability** Refer to definition provided in Section 34.3.1.

- **Service Life** Refer to definition provided in Section 34.3.3.
- **Single Failure Point (SFP)** "The failure of an item that will result in failure of the entire system. Single failure points are normally compensated for by redundancy or an alternative operational procedure" (DAU, 2012, p. B-203). Some people use the term Single Point of Failure (SPF).
- **Storage Life** "Length of time an item can be stored under specified conditions and still meet specified operating requirements" (MIL-HDBK-470A, p. G-15).
- **Unscheduled Maintenance** "Corrective maintenance performed in response to a suspected failure" (MIL-HDBK-470A, p. G-17).
- **Useful Service Life** Refer to definition provided in Section 34.3.3.

34.2 APPROACH TO THIS CHAPTER

Our approach to Chapter 34 is based on what SEs *need-to-know* and *understand* about RMA, not to make you a Subject Matter Expert (SME). Based on that understanding, you should have the requisite knowledge of what to expect and ask of those who are SMEs to enable you to lead a System / Product Development Team (SDT/PDT) to make *informed* technical decisions.

The purpose of Chapter 34 is to establish the fundamentals and nuances of RMA to enable you as an SE to:

1. Collaborate with RMA Engineers and understand their semantics.
2. Identify SYSTEM RMA capabilities required by the Users and End Users.
3. Trade-off RMA capability priorities in terms of *necessity, sufficiency, and affordability*.
4. Translate RMA capability requirements into the SPS and EDS requirements.
5. Learn to recognize improper shorts-cuts and professionally validate their application and validity.

For military systems, approximately 70% of the Total Cost of Ownership (TCO) of a SYSTEM is consumed during the OM&S Phase (DAU, 1997, p. 13–6). The cost implications here range from thousands to billions of dollars or equivalent in other currencies. *Avoid* the naïve notion that RMA is simply a lengthy academic exercise. SYSTEM or PRODUCT success depends on working knowledge of RMA concepts, principles, and practices.

RMA requires having a basic understanding of Life Data Analysis methods, probability theory, network structures, and selection of math models to represent curve-fitting characteristics of the SYSTEM OR ENTITY being analyzed. The application of this knowledge occurs in four forms:

1. Analytical decision support for proposal and System Design activities.
2. Analytical assessment of physical SYSTEM OR ENTITY Lifetime Data via RMA.
3. Validation of RMA models based on actual SYSTEM OR ENTITY performance data.
4. Analytical assessment of field data and failure reporting to support *preventive* and *corrective maintenance* decision-making and potential SYSTEM OR ENTITY design updates and field retrofits.

Specialty Engineering disciplines include RMA based on the *significance* and *seriousness* of safety, legal, financial, and political consequences resulting from the *application* or *misapplication* of RMA by amateurs. Instead employ *qualified*, *competent*, professional Reliability and Maintainability (R&M) Engineers to perform RMA. With proper, *qualified* credentials, these Subject Matter Experts (SMEs) should be able to competently:

1. Know *when* and *where* to apply RMA methodologies.
2. Interpret the results and their subtleties for informed decision-making.
3. Articulate the results and underlying assumptions to enable technical decision makers to make *informed* decisions.

Your job, as an SE is to develop a *working knowledge* of the basic RMA concepts, principles, and practices to enable you to:

- Know when to employ RMA services.
- Gain a level of confidence in the individual(s) performing RMA.
- Understand *how to interpret* and *oversee the application of* the results provided by the SMEs.

A Word of Caution 34.1

Employ Qualified, Competent RMA Professionals

Since RMA competencies and experience are required to understand which formulas to apply to various architectural configurations as well as the risk of *misapplication*, you are advised to employ the services of a *certified professional* RMA SME of your own choice with a *proven* track record, either on your staff or as a consultant. *Do not* attempt to apply RMA practices unless you are properly *trained, certified, and understand what you are doing*.

Therefore, unless you are a trained and qualified R&M SME, as an SE on a project, you should not be spending your time computing equations; it could be an indication of *misplaced priorities*. Leave the work to a qualified R&M Engineer. Your role as an SE is to review the work of others and understand the challenging questions that need to be asked, discussed, and resolved!

Our discussions include terms such as SYSTEM OR ENTITY Level components. Recall that an ENTITY is a generic reference to a PRODUCT, SUBSYSTEM, ASSEMBLY, SUBASSEMBLY, OR PART. Examples of PART Level components include: electrical—resistors, capacitors; mechanical—pistons, valves, shafts, software; chemical—fluids, lubricants, within an ENTITY.

Chapter 34 is structured on two levels of RMA knowledge: (1) its overarching theme and (2) Lifetime Data Functions required to understand the theme. The overarching theme focuses on RCM and CBM. Both of these interdependent topics: (1) represent a convolution of System R&M concepts, (2) are inextricably linked, and (3) ultimately become the major drivers for determining System Availability to perform missions on demand.

Whereas most Enterprise and Engineering paradigms are *misguided* in the belief that the objective of System Reliability is to preserve EQUIPMENT, the goal of RCM is to preserve a SYSTEM OR PRODUCT's capability (function) to successfully complete its mission via insightful System Architecture configurations, component selection or development, removal of latent defects, and preventive and corrective maintenance actions. Total elimination of EQUIPMENT failures is cost-prohibitive and not the objective. Since RCM and CBM require foundations in R&M concepts, our discussion on sequence begins with System Reliability followed by System Maintainability, System Availability, and culminates in a discussion of RCM-CBM.

34.2.1 System Reliability Section Overview

We begin with Development of Reliable Systems and Products—System Reliability. As an introduction, we define *reliability, mission reliability,* and *equipment reliability*. Since a SYSTEM or ENTITY's Reliability is determined by its failure characteristics, we define what a *failure* is and address it in terms of its *mission critical* and *safety critical* context.

The foundation for System Reliability resides in understanding statistical Lifetime Data Distributions. For some, this may be a refresher from statistics courses. Since the Reliability of systems, products, and services is: (1) conditional and (2) can be characterized by statistical failure distributions, we can model them mathematically. Lifetime Data Distributions such as the Exponential, Gaussian Normal, Log-Normal, and Weibull curve fitting enable us to model a SYSTEM or PRODUCT's lifetime characteristics.

On the basis of an understanding of Lifetime Data Distributions, we shift into understanding Lifetime Data Analysis based on Lifetime Data Functions to characterize the Reliability of a SYSTEM or PRODUCT. The key functions include: Probability Distribution Function (PDF), Cumulative Distribution Function (CDF), the Reliability Function, the Failure Rate Function, the Mean Life Function, the Median Life Function, and the Mode Life Function.

We next introduce the Bathtub Concept, a controversial instructional explanation for the application of Lifetime Data profiles. The discussion first addresses its Decreasing, Stabilized, and Increasing Failure Periods, their validity and relevance to Reliability Engineering (RE) and then addresses challenges to the validity of the Bathtub Concept.

Assessment of SYSTEM or ENTITY Reliability requires insightful knowledge of the SYSTEM's architecture, which forms the basis for three types of Reliability configurations used to assess System Reliability: the Series, Parallel, and Series–Parallel network constructs.

The scope of System Reliability requires more than simply knowledge and application of RMA Lifetime Data Analysis and equations. The question is: *do you understand:*

1. *The ramifications of a* SYSTEM *or* PRODUCT'S *failure modes and effects*?
2. *How to mitigate the failure modes and effects to achieve the SPS reliability requirements*?

We conclude the System Reliability section with an introduction and discussion of Failure Modes & Effects Analysis (FMEA) and Failure Modes & Effects Criticality Analysis (FMECA).

34.2.2 System Maintainability Section Overview

You can develop reliable SYSTEMS and PRODUCTS; however, if you do not maintain them, the reliability is rendered *useless*. Our discussion of System Maintainability begins with the ConOps Maintainability Concept formulated in Table 6.1.

Analytically, we introduce and define the high-level concepts of SYSTEM or PRODUCT Uptime and Downtime. Then, we address *Preventive Maintenance* (PM) and its three types followed by a discussion of Corrective Maintenance. Since Preventive and *corrective maintenance* actions require various levels of expertise and resources, we introduce the two primary levels of maintenance: Field Level Maintenance - Organizational and Intermediate and (2) Depot or Original Equipment (OEM) Manufacturer Level Maintenance and the need for a Failure Reporting, Analysis, and Corrective Action System (FRACAS).

Given an understanding of System Maintainability in the Enterprise context, we introduce a discussion of the metrics or Measures of Maintainability for measuring *maintenance*

action performance. Then, conclude with sources of maintainability data followed by a brief discussion of Logistic Support Analysis (LSA) and provisioning of spares for System Sustainability.

34.2.3 Reliability-Centered-Maintenance (RCM)/ Condition-Based-Maintenance (CBM) Section Overview

Given basic understanding of System R&M, those discussions serve as the foundation for the overarching theme of Chapter 34, Reliability-Centered-Maintenance (RCM) and CBM. We define and explore the RCM and CBM concepts.

34.2.4 System Availability Section Overview

Systems, products, and services can be designed to be *reliable and maintainable*; however, *the ultimate question is: will they perform on demand when required by the User*? This brings us to a discussion of System Availability. Since System Developers and Users respectively have accountability for design and field maintenance of a SYSTEM or PRODUCT, which impacts its Availability to perform, we introduce and define the three types of availability: Inherent Availability, Operational Availability, and Achieved Availability.

Let's begin with SYSTEM Reliability.

34.3 SYSTEM RELIABILITY

Principle 34.1

Conditional Reliability Principle

Reliability is the *conditional probability* that a system, product, or service with a given physical operating condition will successfully complete a mission of a specified duration in a prescribed Operating Environment without disruption.

A SYSTEM's Architecture provides the foundational framework for achieving SYSTEM success. Achievement of success is ultimately determined by: (1) the selection of an optimal System Architecture framework, (2) proper selection of components to implement the framework, and (3) component reliabilities to provide the durability required to accomplish missions in a prescribed Operating Environments. Individual contributions of each multi-level Entity's reliability impact System Maintainability—ease of maintenance—and System Availability—ability to perform missions on demand. EQUIPMENT Element performance and reliability contributions integrate with PERSONNEL, PROCEDURAL DATA, and MISSION RESOURCES, at the MISSION SYSTEM and ENABLING SYSTEM Levels and subsequently impact Enterprise System of Interest (SOI) Mission Reliability and success.

To better understand what Reliability is, let's begin with an Engineering definition.

34.3.1 What is Reliability?

Bazovsky (1961) defines *reliability* as:

- **Reliability** "The conditional probability, at a given confidence level, that the equipment will perform its intended functions satisfactorily or without failure, i.e., within specified performance limits, at a given age, for a specified length of time, function period, or mission time, when used in the manner and for the purpose intended while operating under the specified application and operation environments with their associated stress levels" Bazovsky (1961, p. 19).

In general, people read the words in definitions of *reliability*. However, in the transition from reading to comprehending, internal paradigms filter and distort the meaning to a different interpretation. Referring to Bazovsky's *reliability* definition, where formal Reliability education is lacking, several paradigms illustrate this point.

- **Paradigm #1—Reliability is a Probability of Failure (False)** Observe Bazovsky's Reliability definition phrase "… probability, at a given confidence level, that the equipment will perform …" Engineers often erroneously believe that Reliability represents the probability that a system, product, or service will *fail*. Embedding "fail" or "failure" into a Reliability definition automatically biases one's mental processes and ingrains a paradigm that is very difficult to shift, especially at the project and Enterprise levels.
- **Paradigm #2—Failure Means Total (False)** Observe Bazovsky's Reliability definition phrase "… without failure, i.e., within specified performance limits …" As a result of Paradigm #1, people often mistakenly believe that a *failure* represents a System or Entity that has experienced a total failure – self-destructed - or is inoperable. A *failure* simply means that the performance of a System or Entity is outside its specified SPS or EDS requirements limits. As we shall see in the next section, a failure encompasses any performance degradation as a result of *wear and tear* due to stress, friction, fatigue, misalignment or out of calibration, overheating, deterioration, age, and so forth even though it may still be "operable."
- **Paradigm #3—Reliability is a "Prediction" (False)** Reliability at best is an estimate based on a specified set of System, mission, and OPERATING ENVIRONMENT conditions. Here's the irony: Enterprises and projects "estimate" costs due to *uncertainty* – risk – of external organizations and events beyond their level of control.

Yet, Reliability Engineers "predict" the Reliability of a system, product, or service as if, for example, they can control external Enterprise and Engineered Systems interacting with the system and the OPERATING ENVIRONMENT such as weather phenomena.

- **Paradigm #4—Reliability is a Probability (Partial Truth)** Observe Bazovsky's Reliability definition phrase "… at a given age, for a specified length of time, function period, or mission time, when used in the manner and for the purpose intended while operating under the specified application and operation environments with their associated stress levels." Reliability is more than an abstract *probability percentage* for two reasons:
  1. Reliability is a *conditional probability* based on constraints specified by the SPS or EDS such as initial conditions - *operating condition* the beginning of a mission, *mission duration*, and a *prescribed* OPERATING ENVIRONMENT.
  2. As a *conditional probability*, failure to communicate the Reliability estimate without qualifying operating condition, mission duration, and OPERATING ENVIRONMENT is *incomplete information* and effectively *invalidates* the estimate.
- **Paradigm #5—Reliability, Maintainability, and Availability are All About Equations (False)** As a *conditional probability*, mathematical equations are an integral part of Reliability but *only* as a *supporting tool* to provide data to make and validate *informed* decisions. RMA requires knowledge and implementation of all the *concepts, principles*, and *practices* in Chapters 3 – 33. Examples include System or Product: Use Cases (UCs), concept formulation & development; phases, modes, and states of operation; architecting and interfaces; User-Centric System Design (UCSD), and others. As those concepts and decisions are being developed, *then and only then* do equations have relevance … as a *dependent analytical tool* to support RMA estimates and Modeling & Simulation (M&S). Equations are *useless* unless you have data to select the right equations, component characteristics, and so forth. Those data originate from Chapters 3 - 33, not vice versa.
- **Paradigm #6—Reliability is Components that Will Not Fail (False)** Most people, especially Engineers and managers, erroneously perceive Reliability to be designing components that will not fail. Every Enterprise or Engineered System or Product inevitably will fail due to degradation, wear, deterioration, weak materials, or lack of use and maintenance depending on its frequency of usage, misuse, abuse, and OPERATING ENVIRONMENT conditions. Reliability is not about *preservation* of components; this reflects an "Engineering the Box" (Chapter 1) mindset. Instead, Reliability is about ensuring the *continuity of mission operations* without disruption - "Engineering the System." This requires insightful *collaboration* between multi-discipline SE, System Architecting, and Specialty Engineering such as RMA, Human Factors (HF), and Safety. The objective is to select an *optimal* System or Product Architecture from a viable set of candidate alternatives, select components, and prescribe User operations and training. Collectively, when performed *properly* and *safely*, a System or Product has a higher probability of achieving the User's mission objectives within technical, technology, cost, and schedule constraints with acceptable risk while minimizing the Total Cost of Ownership (TCO) over the System / Product Life Cycle.

Although not explicitly stated in the definition, the implication is that *at delivery*, the Reliability of a SYSTEM or ENTITY at $t = 0$ should be "X" for a given SPS or EDS performance, *useful service life*, and OPERATING ENVIRONMENT conditions requirements. Once the SYSTEM or ENTITY leaves the System Developer's or manufacturer's facility and has been: (1) accepted by the System Acquirer or User and (2) placed into field service operation, its Reliability *diminishes* over its *useful service life* due to usage and deterioration of component physical properties.

34.3.1.1 Reliability is a Conditional Probability

The initial phrase in Bazovsky's (1961, p. 19) definition clearly establishes the context of reliability in terms of its *conditional probability*. *What does this mean*? Reliability is "conditional" based on the assumption that the SYSTEM or ENTITY is (1) fully operational, (2) compliant with its SPS or EDS, and (3) being operated properly and safely in a prescribed OPERATING ENVIRONMENT.

34.3.1.2 Mission Reliability versus Equipment Reliability

Bazovsky's (1961, p. 19) *Reliability* definition focuses on "equipment." However, Reliability is contextual relative to how the SYSTEM is bounded (Figure 8.1) and specified. In general, Enterprise Systems are accountable for performing and successfully accomplishing missions—Mission Assurance. This requires Enterprise assets - SOI MISSION SYSTEMS and ENABLING SYSTEMS - each with a required *mission reliability* defined as:

- **Mission Reliability**—The *conditional* probability that a MISSION SYSTEM or ENABLING SYSTEM—each composed of the integrated set of PERSONNEL, EQUIPMENT, PROCEDURAL DATA, MISSION RESOURCES, SYSTEM RESPONSES, and FACILITIES (Figure 9.2)—will accomplish its assigned mission of a specified duration and OPERATING ENVIRONMENT without disruption.

Since EQUIPMENT is often acquired externally from System Developers, System Acquirers specify its reliability in the SPS as a contributory measure (Figure 5.3) for achieving the Enterprise level Mission Reliability. So *what is EQUIPMENT Reliability?*

- **EQUIPMENT Reliability**—The *conditional* probability that the EQUIPMENT Element—HARDWARE and SOFTWARE—will deliver its capabilities *without failure or disruption* in support of a MISSION SYSTEM or ENABLING SYSTEM's mission of a specified duration without disruption when subjected to a set of prescribed OPERATING ENVIRONMENT conditions.

To illustrate *Systems Thinking* (Chapter 1) in terms of EQUIPMENT Reliability *versus* Mission Reliability, consider the following example:

Example 34.1 **Automobile Trip EQUIPMENT versus Mission Reliability**

An automobile operates on ideal roads without failure—EQUIPMENT Reliability—versus the *random probability* of *unavoidably* running over a board with a nail pointing upward on the road and causing a tire to blow out. That gets into *Mission Reliability* applicable to the MISSION SYSTEM under the Command and Control (C2) by the driver—PERSONNEL ELEMENT—of the EQUIPMENT Element. Can a car recognize a road hazard? In general, no, unless it has special capabilities to detect objects such as approaching another vehicle from behind or an object when moving in reverse. Can its driver recognize a road hazard? That depends on a number of factors such as time of day, size of object, reaction time, and so forth. In that context, the Mission Reliability of the Automobile System - EQUIPMENT and PERSONNEL (Driver) Elements - is a conditional probability dependent on the Driver's ability to see, detect, and avoid road hazards and the EQUIPMENT Element's design – Center of Gravity (CG), handling, traction control, and object detection subsystems, if installed.

Some SYSTEMS or PRODUCTS employ resources such as fuel that have a *Useful Service Life* – xx miles or *shelf life* of "n" months on specific OPERATING ENVIRONMENT conditions. However, this entails resource *planning* and *depletion* versus *failure*. Consider the following example:

Example 34.2 **Resource Depletion—Mission Failures versus Equipment Failures**

Automobile fuel tanks are designed to store X gallons or liters of fuel. On average, the fuel stored in the tank has a Useful Service Life of XXX miles or kilometers and time—weeks or months determined by rate of consumption or shelf life. If the fuel supply becomes depleted during a mission, you have a Mission failure – disruption - due to the Personnel Element - Driver - risk in properly planning and recognizing the need to replenish the fuel. This is not an Equipment Element failure unless the tank or line develops a leak, becomes obstructed, or fault sensor measurements. As an Equipment Element design compensating provision, most automobiles are designed with dashboard displays to indicate fuel reserves, potential driving range to ensure continuity of mission, and caution indicators when fuel reserves diminish below a specific threshold.

Although the concepts of EQUIPMENT Reliability versus Mission Reliability have different contexts, they have one thing in common, *What Constitutes a Failure*?

34.3.1.3 Reliability & Maintainability (R&M) Engineering[1]

Principle 34.2 **R&M Engineering Principle**

The objective of R&M Engineering is to ensure continuity of Mission System operations and capabilities to successfully complete a mission without disruption and accomplish its performance-based objectives, not keep the Equipment Element components from failing.

R&M Engineering is considered a Specialty Engineering discipline. Depending on the size of an Enterprise and complexity of the system, product, and service it produces, R&M may be performed by an individual specialist, a Team, or by external consulting Enterprises.

The purpose of R&M Engineering is not to preserve EQUIPMENT or to keep it from failing. It is to *avoid* the effects and consequences of failure to ensure *continuity of capability* to accomplish its mission. This is a *critical cornerstone* for R&M Engineering including the RCM and CBM concepts introduced later in the chapter.

Enterprises and Engineers are often characterized by a paradigm that *erroneously* views the purpose of R&M is to "evaluate and assess" a System Design Solution "after the fact." Then, suggest design changes – but not too many - to *prevent* or *minimize* EQUIPMENT failures. Objective evidence of this paradigm requires simple observation of project tasking of R&M Engineers' activities and work products. This paradigm is narrow-scoped and has been flawed for decades by Project Management, Engineering, Functional Management, and Executive Management.

The reality is, R&M Engineering should be an integral part of informed technical decision-making during the proposal phase for a System or Product and throughout the System Development Phase (Figure 12.2). Example activities include:

[1]For information concerning the early history and evolution of Reliability Engineering, refer to Kececioglu (2002, pp. 43–52).

1. Participate in the development and analysis of R&M specification requirements.
2. Collaborate with the System Architect to formulate, select, and develop a System Architecture that will meet those requirements.
3. Collaborate with designers to select components.
4. Analyze and evaluate the R&M of the *evolving* and *maturing* physical System Design Solution for *compliance* to SPS RMA requirements.
5. Optimize RMA to achieve the lowest TCO solution Figure 34.24 that has a risk level that is acceptable to the User.

By virtue of this paradigm, Engineers tend to view Principle 34.2 in terms of "Engineering the (EQUIPMENT) Box." Reread Principle 34.2. The context and scope of Principle 34.2 is "Engineering the System," which includes "Engineering the Box." Here's why.

- First, Principle 34.2 addresses the User's Mission Reliability that includes all of the System of Interest (SOI) MISSION SYSTEM and ENABLING System Elements—EQUIPMENT, PERSONNEL, MISSION RESOURCES, PROCEDURAL DATA, SYSTEM RESPONSES, and FACILITIES in Figure 5.3. Based on that assessment, a question emerges: is this Mission Reliability *realistically affordable*? If not, then *what level of mission reliability is affordable* (Figure 5.3)?
- Second, the User's Mission Reliability decision serves as the baseline for deriving the SOI's—MISSION SYSTEM or ENABLING SYSTEM—RMA—and subsequently the EQUIPMENT RMA requirements specified in the SPS. Remember the System Developer's perspective, the SPS RMA requirements become the baseline for system *optimization* and SPS compliance.

On the basis of these points, your job as an SE is to ensure that R&M Engineering is an integral part of every project throughout the System Development phase (Figure 12.2) beginning with the proposal. R&M Engineers should be core members of the project's System Development Team (SDT) of Systems Engineering & Integration Team (SEIT), the highest-level technical team.

Based on this discussion of Mission and EQUIPMENT Reliability, let's shift our focus to a topic common to both – Failure.

34.3.1.4 What Constitutes a Failure? Definitions of a failure are often contextual:

- **Failure** (Condition)—A *condition* in which the performance of a SYSTEM or ENTITY has degraded or deteriorated to the point where it is no longer compliant with its specification requirements.
- **Failure** (Event)—The *event* or *inoperable* state in which any SYSTEM or ENTITY does not or would not perform as specified in the SPS or EDS. Failure events occur in two forms:
  a. **Dependent Failure**: A *premature* SYSTEM or ENTITY failure induced by a fault (Figure 26.8) or failure of an external system. For example, an engine component fails prematurely due to a failure by the User to follow proper lubrication maintenance procedures.
  b. **Independent Failure**: A failure that occurs as a result of an internal fault such as fatigue, corrosion, or fractures (Figure 26.8).

Observe the term *condition* in the first definition. A failure *condition* is time-dependent and a consequence of a *fault* that has materialized. On the basis of Reason's Accident Trajectory Model shown in Figure 24.1, a fault represents a *hazard* that if allowed to materialize under various scenarios becomes a *failure* that may contribute to an *incident* or *accident*.

The materialization of the *fault* into a *failure* can be caused by:

- **Design Defects**—Specification misinterpretation; *latent defects* such as design flaws, errors or deficiency; incompatibility, drift.
- **Component Defects**—Material property flaws, degradation, or deterioration.
- **Manufacturing Defects**—Poor workmanship, improper assembly, inadequate Quality Control (QC) and Assurance (QA).
- **Operational Defects**—Misuse, abuse, or misapplication of the SYSTEM or ENTITY; *stressful* conditions such as overheating, electrical overloads, shock, vibration.
- **Maintenance Defects**—Misalignment, creep, out of calibration, end-of-life failures.
- **Anomaly**—Problem that may not be observable, explicitly identified, or replicated.

As a final note, a failure may be *temporary, intermittent* (anomaly), or *permanent* depending on the SYSTEM or ENTITY; OPERATING ENVIRONMENT; application; or scenario-based conditions. Consider the following example:

Example 34.3 **Intermittent Wiring Failure**

Assume that a vehicle such as an automobile or aircraft has a wiring problem due to a wire's insulation rubbing against metal under specific conditions. If allowed to continue unchecked and mitigated, the insulation could become chaffed possibly

causing a short circuit and/or fire. Under shock and vibration conditions, the wire may only have intermittent contact with the metal. As a result, you have an intermittent condition that exhibits the appearance of an anomaly due to the maintainer's being unable to replicate the problem while the vehicle is moving. This is why periodic visual inspections are critical.

The key point here is: failures present themselves in a number of different forms to different people that have their own perspectives. Having an understanding, opinion, or personal knowledge of what constitutes a failure *does not mean* that everyone on a project shares or agrees with your definition. As an SE, your job is to establish a consensus (Principle 1.3) within the project and preferably within your Enterprise as to what constitutes a *failure*. This brings us to *The Importance of Documenting the Failure Definition*.

34.3.1.5 The Importance of Documenting the Failure Definition

Failure Definition Principle

Principle 34.3 Explicitly define, scope, establish a consensus, and document what constitutes a *failure* among Stakeholders—Enterprise, project, System Acquirer, and User. Then, communicate and ensure everyone understands.

Why is it important to document the definition of failure? Weibull.com (2004) very explicitly articulates *why* you, your project, and Enterprise need to establish a consensus and document the *failure* definition.

> "Another benefit of having universally agreed-upon failure definitions is that it will minimize the tendency to rationalize away failures on certain tests."

As an SE, *how do you ensure everyone has a shared vision as to what constitutes a* failure (Principle 1.3)? You need to:

1. Establish an Enterprise Organizational Standard Process (OSP) consensus definition of a failure for use on projects.
2. Establish and formalize agreement with the System Acquirer and User about the definition of a *failure*.
3. Disseminate the definition across all project teams.

34.3.1.6 Failure Indications Academically, we can define a *failure* as a *non-compliance* with SPS or EDS specification requirements. However, before, during, and following a mission, most systems do not have a magical display of specification compliance of every ENTITY within the System. The result would simply be *impractical* and *cost-prohibitive*. Our best attempts are to:

1. Instrument the SYSTEM or ENTITY to monitor – Monitor, Command, & Control (MC2) (Figure 26.6) - performance conditions of a few *essential*, *mission critical*, and *safety critical* capabilities relative to some specification requirements limits.
2. Perform *preventive maintenance* at pre-defined service intervals.
3. Train the operator and maintainer to perform PRE-MISSION, MISSION, and POST-MISSION real-time and independent inspections such as unusual noises, odors, excessive heat, leaks, chafed wires, fatigue cracks, which may indicate potential pre-failure indications that require *corrective maintenance.*

34.3.1.7 Mission Critical and Safety Critical Items

Mission Reliability Principle

Principle 34.4 *Mission reliability* is like a chain; it is only as reliable as the weakest link in its System Architecture configuration.

Mission-Safety Critical Components Principle

Principle 34.5 Identify, analyze, and label *mission critical* and *safety critical* ENTITY or PART Level components and ensure ease of access for their replacement.

By definition, a component's non-compliance with its specification requirements is a *failure*. The reality is, some components are *essential* to achieving and sustaining EQUIPMENT Element, MISSION SYSTEM, and ENABLING SYSTEM reliability and performance, some are not. Consider the following example:

Automobile Mission Critical-Safety Critical Example

Example 34.4 Prior to leaving on a trip, an automobile should have four tires and a spare in an acceptable condition to travel within the *prescribed* mission distance and OPERATING ENVIRONMENT. During travel, if a flat occurs, the mission can continue, though with some level of elevated risk due to use of the spare until the failed tire is repaired. Tires installed on a car are irrefutably *mission critical* items to ensure achieving *travel* mission objectives. The question is: *when does the set of tires become mission critical? Prior to leaving on a long trip*? When a flat occurs and the spare is used to replace it? The assumption here is that tires are not readily available along the route.

What if the automobile's headlamp is inoperable? Is it a mission critical component? It depends on the situation. If the vehicle's headlamp fails during daylight hours, it may not be *mission critical* unless lighting conditions or compliance with the law necessitate it being activated. *What*

about nighttime hours or in dense fog at night or in the daytime? *Under these conditions, is it both a mission critical and a safety critical item*?

What if the automobile's tail light is inoperable? Is it a mission critical item? It depends on the situation. Indirectly, it is mission critical if you stop and a rear-end collision occurs. Then, the mission may be in jeopardy. If the vehicle's tail light goes out during daylight hours, it may not be *mission critical*; however, it is safety critical as required by the law. *What about nighttime hours or in dense fog at night or in the daytime*? *Under these conditions, is it both a mission critical and a safety critical item*?

Given these fundamentals, let's advance our discussion with an introduction to System Reliability—Strategic Implementation Precepts (Mettler-Wasson, 2006).

34.3.1.8 System Reliability—Strategic Implementation Precepts Although not explicitly addressed in most literature, System Reliability is founded on a multi-faceted philosophy that includes several *precepts:*

- **Precept #1**—Every SYSTEM or ENTITY has a Probability of Success, $P_{SUCCESS}$, that is determined by its Pre-Mission Operational Health and Status (OH&S); mission duration, Uses Case (UCs) and scenarios, and prescribed set of mission OPERATING ENVIRONMENT conditions.
- **Precept #2**—SYSTEM, ENTITIES, or PARTS have an *inherent failure rate* at delivery due to *latent defects* such as design errors and flaws, component/material properties, and workmanship processes and methods.
- **Precept #3**—A SYSTEM or ENTITY level component's RMA can be approximated and characterized with various mathematical distributions and models.
- **Precept #4**—With proper attention, *latent defects* within a specific SYSTEM or ENTITY are eliminated over time via *Condition-Based Monitoring (CBM)* and *corrective maintenance actions* when identified.
- **Precept #5**—By eliminating potential fail points such as *weak* or *failed components*, the SYSTEM or ENTITY *failure rate* decreases, thereby improving the reliability, assuming the SYSTEM is deployed, operated, maintained and sustained according to the System Developer's instructions.
- **Precept #6**—At some point during a mission or *useful service life* of a SYSTEM or ENTITY, failure rates begin to *increase* due to the *combinational* and *cumulative* effects of component failure rates resulting from physical interactions, fatigue and wear-out, and component deterioration due to OPERATING ENVIRONMENT stresses and conditions.
- **Precept #7**—These effects are minimized and the SYSTEM, ENTITY, PART'S *useful service life* extended via a proactive program of system training, proper use, handling, and timely *preventive* and *corrective maintenance* actions at specified operating intervals.

34.3.1.9 The Strategic and Tactical Importance of Fault Elimination

Fault Elimination Principle

Principle 34.6 Every fault—*latent defect*—left undiscovered in a System Design Solution or its implementation is a potential risk that may jeopardize accomplishment of the mission, injury or loss of life to the User and public, result in damage to the SYSTEM OR PRODUCT, or damage to the environment.

When SYSTEMS or PRODUCTS fail, they serve no useful purpose and provide no value to the User. In the case of larger, more complex systems, if they are not operating, they are not *producing* revenue. Instead, they are *consuming* revenue for repairs and are reducing profitability.

Chapter 13, Table 13.1 illustrates what is often referred to as the 100X Rule illustrating the exponentially increasing cost-to-correct errors—faults (Figure 13.1). When a *fault* materializes, the revenue-profitability stream *diminishes*. Therefore, the strategy for eliminating faults (hazards) (Figure 24.1) that can become failures must begin early in the System Development Phase of the System/Product Life Cycle. Figure 13.2 introduced earlier illustrates the importance of reducing *latent defects*—faults—early in the System Development Phase (Figure 12.2). This last point will be addressed later in our discussion of the Bathtub Concept Decreasing Failures Rates (DFR). Elimination of faults early in the System Development Phase (Figure 12.2) provides both tactical and strategic benefits (Figure 13.2).

- **Tactical**—Elimination of faults reduces rework, scrapped ENTITIES and PARTS, reduces technical risk allowing more time for testing and achievement of delivery schedules and systems, products, or services that are compliant with specifications.
- **Strategic**—Delivery of systems, products, or services free of *latent defects* that meet specification requirements is *highly valued* and *remembered* by customers in acquiring new systems.

34.3.1.10 How Complexity Affects System Reliability Enterprises and individuals can debate the importance of topics such as Design for Reliability; yet fail to comprehend the *magnitude* and *significance* of System Reliability estimates until an incident, accident, or catastrophe occurs. At that point it is too late to debate. In general, one of the key characteristics of professionals with real-world experience

TABLE 34.1 How Complexity Affects System Reliability*

Number of Critical Components	Individual Component Reliability (%)			
	99.999	99.99	99.9	99.0
	System Reliability (%)			
10	99.99	99.90	99.00	90.44
100	99.90	99.01	90.48	36.60
250	99.75	97.53	77.87	8.11
500	99.50	95.12	60.64	0.66
1,000	99.01	90.48	36.77	<0.1
1,000	90.48	36.79	<0.1	<0.1
10,0000	36.79	<0.1	<0.1	<0.1

*It is assumed that all critical components are reliability wise in series.
Source: Kececioglu, 2002, Table 1.4, p. 12 Used with Permission.

such as Engineers, physicists, doctors, is *comprehension* and *appreciation* of the *magnitude* and *ramifications* of their decisions. In the case of System Reliability, Kececioglu (2002) addressed this issue very effectively as shown in Table 34.1. Observe the impact on overall System Reliability due to Individual Component marginal reliability changes as a function of SYSTEM *complexity*—Number of Critical Components.

Heading 34.1 Engineers emerging from SE courses often erroneously perceive RMA as *plugging and chugging* equations. Equations for RMA are simply tools to compute estimates. The reality is R&M is part of a larger domain referred to as *Life* or *Lifetime Data Analysis*. Therefore, to shift this narrow-focused paradigm, let's establish a proper foundation in Lifetime Data Analysis Concepts.

34.3.2 Lifetime Data Analysis Concepts and Distributions

One of the challenges in understanding System Reliability is an exclusive focus on the topic without understanding its context within a higher-level realm, Life Data Analysis. Reliability is one of the key Lifetime Data functions. To better understand Lifetime Data Concepts, let's use a simple example as an illustration. Figure 34.1 provides a graphical view.

Example 34.5 **Fundamental Lifetime Data Concepts**

Let's assume that we conduct a test with a statistically valid sample quantity of a System or Entity and allow the test to Run-to-Failure (RTF) (Figure 34.1 Panel A) until all the Units-Under-Test (UUTs) fail. We take of sample of 31 units, place them in test fixtures and controlled OPERATING ENVIRONMENT, activate all units simultaneously, await each one to fail, and record their RTF Elapsed Time as shown in Panel B. Then, plot the discrete test results as shown in Panel C. Observe that the Panel C plot represents the *failure frequency* or density of discrete test values. As a profile, the Failure Density plot represents the distribution of *discrete*, *instantaneous*, Failure Rates, $\lambda(t)$, as a function of time.

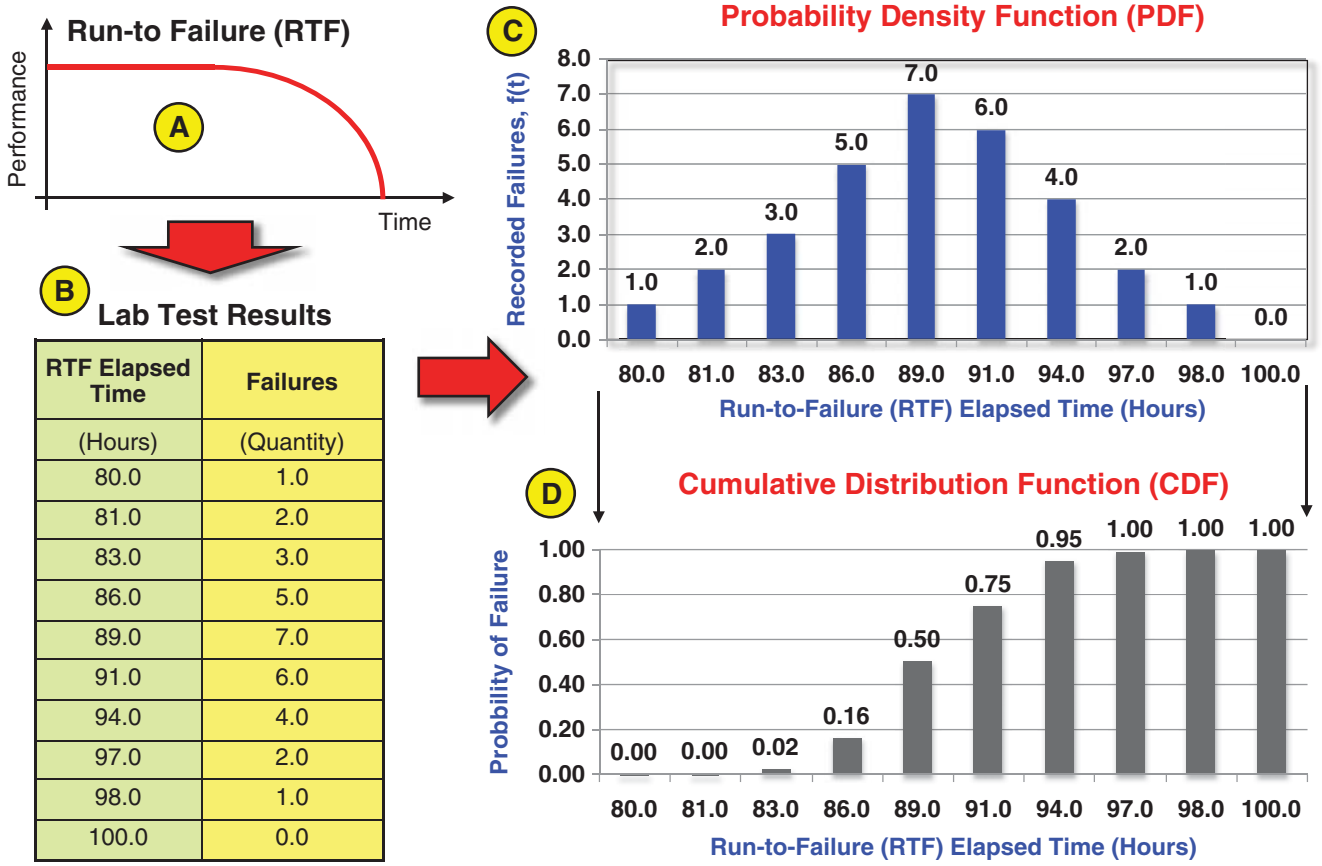
Analyzing Panel C, we observe that the first failure occurred at $t = 80.0$ hours and the last failure at $t = 98.0$ hours. Observe that at $t = 89.0$ hours, 7 units failed. Additionally, we see that the failure distribution approximates a bell-shaped Gaussian (Normal) Distribution with a central mean at $t = 89.0$ hours. Collectively, we can state that the MTTF of the 20 units is 89.0 hours. Since the instantaneous Failure Rates, $\lambda(t)$, distribution is *conditional* based on each UUT's operating condition, statistically we refer to it as a Probability Density Function (PDF). The PDF provides information from which the *likelihood* or *probability* of failure at a specific instant of time, t, can be computed.

As discrete Failure Rates, $\lambda(t)$, suppose we are asked: from $t = 80.0$ hours until $t < 98.0$ hours, how many units can we expect to fail at a specific instant in time? In the range from $t = 80.0$ hours up to $t = 98.0$ hours, we can say that at any instant in time a portion of the 31 units have failed—UnitsFailed; others continue to survive—UnitsSurviving. Net result: UnitsFailed + UnitsSurviving = 31. Panel C illustrates that we can also *manually compute* the *cumulative* quantity of failures as a function of time – the Cumulative Distribution Function (CDF), F(t) - over the interval from 80.0 hours $< t <$ 98.0 hours.

Manual computations may be *acceptable* in some applications; however, they are *insufficient* where Mission Reliability, System Reliability, and cost are Critical Operational and Technical Issues (COIs/CTIs). For example, the cost of maintenance for large populations of production units can be very expensive if performed *prematurely*. Conversely, if maintenance is not performed in advance of failure conditions (Figure 34.25), *Mission Reliability* and *System Reliability* become High Risk, which could jeopardize the mission and potentially have *catastrophic* results to the User, the EQUIPMENT, the public, or the environment.

So, what is the *optimal* solution? We know that Failure Rate, $\lambda(t)$, profiles exhibit characteristics similar to mathematical curves. Therefore, the answer is to *exploit* mathematical tools. These tools enable us to scientifically refine the Just-in-Time (JIT) *optimal* Reliability and Maintenance (R&M) solution (Figure 34.24).

How do we accomplish this? The answer resides in Lifetime Data Functions and their distributions. The distributions enable us to mathematically characterize Failure Rate $\lambda(t)$, profiles via a "curve fitting" transfer function *validated* by actual test sample results. Mathematical transfer functions provide a *continuum* of Failure Rates, $\lambda(t)$, and *avoid* costly approximations via extrapolations. Therefore, Lifetime Data



RTF Elapsed Time	Failures
(Hours)	(Quantity)
80.0	1.0
81.0	2.0
83.0	3.0
86.0	5.0
89.0	7.0
91.0	6.0
94.0	4.0
97.0	2.0
98.0	1.0
100.0	0.0

Figure 34.1 Failure Distribution Profile Examples Profile Example Illustrating the Probability Density Function (PDF) and the Cumulative Density Function (CDF)

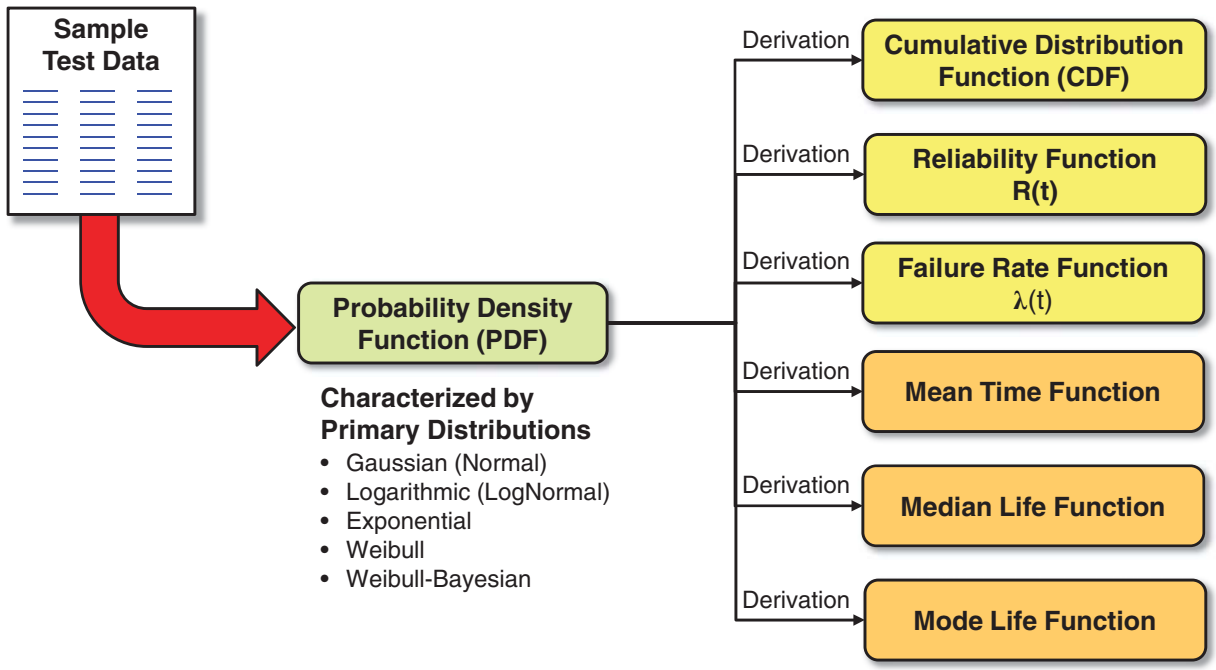


Figure 34.2 Life Data Functions and Their Traceability

Functions and Distributions will be the subject of our discussions in this section.

First, a word about Failure Rate, $\lambda(t)$.

34.3.2.1 Understanding the Failure Rate, $\lambda(t)$

Analytically, our discussion of failures requires more than casual references to events. We need to be able to *quantify*—bound and specify—a *failure* in terms of its probability of occurrence, $P_x(t)$. However, since manufactured PART Level components exhibit inherent variations in their material properties and characteristics that are later stressed (Figure 34.11) by OPERATING ENVIRONMENT conditions, loading effects, operational cycles, we can expect *random* failures to occur over some time period as illustrated in Figure 34.1 Panel C. Therefore, a sample of one PART is *insufficient* for deriving inferences about a larger population of identical PARTS. As a result, we subject a statistically valid sample of identical components to identical operating conditions to observe and characterize their failure rate profiles and enable us to mathematically model their failure characteristics. We mathematically represent the Failure Rate of a distribution as:

$$\text{Failure Rate} = \lambda(t) \quad \text{for} \quad t \geq 0 \tag{34.1}$$

where:

- $\lambda(t)$ represents the *frequency of failure* or *failure density* as a function of: (1) a specified set of OPERATING ENVIRONMENT conditions and (2) the continuous *random* variable time, t.

Author's Note 34.1

Failure Rates

In a later discussion of the Probability Density Function (PDF) we will address the need to refine *failure rate* into two key types: Instantaneous Failure Rate, $\lambda(t)_{\text{Inst}}$, and the Average Failure Rate, $\lambda(t)_{\text{Avg}}$.

34.3.2.2 Lifetime Data Analysis and Distributions

Principle 34.7

Lifetime Data Functions Principle

Every system, product, or service is characterized by seven Lifetime Data Functions:

1. The Probability Density Function (PDF)
2. The Cumulative Distribution Function (CDF), F(t)
3. The Reliability Function, $R(t)$
4. The Failure Rate Function, $\lambda(t)$
5. The Mean Time Function
6. The Median Life Function
7. The Mode Life Function

Principle 34.8

Lifetime Data Profile Principle

Every SYSTEM or ENTITY has a unique Lifetime Data Distribution profile that characterizes its failure frequency or density as a function of the continuous variable, *time*.

If you analyze the life profiles exhibited by individual mechanical, electrical, fluid components and materials, you will find a variety of characteristic curves such as those shown in Figure 34.3. Since each curve characterizes the life profile of specific types of a SYSTEM or ENTITY, we refer to them as Lifetime Data Distributions. When you integrate these—PARTS → SUBASSEMBLIES → ASSEMBLIES → SUBSYSTEMS → PRODUCTS—into a higher level SYSTEM Reliability, what emerges is an overall SYSTEM Level Input/Output (I/O) *transfer function*. Each *transfer function* has its own unique characteristic profile that is often modeled using a Weibull Distribution discussed later.

Across the spectrum of Figure 34.3 characteristic profiles, Life Data Analysis applies and customizes mathematical models to achieve a "best fit" approximation to the physical failure distribution characteristics of a SYSTEM or ENTITY. The most commonly used Lifetime Data distributions include: (1) the Gaussian (Normal) Distribution (Figure 34.3 Panel A), (2) the Log-Normal Distribution (Figure 34.3 Panel B), (3) the CDF (Figure 34.3 Panel C), (4) the Exponential Distribution (Figure 34.3 Panel D), and (5) the Weibull Distribution (Figure 34.4). A sixth distribution, Bayesian-Weibull, is also used. The reality is: most Run-to-Failure (RTF) reliability tests do not neatly conform to these distributions. The distributions are intended to reasonably approximate the statistical characteristics of the set of UUTs failure data.

A Word of Caution 34.2

Proper Selection of a SYSTEM or ENTITY's Failure Distribution Profile

One of the challenges in Lifetime Data Analysis is selecting a data distribution profile that *accurately* represents a "best fit" to a component's characteristic profile.

Sometimes for expediency, R&M Engineers and others will employ a distribution such as the Exponential Distribution to get an interim "Quick look" estimate. Unfortunately, this becomes *the* analytical result that is rationalized as "good enough." Besides, who is going to challenge it!

From an Engineering best practices perspective, exercise caution about life data distribution estimates and ensure they are rationalized and documented for future reference!! If the R&M Engineer is *unwilling* to do this, it should immediately raise a flag of concern.

Given this introduction to Lifetime Data Functions, let's begin with the PDF.

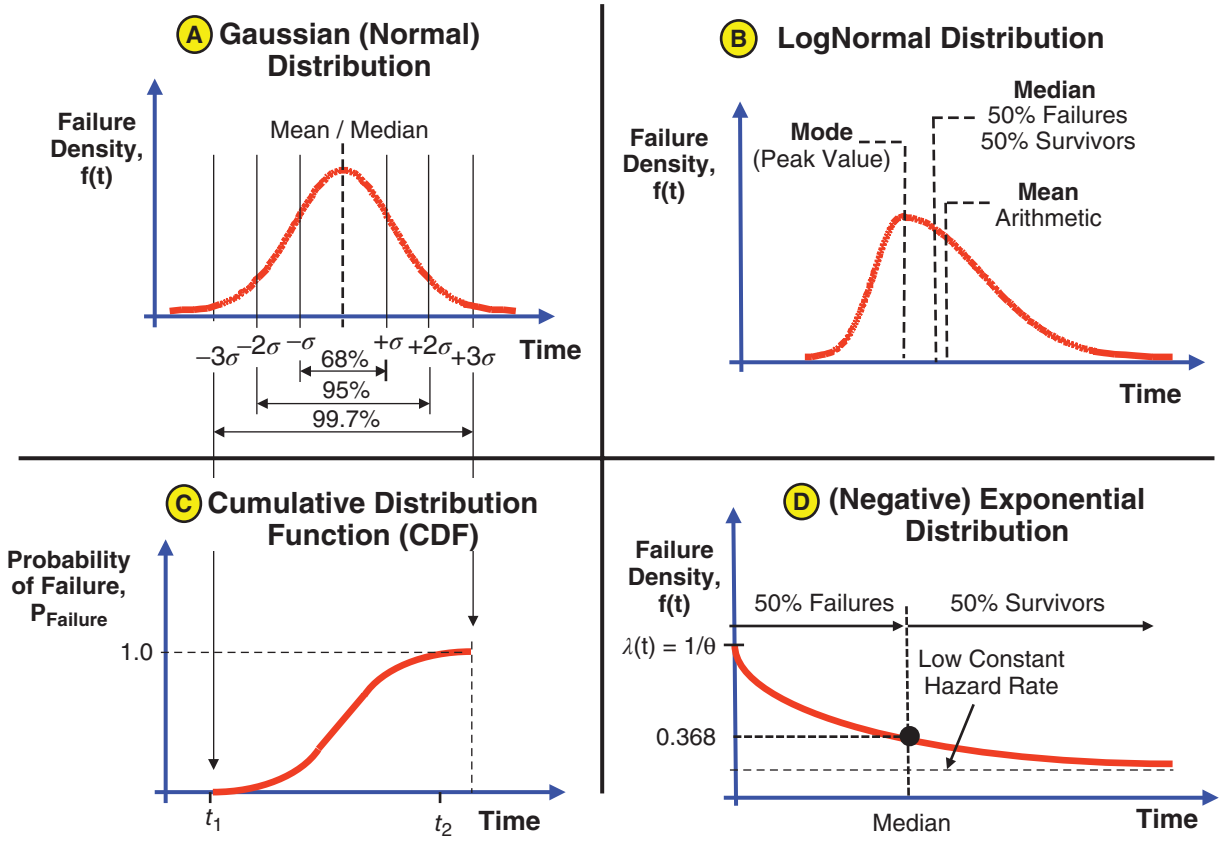


Figure 34.3 Basic Types of Lifetime Data Analysis Distributions

34.3.2.3 The Probability Density Function (PDF)

Probability Density Function (PDF) Principle

Principle 34.9 The Probability Density Function (PDF) serves as the foundation for deriving all other Lifetime Data Distributions—the Cumulative Distribution Function (CDF), the Reliability Function ($R(t)$), the Failure Rate Function ($\lambda(t)$), the Mean Time Function, the Median Life Function, and the Mode Life Function.

PDF Profiles Principle

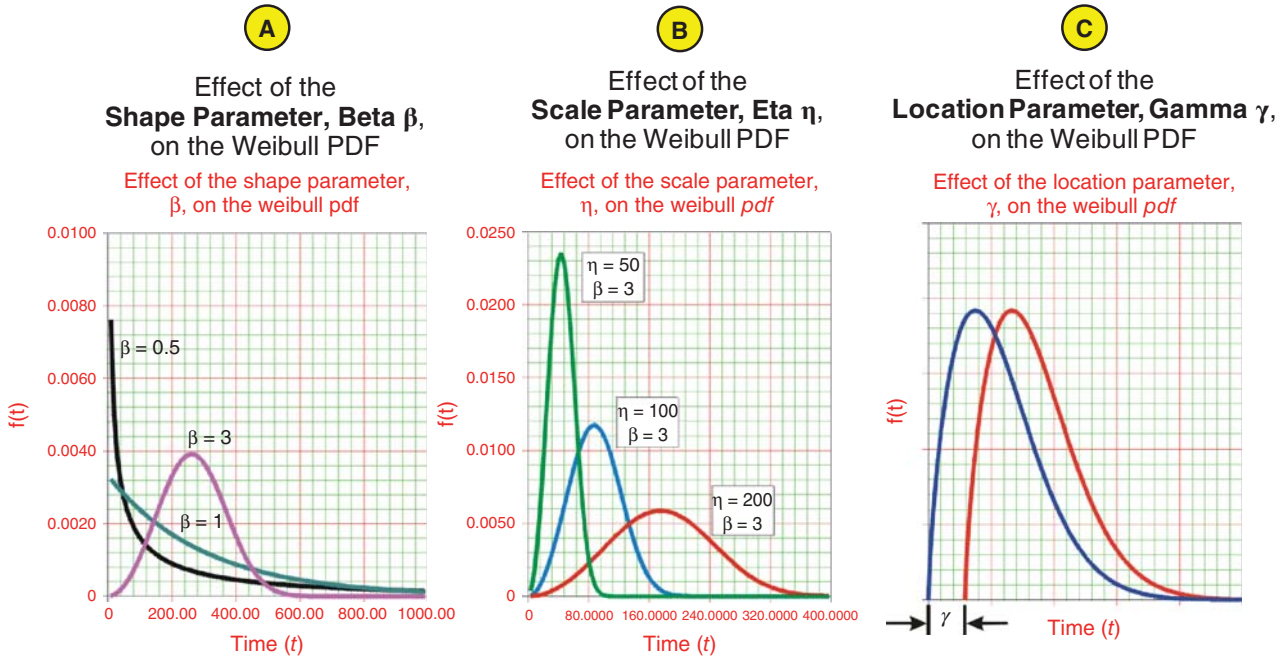
Principle 34.10 The Probability Density Function (PDF) is characterized by five primary Lifetime Data Distributions:

1. The Exponential Distribution
2. The Gaussian (Normal) Distribution
3. The Log-Normal Distribution
4. The Weibull Distribution
5. The Bayesian-Weibull Distribution

Lifetime Data Distributions are expressed in terms of the Failure PDF, $f(t)$. In fact, the PDF serves as the *cornerstone* for deriving the Cumulative Distribution Function (CDF), the Reliability Function, the Function, the Mean Time Function, the Median Life Function, and the Mode Life function as illustrated in Figure 34.2. Observe how the PDF can be characterized by a variety of commonly used distributions such as Gaussian (Normal), LogNormal, Exponential, Weibull, and Weibull-Bayesian. To better understand Lifetime Distributions, let's begin with the PDF.

As its name infers, the PDF is used to express the probability of a failure at time, t. Think of a PDF profile as a *histogram* with a series of Δt vertical *time increments* as shown in Figure 34.1 Panel C. The magnitude of each Δt time increment represents the *quantity* of failures expected to occur at a specific instance of time. From a probability perspective, the area underneath the PDF represents the *probability of failure* for a SYSTEM or ENTITY as a function of the continuous variable time, t (Figure 34.1 Panel B).

The PDF expresses an *estimate* of the probability of an event based on observations, objective evidence—measured data, and validation of actual SYSTEM or ENTITY sample or field data. In general, the PDF enables R&M Engineers to answer the question: *At a specific instance of time, $t = t_x$, what is the probability that a SYSTEM or ENTITY X will fail*? For example, in Figure 34.2 Panel C if: (1) laboratory or field data indicate a specific component has a Failure Rate, $\lambda(t)$, of 7 units at $t = 89.0$ hours and (2) the elapsed time of $t = 89.0$ hours will occur during a System's mission, you should



References: Used by permission.

- Source: Weibull.com (http://www.weibull.com/basics/parameters.htm)
- Panel A - www.weibull.com/basics/weibull_beta.gif
- Panel B - www.weibull.com/basics/weibull_eta.gif
- Panel C – www.weibull.com/basics/weibull_gamma.gif

Figure 34.4 Weibull Distribution Examples Illustrating the Shape, Scale, and Location Parameter Effects (Source: Weibull.com, 2014 – Used with Permission)

consider performing a preventive or corrective maintenance action prior to the mission.

34.3.2.3.1 The Exponential Distribution PDF One of the simplest and most commonly used distributions is the Exponential Distribution shown in Figure 34.3 Panel D. Unfortunately, as a result of its simplicity, the Exponential Distribution is one of the most *misapplied* distributions and is often applied to SYSTEMS or ENTITIES that exhibit a *different* characteristic profile such as a Normal or LogNormal Distribution.

The Exponential Distribution occurs in two forms: (1) a one-parameter distribution and (2) a two-parameter distribution. The two-parameter distribution, which is a refinement of the one-parameter distribution, accommodates distributions that are *offset* along the time axis away from the origin.

THE TWO-PARAMETER EXPONENTIAL PDF DISTRIBUTION Mathematically, we can express the Exponential Distribution PDF as a Two-Parameter function:

$$f(t) = \lambda e^{-(t-\gamma)} \quad \text{for } f(t) \geq 0,\ t \geq 0, \lambda > 0 \qquad (34.2)$$

where:

- $f(t)$ characterizes the Exponential PDF over a specific time interval.
- λ(Lambda) represents the Failure Density at time, t.
- λ(Lambda) represents the Location offset parameter representing the curve start point to the right of the origin.
- t represents the Elapsed Time in operational hours, cycles, and so forth.

ONE-PARAMETER EXPONENTIAL PDF DISTRIBUTION If the Location Offset parameter, γ, $= 0$ in Eq. 34.2, the result is a One-Parameter Exponential PDF:

$$f(t) = \lambda e^{-\lambda t} \quad \text{for} \quad f(t) \geq 0,\ t \geq 0, \lambda > 0 \qquad (34.3)$$

REPAIRABLE, NON-REPAIRABLE, AND REPLACEMENT SYSTEMS

MTBF-MTTF Principle

Principle 34.11 MTBF represents the Mean Life (θ) of a statistically valid sample of a population of *repairable* items, MTTF represents the Mean Life (μ) of a statistically valid sample of a population of *non-repairable* items.

The Failure Rate, $\lambda(t)$, represents the quantity of failures per unit of measurement such as failures/hour, failures/cycle, and so forth. We know that MTTF (μ) represents the *mean* time of all failures within a SYSTEM OR ENTITY's Exponential PDF profile. However, the value of the Failure Density, $\lambda(t)$ is contextual depending on whether the SYSTEM or ENTITY is

repairable or *non-repairable*. Let's define what is meant by a *repairable* SYSTEM or ENTITY:

- **Repairable Item** "An item of a durable nature which has been determined by the application of engineering, economic, and other factors to be the type of item feasible for *restoration* to a serviceable condition through regular repair procedures" (DAU, 2012, p. B-190).

Observe the operative term *restoration* in the Repairable Item definition. Due to levels of abstraction and integration, these terms are contextual. For example, a higher level SYSTEM or ENTITY may fail due to a lower level PART failure. Hypothetically, one could say that the higher level SYSTEM or ENTITY is a *repairable item* via *restoration*.

In contrast, the failed PART, which may not be restorable may be corrected with (1) an identical Part Number from the same manufacturer or (2) with a *replacement item*, which may not be identical to the ENTITY that has failed but conforms to its electro-mechanical interface and space boundary conditions. For example, *replacement* parts in the commercial marketplace may only be available from "After Market" vendors. Let's define the term:

- **Replacement Item** "An item which is *replaceable* with another item, but which may differ physically from the original item in which the installation of the replacement item requires operations such as drilling, reaming, cutting, filing, shimming, etc., in addition to the normal application and methods of attachment" (MIL-STD-480B, p. 11).

Given an understanding of *repairable* and *replacement* SYSTEMS or ENTITIES, let's differentiate the MTBF for *repairable* systems versus MTTF for *non-repairable* systems.

Figure 34.1 Panel C illustrates the Time-to-Failure (TTF) or Run-to-Failure (RTF) concept, which *applies to both* repairable and non-repairable systems but ends there. *Non-Repairable* systems are single use, Removed and Replaced (R&R), and disposed. So, MTTF appropriately describes their TTF.

R*epairable* systems, which are reusable, have a different context of usage. In those cases, we are interested in: (1) the *mean* time period a SYSTEM or ENTITY will operate before it is expected to fail – MTTF - and (2) the mean time period we can expect for repairs – MTTR – to restore it to an operable condition enabling a Return to Service (RTS). Mathematically, we can express MTBF as:

$$\text{MTBF} = \text{MTTF} + \text{MTTR} \tag{34.4}$$

Therefore, MTBF > MTTF for repairable SYSTEMS or ENTITIES. Despite that reality and the approximate similarity of the magnitudes some Engineers *erroneously* equate MTBF of *repairable* items as being equivalent to the MTTF for *non-repairable* items. So, *how does this impact the Exponential PDF*?

We know that RTF or TTF and Failure Rate, $\lambda(t)$, are inversely proportional. Therefore, for *repairable* systems, $\lambda(t)$ is the *reciprocal* of Mean Time Between Failure (MTBF), which is represented by the Greek letter Theta (θ). We express it mathematically as:

$$\lambda(t)_{\text{repairable}} = \frac{1}{\text{MTBF}} = \frac{1}{\theta} \tag{34.5}$$

For *non-repairable* systems, $\lambda(t)$ is the *reciprocal* of the MTTF represented by the Greek letter Mu (μ). We express it mathematically as:

$$\lambda(t)_{\text{nonrepairable}} = \frac{1}{\text{MTTF}} = \frac{1}{\mu} \tag{34.6}$$

For *repairable* systems, substituting Eq. 34.5 in Eq. 34.3 yields the following result:

$$f(t)_{\text{Repairable}} = \frac{1}{\theta} e^{-t/\theta} \quad \text{for } f(t) \geq 0,\ t \geq 0,\ \theta \geq 0 \tag{34.7}$$

For non-*repairable* systems, substituting Eq. 34.6 in Eq. 34.3 yields the following result:

$$f(t)_{\text{nonrepairable}} = \frac{1}{\mu} e^{-t/\mu} \quad \text{for } f(t) \geq 0,\ t \geq 0,\ \mu \geq 0 \tag{34.8}$$

Since MTBF > MTTF (Eq. 34.4), therefore $f(t)_{\text{Repairable}} \neq f(t)_{\text{Unrepairable}}$.

THE EXPONENTIAL PDF SCALING PARAMETER, GAMMA (γ) Our discussion of the Failure Rate, $\lambda(t)$, as a magnitude also has noteworthy *graphical* implications. Specifically, at $t = 0$, $f(t)$ has a y-intercept value of $1/\theta$ in Eq. 34.7 or $1/\mu$ in Eq. 34.8. Therefore, $\lambda(t) = 0$ serves as a *Scaling Parameter* for the Exponential PDF Distribution.

EXPONENTIAL PDF APPLICATION The Exponential Distribution PDF is typically used with SYSTEMS or ENTITIES that can be characterized by a Negative Exponential Distribution. Nelson (1990) observes that based on his experience the Exponential Distribution:

- "… described the life of insulating oils and fluids (dielectrics) and certain materials and products" Nelson (1990, p. 53).
- "… characterizes only 10–15% of products in the lower tail of the distribution" Nelson (1990, p. 54).

Nelson (1990) advises that the Weibull Distribution, which is addressed in Figure 34.4, or another distribution may be more appropriate.

THE GAUSSIAN (NORMAL) LIFETIME DISTRIBUTION Some SYSTEM, ENTITIES, or PARTS exhibit failure characteristic profiles that resemble the Gaussian (Normal) Distribution such as the one illustrated in Figure 34.3 Panel A. Graphically, the bell-shaped, symmetrical curve consists of a central mean with diminishing values that taper down and extend toward $-\infty$ and $+\infty$.

The Gaussian (Normal) Distribution is used to plot the failure frequency distribution of SYSTEMS or ENTITIES that exhibit characteristics that match the profile such as light bulbs. From a Life Data perspective, the frequency of failures, which vary as a function of the *continuous* random variable time t and specific type of component characterizes its PDF profile.

Mathematically, the Gaussian (Normal) Distribution PDF can be expressed as:

$$f(t) \equiv \frac{1}{\sigma\sqrt{2\pi}} e^{1/2\left(\frac{t-\mu}{\sigma}\right)^2} \quad \text{for } t \geq 0 \qquad (34.9)$$

where:

- $f(t)$ = Exponential PDF
- σ = Standard deviation
- μ = Central mean of the distribution
- t = Continuous variable representing lapsed time

Since the Gaussian (Normal) Lifetime Distribution is symmetrical, its *median* and *mode* are all coincidental with its *mean*, μ, shown in Figure 34.3 Panel A.

34.3.2.3.2 The Log-Normal Lifetime Distribution Failures do not always occur as *symmetrical* PDFs. Although a Failure PDF exhibits a *mean*, a *mode*, and a *median*, one side may be skewed leaving an *unsymmetrical* appearance as shown in Figure 34.3 Panel B. Since the failure frequency occurrences resemble a Gaussian (Normal) Distribution but are dispersed as a logarithmic function of the *continuous* random variable *time t,* the profile is referred to as a Log-Normal Distribution.

Log-Normal distributions are often used in applications such as fatigue-based cycles to failure, material strengths, variable loading effects, and corrective maintenance actions. Mathematically, the PDF of the Log-Normal Distribution is defined as:

$$f(t) = \frac{1}{\sigma t\sqrt{2\pi}} exp\left[-\frac{1}{2}\left(\frac{\text{In}t-\mu}{\sigma}\right)^2\right]$$
$$\text{For } -\infty < \mu < \infty, t > 0, \sigma > 0 \qquad (34.10)$$

where:

- t' = natural log of t representing the values for Time to Failure (TTF)
- μ' = Central Mean of the natural logarithms of the TTF
- σ' = Standard deviation of the natural logarithms of the TTF

34.3.2.3.3 The Weibull Lifetime Distribution Observe the locations of the Log-Normal Distribution's *mean*, *median*, and *mode*. *Lifetime data* do not always conveniently fit distributions such as the Exponential, Gaussian (Normal), or Log-Normal. *Curve fitting* or *shaping* becomes the prescriptive method for ensuring a "best fit" distribution based on sampled data.

One method for shaping the distribution to achieve a best-fit match for the sampled data is the Weibull Distribution created by Dr. E. H. Waloddi Weibull, a Swedish engineer, scientist, and mathematician (Wikipedia, 2013a). Whereas most distributions have a unique profile appearance, the three control parameters—*scale, shape,* and *location*—provide a powerful combination and flexibility to recreate most Life Data distributions, shown in Figure 34.4.

THE THREE-PARAMETER WEIBULL DISTRIBUTION As a Three-Parameter Weibull Distribution, we can express PDF mathematically as (ReliaWiki.org, 2013a):

$$f(t) = \frac{\beta}{\eta}\left(\frac{t-\gamma}{\eta}\right)^{\beta-1} e^{-\left(\frac{t-\gamma}{\eta}\right)^{\beta}} \quad \text{for} \quad t \geq 0 \qquad (34.11)$$

where:

- β (Beta) = Shape parameter
- η (Eta) = Scale parameter
- γ(Gamma) = Location offset parameter

To better understand each of these parameters, let's describe it graphically using Figure 34.4:

- **Shape Parameter (Beta, β)**—A *unitless* quantity that controls the *shape* of the distribution. For applications such as the Gaussian (Normal) Distribution, the shape would be symmetrical resembling a bell-shaped curve. Figure 34.4 Panel A illustrates some of the effects as the Shape Parameter (Beta, β) is assigned different values. The Shape Parameter can be very useful, especially in characterizing the Failure Rate, $\lambda(t)$, and Reliability Function, $R(t)$, introduced later. Speaks (2013, p. 5) makes several points concerning the Weibull Shape Parameter:
  - For $\beta < 1$, the shape of Weibull Distribution models approximate the early failure profiles—Bathtub Concept Decreasing Failure Region (DFR) introduced later.

- For $\beta = 1$, the shape of Weibull Distribution models approximate the Exponential PDF—Bathtub Concept Stabilized Failure Region (SFR) introduced later.
- For $\beta = 3$, the shape of Weibull Distribution models approximate the Gaussian (Normal) PDF.
- For $\beta = 10$, the shape of Weibull Distribution models approximate the End-of-Life Wear-out—Bathtub Concept Increasing Failure Region (IFR) introduced later.

- **Scale Parameter (Eta, η)**—A *unitless* quantity that controls the *scale* of the Weibull Distribution in terms of its *width*—standard deviation. If we had constructed a Gaussian (Normal) Distribution using Eq. 34.9, the Scale Parameter (Eta, η) would be determined by the distribution's standard deviation. Figure 34.4 Panel B illustrates some of the effects as Eta is assigned different values such as 50, 100, and 200. Observe the pattern $1/\eta$. You should recognize the pattern as the Failure Rate, $\lambda(t)$ expressed in Eqs. 34.7 and 34.8 for repairable and non-repairable systems.
- **Location Parameter (Gamma, γ)**—A *unitless* quantity that controls the *location* of the distribution relative to the origin ($t = 0$) along the abscissa of the graph. If we had constructed a Gaussian (Normal) Distribution the Location Parameter, Gamma (γ), represents the *offset* location of the start of the curve along the horizontal *time* axis. Figure 34.4 Panel C illustrates one of the offset effects as Gamma is assigned a different value.

How are the Scale (Beta), Shape (Eta), and Location (Gamma) parameters determined? In general, the answer depends on the data and the distribution profile selected. A System Analyst could approach the task on a *trial-and-error basis*. Reliawiki.org (2013c) suggests consideration of methods such as: probability plotting, Rank Regression on X (RRX), Rank Regression on Y (RRY), and Maximum Likelihood Estimation (MLE).

THE TWO-PARAMETER WEIBULL DISTRIBUTION For applications in which there is no time-based location offset ($\gamma = 0$), the PDF becomes a Two-Parameter Weibull Distribution. We can express the PDF as a function of the Shape Parameter, Beta (β), the Scale Parameter, Gamma (γ), and the *continuous* variable time, t, as (ReliaWiki.org, 2013a):

$$f(t) = \frac{\beta}{\eta}\left(\frac{t}{\eta}\right)^{\beta-1} e^{-\left(\frac{t}{\eta}\right)^{\beta}} \quad \text{for } t \geq 0 \tag{34.12}$$

THE ONE-PARAMETER WEIBULL DISTRIBUTION For SYSTEM or ENTITY PDFs that are consistent, β can be a known constant based on sampled data unique to your SYSTEM or ENTITY. Mathematically, Eq. 34.11 can be expressed as a One-Parameter Weibull PDF as a function of the Scale Parameter, Gamma (γ), and the *continuous* variable time, t, as (ReliaWiki.org, 2013a):

$$f(t) = \frac{k}{\eta}\left(\frac{t}{\eta}\right)^{k-1} e^{-\left(\frac{t}{\eta}\right)^{k}} \quad \text{for } t \geq 0 \tag{34.13}$$

where:

- k = known constant.

34.3.2.3.4 The Bayesian-Weibull Lifetime Distribution One other type of PDF distribution used is the Bayesian-Weibull Distribution. The context of this distribution is based on prior knowledge of the shape of the PDF for a specific ENTITY by the System Analyst. Specifically, the Shape Parameter, Beta, (β). Construction of the distribution includes consideration of the "variation and uncertainty" that may exist in the Shape Parameter, Beta (β) (Reliawiki.org, 2013b).

34.3.2.3.5 Lifetime Data Distributions Summary In summary, Lifetime Data analysis and mathematical statistics provide powerful tools to model various types of failure distributions. These include: the Exponential, Gaussian (Normal), Log-Normal, Weibull, and Bayesian-Weibull Distributions. For additional information about various Life Data Distributions, refer to NIST (2013, Section 1.3.6.6).

34.3.2.3.6 The Cumulative Distribution Function (CDF) Let's assume an Enterprise has X units of a SYSTEM or PRODUCT in field operation. A question might be: *how many units do we expect to fail by time, $t = t_x$*? The context of this question is not about a specific *instant* of time; it is about the *cumulative* failures over the time interval *from* $t = 0$ until $t = t_x$. This brings us to our discussion of the CDF, $F(t)$.

Author's Note 34.2

$F(t)$ versus $f(t)$ Capitalization, $Q(t)$ versus $F(t)$

Observe that the PDF is characterized by a *lower case* $f(t)$ and the CDF by a *capital* $F(t)$. Some textbooks use the parameter $Q(t)$ in lieu of $F(t)$.

To better understand the CDF concept, let's use the Gaussian (Normal) Distribution illustration shown in Figure 34.3 Panel A as a precursor to our mathematical discussion. Since we are dealing with probability, we know that the *area* underneath the Lifetime Data Distribution profile is equal 1. To understand the probability of failure expected to occur over the time interval $0 \geq t \geq t_x$, we integrate the Exponential Failure PDF, $f(t)$.

The cumulative quantity of failures over the time interval *from* $t = 0$ to $t = t_x$ are shown in Figure 34.1 Panel D. This enables us to establish the relationship between the Gaussian

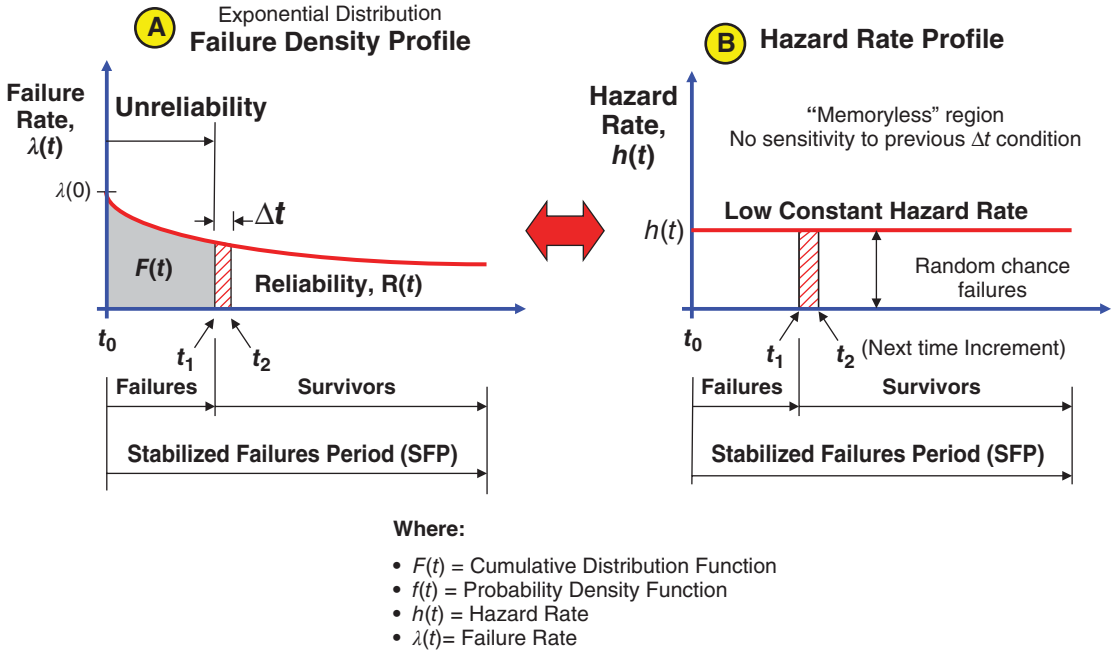


Figure 34.5 Differences Between the Failure Density Profile, *f(t)*, and the Hazard Rate, *h(t)*, Profile

(Normal) LogNormal, and Exponential Distributions, $F(t)$ over a specified time interval. To characterize the relationship mathematically:

$$F(t_a \leq t \leq t_b) = \int_{t_a}^{t_b} f(t)dt$$

$$\text{for } 0 \leq t_a \leq t \leq t_b \text{ over the range 0 to } +\infty \tag{34.14}$$

where:

- $f(t)$ = Failure PDF as a function of time, t.
- t = *Continuous* random variable.

Observe the number of failures to the *left* of a line represented by time, t1, in the PDF distribution shown in Figure 34.5 Panel A. The areas under this portion of the PDF represents what is referred to as the SYSTEM or ENTITY's *unreliability*. We will address this point in detail later in our next topic, the Reliability Function.

For an Exponential PDF substituting Eq. 34.3 in Eq. 34.14, we can express the CDF as:

$$F(t) = \int_0^t \lambda e^{-\lambda t} dt \quad \text{for } t \geq 0 \tag{34.15}$$

In summary, we have:

- Introduced the CDF, $F(t)$, in terms of its relationship to the PDF - Gaussian (Normal), LogNormal, Exponential, Weibull, and Weibull-Bayesian Distributions.
- Defined the CDF, $F(t)$, as a measure of the System or Entity's Probability of Failure, P_{Failure} or *unreliability* – as a function of the continuous variable time, t.

This last point is very important as we go forward to our next topic, The Reliability Function.

34.3.2.4 The Reliability Function Whereas engineers, managers, and executives, prematurely launch into computing the Reliability for a SYSTEM or ENTITY, you should *recognize* and *appreciate* that Reliability is just one of the Seven Life Data functions (Principle 34.7) required to understand and perform R&M.

To better understand the concept and computation of the Reliability Function, let's build on the SYSTEM or ENTITY's *unreliability* introduced in the preceding discussion. To understand *reliability*, we need to define its complement *unreliability*.

Using probability theory, we can state that the Probability of Success, P_{Success}, and the Probability of Failure, P_{Failure}, as a function of elapsed time are complementary. Since the area underneath the PDF distribution is normalized to 1.0, we can mathematically express this relationship as a function of the continuous variable *time*, t, as:

$$P_{\text{Success}}(t) \equiv 1.0 - P_{\text{Failure}}(t) \quad \text{for } t \geq 0 \tag{34.16}$$

If the degree of *success* represents the *reliability* of a SYSTEM or ENTITY at a specific instant in time for a prescribed set of OPERATING ENVIRONMENT conditions, we equate:

$$P_{\text{Success}}(t) \equiv \text{Reliability},\ R(t) \quad \text{for } t \geq 0 \tag{34.17}$$

Failures, as random time and condition-based functions, can occur throughout a mission. A SYSTEM or component's Probability of Failure, P_{Failure} or Mission Unreliability, is represented by its CDF, $F(t)$, at a specific instant in time. Therefore, by substituting Eq. 34.17 into Eq. 34.16 we can express Reliability $R(t)$ as:

$$R(t) \equiv 1 - P_{\text{Failure}}(t) \quad \text{for } t \geq 0 \tag{34.18}$$

We can state that $R(t)$ represents the *probability* that a SYSTEM OR ENTITY will *survive* a planned mission of a specific duration under a prescribed set of OPERATING ENVIRONMENT conditions without failure. We refer to this bounded condition as the *System Reliability*, which is also referred to as the *Survival Function*.

Substituting Eq. 34.14 in Eq. 34.18, we can express reliability, $R(t)$, over the interval *from* t_0, the installation time, to t_1, the current time, in terms of the CDF, $F(t)$, as:

$$R(t) = 1 - F(t) = 1 - \int_{t0}^{\infty} f(t)dt \tag{34.19}$$

where:

$f(t)$ = Exponential Failure PDF.

Since the total area under the PDF curve is unity (1.0), we can substitute Eq. 34.3 into Eq. 34.19 for the interval $t_0 \rightarrow t_\infty$:

$$R(t) = 1 - \int_{t0}^{\infty} \lambda e^{-\lambda t} dt \tag{34.20}$$

Engineers often assume that the Mean Life (θ) of a *repairable* SYSTEM or ENTITY represents an instant in time when 50% of the units have *failed* and 50% have *survived*. This is *erroneous*! In fact, it reflects a *convolution* of two different concepts—Mean Life (θ) versus the Lifetime Data Median Life Function introduced in a later section. The Exponential Distribution Function exemplifies this point. The question that should be asked is: *For a* SYSTEM OR ENTITY *characterized by the Exponential Distribution, what is its Reliability—probability of units "surviving"—beyond its Mean Life (θ)?*

Let's assume a SYSTEM or ENTITY exhibits a constant failure rate, $\lambda(t)$, as illustrated in Figure 34.3 Panel D. If we substitute $1/\theta$ (Eq. 34.5) for λ in Eq. 34.20, set $t = \theta$ (Mean Life), and integrate over the time interval from $t_0 \rightarrow t$

$$R(t) = 1 - \int_0^t \frac{1}{\theta} e^{-\frac{t}{\theta}} dt$$

$$R(t) = 1 - [1 - e^{-1}]$$

$$R(t) = 1 - 0.632 = 0.368 \tag{34.21}$$

Therefore, the Reliability, $R(t)$, for a SYSTEM or ENTITY characterized by an Exponential Distribution PDF when 50% of the units have failed (Median) and 50% survive is only 36.8%. This means that:

- 63.2% of the area under the PDF curve lies to the *left* of the Median Life representing units *failed*;
- 36.8% of the area under the PDF curve lies to the *right* of the Median Life representing units *surviving*.

The preceding example clearly illustrates that the Reliability, $R(t)$, for an Exponential PDF Distribution beyond the Mean Life (θ) – 0.368—is not the same as the Median Life Function in which 50% of the units survive either side of the Mean Life (θ).

34.3.2.4.1 Reliability Predictions or Estimates?

RMA Estimates Principle

Principle 34.12 Reliability, Maintainability, and Availability (RMA) are time-variant assessments that produce *estimates*, not *predictions*, due to the *uncertainty* in a SYSTEM OR ENTITY's physical operating condition and mission OPERATING ENVIRONMENT conditions.

Reliability employs statistical methods and techniques based on *approximations* and *assumptions* about a population of physical entities at all levels of abstraction—SYSTEM, PRODUCTS, SUBSYSTEMS, ASSEMBLIES, SUBASSEMBLIES, and PARTS. As a result, you will hear people comment about predicting the reliability of a SYSTEM or ENTITY.

On inspection, the notion of *predictions* is indicative of soothsayers and wizards. The context here is one of *informed* decision-making caveated by assumptions based on *observed and measured factual evidence* employed in and provided by mathematical Modeling and Simulation (M&S). Therefore, if Reliability theory is based on assumptions, observations, approximations, uncertainities, and probabilities, do you *predict* or *estimate* the reliability of a SYSTEM or ENTITY? We *estimate* System Development costs; yet, make Reliability *predictions*? Weibull.com (2013) articulates the point well.

> "Because life data analysis results are estimates based on the observed lifetimes of a sampling of units, there is uncertainty in the results due to the limited sample sizes."

This text refers to computations of a SYSTEM OR ENTITY'S RMA as *estimates*. Computations, however, in the absence of seasoned knowledge, experience, and wisdom are subject to the old cliché: *A fool with a tool is still a fool* (anonymous). When applied correctly by a competent, qualified, R&M Engineer, *estimates* can be as cost-effective and accurate as in-depth analyses that require assumptions and consume extensive resources without necessarily improving the reliability of the product. Reliability *estimates* coupled with "worst case" analyses can be very effective.

RELIABILITY FUNCTION SUMMARY The preceding discussion brings us to a key closing point:

Always bound and express Reliability Estimates in terms of five key criteria:

- **Criteria #1** – A prescribed set of Operating Environment conditions.
- **Criteria #2** – A bounded mission time duration or remaining mission time.
- **Criteria #3** – The operating condition of the Mission System Elements – PERSONNEL, EQUIPMENT, MISSION RESOURCES, PROCEDURAL DATA, and SYSTEM RESPONSES. (Plus Facilities for Enabling Systems)
- **Criteria #4** – The elapsed operating time since the start of the mission.
- **Criteria #5** – A probability of success in completing a defined mission without disruption.

Avoid using MTTF or MTBF as the reliability requirement without bounding these conditions.

How to Invalidate a Reliability Estimate and Violate Principle 34.6

Author's Note 34.3 As a follow-up to Principle 30.2, when stating Reliability in any technical document, ensure that the Reliability Estimate Criteria listed above are stated on the same page as the Reliability estimate. Separating the information across multiple pages effectively *invalidates* the estimate and violates Principle 30.2.

In summary, we have illustrated how the Reliability Function is derived based on a SYSTEM OR ENTITY'S PDF. This leads to the next of the Five Life Data Functions, the Instantaneous Failure Rate Function.

34.3.2.5 The Failure Rate Function, λ(t) A general discussion of Failure Rate, $\lambda(t)$, soon leads to the need to refine the term for more explicit contexts. The need for refinement is driven by two key questions:

1. *How many units can we expect to fail at a specific instant in time during a mission*?
2. *How many units can we expect to fail over a period of time*?

These questions require that we further refine Failure Rate, $\lambda(t)$, in general terms into more specific terms. The terms are Instantaneous Failure Rate, $\lambda(t)_{\text{Inst}}$, and the Average Failure Rate, $\lambda(t)_{\text{Avg}}$.

Most textbooks refer to the Failure Rate, $\lambda(t)$, generically without regard to context. For example, there is a significant difference between Instantaneous Failure Rate, $\lambda(t)_{\text{Inst}}$, versus Average Failure Rate, $\lambda(t)_{\text{Avg}}$. Let's define each of these terms:

- **Instantaneous Failure Rate** A metric that represents the *frequency of failure* or *failure density* as a function of the continuous random variable *time*, t at a specific instant in time.
- **Average Failure Rate** "The total number of failures within an item population, divided by the total number of life units expended by that population, during a particular measurement period under stated conditions" (MIL-HDBK-338B, p. 3–7).

NIST (2013, Section 8.1.2.3) notes, "The failure rate is the rate at which the population survivors at any given instant are 'falling off the cliff.'" For example, the continuous variations on an automobile's velocity indicated by its speedometer—miles per hour (mph) or kilometers per hour (km/h)—are analogous to the Instantaneous Failure Rate, $\lambda(t)_{\text{Inst}}$, at a specific instant of time.

Most people think in terms of *average failure rates*, $\lambda(t)_{\text{Avg}}$, per *unit of time* such as hours, weeks, months, or cycles. For example, how many failures have occurred based on the total operational hours of all units of a SYSTEM OR ENTITY—*average* quantity of failures per operational hour. Consider the following example:

Average Failure Rate, $\lambda(t)_{\text{Avg}}$

Example 34.6 Assume a trucking company has a fleet of 100 trucks (units) operating 8 hours per day. During a 30-day period, three trucks experience failures that require removal from service and corrective action. Therefore, for that specific month, the *Average Failure Rate* is:

- 3/100 or 0.03 failures per truck per month.

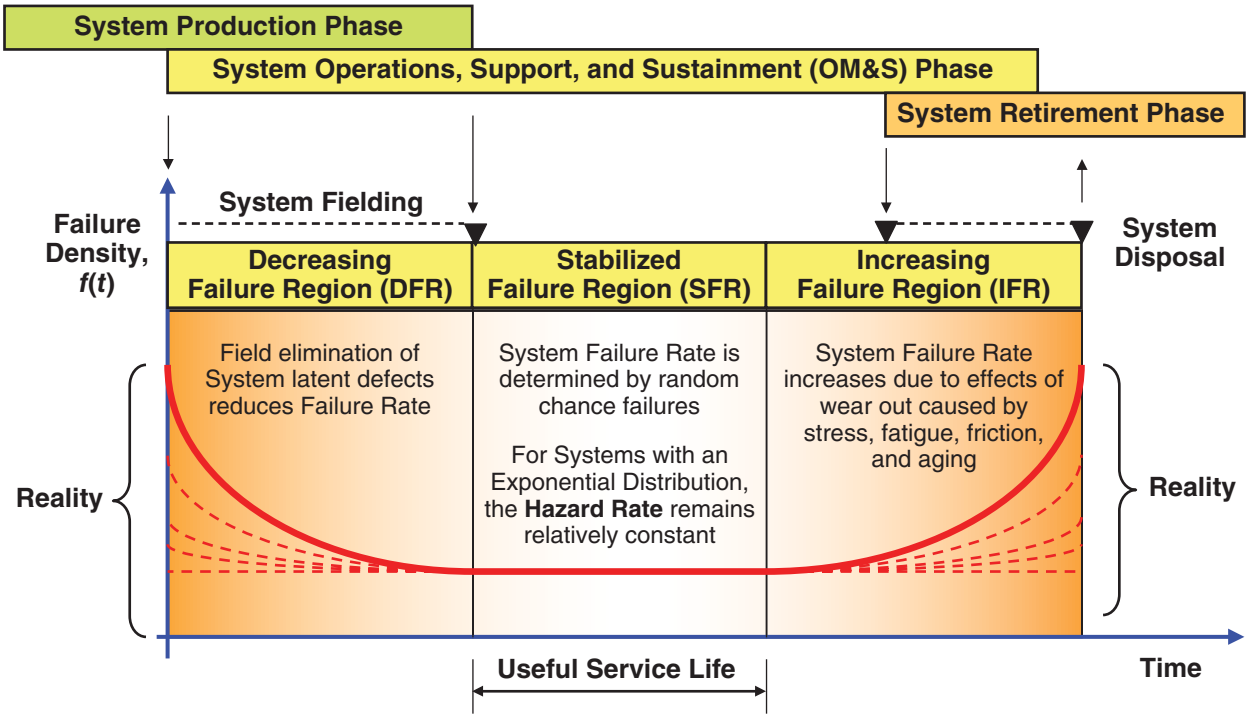


Figure 34.6 Bathtub Curve Concept Illustrating the Decreasing, Stabilized, and Increasing Failure Regions (DFR, SFR, and IFR)

- 0.001 failures per truck per day.
- 0.000125 failures per truck per hour.

Observe the importance of *qualifying* and *bounding* the *context* such as average failure rate per truck, per month, per day, or per hour.

As another example, suppose Figure 34.1 Panel C representing the RTF of a System or Product. The Average Failure Rate, $\lambda(t)_{Avg}$, is:

$$\lambda(t)_{Avg} = \frac{31 \text{ failures (100\% of units)}}{98.0 - 80.0 \text{ hours}}$$
$$= 0.316 \text{ failures per hour}$$

Now, contrast $\lambda(t)_{Avg}$, with The Instantaneous Failure Rate, $\lambda(t)_{Inst}$, of 7.0 units that occurred at $t = 89.0$ hours. Additionally, recognize that the failures are "clustered" between 70.0 hours $\leq \lambda(t)_{Avg} \leq$ 98.0 hours leaving a void between 0.0 $\leq \lambda(t)_{Avg} <$ 80.0 hours. As a result, $\lambda(t)_{Avg}$ serves no useful purpose. In contrast, the Average Failure Rate, $\lambda(t)_{Avg}$, has some relative value over the range 80.0 hours $\leq \lambda(t)_{Avg}$ 98.0 hours when its value is 1.722 failures per hour.

One of the objectives of Lifetime Data Distributions is to define the characteristic PDF Distribution of a SYSTEM or ENTITY's failures. Then, being able to detect those failures in sufficient time to take *preventive* or *corrective* action. Despite their apparent simplicity, the concepts of *failure density as a function of time—PDF, average failure rate* ($\lambda(t)_{Avg}$), and *instantaneous failure* or *hazard rate*—are often one of the most confusing aspects for many Engineers concerning Life Data Analysis. Understanding the context of these terms and their application is a key foundation for Lifetime Data Analysis.

In summary, if a SYSTEM or PRODUCT has a demanding mission requirement to operate continuously for 6 hours, and has a *likely probability* that it will fail during that period, then you should consider replacing any suspect ENTITIES or PARTS before the mission, especially if the component is *mission* or *safety critical*. Examples of *mission* or *safety critical items* include medical devices, bakery ovens, and aircraft.

Our discussion up to this point consisted of a review of statistical distributions applied to R&M. Our next topic, the Hazard Function focuses on the *Useful Service Life* or Stabilized Failure Region (SFR) of the Bathtub Curve (Figure 34.6) characterized by Exponential PDFs.

34.3.2.6 The Hazard Function h(t)

Principle 34.13

Hazard Rate Principle

SYSTEMS or components that are characterized by Negative Exponential PDF Distributions mathematically exhibit a *constant* Hazard Rate, $h(t)$, during its Useful Service Life based on random, chance failures.

One of the challenging aspects of Reliability is whether a System or Entity will successfully complete the next mission without failure. *If the Reliability, R(t), of a System or Entity is 0.9 based on its current condition, what is its current reliability - probability of successfully completing a mission with a duration of Δt without failure for a given set of* Operating Environment *conditions*?

This question places a *condition* on the Reliability Function. As a result, it is sometimes referred to as the *Conditional Reliability Function*; others refer to it as the *Hazard Function*. To add to the confusion, some people refer to the Hazard Function as the *Failure Rate Function*. Figure 34.5 Panels A and B illustrate differences in the two terms.

The Hazard Function, $h(t)$, is defined as the ratio between a System or Entity's Failure Rate, $\lambda(t)$, to its Reliability, $R(t)$, as a function of the continuous variable *time*, t. The general equation for the Hazard Function is:

$$h(t) = \frac{f(t)}{R(t)} = \frac{f(t)}{1 - F(t)} \quad \text{for } t \geq 0 \tag{34.22}$$

where:

- $h(t)$ = Hazard Rate.

NIST (2013, Section 8.1.2.3) notes that: "The failure rate is sometimes called a 'conditional failure rate' since the denominator $1 - F(t)$ (i.e., the population survivors) converts the expression into a *conditional* rate, given survival past time, t."

A Word of Caution 34.3

NIST (2013, Section 8.1.2.3) refers to the "Failure (or Hazard) Rate" and states, "The failure rate is defined for *non-repairable* populations as the (instantaneous) rate of failure for the survivors to time t during the next instant of time." Despite the convolution of terms "Failure (or Hazard) Rate" in the title, the descriptions address the Hazard Rate for Systems or Products characterized by Exponential Distributions. The Hazard Rate, $h(t)$, is a ratio-based metric, which is different from the Failure Rate, $\lambda(t)$, that characterizes other types of non-Exponential Distribution Systems or Products.

For Exponential Distributions, we substitute Eq. 34.3 into Eq. 34.22:

$$h(t) = \frac{f(t)}{1 - F(t)} = \frac{\lambda e^{-\lambda t}}{1 - (1 - \lambda e^{-\lambda t})} \tag{34.23}$$

$$h(t) = \frac{\lambda e^{-\lambda t}}{e^{-\lambda t}} \quad \text{for } t \geq 0$$

Resulting in:

$$h(t) = \lambda \quad \text{for } t \geq 0 \tag{34.24}$$

Eq. 34.24 indicates that for Exponential Distributions, the *Hazard Rate* is constant thoughout the Stabilized Failure Region (SFR) (Figure 34.7). Because of the *constant failure rate*, statisticians describe the condition as "memory-less" and not influenced by failures until the current time period. This is in contrast to the past human condition, for example,

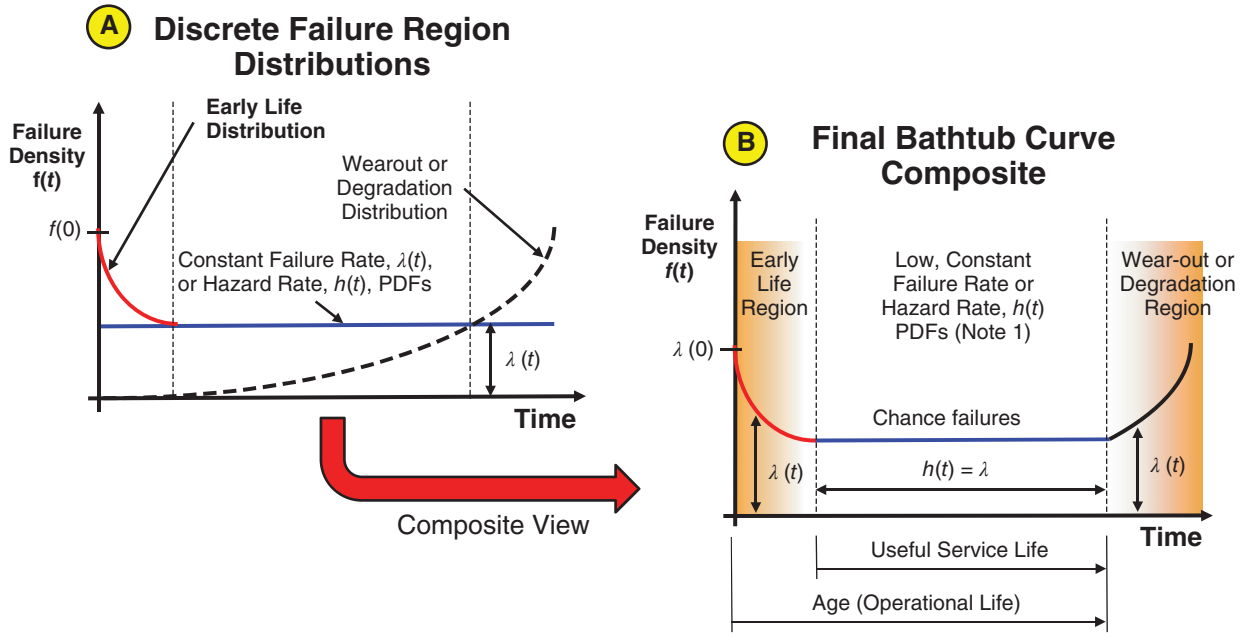


Figure 34.7 Bathtub Curve Illustration Depicting Its Different Failure Distribution Profiles Failure

in which our bodies are dependent on and influenced by the condition in the preceding time period—"memory."

What is the value in knowing the Hazard Rate? Reliawiki.org (2012) states that the metric " … is useful in characterizing the failure behavior of a component, determining maintenance crew allocation, planning for spares provisioning, etc."

If we express the relationship of $\lambda(t)$ as a function of Reliability, $R(t,\ t+\Delta t)$ over the interval from t to $t+\Delta t$:

$$\lambda(t) = \frac{R(t_1) - R(t_2)}{(t_2 - t_1) \cdot R(t_1)} \quad \text{for } t \geq 0 \tag{34.25}$$

or

$$\lambda(t) = \frac{R(t) - R(t+\Delta t)}{(\Delta t) R(t)} \quad \text{for } t \geq 0 \tag{34.26}$$

Author's Note 34.4

Operational Age Notation

Since the key question concerns Hazard Rate, the Reliability from the current time, t_1, until the completion of a mission, t_2, some authors refer to: (1) the current time, t_1, as T representing the operational age in hours and (2) t_2 as the next Δt increment.

Remember—As an example, Failure Rate units consist of measurements such as: (1) failures per million hours, (2) failures per month, or (3) failures per cycle. Failure Rate, $\lambda(t)$, that characterizes a PDF represents the *failure density* as a function of the continuous variable time, t; the CDF represents the cumulative probability of a failure of a System or Entity over time.

34.3.2.7 The Mean Time Functions—MTTF and MTBF

For any Lifetime Data distribution characterizing a population of units, we need to know the expected *mean life* represented by Time to Failure (TTF). Lifetime Data Analysis accommodates this need via the Mean Life Function, which enables us to model and estimate a System or Entity's MTTF, (μ). However, MTTF does not enable us to differentiate if the item is *repairable* or *non-repairable*. This means pre-defined decisions must be made during component selection (Chapter 16) designating Entities that are considered *repairable* versus *unrepairable*.

34.3.2.7.1 MTBF Repairable Items Since *repairable* Systems or Entities, by definition, can be repaired, the metric for expressing their average is MTBF (θ), which is the inverse of the Failure Rate, $\lambda(t)$. Observe the operative phrase "between failures" in MTBF. Recognize that failures for *repairable* Entities have two contexts: (1) the MTBF of a System or Entity, in general, and (2) the MTBF of a specific Part Level failure within the System or Entity. MIL-HDBK-470A defines MTBF as:

- Mean time Between Failure (MTBF) "A basic measure of reliability for repairable items. The mean number of life units during which all parts of the item perform within their specified limits, during a particular measurement interval under stated conditions" (MIL-HDBK-470A, p. G-11).

34.3.2.7.2 MTTF for Repairable and Replaceable Items *Non-repairable items*, by definition, are *unrepairable*. They are Removed and Replaced (R&R) and disposed of in accordance with federal, state, and local regulations. Since a *non-repairable* System or Entity *fails only once*, the operative metric is MTTF.

34.3.2.7.3 Computation of MTBF and MTTF

Principle 34.14

Expected Life Principle

The *expected life* of a System or Entity is the *arithmetic average* of the actual lifetimes of a population of units currently in field use.

Mathematically, we can express the Expected Life, $E(\overline{T})$, or MTTF, as a metric based on the Failure PDF, $f(t)$, and the continuous random variable *time*, t, as:

$$E(\overline{T}) = \int_0^{\infty} t \cdot f(t) dt \quad \text{for } t \geq 0 \tag{34.27}$$

where:

$E(\overline{T})$ = MTTF Common to *repairable* and *non-repairable* items.

Observe that the integral runs over the range from $t = 0$ to $+\infty$. In this context, $E(T)$ represents the arithmetic average of the population of fielded units in operation.

For the Exponential Distribution:

$$E(\overline{T}) = \text{MTTF}(\lambda) = \int_0^{\infty} t \cdot \lambda e^{-\lambda t} dt \quad \text{for } t \geq 0 \tag{34.28}$$

$$E(\overline{T}) = \frac{1}{\lambda} \quad \text{for } t \geq 0 \tag{34.29}$$

Similarly, for *repairable* Systems or Entities the MTBF, (θ) (Eq. 34.5), is expressed as:

$$E(\overline{T}) = \frac{1}{\theta} \quad \text{for } t \geq 0 \tag{34.30}$$

A Word of Caution 34.4

Observe that MTTF, as a magnitude, provides no information concerning a System or Entity's Lifetime Data Distribution. Two cautionary points:

- Reliawiki.org (2012) notes "Because vastly different distributions can have identical means, it is unwise to use the MTTF as the sole measure of the reliability of a component."
- Nelson (1990, p. 56) emphasizes that "expected life" (Principle 34.16) statistically is not the same as "anticipated life."

On inspection, MTBF and MTTF appear to be the same. However, there are differences. You should recall our earlier discussion following Eq. 34.4 that MTBF > MTTF.

We can express MTBF, θ, mathematically for *repairable* items as:

$$\text{MTBF}\ (\theta) = \frac{\text{T}}{\text{Qty. of Failures}} \quad T \geq 0 \qquad (34.31)$$

where:

T = Total operational hours of the fielded population of units.

Recognize that the assumption here is that all UUTs have failed. This is not always the case. Speaks (2005), for example, refers to *suspensions*. A *suspension* represents a test condition in which "a destructive test or observation has been completed without observing a failure" Speaks (2005, p. 7).

The computation for MTTF (μ) for *non-repairable* Systems or Entities is expressed mathematically as:

$$\text{MTTF}(\mu) = \frac{T}{\text{Qty. of UUTs}} \quad \text{for } T \geq 0 \qquad (34.32)$$

where:

T = Total hours of operation of the UUT population before failure and replacement.

The *mean*, (μ) of a statistical PDF distribution is computed by summing the multiplicative products of each discrete value of x by its probability of occurrence, $P(x)$.

$$\mu = \sum x \cdot P(x) \qquad (34.33)$$

To illustrate how the *mean* is calculated, consider the following example:

Example 34.7

Let's assume we need to determine the mean of a set of test data we have collected. The resulting data values are: 8, 11, 12, 15, 18, 39, 54. To calculate the mean of the data set, we do so as follows:

$$\text{Mean} = \frac{8 + 11 + 12 + 15 + 18 + 39 + 54}{7} = 22.4$$

34.3.2.8 The Median Life Function

Principle 34.15

Median Life Principle

The *median* of a System or Entity's PDF represents a specific instant of time when 50% of identical units are estimated to have *failed* and 50% continue to *survive* and *operate*.

For a System or Entity that exhibits symmetry in their failure profiles such as the Gaussian (Normal) Distribution, the mean and the median are *coincidental* (Figure 34.3 Panel A). The real world, however, does not always exhibit symmetry resulting in failure profiles such as the Log-Normal Distribution (Figure 34.3 Panel B). Since the area under any PDF to the left of the current time, t, represents the probability that X units have failed—*unreliability*, $F(t)$ or $Q(t)$ (Figure 34.5 Panel A), a key question is: *when will F(t) and R(t) be simultaneously at 50%* ?

The dividing line when the area under the curve is evenly divided is referred to as the *median* or *centroid*. The median is designated as the *Median Life*. For Lifetime Distributions, the continuous random variable, t, reaches a point in which there is a 50% probability that half of the units have *failed* and half continue to *survive*. As a contrast to the Mean Life Function in Example 34.7, consider Example 34.8 for the computing the *median* of a data set.

Example 34.8

Median of a Data Set

In Example 34.7, the data set values were: 8, 11, 12, 15, 18, 39, 54. Since there were seven values, Value 15 represents the Median with identical quantities of data items on above and below it.

Mathematically, the Median Life Function is computed by *equating* the integral of the PDF, $f(t)$, equal to 0.5 for a time interval between $t = 0$ and the mean Age, $\overline{T}$, in hours, cycles, or actuations of a System or Entity:

$$\int_{-\infty}^{\overline{T}} f(t)dt = 0.5 \quad \text{for} \quad -\infty \leq t \qquad (34.34)$$

34.3.2.9 The Mode Life Function

Principle 34.16

Mode Life Principle

The mode of a System or Entity's PDF occurs at the *peak* failure density (point of inflection) where the *maximum* number of failures is expected to occur at a specific instant in time.

For a given PDF, another question is: *at what point in time is the Lifetime Data Distribution expected to peak as a function of the continuous random variable time, t?* We refer to this as the *mode* of a Lifetime Data Distribution

(Figure 34.3 Panel B). The *mode* of a statistical distribution based on a continuous random variable time, t, is defined as the time in which the PDF is at a peak.

Since the *mode* represents the *peak* of a PDF distribution, we locate the *point of inflection* in which the Failure PDF, $f(t)$, reaches a *maximum*, assuming the distribution has only one peak. Therefore, we express the *mode* as the differential of the Failure PDF, $f(t)$, for the continuous random variable *time*, t, as:

$$\frac{d[f(t)]}{dt} = 0 \quad \text{for} \quad -\infty \leq t \tag{34.35}$$

Then, solve for *time*, t, for a specific Lifetime Data Distribution's $f(t)$.

34.3.2.10 Summary—Lifetime Data Functions In summary, we have introduced the basic Lifetime Data Functions. For an additional graphical summary of the various types of Lifetime Data functions—Exponential, LogNormal, Gaussian (Normal), and Weibull, refer to MIL-HDBK-338B Figure 5.3-1, p. 5–9.

This brings us to our next topic, Service Life Concepts.

34.3.3 Service Life Concepts

HUMAN SYSTEMS—Enterprise and Engineered—exhibit *service life profiles* characterized by a level of infant mortality early followed by growth and stability and finally aging. This leads to a key question: *What is the service life of a* SYSTEM *or* ENTITY? The Defense Acquisition University (DAU) (2012, p. B-201) defines *service life* as follows:

- **Service Life** "Quantifies the average or mean life of the item. There is no general formula for the computation. Often refers to the mean life between overhauls, the mandatory replacement time, or the total usefulness of the item in respect to the (system or product) it supports; that is, from first inception of the (system or product) until final phaseout" (Adapted from DAU, 2012, p. B-201).

In comparison, MIL-HDBK-470A employs the term *Useful Life*:

- **Useful Life** "The number of life units from manufacture to when the item has an unrepairable failure or unacceptable failure rate. Also, the period of time before the failure rate increases due to wear-out" (MIL-HDBK-470A, p. G-17).

To better understand the application of Lifetime Data Distributions to *Service Life* or *Useful* Life, textbooks often employ a concept referred to as the Bathtub Concept, our next topic.

34.3.4 The Debatable Bathtub Curve Concept

Generally, people tend to think of Service Life or Useful Life of a SYSTEM OR ENTITY as having an "out-of-the-box" reliability of 100%. In general, this statement is true, at least for most of the ENTITIES or PARTS. However, SYSTEMS or PRODUCTS often experience early failures in their lives based on the factors discussed in *What Constitutes a Failure*. Building on Reliability Precepts #1 – #7, the assumption is that if we eliminate failures, System Reliability should improve. Conceptually, this is true; however, even that is dependent on the condition of the other Parts and that continuous variable, *time*. The reality is SYSTEMS, ENTITIES, and PARTS, especially mechanical ENTITIES or PARTS, are always in a state of *wear-out* and eventually fail due to use, stress, deterioration, fatigue, unless they are maintained through *preventive* and *corrective maintenance* action.

As a way of explaining these dependencies, textbook discussions of reliability often introduce the concept of *service life profiles* via a model referred to as the Bathtub Curve. The name is derived from the characteristic Bathtub Curve profile as illustrated in Figure 34.6. Conceptually, the Bathtub Curve represents a plot of failure rates over the active *service life* of MISSION SYSTEM or ENABLING SYSTEM EQUIPMENT.

34.3.4.1 Introduction to the Bathtub Curve Observe that the heading of this section is titled "The Debatable Bathtub Curve Concept". There are those who challenge the *validity* of the Bathtub Curve based on a wide dispersal of factual evidence, especially among electronic and mechanical PARTS. Proof in the form of factual evidence resides in industry proprietary data, which are not publically available, assuming Enterprises invest in tracking it. As a result, *controversy* surrounds the Bathtub Curve.

Where does truth reside? Probably in various aspects of all the arguments. The reality is: the Bathtub Curve has elements of truth and instructional value. *If it is controversial, why are we presenting it here*? There are two reasons:

- First, from an educational perspective, you need to understand key concepts and challenges to their validity.
- Second, from an instructional perspective, the Bathtub Curve has a merit—its *pros* outweigh *cons*—as a foundational concept for Reliability.

What is important for you is to understand the Bathtub Concept, its relevance, and issues. Then, draw your own conclusions for validation within your own Enterprise regarding the systems, products, or services it produces.

Given this understanding, let's proceed with a description of the Bathtub Curve.

34.3.4.2 Bathtub Curve Overview Description

In general, if you analyze a sampling of any type of new, mass-produced SYSTEM or PRODUCT, you will find that most operate right "out of the box," or "off the parking lot." For those that fail, the retailers, dealers, take *corrective action* to eliminate the faults or failures. Typically, when corrected, the owner enjoys a relatively event-free Useful Service Life from the SYSTEM or PRODUCT, assuming they perform *preventive* and *corrective* maintenance prescribed by the System Developer or manufacturer. Over a long service life of several years, PARTS begin to *wear out* and fail unless they are maintained. Over a large population of SYSTEMS or PRODUCTS, analysts drew inferences and began to create a model of the typical SYSTEM or PRODUCT's Lifetime Data Distribution. The model became known as the Bathtub Curve.

The Bathtub Curve shown in Figure 34.6 represents a notional paradigm, a model for thinking. Unfortunately, like most paradigms, the Bathtub Curve has become *erroneously* ingrained as a "one-size-fits-all" mindset believed to be *universally applicable* to all SYSTEMS and PRODUCTS. This is false! Most experts agree that field data, though limited, suggest that the failure rate profiles vary significantly from one type of SYSTEM or PRODUCT to another across the marketplace.

Our treatment of the Bathtub Curve will simply be as *an instructional model* for discussion purposes only. As a final note, the Bathtub Curve applies to any SYSTEM or ENTITY composed of levels of integrated PARTS; it is not intended as a PART Level explanation. Wilkins (2002) cautions "does not depict the failure rate of a single item. Instead, the curve describes the relative failure rate of an entire population of products over time."

34.3.4.3 Bathtub Curve Profile Structure

Principle 34.17

Bathtub Curve Principle

The Bathtub Concept is a notional concept based on a piecewise composite, end-to-end structure comprised of a sequence of three different Lifetime Data Distributions: (1) Decreasing Failures Region (DFR), (2) Stabilized Failures Region (SFR)—Useful Service Life, and (3) Increasing Failures Region (IFR).

The Bathtub Curve is actually a *piece-wise composition* of three different Lifetime Data Distributions as shown in Figure 34.7 Panel A. The net result is a composite characteristic profile such as the one shown in Figure 34.7 Panel B.

Remember—Figure 34.6 represents a *generalization* applicable to a SYSTEM or PRODUCT in terms of its: (1) profile appearance and (2) duration as a function of time. In summary, observe several key points about Figure 34.6:

1. **Period of Decreasing Failure Region (DFR)**—Represents a period of *conditional* decreasing failure rates, $\lambda(t)_{DFR}$, due to *inherent* latent defects such as design errors, flaws, and deficiencies; workmanship issues, component and material integrity that can often be characterized by a *negative* Exponential Failure PDF as a function of the continuous variable time, t
2. **Period of Stabilized Failure Region (SFR)**—Represents a period of *random chance Failure Rate*, $\lambda(t)$, that is relatively constant or Hazard Rate, $h(t)$, for Systems or Products that exhibit a negative Exponential PDF as a function of the continuous variable time, t.
3. **Period of Increasing Failure Region (IFR)**—Represents a period of *conditional* increasing failure rates, $\lambda(t)_{IFR}$, due to component wear-out that can often be characterized by an *increasing* Exponential Failure PDF as a function of the continuous variable time, t.

Given this overview description of the Bathtub Curve, let's address a brief history of the Bathtub Curve's Origin.

34.3.4.4 A Brief History of the Bathtub Curve's Origins

The origins of the Bathtub Curve are deeply rooted over several centuries in Life Data Analysis, not in Engineering. Klutke et al. (2003, p. 125) cite: (1) work by E. Halley in 1693 concerning actuarial life-table analysis that uses a similar profile and (2) modern day textbooks from the 1970s. Machol et al. (1965, pp. 33–3) address the concept.

As military systems became more complex in the 20th century, especially around World War II, Reliability Engineers dealing with failures related to electronic vacuum tubes, and the need to predict failures became interested in the Bathtub Curve application to systems. Smith (1992) notes that the origin of the Bathtub Curve dates back to the 1940s and 1950s with the embryonic stages of Reliability Engineering. Early attempts to characterize failure rates of electronic components led to the formulation of the Bathtub Curve. Examples of these components included vacuum tube and early semiconductor technologies that exhibited high failure rates.

Although the Bathtub Curve provides a *conceptual framework* for discussion, Reliability SMEs question the *validity* of the Bathtub Curve in today's world. Due to the higher component reliabilities available today, most systems and products become *obsolete* and are Removed and Replaced (R&R) or retired/disposed long before their *useful service life* profile reaches the Increasing Failure Region (IFR). Most computer hardware will last years or decades. Yet, as Moore's Law (Moore, 1965) postulated, computer technology advancements drive the need to upgrade or replace computers every 2–3 years. Kaplan, (2011) indicates that Intel

encountered yield problems that blemished its remarkable performance of Moore's Law after 45 years.

Klutke (2003, pp. 125–129) and others challenge the Bathtub Curve's *validity* and provide objective critiques. Notionally and conceptually, it may have some relevance in mechanical systems, anecdotal electronic system field data refute the right-hand side of the Bathtub Curve because EQUIPMENT such as computers become technologically obsolete long before the electronic components fail. Refer to Klutke et al. (2003, pp. 125–129) for details of their critique.

Smith (1992) notes that the Bathtub Curve may provide an appropriate profile for a few components. However, *the Bathtub Curve has been assumed to be applicable to more components than is supported by actual field data measurements*. Large, statistically valid sample sizes are required to establish age-reliability characteristics of components. Often, large populations of data are difficult to obtain due to being proprietary to System Developers, assuming they exist—tracked and analyzed. Corporate warranties for current fielded products tend to motivate Lifetime Data records tracking not only for current designs but future designs as well. Leitch (1988, p. 24) notes that the Bathtub Curve is representative of the lifetime Data Distribution of automobiles.

Nelson (1990, p. 70) challenges the *validity* of the Bathtub Curve. He notes that in his experience, the Bathtub Curve describes only 10–15% of applications. Typically, these applications consist of "products with competing failure modes."

Anecdotal evidence suggests that most reliability work is based on the Stabilized Failure Region (SFR), primarily due to the simplicity of dealing with the *constant hazard rate* for *negative Exponential Distributions*. Although any *decreasing* or *increasing* Exponential Distribution can be used to model the three failure rate regions, the Weibull distribution (Figure 34.4) typically provides more flexibility in accurately *shaping* the characteristic profile.

One of the challenges to the Bathtub Curve is its shape. Most SMEs generally agree that electronic SYSTEMS or PRODUCTS such as consumer products become obsolete long before the device wears out or fails and are simply discarded and sent to a landfill or salvaged for exotic metals. Other than latent workmanship defects, they generally have no moving parts. Figure 34.8 Panel A illustrates this point.

Mechanical systems are a different matter. They incur wear from the moment they are activated. Their wear continues at varying rates depending on OPERATING ENVIRONMENT conditions and maintenance such as lubrication, filter replacement, corrosion removal, inspection for fatigue, and cracks. Their failure characteristic profiles may be similar to Figure 34.8 Panel B. Now, contrast the Useful Service Life of *electronic* systems (Figure 34.8 Panel A) with *mechanical* systems (Figure 34.8 Panel B). For additional tabular information concerning the Environmental Effects

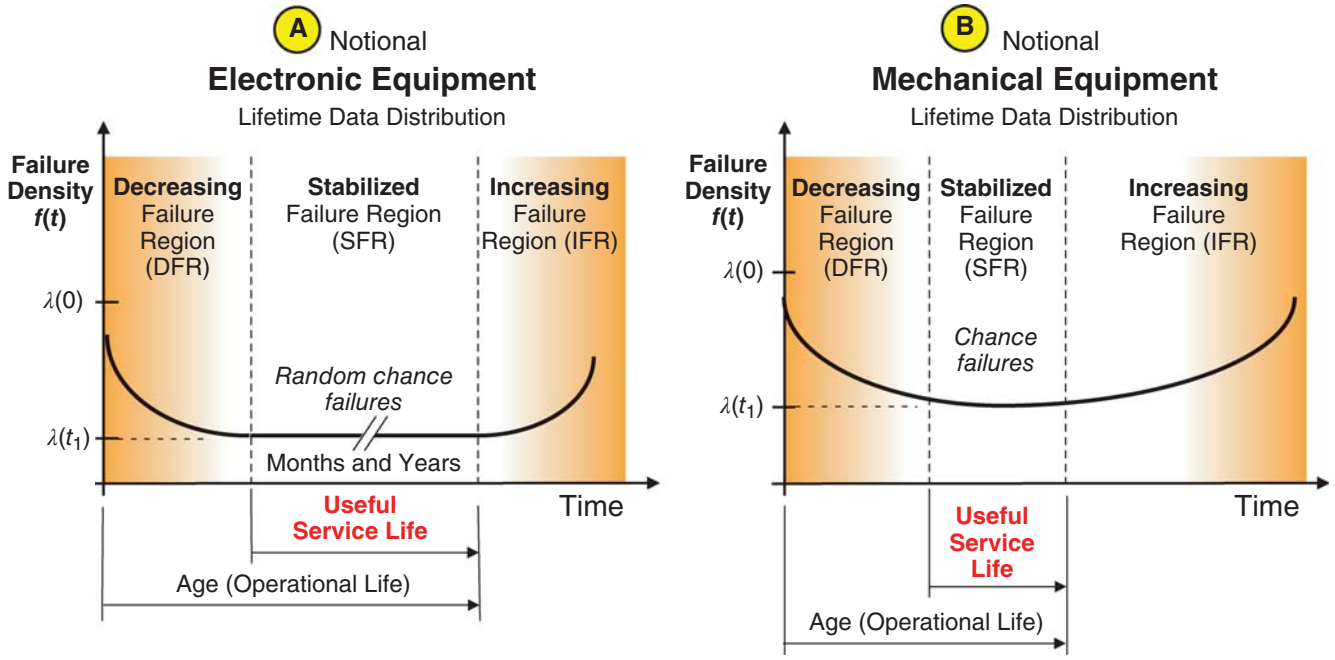


Figure 34.8 Notional Comparison Contrasting the Shape of the Bathtub Curve for Electronic versus Mechanical Equipment Failure Rate Profiles

on Mechanical Design for various environments, principal effects, corrective actions, and descriptions, refer to the SRC (2001).

Based on these observations, the validity of the Bathtub Curve may better serve as an instructional tool for describing a generalized system with three different Lifetime Data Distribution regions rather than a "one-size-fits-all" paradigm for every type SYSTEM or PRODUCT.

Given the history of the Bathtub Curve, let's explore each of its three failure regions.

34.3.4.5 Bathtub Curve—Decreasing Failures Region (DFR)

Principle 34.18

Early Failures Principle

Every system, product, or service when placed in *active service* exhibits a failure rate following manufacture that is a function of its residual latent design defects, weak components and material properties, quality of workmanship, User learning curve, and OPERATING ENVIRONMENT.

Principle 34.19

Early Failures Elimination Principle

Proactive *detection* and *elimination* of latent defects results in a diminishing failure rate up to a point of chance failures due to usage, stress (Figure 34.11), and OPERATING ENVIRONMENT conditions.

The DFR is often referred to as the Early Life Region and begins at SYSTEM or PRODUCT release, deployments, installation, and checkout. When power is applied, failures may occur due to *latent defects* such as design errors, flaws, deficiencies, poor workmanship, and weak components or materials. Radle and Bradicich (2013) note that sources of failure also originate from damage incurred "during shipping, storage, and installation." On the basis of this point, *is there any question why an SPS and other outlines include a section for Packaging, Handling, Storage, and Transportation (PHS&T) requirements*?

During this time, the System Developer may be *debugging* System Design and maintenance issues to identify latent defects and take corrective action. One of the key objectives of System Integration, Test, and Evaluation (SITE) (Chapter 28) is to discover and eliminate latent defects, weak components, and types of failures. Some Enterprises subject each deliverable item to accelerated test cycles to drive out early failures summarily characterized as or conveniently blamed on "infant mortality."

Observe the context mentioned above. During Developmental Design Verification (Chapter 13), testers deal with several technical issues simultaneously. For example:

1. Does the SYSTEM or ENTITY Design comply with its SPS or Entity Development Specification (EDS)? Challenges include: latent defects such as interpretation of specification requirements, design errors, flaws, and deficiencies.
2. Manufacturing latent defects such as poor workmanship, weak components.

When the Developmental Design Verification is complete and proven, the only remaining issue focuses on Product Verification concerning manufacturing latent defects of a single unit – Serial Number (S/N) – of a System or Entity. As noted above, accelerated testing during manufacturing is one method for driving out *marginally* weak components.

One of the ironies of the Bathtub Concept is the *shock* and *surprise* exhibited by the *ad hoc* SDBTF-DPM Paradigm Enterprises (Chapter 2). When confronted by the User on discovery of an unusually large quantity of initial failures of their system, product, or service at delivery (Figure 13.2):

> "How could this happen? You stated in your proposal that if we selected your Enterprise, we could rest assured that we were hiring the best SE capable Enterprise?"

Most textbooks and authorities launch into *academic* descriptions of the Bathtub Curve and focus exclusively on "Engineering's role in analyzing field data and correcting design problems." Here's the issue. Observe where the PDF intersects the vertical Failure Rate, $\lambda(t)$ axis in Figure 34.9. *Where do you suppose the RMA decisions were made that impact the* $\lambda(t = 0)$ *magnitude of the Failure Rate*? Answer: From the three processes to the left: System Design; Component Procurement and Development; and SITE (Figure 12.2)! … *and there is shock and surprise*? Fortunately, most Users see through the rhetoric.

Assuming Engineering has performed its role, designing the SYSTEM or PRODUCT *effectively* leaves residual component selection, quality, and manufacturing defects. Strategies such as Environmental Stress Screening (ESS) of ENTITIES and PARTS are used to "burn in" and eliminate weak components.

Another strategy to eliminate *latent defects* is Low-Rate Initial Production (LRIP). LRIP produces a limited quantity of test articles for field usage by the User. This provides an opportunity to not only validate the SYSTEM or ENTITY but also identify any residual latent Developmental Design defects (Figure 13.2) for corrective action before commitment to Production. The intent is to shift the Exponential Distribution Failure Rate, $\lambda(t)$, downward before delivery to the User as noted by the dashed curve. Figure 34.9 illustrates the point.

Let's assume that during System Design Verification, a SYSTEM or PRODUCT has an early Failure Rate of $\lambda_0(t_0)$. Corrective actions are taken to complete an initial phase of Design Verification concerning specification compliance. At

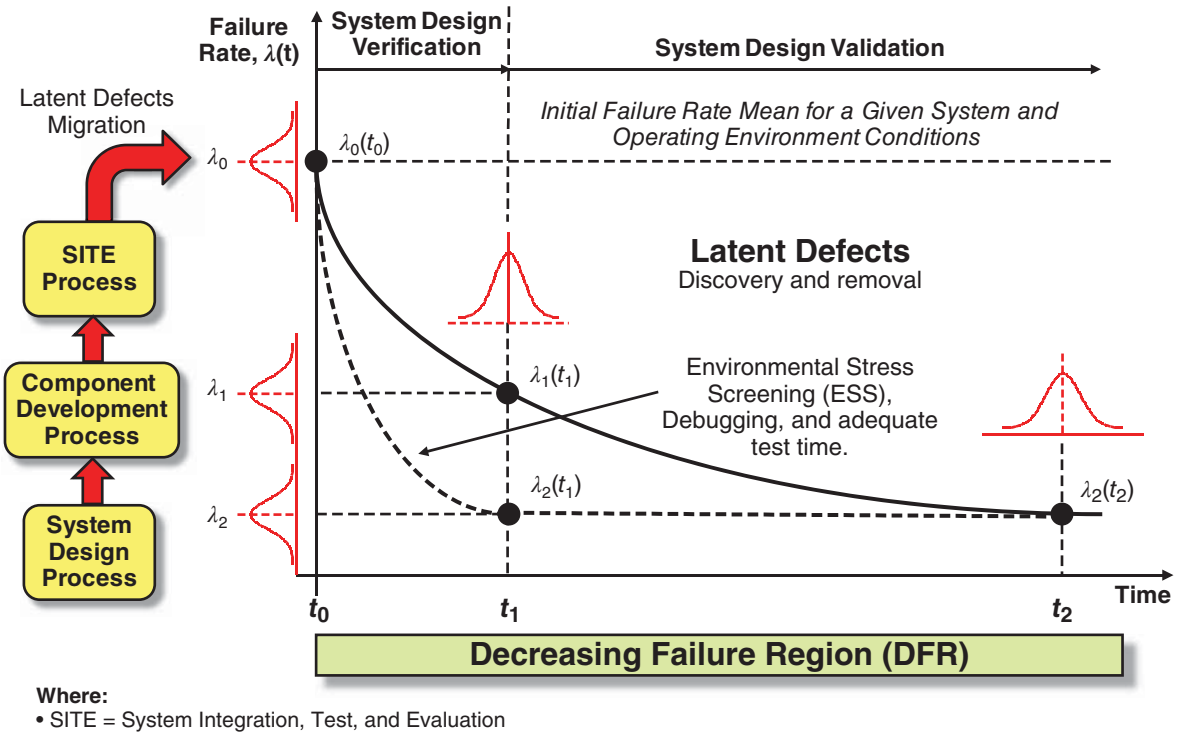


Figure 34.9 Decreasing Failure Rate (DFR) Strategy for Latent Defects Removal During System Integration, Test, and Evaluation (SITE) Prior to Delivery

the end of the System Design Verification, the Failure Rate has been reduced to $\lambda_1(t_1)$ as preparations are made for System Design Validation. As a result, LRIP is used to produce a small quantity of XX units for System Validation. During System Validation, failures are recorded and corrective actions are taken. On completion of System Design Validation, the Failure Rate has been reduced to $\lambda_2(t_2)$. Since the Failure Rate has been reduced from $\lambda_0(t_0)$ to $\lambda_2(t_2)$, a decision is made to release the System or Product. In contrast, employing strategies such as proactive latent defects removal earlier in the System Design and Component Procurement & Development Processes, Environmental Stress Screening (ESS), and adequate SITE time coupled with a competent workforce can make significant reductions in the Failure Rate from $\lambda_0(t_0)$ to $\lambda_2(t_1)$ – dashed line – as well as the cost-to-correct the latent defects (Table 13.1).

In summary, the strategy is to plan for a LRIP, collect and analyze field data, understand the failure profile, take corrective actions such as replace or rework faulty components or poor design, and then proceed with Full-Scale Development (FSD) and SYSTEM or PRODUCT release.

34.3.4.6 Bathtub Curve—Stabilized Failures Region (SFR) The SFR, which is sometimes referred to as the *Constant Failure Region*, begins when the Instantaneous Failure Rate, $\lambda(t)_{\text{Inst}}$, remains relatively *stable* but not necessarily *constant*. NIST (2013, Section 8.1.2.4) refers to this region as the "Intrinsic Failure Period." From a Lifetime Data perspective, this period is referred to as the *Useful Service Life* for the SYSTEM or PRODUCT.

A Word of Caution 34.5

Constant Failure Region Misnomer

Exercise caution when referring to this period as the Constant Failure Region. Technically, it is the Stabilized Failure Region (SFR) Region, $h(t)$, not Constant **Failure** Rate Region, $\lambda(t)$. A stable Hazard Rate applies *only* to SYSTEMS or ENTITIES that exhibit an Exponential Distribution PDF, $f(t)$, shown in Figure 34.3 Panel D.

For those SYSTEMS or ENTITIES that exhibit an Exponential Failure PDF, the SFR is characterized by:

1. The Instantaneous Failure Rate, $\lambda(t)_{\text{Inst}}$, which is:
   a. Random as a function of the continuous variable, *time*.
   b. Low and continues to diminish slightly over time as illustrated in Figure 34.5 Panel A.
2. The Hazard Rate, $h(t)$, which is constant as shown by Eq. 34.24.

Therefore, general characterizations for the Bathtub Curve's HRR midsection as being a "Constant Failure

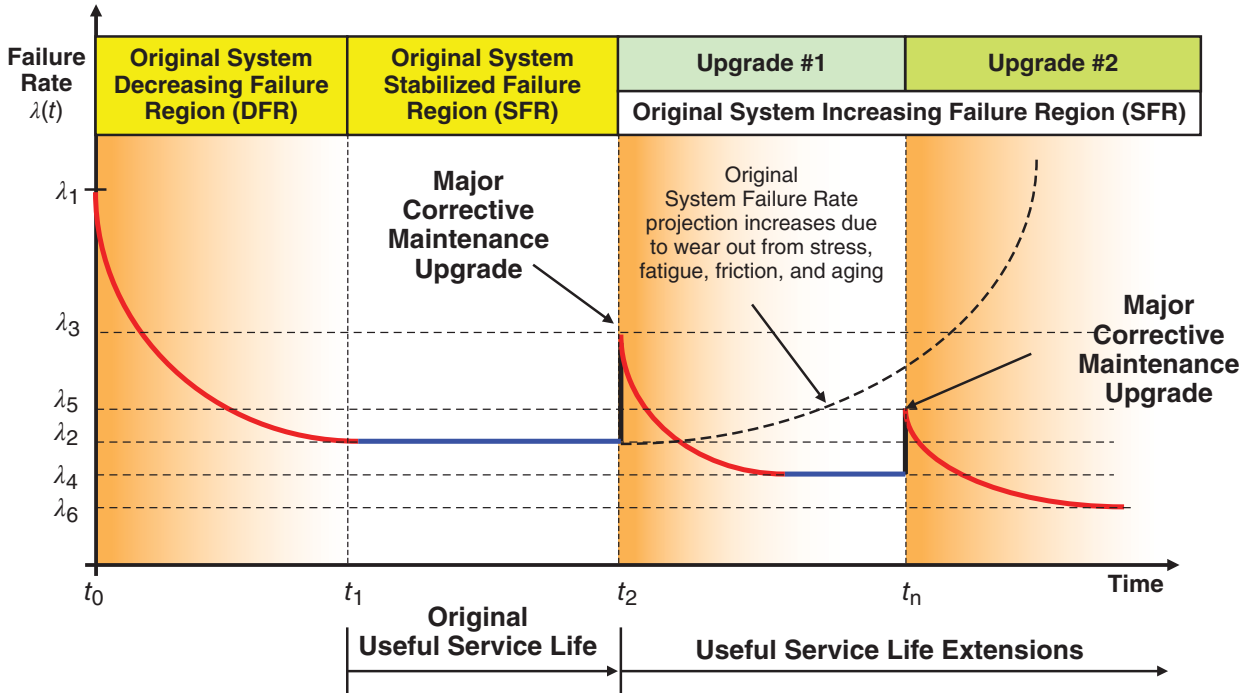


Figure 34.10 Equipment Service Life Extension Program (SLEP) Strategy

Region" is only true mathematically for SYSTEMS or ENTITIES that exhibit an Exponential Distribution PDF.

A Word of Caution 34.6

Wilkins (2002) *cautions* "Do not assume that a product will exhibit a constant failure rate. Use life testing and/or field data and life distribution analysis to determine how your product behaves over its expected lifetime."

Failures in the Stabilized Failure Region (SFR) occur *randomly* by chance and tend to remain relatively *stable* over a finite period of time up to several years. As a result, most SYSTEM OR ENTITY Reliability estimates are based on this period and easy math. In contrast, the math for the Decreasing Failure Region (DFR) and later Increasing Failures Region (IFR) tend to be more complex. Adding to Wilkins' (2002) (A Word of Caution 34.4):

A Word of Caution 34.7

Inappropriate Application of the PDF

"The exponential distribution and the related Mean Time Between Failures (MTBF) metric are appropriate for analyzing data for a product in the 'normal life' period, which is characterized by a constant failure rate. But be careful—many people have "imposed" a constant failure rate model on products that should be characterized by increasing or decreasing failure rates, just because the exponential distribution is an easy model to use." Wilkins (2002)

During the SFP, periodic *preventive* and *corrective* maintenance actions are performed to:

1. Eliminate any random time-based failures.
2. Prolong the Useful Service Life of the SYSTEM or ENTITY. (Figure 34.10)

Despite disciplined maintenance, over time the SYSTEM or PRODUCT failures begin to increase signaling the beginning of the Increasing Failure Region (IFR).

34.3.4.7 Bathtub Curve—Increasing Failure Region (IFR)

Principle 34.20

Wear-out Principle

Every system, product, or service in active service exhibits a failure rate that increases with age due to wear-out; degradation of components and their mass properties; and User use, misuse and abuse, and misapplication in its OPERATING ENVIRONMENT.

The Increasing Failures Region (IFR), which is sometimes referred to as the Wear-out or End-of-Life Region, begins when the Failure Rate, $\lambda(t)$, begins to increase. The increase is due to PART wear-out by deterioration, fatigue, depletion of materials, resulting from environmental, frictional, and stress loading conditions over time.

Remember—The Bathtub Curve reflects SYSTEM or PRODUCT Level failures, not individual PARTS *per se* (Caution 34.4). However, a single PART Level component may

fail leading to a chain of reaction of failure effects that result in an overall accident or incident (Figure 24.1). As a result, a SYSTEM or PRODUCT is only as reliable as its weakest mission critical System Element and component (Principal 34.4). Wilkins (2002) makes the following observation:

> "The shortest-lived component will determine the location of the wear-out time in a given product. In designing a product, the engineer must assure that the shortest-lived component lasts long enough to provide a useful service life. If the component is easily replaced, such as tires, replacement may be expected and will not degrade the perception of the product's reliability. If the component is not easily replaced and not expected to fail, failure will cause customer dissatisfaction." (Figure 5.2)

The essence of *wear-out* failures is their impact of "wear and tear" *stress* on a PART's remaining *strength* to perform and withstand the stress. Table 34.2 provides a listing of OPERATING ENVIRONMENT conditions and stress effects[2].

Example 34.9

Wear-out Failures

Examples of wear-out failures include: mechanical bearings that fail due to loading and frictional effects, mechanical bushings that wear through due to loading and friction; solder joints that crack; highway bridge beams that crack due to loading, thermal expansion and contraction, and corrosion; electronic cable conductors that break due to constant back and forth movement; electrical switches and relays that fail due to constant use.

The Reliability Information Analysis Center (RIAC) (2004) provides an illustration of the effects of *Stress* versus *Strength* relationships in (Figure 34.11). O'Connor (1991, p.5) refers to these as *Load* versus *Strength* relationships.

Since Chapter 34 focuses on Lifetime Data Distributions, the general appearance of Figure 34.11 may appear to be just another failure profile of two intersecting distributions. The point being communicated may not be apparent. Take another look and think of it in this manner.

Principle 34.21

Strength-Stress Principle

Every SYSTEM or ENTITY should be selected with a *strength* design margin that exceeds its prescribed OPERATING ENVIRONMENT conditions and *stresses*.

[2]Refer to NASA PD-EC-1101 (1995) for a more detailed listing of Environmental Factors.

Humans, Enterprises, and Engineered Systems possess an inherent level of *capability* to perform—*strength*. When external forces—*stresses*—exceed strength, the capability degrades and inevitably fails. That is the central theme of Figure 34.11. SYSTEMS or ENTITIES are Engineered and designed to withstand exposure to *forces* and *conditions* in their OPERATING ENVIRONMENT prior to, during, and after a mission. When those forces and conditions—*stresses*—exceed their inherent capabilities—*strength*, either instantaneously or over time, SYSTEM OR ENTITY deteriorates, wears out, collapses, or simply fails.

Figure 34.11 Panel A illustrates how the range of *stressful* forces or conditions distribution overlaps the *strength* distribution. Observe the size of the overlapping shaded area between the stress-strength distributions. The shaded area clearly indicates a region that exhibits a *higher probability* of the item's *stresses* exceeding *strength*.

Now, consider what happens when a SYSTEM OR ENTITY is Engineered with a *safety factor* (Chapter 31) to ensure that its *strength* tolerance exceeds the worst-case *stress* conditions of its OPERATING ENVIRONMENT. Figure 34.11 Panel B provides an illustration. Observe how much the shaded area has shrunk indicating a *lesser probability* that OPERATING ENVIRONMENT stresses will exceed the component's strength.

Besides the obvious, *what is the point here*? SYSTEMS or ENTITIES that: (1) consist of a robust architectural network configurations – Series, Parallel, Series-Parallel (Figure 34.13) - discussed later, (2) have been Engineered with a *safety factor* (Chapter 31) that drives PART selection, and (3) have undergone strong Verification and Validation (V&V) activities and PARTS "burn-in" methods, the results should be analogous to Figure 34.11 Panel B.

Returning to the Increasing Failure Region (Figure 34.6), *corrective maintenance* costs begin to escalate significantly driven by THE ENTITY's R&R. Eventually, the System Owner is confronted with a decision that involves two trade-offs: (1) *refurbish* or *upgrade* the existing SYSTEM or PRODUCT or (2) purchase a new one and retire/dispose of the existing SYSTEM or PRODUCT.

34.3.4.7.1 A Word about MTBF and the Bathtub Curve

Most people tend to think of MTBF as universally applicable to all regions of the Bathtub Curve. However, a word of caution:

A Word of Caution 34.8

Radle and Bradicich (2013) point out that the scope of MTBF applies *only* to *random* chance failures that occur within the Useful Service Life—SFR Region—before wear-out. MTBF *does not* account for Early Life or Wear-Out failures.

TABLE 34.2 Examples of Operating Environment Conditions and Stress Effects

Environmental Conditions	Environmental Stress Effects
• Acceleration • Altitude • Dust/Sand • EMI • Fungus microbes • Humidity • Improper Assembly • Hail/Sleet • Ice/Snow • Noise—Acoustical • Overload • Radiation—Solar • Rain/Mist • Shock—Loads • Shock—Pyrotechnic • Salt/Fog • Temperature—Extremes • Temperature—Shock cycling • Temperature—Altitude • Vibration—Random • Vibration—Sine • Wind	• Abrasion • Absorption • Aging (oxidation) • Arching • Buckling • Chafing • Clogging • Compression • Contamination • Corona • Corrosion • Cracks/Fractures • Deformation • Degradation • Deterioration • Dissimilar metal interactions • Etching • Evaporation • Expansion/Contraction—Thermal • Fatigue • Erosion • Interference • Jamming • Leakage • Misalignment/Creep • Outgassing • Puncture • Resonance • Rupture • Tension • Wear

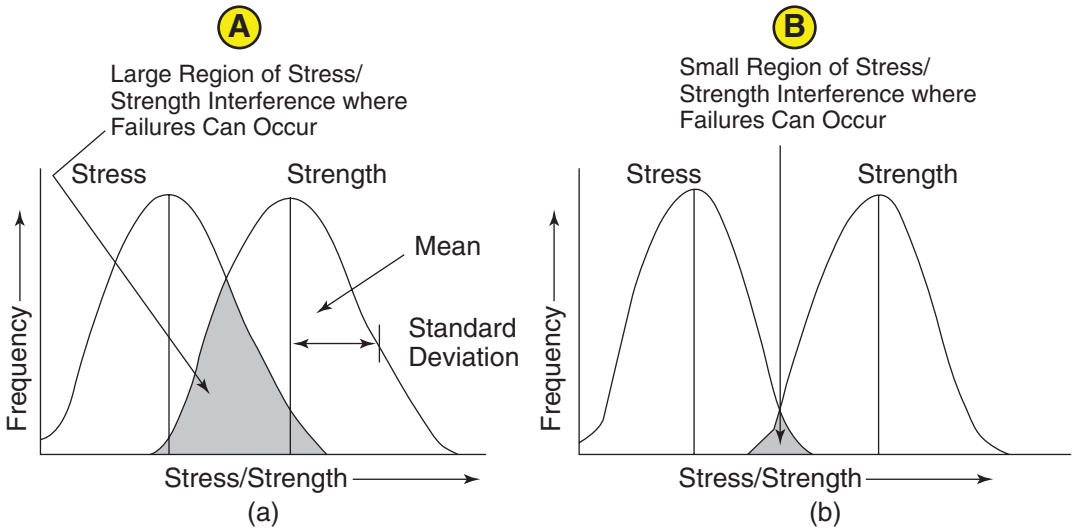


Figure 34.11 Part Stress versus Strength Relationship (Source: The Reliability Information Analysis Center (RIAC), 2004, Used with Permission)

They also caution that MTBF *should not* be used to calculate the *duration* of the Useful Service Life of a System or one of its Entities.

34.3.4.8 Service Life Extension Program (SLEP)

Principle 34.22

Useful Service Life Extension Principle

The Useful Service Life of a system, product, or service can be extended by proactive, Just-in-Time (JIT) sequences of *preventive* and *corrective* maintenance actions and upgrades.

If the System Owner decides to pursue a *refurbishment* or *upgrade*, the Useful Service Life typically can be extended via a Service Life Extension Program (SLEP) as shown in Figure 34.10. Otherwise, the System or Product progresses into the Increasing Failure Region (IFR) indicated by the dashed line and failures begin to increase. Key Point: determine when the IFR is expected to occur and perform the *refurbishment* or *upgrade* before that point in time.

The DAU, for example, defines a Service Life Extension Program (SLEP) as follows:

- **SLEP** "Modification(s) to fielded systems undertaken to extend the life of the system beyond what was previously planned" (DAU, 2012, p. B-202).

Referring to Figure 34.10, let's assume the SFR has a *random* Failure Rate, $\lambda(t_1 - t_2) = \lambda_2$, over the time interval. At t_2, a new technology upgrade is installed resulting in an *increased* Failure Rate, $\lambda_2(t_2)$, initially due to the newly installed components. *Corrective actions* are performed to remove any residual latent defects bringing the interval Failure Rate, $\lambda(t_2–t_n)$ down to λ_4. At t_{n+}, another upgrade is installed and the process repeats itself.

Recognizing that is an ideal scenario, *how do we know that the overall System Reliability Instantaneous Failure Rate will taper down to λ_4 and λ_6*? We do not know. The presumption here is that if you have an existing System or Product that has significant Stabilized Failure Region (SFR) remaining, new technology upgrades should result in *increased* reliability resulting in *less* maintenance. The reverse could happen as well, especially if the System Developer is not familiar with the technology, the technology is immature, or misapplied.

34.3.4.9 System or Product Shelf Life

Principle 34.23

Shelf-Life Principle

Every System or Entity *inherently* has a "use or lose" shelf-life that *diminishes* due to the *deterioration* of material mass properties over time and *exposure* to environmental condition extremes—temperature, humidity, and other factors.

Our description of the Bathtub Curve assumes that a System or Product is: (1) fully operational and (2) employed by the User from the time it is released. However, some Systems or Products may be placed in storage that may or may not be environmentally controlled. In some cases, Part Level components such as lubricant seals, gaskets, and materials have a "shelf life" and naturally deteriorate over time. When taken out of storage, the "new and unused" System or Product may experience failures—*stress* versus *strength* (Figure 34.11)—based on a condition that could be *incorrectly* categorized as *wear-out*.

34.3.4.10 Bathtub Curve Cautionary Note The key theme from our discussion of the Bathtub Curve is as follows:

- Each region of the Bathtub Curve is characterized by *different* Lifetime Data Distributions (Figure 34.6) that vary by System or Entity electronic versus mechanical devices.
- Most reliability estimates are based on the SFR consisting of a *constant* Hazard Rate, $h(t)$, which is *only applicable* to Systems or Entities with Exponential Distributions.

To illustrate how the Hazard Rate, $h(t)$ is *misapplied* (Caution 34.5) in calculating System Reliability estimates, consider the following example:

Example 34.10

Misapplication of the Constant Hazard Rate, *h(t)*

To illustrate how the Constant Hazard Rate is *misapplied* by the assumption that the Bathtub Curve's three failure regions are represented by a Constant Hazard Rate, $h(t)$, consider Figure 34.12 concerning human mortality rates.

Using the Weibull Model as a reference for characterizing the true profile of human mortality, observe how the *improper application* of a Constant Hazard Rate, $h(t)$, would:

- O*verestimate* the true mortality rate for $T(\text{Age}) \leq 60$ years old.
- *Underestimate* the true mortality rate for $60 < T(\text{ Age}) < 100+$ years old.

In summary, make sure you understand the Lifetime Data characteristics of the System or Entity *before* making assumptions about the Lifetime Data Distribution to use (Caution 34.6). Then, *validate* that the Lifetime Data Distribution accurately reflects the data.

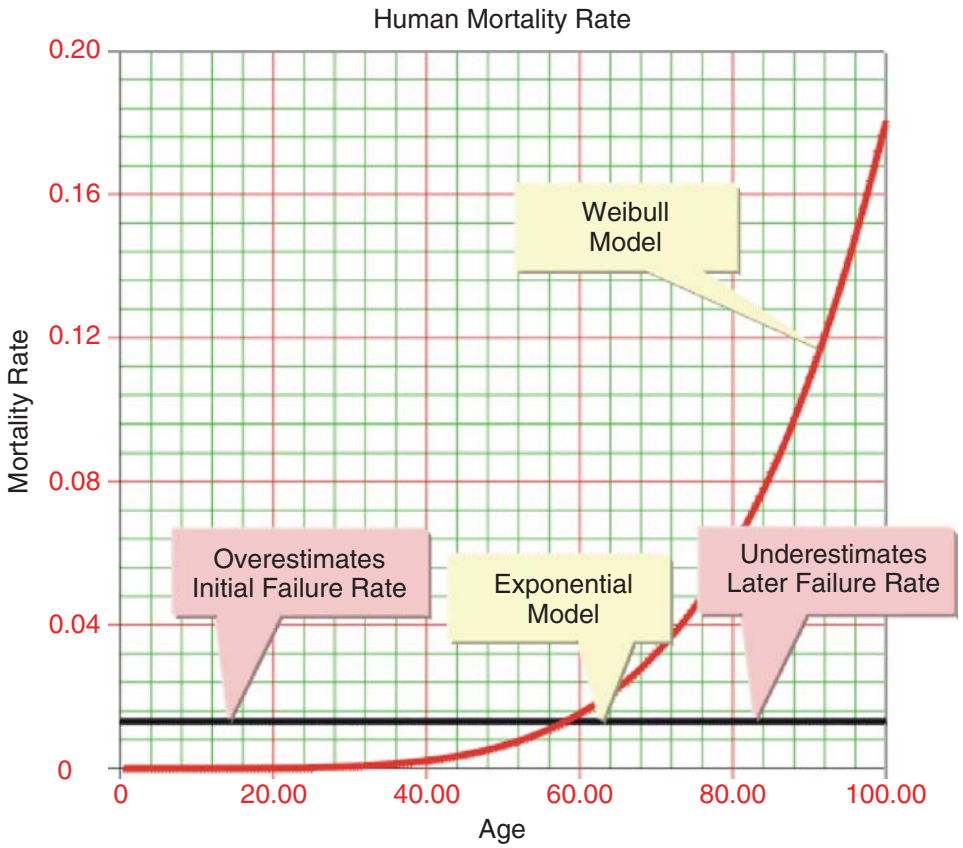


Figure 34.12 Illustration of Misapplication of the Constant Hazard Rate, $h(t)$, to Human Mortality (ReliaSoft, 2001—Used with Permission)

Heading 34.2 At this juncture, we have established a foundation in understanding Lifetime Data Distributions in terms of selecting SYSTEMS or ENTITIES to integrate into HIGHER-LEVEL SYSTEMS (Chapter 9). As Entities we can arrange them in different configurations—System Architecting—to achieve SYSTEM or PRODUCT outcomes and performance. How those configurations are structured in combination with ENTITY or PART failures characteristics determines a SYSTEM or PRODUCT's Reliability. This brings us to our next topic, System Architecting (Chapter 26) Reliability Network Configurations.

34.3.5 Architecting Reliable Systems and Products

A paradigm that pervades most Engineering Enterprises is the notion that we architect a SYSTEM or PRODUCT and then initiate the System Design. When parts lists become available, we initiate a Parts Count Estimate introduced later to determine the SYSTEM or ENTITY design will comply with the System RMA requirements specified in the SPS. Observe two key points here:

1. Someone accorded the title of System Architect creates a vision of an elegant architecture without spending due diligence time (Mini-Case Study 26.1) to determine if the SYSTEM or ENTITY will meet the SPS or EDS RMA requirements.
2. Designers spend hours creating a multi-level design. Then, emerge with PARTS Lists to see if by chance the result will be reliable.

What is wrong with this picture?

System Architectures are not created in vacuum. They require insightful knowledge not only of the capabilities to be provided but also how to maintain continuity and avoid disruption to system capabilities to ensure completion of the mission. That does not mean that SUBSYSTEMS, ASSEMBLIES, SUBASSEMBLIES, or PARTS will not fail. It means that *if they fail*, the SYSTEM or PRODUCT will continue to deliver those capabilities without disruption. To illustrate this point, consider the following example:

Delivery of Reliable Mission Performance With Minimal Disruption

Example 34.11 Laptop computers are designed to operate from a *primary* power source such as 110 vac, 60 Hz.

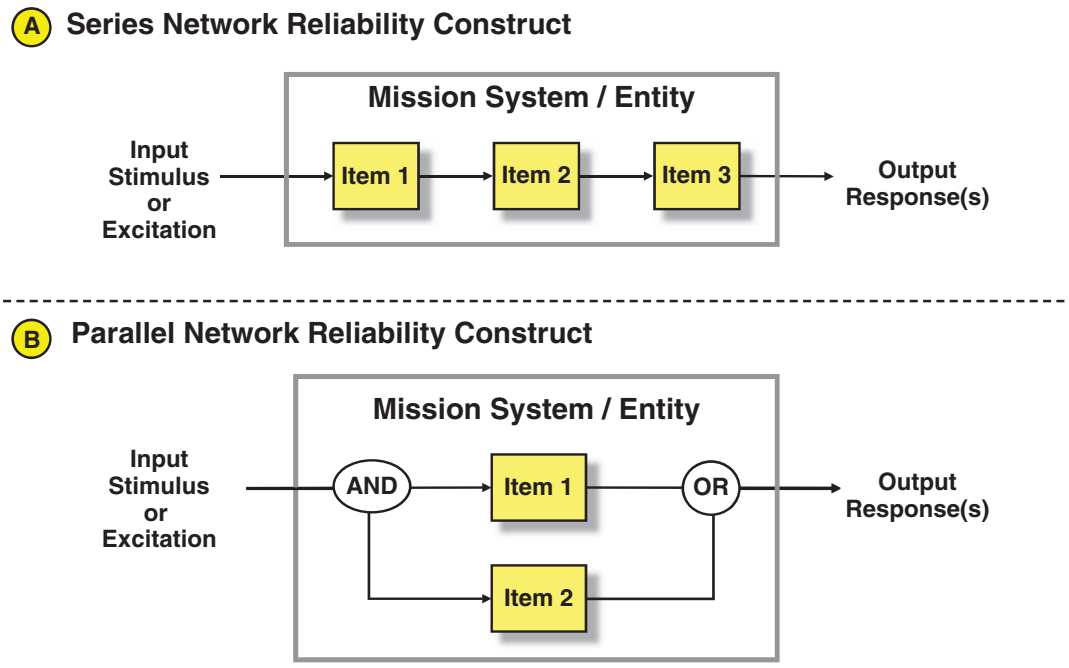


Figure 34.13 Examples of Series and Parallel Network Configuration Constructs

An internal, rechargeable battery is included to serve as a back-up power source for remote operation when the 110 vac power is not available. The same is true for a desktop computer with a Universal Power Supply (UPS) as an external back-up power source.

From a System Reliability perspective, if a laptop or desktop computer loses primary power, the SYSTEM shuts down, resulting in a potential loss of data and frustration by its User. That is, the User's mission has been *disrupted* and obviously not completed.

As a result, laptop and desktop computers are designed to switch over to back-up power sources, if available. Although the User's mission may be disrupted and inconvenienced by the power failure, the performance-based mission outcome is to preserve their data by having *sufficient* time to store the data and close any open files. Since non-emergency power failures typically last from a few minutes to a few hours, it is conceivable that if the back-up power source has sufficient energy, power may be restored before the battery has lost its energy thereby avoiding disruption to the User's mission.

Architecting reliable SYSTEMS or PRODUCTS requires insightful knowledge into how various architectural configurations—networks—affect System Reliability. From a System Development perspective, we normally begin with allocation of System RMA requirements from the SPS to the System Element Architecture (Figure 8.13)—PERSONNEL, EQUIPMENT, and so on. Then, within the EQUIPMENT Element, allocate its SPS RMA requirements to PRODUCTS → SUBSYSTEMS, and so on. However, that approach is dependent on understanding *how* System Reliability is calculated based on the architectural network configurations. Given that dependency, let's reverse the normal approach and begin with some elemental building blocks.

Architecturally, System Reliability estimates are calculated based on the three types of architectural network configurations: (1) Series, (2) Parallel, and (3) Series–Parallel. If the titles sound familiar, the computations are similar to Electrical Engineering network configurations.

34.3.5.1 Series Network Configuration Reliability

Principle 34.24 **Series Network Reliability Estimates**

The *reliability* of a Series Network Configuration, R_{Series}, composed of "n" components is computed as the product of the reliabilities in the general form of $R_{\text{Series}}(t) = [R_1(t)\ R_2(t)\ R_3(t)\ \ldots\ R_n(t)]$.

A *Series Network Configuration* consists of two or more ENTITIES connected in *series* as shown in Figure 34.13 Panel A.

Mathematically, we express this relationship as follows:

$$R_{\text{Series}}(t) = [R_1(t) \cdot R_2(t) \cdot R_3(t)\ \ldots\ R_n(t)] \qquad (34.36)$$

where:

- $R_{\text{Series}}(t)$ = Overall reliability of the Series Network Configuration

- $R_n(t)$ = Reliability of Component #n

Substituting Eq. 34.3 in Eq. 34.36:

$$R_{\text{Series}}(t) = e^{-(\lambda_1 + \lambda_2 \ \lambda_3 \ \ldots \ \lambda_n)(t)} \tag{34.37}$$

where:

- λ_n = Instantaneous Failure Rate, $\lambda(t)$, of the nth component as a function of time, t.
- t = Continuous variable.

Consider the following example:

Example 34.12 **Series Network Configuration Example**

Assume a simple SYSTEM with three SUBSYSTEMS connected in a Series Network Configuration as shown in Figure 34.13 Panel A. SUBSYSTEM #1 $R(t) = 0.985$, SUBSYSTEM #1 $R(t) = 0.995$, SUBSYSTEM #1 $R(t) = 0.99$. *What is the Reliability for the Series System*?

Applying Eq. 34.36:

$$\text{Series System } R(t) = [R_1(t) \cdot R_2(t) \cdot R_3(t)]$$

$$\text{Series System } R(t) = (0.985) \cdot (0.995) \cdot (0.99)$$

Therefore, Series System $R(t) = 0.97$ subject to EQUIPMENT and OPERATING ENVIRONMENT conditions.

34.3.5.2 Parallel Network Configuration Reliability

Principle 34.25 **Parallel Network Reliability Principle**

The reliability of a Parallel Network Configuration, R_{Parallel} is computed as a mathematical series of component reliabilities, $R(t)$ in the general form $1 - [(1 - R_1(t))(1 - R_2(t)) \ \ldots \ (1 - R_n(t))]$.

The second type of reliability network configuration consists of two or more entities connected in *parallel* as illustrated in Figure 34.13 Panel B. We refer to this construct as a *Parallel Reliability Network* and express this relationship as follows:

$$R_{\text{Parallel}} = (R_1(t) + R_2(t)) - R_1(t)R_2(t)$$

For one-out-of-two redundancies (34.38)

where:

- R_{Parallel} = Overall reliability of a bounded Parallel Network Configuration
- R_1 = Reliability of Component #1
- R_2 = Reliability of Component #2

Eq. 34.38 can become unwieldy, especially for large quantities of components. However, we can simplify the equation by restructuring it in the following form:

$$R_{\text{Parallel}} = 1 - [(1 - R_1(t))(1 - R_2(t)) \ \ldots \ (1 - R_n(t))] \tag{34.39}$$

where:

- Each $(1 - R(t))$ term represents the respective Parallel Component's *unreliability*.
- Eq. 34.39 assumes that *1-out-of-n* components are fully operational.

Observe the phrase "*1-out-of-n* Components are fully operational." For *mission* and *safety critical* systems such as aircraft, the Space Shuttle, medical devices or back-up generators, deliverable System procedures may specify that a *minimum* of *k-out-of-n* units are *fully operational* within SPS RMA requirements to properly continue with a mission. The total quantity of redundant components and quantity working simultaneously are system-dependent. Eq. 34.39 represents *1-out-of-n*. Examples include *2-out-of-4* and *2-out-of-5* combinations. Each set of equations is different.[3]

While each additional ENTITY in Eq. 34.39 interjects an additional component subject to failure and maintenance, the Parallel Network Configuration *increases* the overall System Reliability. To illustrate this point further, consider the illustrations shown in of Figure 34.14 Panels A–C. Note how the addition of *redundant* entities with *identical* reliabilities *increases* the overall reliability.

Example 34.13 **Parallel Network Configuration Example**

Assume a simple SYSTEM with three redundant subsystems connected in a Parallel Network Configuration as shown in Figure 34.14 Panel C. SUBSYSTEM #1 $R(t) = 0.985$, SUBSYSTEM #2 $R(t) = 0.995$, SUBSYSTEM #3 $R(t) = 0.99$. *What is the Reliability for the Parallel Network Configuration*?

Applying Eq. 34.39:

$$\text{Parallel System } R(t) = 1 - [(1 - R_1)(1 - R_2)(1 - R_3)]$$

$$\text{Parallel System } R(t) = 1 - [(1 - 0.985)(1 - 0.995)(1 - 0.99)]$$

$$\text{Parallel System } R(t) = 1 - [(0.015)(0.005)(0.010)]$$

Parallel System $R(t)$ = 0.99999 … Subject to EQUIPMENT and OPERATING ENVIRONMENT conditions.

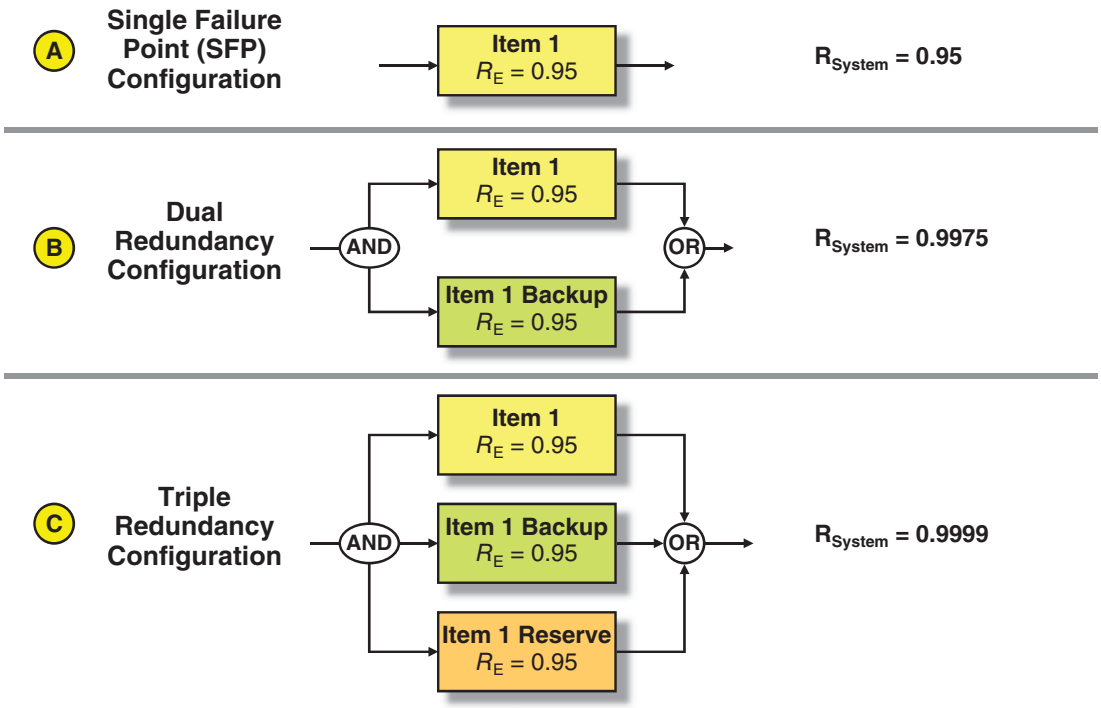


Figure 34.14 Illustration Depicting How Parallel Network Configuration Redundancy Improves System Reliability

We purposely used the same SUBSYSTEMS in the Parallel Network configuration example as we did in the previous Series Network Configuration Example 34.12. Observe how the multiplicative effects Series Network Configuration *reduce* Reliability, $R(t)_{\text{Series}}$, compared to the Parallel Network Configuration reliability, $R(t)_{\text{Parallel}}$. Since network configurations are often complex, R&M Engineers often use a Parts Count Estimate that assumes all PART Level components are connected in Series, which results in a lesser Reliability as a "Quick Look" estimate. If the "Quick Look" Reliability magnitude is greater than the SPS or EDS Reliability requirement, they *assume* that the actual SYSTEM or PRODUCT Reliability will be better.[4]

34.3.5.3 Series–Parallel Network Configuration Reliability The third reliability network construct consists of combinations of Series–Parallel branches as shown in Figure 34.15. We refer to this construct as a *Series–Parallel Network Configuration*. Computation of the Reliability is a multi-step process. To illustrate, consider the following example:

[3]Refer to RAC (2001) for a listing of *k-out-of-n* equations that can be applied to various combinations of requirements.

[4]For additional information, RAC (2001) provides a summary of "Redundancy Equations for Calculating Reliability" for 1-out-of-*n*, 2-out-of-*n*, 3-out-of-*n*, that must be working at any given instant of time.

Example 34.14 **Series–Parallel Network Configuration Problem**

Assume a simple SYSTEM with three redundant Subsystems connected in a Parallel Network Configuration as shown in Figure 34.15. SUBSYSTEM #A1 $R(t) = 0.99$, SUBSYSTEM #A2 $R(t) = 0.975$, SUBSYSTEM #A3 $R(t) = 0.995$, SUBSYSTEM #A4 $R(t) = 0.965$, SUBSYSTEM #B $R(t) = 0.98$, SUBSYSTEM #C1 $R(t) = 0.985$, SUBSYSTEM #C2 $R(t) = 0.995$, SUBSYSTEM #C3 $R(t) = 0.99$. *What is the Reliability for the Series-Parallel* SYSTEM?

Step 1—Compute **Product A Series Network** Reliability (Eq. 34.36) consisting of SUBSYSTEMS A1 and A3:

Product A Series Network (A1 – A3) Reliability,

$$R(t) = [R_{A1}(t) \cdot R_{A3}(t)]$$

Product A Series Network (A1 – A3) Reliability,

$$= [0.99 \cdot 0.995] = 0.985$$

Step 2—Compute **Product A Series Network** Reliability (Eq. 34.36) consisting of SUBSYSTEMS A2 and A4:

Product A Series Network (A2 – A4) Reliability,

$$R(t) = [R_{A2}(t) \cdot R_{A4}(t)]$$

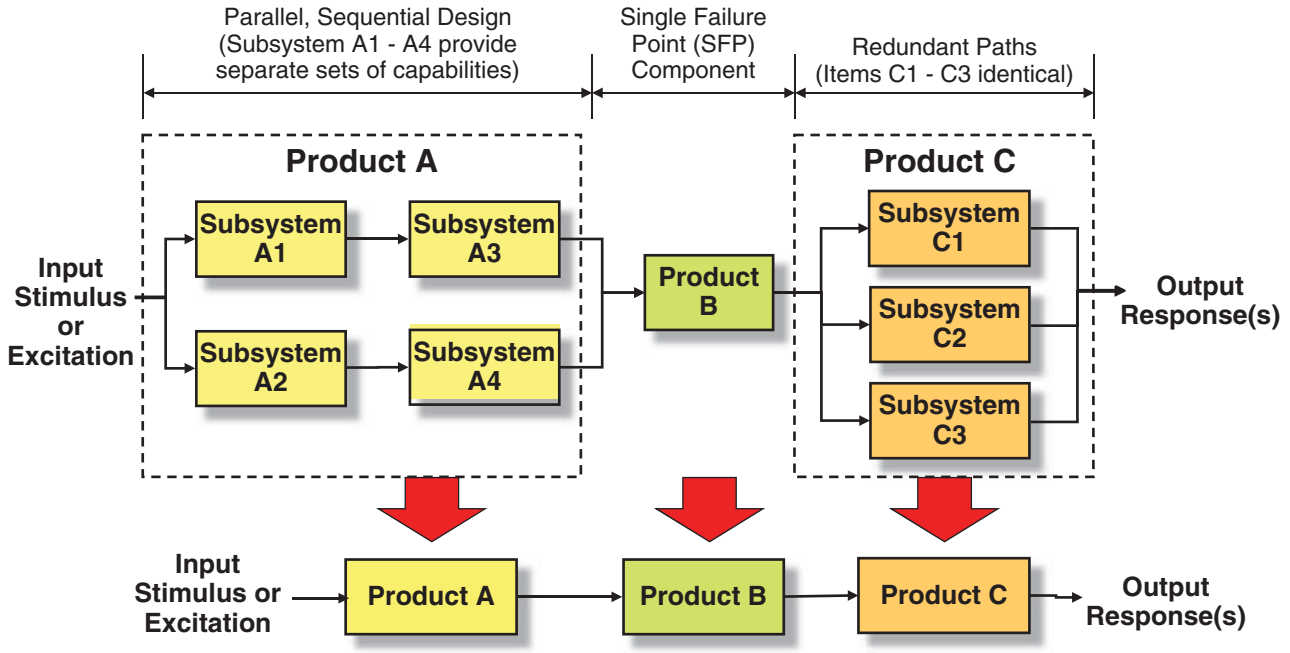


Figure 34.15 Example of a Series–Parallel Reliability Network Configuration

Product A Series Network (A2 – A4) Reliability,

$$R(t) = [0.975 \cdot 0.965] = 0.941$$

Step 3—Compute **Product A** Parallel Network Reliability (Eq. 34.39) consisting of SUBSYSTEMS (A1–A3)||(A2–A4):

Product A Reliability,

$$R(t) = 1 - [(1 - R_{A1||A3})(1 - R_{A2||A4})]$$

Product A Reliability,

$$R(t) = 1 - [(1 - 0.985)(1 - 0.941)]$$

Product A Reliability,

$$R(t) = 0.999$$

Step 4—Compute **Product C** Parallel Network Reliability (Eq. 34.39) consisting of SUBSYSTEMS C1 – C3:

Product C Reliability,

$$R(t) = 1 - [(1 - R_{C1})(1 - R_{C2})(1 - R_{C3})]$$

Product C Reliability,

$$R(t) = 1 - [(1 - 0.985)(1 - 0.995)(1 - 0.99)]$$

Product C Reliability,

$$R(t) = 0.99999$$

Step 5 – Compute **System** Reliability (Eq. 34.36) – Series Network PRODUCT A → PRODUCT B → PRODUCT C:

$$\text{System Reliability, } R(t) = [R_A(t) \cdot R_B(t) \cdot R_C(t)]$$

$$\text{System Reliability, } R(t) = [(0.999)(0.98)(0.99999)]$$

$$\text{System Reliability, } R(t) = 0.979$$

Author's Note 34.5

As a reminder … On the basis of the Series Network Configuration math (Example 34.12), observe that every component added to a SYSTEM or PRODUCT *increases* the *probability* of it experiencing a failure (proportional to the failure rate). Adding redundant—parallel—components increases both maintenance corrective actions and costs; however, it also increases the *likelihood* or *probability* of a successful mission. Although System Reliability has increased, System Maintainability cost increases as well due to the additional hardware and/or software.

34.3.5.4 Application of Reliability Network Modeling

Principle 34.26

Design Modifications Principle

The System Design Solution is not open to *random, ad hoc* modifications up until the day the SYSTEM or PRODUCT is accepted for delivery without major cost, schedule, and risk consequences.

Textbooks typically address System Reliability modeling as an introduction to Series, Parallel, and Series–Parallel network structures and leave the impression that network modeling should occur at all levels of abstraction. Theoretically and ideally, this is true. However, it may be impractical. *Why*? Several reasons:

- **Reason #1**—Reliability Engineers are being subjected to "quick turn-around" cycles to provide immediate System Design decision support feedback. This is due to the *evolving* and *maturing* designs provided by the designers that may be in a constant state of change, especially in *ad hoc* SDBTF-DPM Paradigm Enterprises. That *condition* is a separate matter that needs to be dealt with by management that recognizes the fallacies of the paradigm. Your role as an SE is to "maintain intellectual control of the problem solution" (Principle 1.3) and bring stability to ad hoc decision making (Principle 14.2). Contrary to management views exercising their power and authority, the System Design Solution is not open to *random, ad hoc* modifications up until the day the System or Product is accepted for delivery without major cost, schedule, and risk consequences.
- **Reason #2**—A System is simply a large collection of Part Level components physically integrated into levels of abstraction to achieve a higher level purpose—e.g., emergence (Chapter 3)—*bottom-up*. In general, the System Design Process (Figure 14.9) progresses *top-down* in maturity as a function time. The intent is to bring *stability* to higher-level decisions to enable lower levels to make *informed* decisions and stabilize. As discussed in Chapter 8, a System or Product is simply parts aggregated into Entities at various levels of integration. The Part Level components are the ones that provide the physical data that enable computation of reliability. Therefore, the theoretical foundation for network modeling begins with (1) the Part Level, which is the last level in many cases to be defined and (2) the integrated characteristics of higher-level Entities. This gives little time to establish large, complex, multi-level, network models. The best we can hope for is an *estimation* or *approximation* of reliability.[5]

Given an understanding of the Series–Parallel Network Configuration Models, we are better postured to address the top-down allocation of SPS Reliability requirements to the System or Product's Architectural framework that employs these models.

[5]For additional information concerning reliability modeling, refer to Nicholls (2007).

34.3.6 System Reliability Requirements and Performance Allocation

Reliability & Maintainability (R&M) Allocations Principle

Principle 34.27 Allocation and flow down of System Performance Specification (SPS) R&M requirements to lower levels of Entity Development Specifications (EDSs) should be based on:

1. A Reliability Allocations Block Diagram.
2. Supported by a documented R&M network configuration analysis including assumptions.

The discussions in the earlier part of the System Reliability focused on how System Reliability is used to assess the reliability of a proposed or existing System Design for compliance to SPS or EDS requirements. Effectively, this is an "after-the-fact" assessment as a design *evolves* and *matures*. The challenge is: *How do SEs allocate and flow down the SPS Reliability requirements "up front" to lower* Products, Subsystems, Assemblies, Subassemblies, *and* Parts? The answer is: a *highly iterative process* attempting to balance the allocations based on cost, performance, and risk. R&M tools as well as M&S of architectural network configurations can aid the process. The mechanism for allocating the System Reliability requirements is the Reliability Allocation Block Diagram shown in Figure 34.16.

For an easy starting point, we assume Subsystems #1–#3 operate in a Serial Network Configuration. This may or may not be true. However, for a *worst-case* starting point we assume they are in series. Remember, the System Reliability of a Series Network Configuration *diminishes* rapidly as a multiplicative product using Eq. 34.36. If the worst-case Series estimate represents achievable reliability targets for the Entities, then establish those as initial System Architectural configuration starting points. Otherwise, you will have to either: (1) *reallocate* reliability performance or (2) *innovate* an architecture for the Entity to meet the initial reliability performance target.

34.3.7 Parts Count Reliability Estimates

Parts Count Reliability Estimate Principle

Principle 34.28 Employ Parts Count Reliability Estimates as a "quick look" assessment of a System or Entity's reliability based on generalized assumptions about component type categories, failure rates, quantities, and quality factors operating in a Series Network Configuration.

During proposal efforts, a System Developer as an Offeror needs to be able to estimate a System's Reliability for compliance to the SPS Reliability requirements. The same

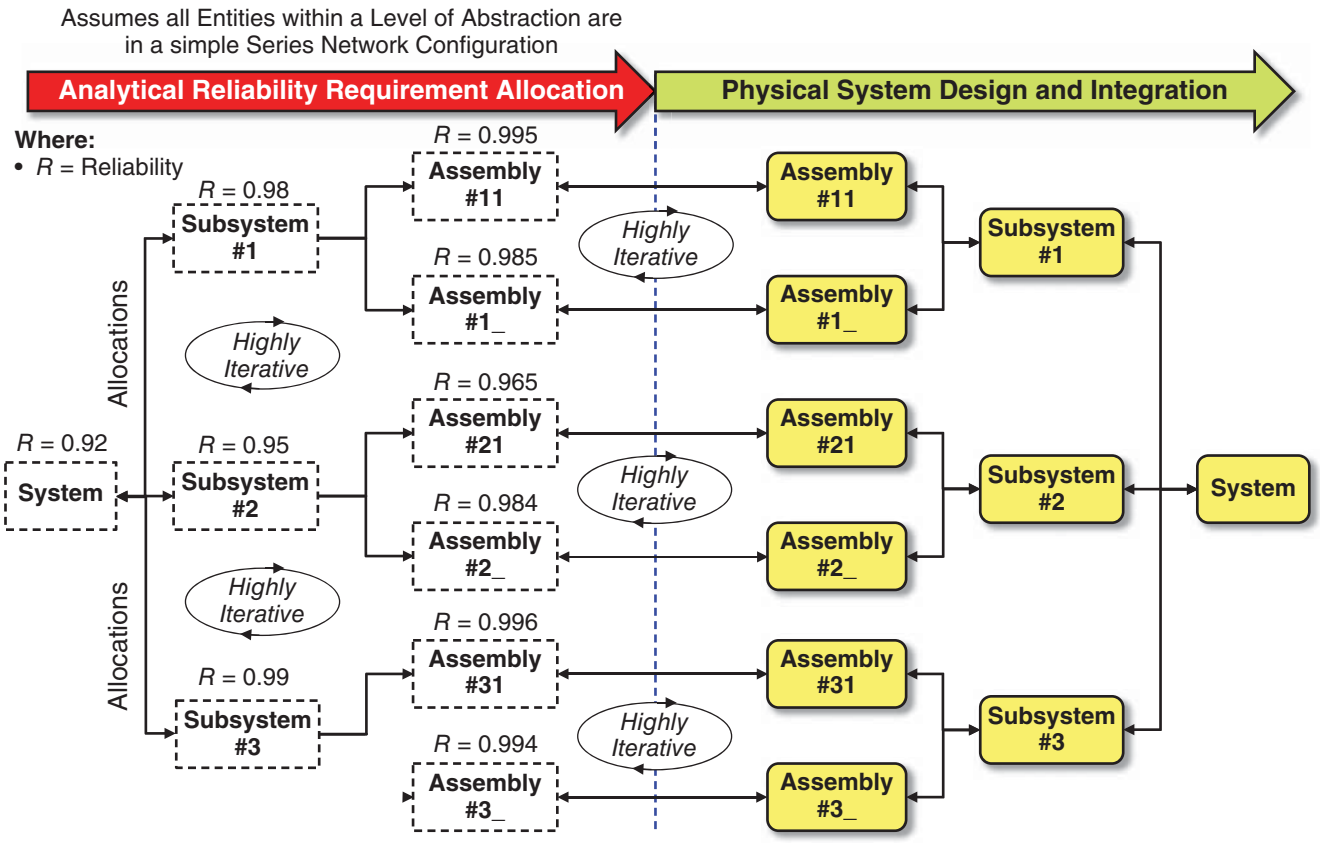


Figure 34.16 Example Illustration of a Reliability Allocation Block Diagram and Implementation

is true for lower levels of System Design in complying with the respective EDS Reliability requirements allocated to its specification. Obviously in these two cases, the SYSTEM or PRODUCT has not been designed; however, preliminary parts lists may exist. *How can we estimate the* SYSTEM *or* ENTITY'S *Reliability*? The answer resides in a method referred to as a *Parts Count Reliability Estimate*. MIL-HDBK-0217F defines the term as follows:

- MIL-HDBK-0217F, in general, provides detailed information about the failure rates of specific types of electronic components such as capacitors, resistors, and semiconductors. **Parts Count Reliability Estimate** "This prediction method is applicable during bid proposal and early design phases when insufficient information is available to use the part stress analysis models shown in the main body of this Handbook … " MIL-HDBK-0217F (1991) (p. A-1).

Parts Count Reliability Estimates are based on generalized:

1. Categories of PARTS—resistors, capacitors, Integrated Circuits (ICs), relays, motors, etc.
2. Failure Rates of PART Types.
3. Quantity of a given PART Type in each category.
4. Quality factors of each PART Type.

Referral Refer to MIL-HDBK-0217F Appendix A p. A1 "Parts Count Reliability Prediction" for details concerning the computation and instructions for distributed ENTITIES—PRODUCTS, SUBSYSTEMS—within the SYSTEM that may have different OPERATING ENVIRONMENT conditions. The appendix includes tabular summaries of: (1) *generic* failure rates for various categories such as microcircuits, discrete semiconductors, inductive and electromechanical devices and (2) quality factors.

For additional information about Electronic Reliability Design estimates, refer to MIL-HDBK-338B.

34.3.8 Failure Modes & Effects Analysis (FMEA)

System Reliability and FMEA-Based Architecting Principle

Principle 34.29 The *robustness* of a SYSTEM or ENTITY'S Architecture (Principle 26.17) is determined by its ability to *detect*, *tolerate*, and *contain* faults

(Principle 26.16) as a result of rigorous Failure Modes & Effects Analysis (FMEA) and *corrective actions* via its *compensating* and *mitigating* actions.

Simply constructing reliability network configuration models of the System Architecture and System Elements elements provides some insights for System Reliability. However, component failure effects range from *benign* and *non-threatening* to *catastrophic*. As a result, SEs need to understand: (1) *how* and in *what ways* a SYSTEM OR PRODUCT *fails* and (2) *what* the potential *ramifications* of failure propagations (Figure 26.8) are in terms of mission completion.

Engineered Systems should also undergo Safety Analysis to assess the risks and potential adverse impacts to the system, general public, and the environment. The Safety Analysis involves conducting a FMEA. *What is an FMEA*? The FAA SEM (2006) and MIL-HDBK-470A define an FMEA as:

- **FMEA** (FAA) "An evaluation process for analyzing and assessing the potential failures in a system, i.e. a systematic method of identifying the failure modes of a system, a constituent piece, or function and determining the effects on the next higher level of the design" (FAA SEM, 2006, Vol. 3, p. B-4).

  FMEA (DoD) "A procedure by which each potential failure mode in a product (system) is analyzed to determine the results or effects thereof on the product and to classify each potential failure mode according to its severity or risk probability number" (MIL-HDBK-470A, p. G-5).

These definitions lead to the question: *what is a failure mode*? MIL-HDBK-470A (p. G-5) defines a *failure mode* as follows:

- "**Failure Mode** The consequence of the mechanism through which the failure occurs, that is, short, open, fracture, excessive wear."

Failure Modes produce *failure effects* that can cause consequential injury or damage. MIL-HDBK-470A defines a *failure effect* as:

- **Failure Effect** "The consequence(s) a failure mode has on the operation, function, or status of an item. Failure effects are typically classified as Local, Next Higher Level, and End Effect" (MIL-HDBK-470A, p. G-5).

Based on these definitions, *what is the purpose of an FMEA*?

34.3.8.1 FMEA Purpose The purpose of the FMEA is to:

1. Understand *how* a SYSTEM or PRODUCT and its components might fail due to *misapplication, misuse,* or *abuse* by Users—operators, maintainers, poor design, or have one or more Single Failure Points (SFPs).
2. Identify, assess, and prioritize failure mode and effects risks.
3. Develop *compensating* or *mitigating actions* to minimize failure mode effects.

Author's Note 34.6

Compensating or Mitigating Actions

Failure effects can be mitigated and compensated in the Equipment Element design or through any of the other System Elements – Personnel (Training), Mission Resources – threat avoidance, or Procedural Data – Instructional or Operator's Manuals. As an example, refer to Mini-Case Study 24.1.

MIL-HDBK-470A (p. 4–34) also notes that:

> "The results of the FMEA are used to determine placement and nature of test points, to develop troubleshooting schemes, to establish design characteristics relative to the ease of maintenance, and to develop *fault detection and isolation* strategies." Refer to (Principle 26.16).

System Reliability encompasses more than simply evaluating the Physical System Design Solution to compute Lifetime Data Distribution metrics to verify compliance. Although this is a *necessary* condition for System Reliability, it is *insufficient* in terms of a sound SE practice to ensure SYSTEM or PRODUCT capabilities *survive* the duration of the mission. The true value in System Reliability estimates depends on *how* the estimates are incorporated as corrective actions into the *evolving* and *maturing* System Design Solution.

An FMEA assesses the total System Design Solution, not just SUBSYSTEMS, ASSEMBLIES, SUBASSEMBLIES, or PARTS. This includes: Physical Domain Solution architecture (Figure 14.1); external and internal interfaces—Human, external system; component reliabilities and performance de-rating assumptions; SFPs; and PART *proximity* to potential failure sources or conditions (Figure 26.12). Each of these conditions or *failure modes* may be self-contained within an ENTITY or even worse propagate beyond the ENTITY's boundaries (Figure 26.8) causing a chain reaction of *failure effects* as addressed earlier in Reasons (1990) Accident Trajectory Model application shown in Figure 24.1.

Another key point: Performing an FMEA illustrates *why* System Architecting requires more than lashing ENTITIES together into a framework and declaring it an *architecture*. The FMEA should include a focus on *eliminating* and *containing* faults within the System Architecture as discussed earlier in Principle 26.16 and Figure 26.8 illustrates.

To better understand what an FMEA is, let's explore its methodology.

34.3.8.2 FMEA Methodology The FMEA methodology is based on an analytical process that identifies a SYSTEM or PRODUCT's potential failure modes, their effects on performance, and determines *compensating* or *mitigating* design actions for correcting or reducing the effects.

Using Figure 34.16 as a reference, Mission performance of a SYSTEM or PRODUCT is driven by its: (1) OPERATING ENVIRONMENT—Threats—HUMAN SYSTEMS, NATURAL, AND INDUCED and (2) Operational Scenarios that impact all Levels of Abstraction. Analysis of: (1) the inherent, multi-level, System Design Solution and (2) the impact of the OPERATING ENVIRONMENT *stresses* on the residual *strength* of components (Figure 34.11) may produce various types of failure modes. Each failure mode may produce Failure Effects that jeopardize completion of the mission or its success.

When failure modes are identified, each is assigned a Unique Identifier for tracking purposes and subjected to a Risk Assessment. As part of a typical Risk assessment, each failure mode's *Probability or Likelihood of Occurrence* is assessed in combination with its Consequences of Failure to determine its Level of Risk. On the basis of the Level of Risk and Severity Classification, Mitigating Actions are formulated and submitted as *corrective actions* for updating: (1) Standard Operating Practices and Procedures (SOPPs) and (2) Design Modifications into the appropriate ENTITIES of the SYSTEM or PRODUCT. On the basis of the FMEA results:

1. SEs can orchestrate *design-mitigating actions* such as redesign and procedural changes (Figure 24.1) that enable cost-effective ways to mitigate the risks of failure mode effects.
2. System operators and maintainers can perform *compensating provisions* (Mini-Case Study 24.1) to recover from potential failure modes by applying *Systems Thinking* and corrective actions.

Note the *compensating provisions* by SYSTEM operators and maintainers. Perhaps one of the best illustrations of *compensating provisions* is captured in Mini-Case Study 24.1. Chuck Yeager applied compensating provisions *to* an F-86 Sabre test pilot aircraft to recover from a potentially fatal crash condition. His quick *Systems Thinking*

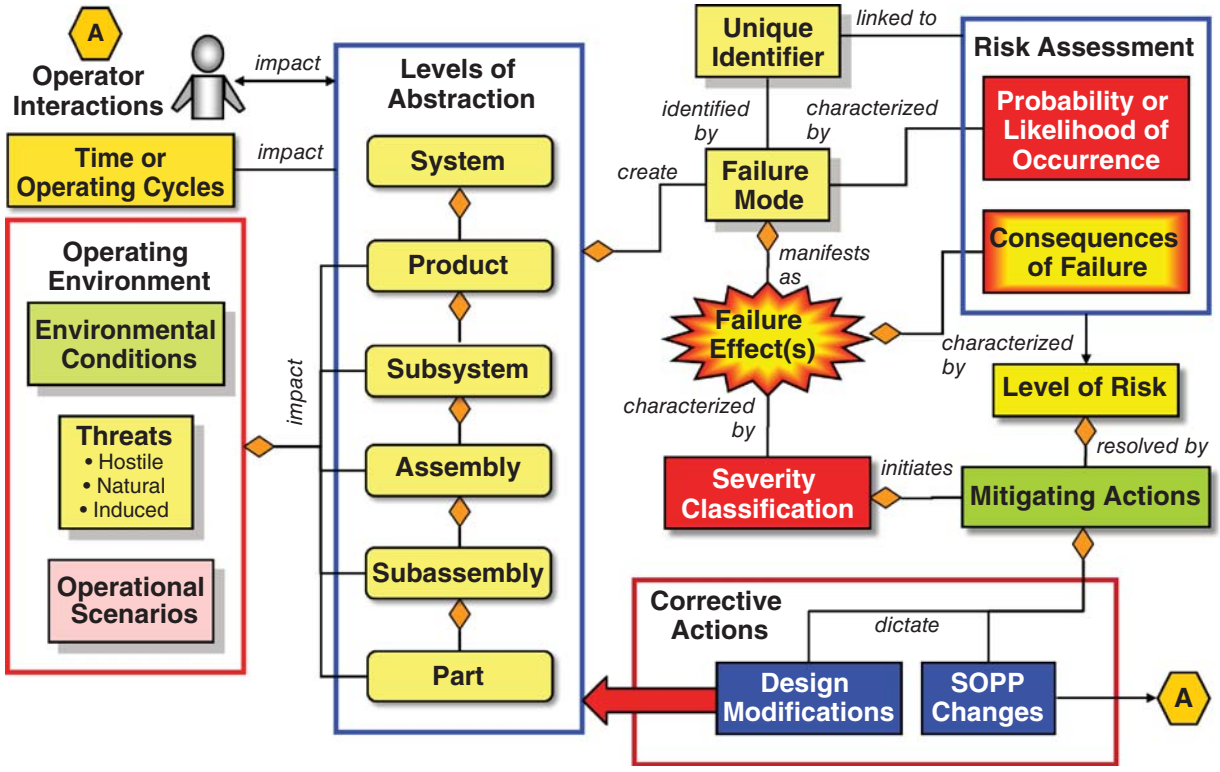


Figure 34.17 Failure Modes & Effects Analysis (FMEA) Entity Relationships

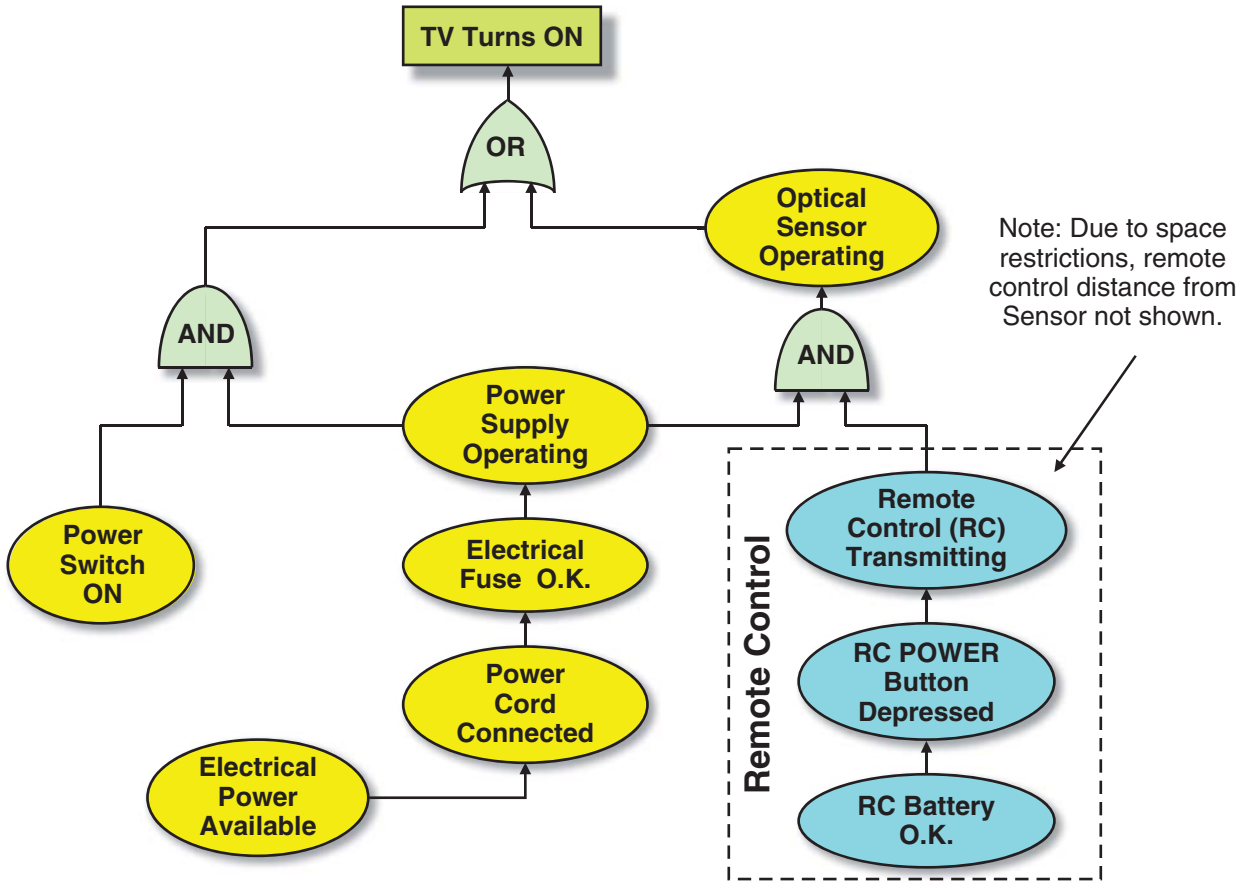


Figure 34.18 Fault Tree Example – Remote Controlled Television Power Activation

(Chapter 1) ultimately lead to solving a mysterious aircraft latent defect - *fault* - that migrated *undetected* in the aircraft's manufacturing process and cost several test pilot lives that lead to *design mitigation actions*.

This brings up a question: *How do we identify failure modes*? M&S of the Physical SYSTEM offers one method via Cause and Effect Analysis. Another method is Fault Tree Analysis (FTA), our next topic.

34.3.8.3 Fault Tree Analysis (FTA) One method for understanding *how* a SYSTEM or ENTITY may fail is to create fault trees. Figure 34.18 provides an example of a simple television hand-held remote. Boolean circuit symbology, such as AND and OR gates, is used to represent the sequence of dependencies and how each impacts the outcome, TV turns On.

Sometimes even FTA and M&S methods do not provide adequate identification of potential failure modes due to potential timing conflicts and OPERATING ENVIRONMENT conditions. As a result, testing becomes the *last line of defense* for detecting and isolating *latent defects*—faults—that can later manifest themselves as failure modes during the System Operations, Maintenance, and Sustainment (OM&S) Phase.

A Word of Caution 34.9

Remember—FTA can provide very valuable insights into the chain of logical and physical dependencies such as electrical, mechanical, and software interactions. As noted in Figure 34.18, the FTA does not provide insights concerning the impact of physical layouts and proximity to component failures that may cause other components to fail *unless* you make the effort to address such.

This point illustrates the critical importance of allocating as much time for SITE—up to 40% or more (Heuristic 28.1). Instead, Enterprises and projects, especially those that exhibit the *ad hoc* SDBTF-DPM Paradigm. These Enterprises allow design teams to wander around delivering the designs late leaving only 10–20% or less of the project schedule for SITE.

Practically, it is impossible to find every conceivable failure mode "on the ground in a controlled test environment." You have to subject the SYSTEM or ENTITY to actual use in its OPERATING ENVIRONMENT to discover *unknown* failure modes.

34.3.8.4 FEMA Risk Assessment Matrix FMEA also requires an insightful risk assessment of a Failure Mode's *probability of occurrence* and the *severity—consequences* of failure. MIL-STD-882E, for example, uses the Risk Assessment Matrix illustrated in Figure 34.19 to assess a potential failure mode's *probability* of *occurrence* and *severity*.

- **Probability of Occurrence Categories**—Represent a *qualitative* measure of the Probability of Occurrence—Frequent, Probable, Occasional, Remote, Improbable, and Eliminated.
- **Severity Categories**—Represent a *qualitative* measure of the potential Failure Mode effects—Catastrophic, Critical, Marginal, and Negligible.

Former MIL-STD-1629A defined *severity* as:

- **Severity** "The consequences of a failure mode. Severity considers the worst potential consequence of a failure, determined by the degree of injury, property damage, or system damage that could ultimately occur." (MIL-STD-1629A, p. 3).

One of the unique features of an FMECA is the assignment of Risk Priority Numbers (RPNs) that can be partitioned into Levels of Risk—Low, Medium, or High Risk. MIL-STD-882E RPNs are calculated as follows:

$$\text{Risk Priority Number (RPN)} = \text{Severity(S)} \cdot \text{Occurrence(O)} \tag{34.40}$$

Observe that Risk Levels are identified in intersecting cells of the *Probability of Occurrence* (rows) versus *Severity* (columns) of Figure 34.19 based on the RPNs.[6]

Dhillon (1999, p. 54) expands Eq. 34.40 to include an additional multiplicative factor, Detection Rate (DR), as shown below and rating criteria (Dhillon, Table 4.3, p. 54).

$$\text{Risk Priority Number (RPN)} = \text{Severity (SR)} \cdot \text{Occurrence (OR)} \cdot \text{Detection (DR)} \tag{34.41}$$

Remember—The FMEA Risk Assessment Matrix simply enables us to identify: (1) the *existence* of a risk and (2) its level of *significance*. That does not necessarily infer that *proactive corrective actions* have been initiated to mitigate the risk in the System Design Solution. To address this point, Enterprises and projects create FMEA worksheets that provide details for tracking Failure Modes. Although the inference here is a paper worksheet, FMEA data should be tracked via an online database accessible to project personnel on a need-to-know basis.

The preceding FMEA overview discussion brings us to our next topic, the FMEA Worksheet and its Data Collection.

34.3.8.5 FMEA Worksheet and Data Collection There are numerous ways of creating an FMEA Form for recording results of the analysis. Although MIL-STD-1629A was

[6]For additional information on these topics, refer to MIL-STD-882E (pp. 10–13) and US Department of the Army TM 5-698-2 (2006).

RISK ASSESSMENT MATRIX

SEVERITY / PROBABILITY	Catastrophic (1)	Critical (2)	Marginal (3)	Negligible (4)
Frequent (A)	High	High	Serious	Medium
Probable (B)	High	High	Serious	Medium
Occasional (C)	High	Serious	Medium	Low
Remote (D)	Serious	Medium	Medium	Low
Improbable (E)	Medium	Medium	Medium	Low
Eliminated (F)	Eliminated			

Figure 34.19 DoD FMEA Risk Assessment Matrix Example (Source: MIL-STD-882E, Table III, p. 12.)

cancelled many years ago and has not been replaced, it is still searchable via the internet.

Figure 34.20 refers presents the FMEA output as a *worksheet*—traditional paper. However, FMEA information should be tracked as Failure Mode-based records in an on-line database. The database should be accessible to all project personnel that have *Need-to-Know* access with specific create, read, modify, or delete access privileges. The *worksheet* is simply an online form that can be used to collect FMEA data and enter it into the database.

34.3.8.6 FMEA Focus Areas As evidenced by the FMEA Risk Assessment Matrix (Figure 34.19), some Failure Modes are significant, have a frequent *Probability of Occurrence*, and pose various levels of *Severity*. One of the challenges due to limited FMEA resources is *over-identification* of failure modes and effects. *How do you determine what is to be included in an FMEA*? DoD 4151.22-M (2011, p. 8) provides the following guidance:

> "… if a failure mode meets one or more of the criteria (listed below), then the failure mode should be included in the analysis:
>
> (a) Those that have happened before.
> (b) Those that have not happened but are real possibilities.
> (c) Those that have not happened and are unlikely to occur but have severe consequences.
> (d) Those currently managed by a failure management strategy."

Observe the reference in the last item above to "a failure management strategy." The FMEA process is *ongoing* throughout the System Development Process. As Failure Modes are identified, they are assessed and tracked via the FMEA database for risk *mitigation* and preferably *elimination.*

34.3.8.7 FMEA Outputs The results of an FMEA consist of recommendations in the form of *compensating or mitigating actions* for a SYSTEM or PRODUCT. From a System Maintainability perspective, MIL-HDBK-470A suggests that the primary outputs of an FMEA include:

- "Identification of single point failures.
- Fail-safe design deficiencies.
- False alarm occurrences.
- Operator/maintenance person safety considerations.
- Potential failure detection methodology, including:
  a. Protective and warning devices.
  b. Failure over-ride features.
  c. Built-In Test (BIT) provisions" (MIL-HBDK-470A, p. 4–34).

FAILURE MODES and EFFECTS ANALYSIS (FMEA)

System: _____ Date: _____
Part Name: _____ Sheet: ___ of ___
Reference Drawing: _____ Compiled By: _____
Mission: _____ Approved By: _____

Item Number	Item / Functional ID	Potential Failure Modes	Failure Effects			Detection Method	Compensating Provision	Severity Class	Remarks
			Local Effects	Next Higher Level	End Effects				

Figure 34.20 Example of a Failure Modes & Effects Analysis (FMEA) Worksheet (Source: US Army TM 5-698-4 DA Form 7610, FMEA Worksheet Flow, August, 2006)

Avoid the notion that an FMEA is just a casual "Go Do" task (Figure 2.12). MIL-HDBK-470A explicitly states:

> "The FMEA should describe the means by which the occurrence of a specific functional failure (failure mode) is detected and localized by the operator or maintenance person" (MIL-HBDK-470A, p. 4–34).

Now, reread the MIL-HBDK-470A quote above. Observe that it didn't say "identify failures"; it said " … describe … how each failure occurrence "is detected and localized by the operator or maintenance person." This is a major paradigm shift from traditional Engineering; recognize and appreciate the difference!

34.3.9 Failure Modes & Effects Criticality Analysis (FMECA)

For Systems or Products that require high levels of reliability such as spacecraft, medical, military, or financial, the FMEA can be expanded to include a Criticality Analysis of specific component reliability and their effects. We refer to this as a FMECA. The US FAA defines an FMECA as:

- **FMECA** (FAA) "An analysis method used to identify potential design weaknesses through a systematic analysis approach that considers all possible ways in which a component may fail (the modes of failure); possible causes for each failure; likely frequency of occurrence; criticality of failure; effects of each failure on systems operation (and on various system components); and any corrective action that may be initiated to prevent (or reduce the probability of) the potential problem from occurring in the future" (FAA SEM, 2006, Vol. 3, pp. B-4, 5).

The FMECA should analytically identify Reliability Critical Items (RCIs) that require specification, selection, and oversight. Consider the following example:

Example 34.15 **Electrostatic Discharge (ESD) and Shelf-Life Considerations**

Failures in aircraft Flight Control System (FCS) sensors such as gyroscopes and accelerometers can result in major *accuracy* and *safety* issues. Therefore, these components should be designated as RCIs.

RCI solutions include: isolating RCI components, specifying higher reliability components, or implementing redundancy with less-reliability components. Redundancy, however, compounds the System's Reliability, and increases the Equipment - Hardware and Software - maintenance costs due to the increased parts counts.

Some Systems or Products also require special considerations such as Electro-Static Discharge (ESD) methods during manufacture and testing to preclude premature failures due to poor manufacturing handling procedures. Additionally, systems that are stored for extended periods of time before usage may have components with a *limited shelf life*. Therefore, factor ESD and shelf-life considerations into System Reliability estimates and design requirements.

FMECAs assess the criticality of Entities and Parts using a Criticality Matrix based on a number of factors that include Failure Mode Criticality Number and an Item Criticality Number.

Referral For additional information about FMECAs, please refer to:

- MIL-STD-1629A, pp. 102–1 to 102–7 for a more detailed description of FMECA development.
- US Army TM 5-698-4 (2006) for detailed descriptions of FMEAs and FMECAs.

34.3.10 System Reliability Summary

As a final point, one of the ironies of System Developer Enterprises is the compelling *need-to-know* the Reliability of the System. Pressures become relentless by executives to refine Reliability estimates to ensure compliance with the SPS—makes perfect Engineering sense. Then, as the physical Subassemblies, Assemblies, Subsystems, and Products are being developed and actual data become available, everyone looses interest in *validating* the original System Reliability estimates until the next project begins. At that point, it comes down to:

1. The professional discipline and initiative of an individual to periodically document even rudimentary reliability data for current and future project reference or …
2. Develop reliability models as legacy designs for use in the next System or Product application.

However, even these tasks are subject to being funded. If they are not funded, they are not performed.

When the next project starts, the *chaos* repeats itself. Objectively, if you work in an Enterprise that has a product line that is periodically updated or leverages and *validates* legacy designs and models with *actual field data*, there is no practical reason *why* you should not have reasonably accurate reliability models of these designs. Otherwise, you may not be in business very long!

34.4 UNDERSTANDING SYSTEM MAINTAINABILITY

From a business perspective, if a SYSTEM or PRODUCT experiences a failure, it produces an internal chain reaction. When a SYSTEM or ENTITY requires maintenance, (1) revenue ceases, and (2) money is expended to restore it operationally, which impacts profitability. The same analogy applies to Users and End Users such as: (1) consumers—who derive pleasure from commercial products such as TVs, smartphones, and devices; (2) medical patients requiring surgery, MRIs, kidney dialysis, and services; (3) transportation vehicle services—aircraft, trains, and (4) telecommunications cable networks.

The *efficiency* and *effectiveness* of maintenance actions to keep a SYSTEM or ENTITY operational and producing revenue is exemplified by its *maintainability*. More specifically, a SYSTEM or ENTITY's *maintainability* is a measure of its capability to be restored from a degraded or *failed state* to a specified level of performance within a designated time standard. Therefore, Maintainability Engineering, like many other disciplines, must be an integral part of the SE decision-making process during the proposal phase and beginning with Contract Award. The starting point begins with formulation of a Maintenance Concept introduced in Table 6.1.

Most textbooks approach Maintainability - like Reliability - with *equationitis*. Equations are mechanistic and *necessary* but *insufficient* for addressing the underlying concepts SEs need to understand to be able to apply them. To address this point, let's begin with a few of the underlying concepts of Maintainability. Figure 34.21 serves as a guide.

- Graph A—Illustrates a concept referred to as the EQUIPMENT Failure Condition Trajectory (Potential-Function Interval) (P-F) Curve addressed later in Figure 34.25. If we allow a SYSTEM OR ENTITY to Run to Failure (RTF), for example, lubricants degrade or breakdown to a point where they need to be replaced. When maintenance is lacking, the lubricants become a *fault* (Figure 26.8) or hazard (Figure 24.1) in the form of friction resulting in heat in mechanical systems. Due to the *unnecessary wear and tear*, the fault propagates (Figure 26.8) into surrounding components that may be *mission* or *safety critical*.
- Graph B—Illustrates the MTTF for *repairable* or *non-repairable* SYSTEMS OR ENTITIES. Let's assume

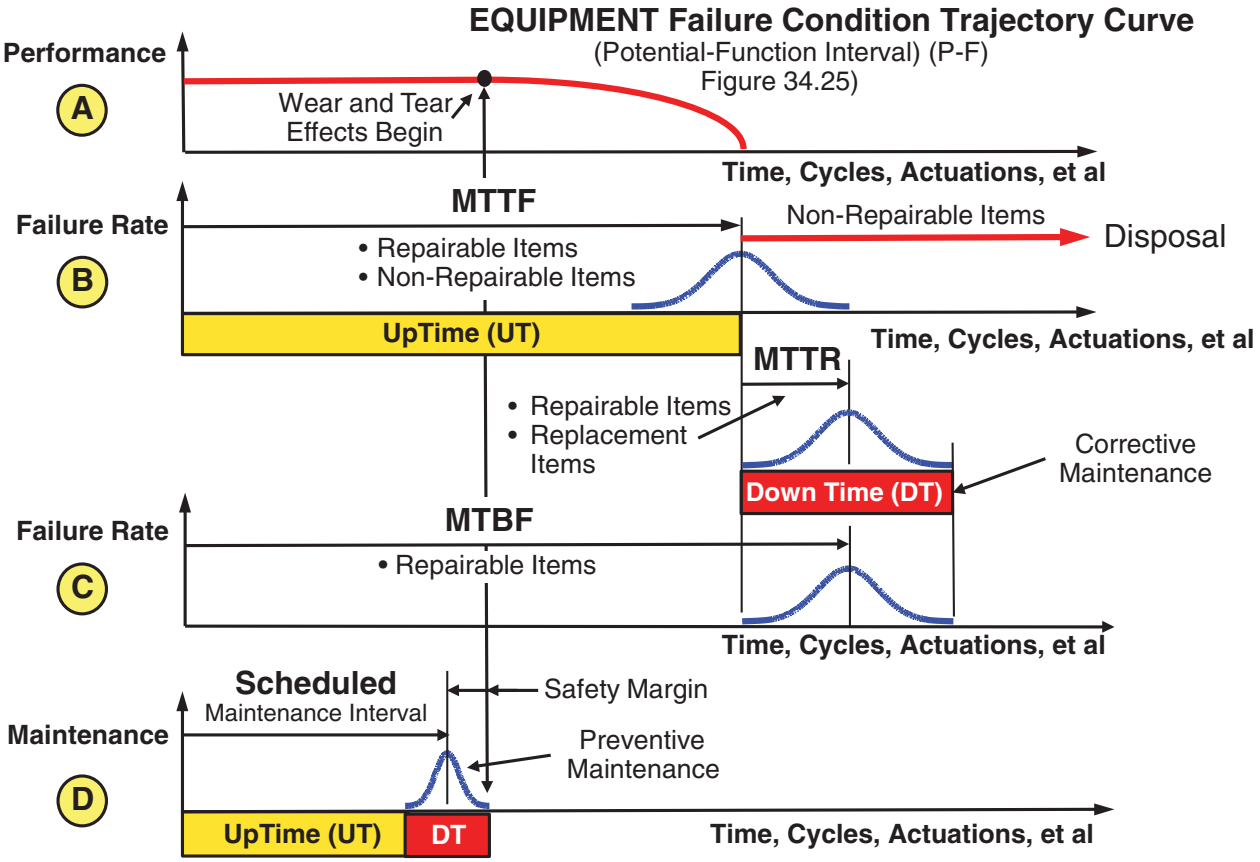


Figure 34.21 Underlying Concepts of Maintainability

that (1) when the P-F Curve (Graph A) hypothetically crosses the horizontal axis indicating failure and (2) the P-F Curve represents a statistically valid sample of SYSTEMS OR ENTITIES. Therefore, the MTTF (Graph B) represents those failures. *Non-repairable* SYSTEMS OR ENTITIES are: (1) *removed* and discarded via disposal processes and (2) replaced by components that may or may not be identical to the Original Equipment Manufacturer (OEM) components but provide the same form, fit, and function as the OEM SYSTEM OR ENTITY.

- Graph C—Illustrates the Mean Time To Restore (MTTR) the SYSTEM OR ENTITY to a specified serviceable condition. When the SYSTEM OR ENTITY is *unavailable* to perform missions, we refer to it as Down Time (DT). Where DT represents the *mean* for a diverse set of SYSTEMS OR ENTITIES, Mean Down Time (MDT) is used as the metric. During MDT, *corrective maintenance* actions are performed to repair and *restore* the SYSTEM OR ENTITY to a serviceable condition. This brings us to a key point for *repairable* SYSTEMS OR ENTITIES: *MTBF represents the MTTF plus MTTR.*
  Obviously, in the hypothetical discussion above, we do not allow SYSTEMS OR ENTITIES to Run-to-Failure (RTF) such as the light bulb example in Figure 34.1. This is where Maintainability becomes a key cornerstone of RMA. From a Risk and Opportunity Management (R&OM) perspective, *how does Maintainability reduce the: (1) risk of failure – risk avoidance – and (2) MDT*? The answer is: *scheduled* maintenance actions (Graph D).
- Graph D—Illustrates how *scheduled* or *periodic* maintenance actions are established based on a safety margin prior to the commencement of *wear and tear* (Graph A). We refer to this activity as *preventive* maintenance. When properly performed in a timely manner, the Useful Service Life of the SYSTEM OR ENTITY is prolonged.

Author's Note 34.7

MTTF – Repairable versus Non-Repairable Systems

Traditionally, R&M relegates MTTF to *non-repairable* items and MTBF to *repairable* items. The reality is: both repairable and non-repairable items have an RTF PDF with a central mean represented by the MTTF. Beyond that point, each is treated differently. *Non-repairable* items are *disposed*; *repairable* items are *restored* to a serviceable condition.

Given this background, let's begin with a discussion of Maintainability versus Maintenance – What is the Difference?

34.4.1 Maintainability versus Maintenance—What Is the Difference?

Principle 34.30

Maintainability versus Maintenance Principle

Maintainability is a system, product, or service *quality factor* attribute. *Maintenance* is a preventive or corrective action *activity*.

Engineers often loosely interchange *maintainability* and *maintenance* without realizing it. Observe that *Maintainability* is an SPS Section 3.2 Operational Performance Characteristic (Table 20.1) whereas *maintenance* is an *activity*. Let's examine definitions of the term from two perspectives of the US FAA and DoD:

- **Maintainability** (FAA) "The measure of the ability of an item to be retained in or restored to specified condition through maintenance performed, at each prescribed level of maintenance and repair, by appropriately skilled personnel using prescribed procedures and resources" (FAA-HDBK-006A, p. 12).
- **Maintainability** (MIL-HDBK-470A) "The relative ease and economy of time and resources with which an item can be retained in, or restored to, a specified condition when maintenance is performed by personnel having specified skill levels, using prescribed procedures and resources, at each prescribed level of maintenance and repair. In this context, it is a function of design" (MIL-HDBK-470A, p. 2–1).

Both definitions are virtually identical. However, notice the subtlety of the FAA's perspective versus the DoD's.

In contrast, *Maintenance* is a User Enterprise work activity that focuses on *repairing* or *servicing* the EQUIPMENT Element. MIL-HDBK-1908B defines *Maintenance* as:

- **Maintenance** "All actions necessary for retaining material in (or restoring it to) a serviceable condition. Maintenance includes servicing, repair, modification, modernization, overhaul, inspection, condition determination, corrosion control, and initial provisioning of support items" (MIL-HDBK-1908B, p. 21).

You can develop highly reliable SYSTEMS or PRODUCTS. However, if they are not maintained properly, System Reliability diminishes rapidly. *How do we establish a System Maintenance approach that will ensure maintenance is performed properly*? The answer begins with the Maintainability Concept.

34.4.2 The Importance of the Maintenance Concept

The cornerstone for planning System Maintainability resides in the SYSTEM's Maintenance Concept (Table 6.1) expressed in the SYSTEM's Concept of Operations (ConOps) document. The ConOps provides an initial starting point for documenting the Maintenance Concept unless there is a compelling need for a stand-alone document such as a required deliverable. Remember, the ConOps documents the key strategy that defines the *who*, *what*, *when*, *where*, and *how* a system, product, or service will be maintained. For example, maintenance of complex systems may be performed by: (1) the User, (2) an internal support organization, (3) external contractors, or (4) a combination of these.

Subsequently, a System Operations, Maintenance, and Support (OM&S) Plan or Integrated Support Plan (ISP) may be required or prepared to document the implementation of the ConOps Maintenance Concept. The details are determined by our next topic, Organizational Maintenance Perspectives.

34.4.3 Organizational Maintenance Perspectives

Organizationally, *System Maintainability* often receives token 'lip service' and usually falls second in priority to *System Reliability.* Naively and erroneously, the managerial paradigm is that if a SYSTEM's reliability is 'good enough' to meet SPS requirements, Maintainability Engineering is *minimized.* This is a comparable analogy to Bernard's (2008, p. 5) erroneous reasoning presented in the Introduction to the chapter.

In general, System OM&S costs for military systems account for approximately 60% - 70% (DAU, 1997, p. 13–6) of a SYSTEM or ENTITY's TCO, especially for complex systems. Since SOFTWARE is an integral item in systems and products today, Post Deployment Software Support (PDSS) Costs and their contributory factors should be a *major* concern *early* beginning at contract award or task initiation. The primary SYSTEM Quality Factors (Table 20.1) that contribute to operational support costs include:

- *System Reliability*—Development costs aimed at achieving mission objectives and reducing the frequency of maintenance.
- *System Maintainability*—Costs associated with the amount of time required to perform *preventive* and *corrective maintenance*.
- *System Availability*—An outcome determined by the SYSTEM or ENTITY's R&M.
- *System Sustainability*—Costs required to sustain system, product, or service operations without disruption.

Once a SYSTEM is accepted by the System Acquirer and fielded, the User lives with the consequences of System Development decision-making and its *accountability* or the lack *thereof* for the R&M factors. Additionally, these considerations drive another factor, *System Availability*.

System Availability represents the level of *operational readiness* of a system, product, or service to perform its mission *on demand* for a given set of operating conditions. When a SYSTEM or ENTITY is *unavailable* in commercial Enterprise environments, it: (1) *is not* generating revenue and (2) even worse, is costing the Enterprise for repairs thereby impacting bottom-line *profitability*.

Engineering textbooks often approach R&M with *equationitis*. Equationitis, like *Analysis Paralysis*, is a human condition created by a preoccupation with equations *without* fully understanding the operational challenges—*problem space*—Users have to address the base set assumptions to be established that lead to the need for equations.

Maintainability is a classic topical example. To illustrate *why* we need to understand User challenges, let's briefly explore an ENABLING SYSTEM scenario.

A system, product, or service is either: (1) operating—performing a mission or supporting training, (2) in storage, (3) awaiting maintenance, (4) being maintained, or (5) awaiting return to *active service*. Every hour spent in *maintenance* equates to lost revenue as in the case of commercial systems such as production machinery, airlines, and so forth. To ensure that the Enterprise System can sustain schedules, extra systems—spares—may be procured, leased, or rented to maintain *continuity of operations* while a SYSTEM or ENTITY is being maintained.

When a SYSTEM or ENTITY failure occurs, you need:

- A supply of spare parts *on-site* along with skilled maintenance technicians who can perform the repair with the least amount of downtime for maintenance.
- A Sustainment Supply Chain (Figure 4.1) to ensure spare PARTS in inventory is adequate.

Therefore, you need some insight concerning: (1) the quantity of spare parts required for a given type of failure, (2) the number of maintenance technicians required "full time" and "part time," (3) the number of maintenance workstations, if applicable, (4) the types and quantities of test equipment, (5) SYSTEM, PRODUCT, and PARTS storage space allocations, (6) procurement ordering systems, (7) logistical support systems. This challenge is compounded by seasonal usage factors for some systems. All of this translates into the TCO. To *minimize* the cost of maintaining an inventory of spare parts, many Enterprises establish strategic partnership agreements or subcontract with tier-level suppliers to provide parts Just-in-Time (JIT), when required. The automotive industry is a classic example and model.

Given this challenge, *how do we minimize the cost of maintenance of the evolving and maturing System Design Solution*? We can:

1. Incorporate a Self-Test or Built-in-Test (BIT) or BIT Equipment (BITE) capability into major items to *detect, isolate,* and *contain* Line Replaceable Unit (LRU) failures (Principle 26.16).
2. Perform *corrective* maintenance on *mission critical* and *safety critical* items during *preventive maintenance* and *corrective maintenance* cycles *before* they fail.
3. Designate LRUs and design the System Design Solution to facilitate LRU Removal and Replacement (R&R).
4. Provide easy access to, quick removal of, and installation of LRUs, and so forth.
5. Incorporate Reliability-Centered Maintenance (RCM) and Condition-Based Maintenance (CBM) capabilities introduced later that *continuously* monitor and assess the SYSTEM's Operational Health & Status (OH&S):
   a. Inform the User operator or maintainer of a SYSTEM or ENTITY 's current condition maintenance actions between missions.
   b. Communicate the need for *corrective maintenance actions* and spare parts prior to and on return from a mission.

Enterprises sometimes view maintenance tasks as nothing more than robotic, *reactionary* corrective actions. Consider the following example:

Impact of Maintenance Culture of Some Enterprises on System Performance

Example 34.16 Maintenance paradigm: *Just follow the maintenance manual. You don't even have to think about it. Remove two bolts, remove and replace (R&R) the defective component, reinstall the two bolts, and you are finished. If it still doesn't work, we'll replace the installed component with another one as many times as we need to until it works.*

Example 34.16 ignores the fact that maybe taking a few extra minutes to ensure the newly installed item had ancillary parts such as washers, assembled and aligned in the right sequence, bolts torqued, or simply was verified under operating conditions (Figure 24.2). All of these factors affect SYSTEM or ENTITY performance and early failure sometimes with chain reaction failure modes and effects as characterized by Figure 24.1.

Maintenance tasks, however, are more than *reactionary* corrective actions. DoD 4151.22-M (2011) makes two key points about defining maintenance tasks:

- "A maintenance task is considered effective when it reduces the risk of failure to an acceptable level.
- The consequences of failure must be used to determine task effectiveness" (DoD 4151.22-M, 2011, p. 9).

Given this overview, let's begin our discussion of System Maintainability beginning with the Levels of Maintenance.

34.4.3.1 Levels of Maintenance

Levels of Maintenance Principle

Principle 34.31 Establish levels of maintenance at geographical locations commensurate with the size of the User community, complexity of the repair, response times, and cost effectiveness.

Enterprises stage where preventive and corrective actions are performed. For example, the US DoD employs a two-level maintenance model (DAU, 2012, p. B-131):

- Depot Level Maintenance (Highest Level)
- Field Level Maintenance
  - Organizational Level Maintenance
  - Intermediate Level Maintenance (Lowest Level)

Let's explore the scope of each of these levels of maintenance.

34.4.3.1.1 Field- or Organizational-Level Maintenance *Field- or Organizational-Level Maintenance* consists of *preventive* and *corrective* maintenance actions performed in the field by a User or their support organization. At a *minimum*, these actions entail Removal and Replacement (R&R) of a failed component at the end of its *Useful Service Life* with a *new, repaired, remanufactured,* or *refurbished* item. In some instances, the component may be repaired at this level with Common Support Equipment (CSE) tools (Chapter 8). The Enterprise performing the Field or Organizational Maintenance may be internal to the User's Enterprise or contractor support referred to as Contract Logistics Support (CLS).

34.4.3.1.2 Intermediate Maintenance *Intermediate Maintenance* consists of corrective maintenance actions that require *removal and repair* or *replacement* of a failed SYSTEM or ENTITY by the User or CLS personnel and return to a central or regional repair facility. The facility may have specialized Peculiar Support Equipment (PSE) (Chapter 8) to accomplish the maintenance action. When the *repaired* item is returned to the field for *active service*, the User or CLS personnel perform Field or Organizational Maintenance to reinstall, check out, and verify the appropriate level of operation.

34.4.3.1.3 Depot or Factory Maintenance *Factory maintenance* actions are those performed by a depot, Original Equipment Manufacturer (OEM) or System Developer. Failed items may be escalated by the User or CLS: (1) via Field Level Maintenance - Organizational or Intermediate or (2) directly to a depot or OEM that has the specialized PSE (Chapter 8) and expertise required to determine the *cause of failure*. This may include site visits back to the User to interview personnel, review records, and inspect other items that may provide insights into the circumstances and operating conditions leading to the *contributory, probable,* or *root cause* (Figure 24.1) for the required maintenance action.

34.4.3.1.4 Failure Reporting and Corrective Action System (FRACAS)

FRACAS Principle

Principle 34.32 Institute a Failure Reporting and Corrective Action System (FRACAS) to track corrective maintenance actions, and data for analysis, defect elimination, and performance improvements.

When a SYSTEM or ENTITY is developed, R&M Engineers employ multi-discipline SE designs, Engineering Bills of Materials (EBOMs), or vendor/manufacturer data sheets to construct R&M mathematical models to estimate a SYSTEM'S or ENTITY's reliability. These models, which are based on the SE Process Physical Domain (Design) Solution (Figure 14.1), provide insights related to the *premature* wear, thermal stress, adequacy of periodic inspections, and performance metrics tracking.

FRACAS data records provide early indications of potential *wear-out* (Figure 34.7 Panel B) that may lead to *premature* failure. In turn, *mitigating* or *compensating* actions are made to improve the System Design or select alternative components to achieve SPS or EDS requirements compliance.

Once the SYSTEM is fielded, *actual* field data should be used to further *refine* and *validate* the M&S *physical* models. Therefore, *how can these data be obtained and tracked*? The solution is a FRACAS. MIL-HDBK-470A and the FAA SEM (2006, pp. 4.8-30/31) define a FRACAS as:

- **FRACAS** (MIL-HDBK-470A) "A closed-loop data reporting system for the purpose of systematically recording, analyzing, and resolving equipment reliability AND maintainability problems and failures" MIL-HDBK-470A (pp. 4–58).
- **FRACAS** (FAA SEM, 2006) "Tracking, analyzing, and correcting problems are key activities of an RMA program. The scope of this activity should be based on system complexity and maturity, environmental constraints, testing regimen, definition of reportable failures, and organizational roles within the FRACAS … " (FAA SEM, 2006, pp. 4.8-30-31).

Depending on Enterprise plans for a SYSTEM or ENTITY, the FRACAS provides an archival history of maintenance *actions* for each component and *replacements* with new components. Field data can enable the Enterprise to confidently and cost effectively extend the *useful service life* and employment of the asset (Figure 34.10).

Understand the Context of FRACAS Data

Author's Note 34.8 When you review FRACAS data, recognize that you need to understand the *context of any failure*. Depending on the discipline and rigor instilled in maintenance personnel, *errors* such as ordering and installing the *incorrect* part may sometimes be recorded as a failure, which initiates yet another maintenance action. Refer to Example 34.16 Maintenance Culture in Some Enterprises.

Additionally, a FRACAS may not include maintenance time tracking.

34.4.4 Maintainability Anthropometrics and Biomechanics

Textbooks often approach Maintainability in terms of equation-based computations. However, Human *anthropometrics, biomechanics* addressed in Chapter 24 are a major contributor to *successful* maintainability outcomes and performance metrics.

SYSTEMS and SUBSYSTEMS must be designed not only for the Mission System Users User but also for the Enabling Systems Maintenance PERSONNEL that also have to operate and perform maintenance. The ability of PERSONNEL – maintainers -to easily reach, twist, turn, open/close, calibrate/align, adjust, components via access doors, portals, and spaces to R&R ENTITIES—SUBSYSTEMS, ASSEMBLIES, SUBASSEMBLIES, and PARTS - is critical for success. The effectiveness in performing maintenance tasks impacts maintenance time and subsequently labor costs, etc.

Maintainability Estimates Principle

Principle 34.33 Maintainability estimates should be documented and supported by *assumptions* and/or actual performance measurement *data* to substantiate the results; otherwise the estimate is nothing more than personal opinion.

Recognize that the computation of System Maintainability equations assume that a baseline set of human *anthropometrics* and *biomechanics* (Figure 24.8) of the Maintenance

PERSONNEL have been established as a *precondition*. In fact, most Maintainability metrics are simply generic numbers; the knowledge, skill sets, tools, or EQUIPMENT that may be available are assumed based on *experiential* field data derived from actual task performance. This is in sharp contrast to Boehm's algorithmic Constructive Cost Model (COCOMO) Model (Boehm, 1981) for estimating Software Development effort, which factors in these considerations. Therefore, Maintainability estimates should explicitly document all *assumptions* and/or performance data measurements of these items with the estimate.

Recognizing that all maintenance PERSONNEL do not perform at the same level of proficiency, training programs are established to provide consistency and uniformity in knowledge and performance.

Commonly Used System Maintainability Terms and Equations

Author's Note 34.9 System Maintainability is characterized by as many terms as there are Enterprises. Every business domain has its own versions and nuances. Our discussions will introduce a few of the most *commonly used* terms. You are encouraged to pursue additional personal study concerning terms unique to your industry.

34.4.5 System Uptime (UT) and Downtime (DT)

From a *failure* perspective, a system, product, or service is either in a state of (1) performing missions, (2) *operational readiness* to conduct its missions (Chapter 7), (3) *degraded performance,* or (4) *inoperable*. In this context, Users have a *need-to-know* if the SYSTEM is "Up," meaning operational, or "Down" for maintenance." UpTime and and DownTime are terms frequently used, especially for Enterprises deploying SYSTEMS to perform missions such as computer centers, web sites, and transportation vehicles.

To better understand the meaning of *Uptime* versus *Downtime*, let's define the terms:

34.4.5.1 Mean Up Time (MUT) or $\overline{\text{UT}}$ If a system, product, or service is fully operational with no maintenance issues, it is considered "Up." The *elapsed* time since its last maintenance action is referred to as Uptime, which is defined as:

- **Uptime (UT)** "That element of Active Time during which an item is in condition to perform its required functions … " (MIL-HDBK-470A, p. G-17).

Figure 34.22 serves as an illustration of Uptime and Downtime. *Uptime*, as an abstract label, does not infer

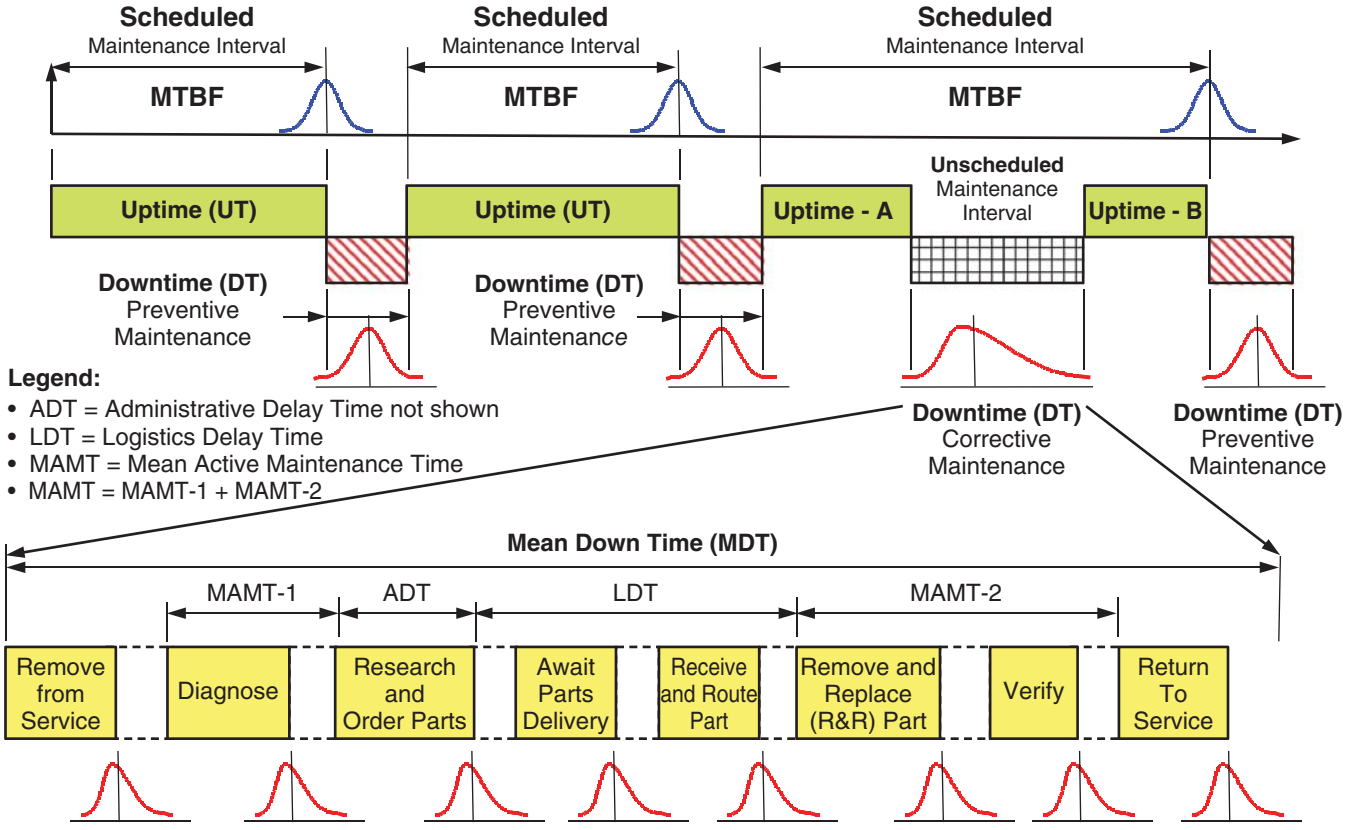


Figure 34.22 Composite Overview Representing System Maintenance Cycles

mathematically or *statistically* the age a SYSTEM or ENTITY. A more appropriate metric is Mean Uptime (MUT) or Mean Down Time (MDT), our next topics.

34.4.5.2 MDT or $\overline{\text{DT}}$ When a SYSTEM or ENTITY fails or requires maintenance, Users want to know: *how long the SYSTEM will be out of service—Downtime*? In the commercial world, System Downtime means an interruption to the revenue stream. In the military and medical fields, *DownTime* can mean the difference in life or death situations. MIL-HDBK-470A, for example, offers several definitions concerning *DT*, *System Downtime,* and *MDT* as follows:

- **Downtime (DT)** "That element of time during which an item is in an operational inventory but is not in condition to perform its required function" (MIL-HDBK-470A, p. G-4).
- **System Downtime** "The time interval between the commencement of work on a system (product) malfunction and the time when the system has been repaired and/or checked by the maintenance person, and no further maintenance activity is executed" (MIL-HDBK-470A, p. G-16).
- **Mean Downtime (MDT)** "The average time a system is *unavailable* for use due to a failure. Time includes the actual repair time plus all delay time associated with a repair person arriving with the appropriate replacement parts" (MIL-HDBK-470A, p. G-10).

The concept of MDT is appropriate for overview purposes; however, MDT is too abstract and needs further clarification to help us account for the activities to be performed. Any time a SYSTEM or ENTITY requires maintenance, they often refer to its condition as "down for maintenance" irrespective of whether maintenance is *actively* being performed or not. As a result, we need to clarify what occurs during maintenance to enable us to project when a SYSTEM or ENTITY will be returned to active service in a state of *operational readiness* to conduct missions.

MDT is comprised of three time-based activities: (1) Mean Active Maintenance Time (MAMT), (2) Mean Administrative Delay Time, (MADT or $\overline{\text{ADT}}$), and (3) Mean Logistic Delay Time (MLDT or $\overline{\text{LDT}}$). Mathematically, we express MDT ($\overline{\text{DT}}$) as follows:

$$\text{MDT}\ (\overline{\text{DT}}) = \overline{\text{MAMT}} + \overline{\text{LDT}} + \overline{\text{ADT}} \tag{34.42}$$

where:

- MDT = Mean Down Time
- MAMT = Mean Active Maintenance Time (Eq. 34.43)
- MADT = Mean Administrative Delay Time
- MLDT = Mean Logistics Delay Time

A Word of Caution 34.10

Mean values for $\overline{\text{ADT}}$ and $\overline{\text{LDT}}$ are fine for answering *general* questions concerning how much time will be required to order and receive parts for a repair. Some commonly used parts may be in stock locally or at a distributor; others may only be available from the Original Equipment Manufacturer (OEM), which could take considerably longer.

Such is the case if the delivery of various parts were represented by the Log-Normal Distribution shown in Figure 34.3 Panel B. Recognize the differences and employ the *mean* or *discrete* values for $\overline{\text{LDT}}$ or LDT as well as $\overline{\text{ADT}}$ or ADT, whichever is appropriate for your project. Hypothetically, if $\overline{\text{ADT}}$ = 1.6 hours for ordering PARTS as a *mean* and ADT = 2.1 hours for ordering a specific PART, does it make a difference? It depends on the application and order frequency; time is money!

Observe that in Eq. 34.42, the User or System Acquirer: (1) controls the $\overline{\text{M}}$ via the SPS and (2) has Enterprise *accountability* for $\overline{\text{ADT}}$ and $\overline{\text{LDT}}$. The System Developer has no control—*zero*—over the User's own Maintenance and Sustainment Systems. We will address this topic further in System Availability concerning the formation of a Logistics Support Working Group (LSWG).

Now, let's shift our focus to defining and scoping each of MDT's three elements—MAMT, LDT, and ADT.

34.4.5.3 MAMT ($\overline{\text{M}}$) As the multi-level System Design Solution evolves and matures, SEs need to know: *what is the mean time to accomplish a preventive or a corrective maintenance action*? The solution resides in an metric referred to as the MAMT ($\overline{\text{M}}$).

Mathematically, MAMT is computed as follows:

$$\text{MAMT} = \frac{f_{\text{PMT}}(\overline{\text{M}}_{\text{PMT}}) + \lambda(\overline{\text{M}}_{\text{CMT}})}{f_{\text{PM}} + \lambda} \tag{34.43}$$

where:

- f_{PM} = Frequency of PM
- λ = Failure density or frequency of corrective maintenance (Eq. 34.47)
- $\overline{\text{M}}_{\text{PMT}}$ =Mean Preventive Maintenance Time (Eq. 34.46)
- $\overline{\text{M}}_{\text{CMT}}$ = Mean Corrective Maintenance Time (Eq. 34.48)

MIL-HDBK-470A, for example, defines another term, *Active Corrective Maintenance Time (ACMT)* as:

- **ACMT** "That part of active maintenance time during which actions of corrective maintenance are performed

on an item. Excluded are logistics and administrative delays (e.g., awaiting parts, shift change, etc.)" (MIL-HDBK-470A, p. G-1).

34.4.5.3.1 Logistics Delay Time (LDT) LDT is the *actual* time required to perform actions that primarily support corrective maintenance but is applicable to PM if spare parts inventory becomes depleted. LDT examples include: (1) research for replacement or equivalent parts, (2) procurement and tracking of parts orders, (3) delivery of parts from the supplier, (4) receiving inspection of parts, (5) routing to a maintenance technician, and (6) reordering of incorrect parts.

The DAU (2012) refers to Mean Logistics Delay Time, MLDT ($\overline{\text{LDT}}$) and defines it as:

- **MLDT** "Indicator of the average time a system is awaiting maintenance and generally includes time for locating parts and tools; locating, setting up, or calibrating test equipment; dispatching personnel; reviewing technical manuals; complying with supply procedures; and awaiting transportation. The MLDT is largely dependent upon the (User's) Logistics Support (LS) structure and environment." (DAU, 2012, p. B-139).

34.4.5.3.2 Administrative Delay Time (ADT) ADT is the *actual* time spent on activities such as staffing the Maintenance organization; training personnel; vacation, holiday, and sick leave; and so forth. ADT is sometimes expressed in terms of Mean Administrative Delay Time, MADT ($\overline{\text{ADT}}$). For example, MIL-HDBK-470A defines *Administrative Time* as:

- **Administrative Time** "That element of delay time, not included in the supply delay time." (MIL-HDBK-470A, p. G-1).

34.4.5.3.3 UT Ratio Since systems are used to perform missions that are impacted if the System is Down, Users are very interested in what is referred to as the *Uptime Ratio*. MIL-HDBK-470A, p. G-17 defines Uptime Ratio as:

- **UT Ratio** "A composite measure of operational availability and dependability that includes the combined effects of item design, installation, quality, environment, operation, maintenance, repair, and logistics support: The quotient of uptime divided by the sum of uptime and downtime.)"

This definition enables us to mathematically establish the UT Ratio as:

$$\text{UT Ratio} = \frac{\text{UT}}{\text{DT} + \text{UT}} \tag{34.44}$$

Again, UT and DT are simply labels for discrete data. A more appropriate representation of Uptime Ratio is to express Eq. 34.44 in terms of their *mean* values. Here, the two terms are more indicative of their mathematical and statistical distributions.

$$\text{UT Ratio} = \frac{\text{Mean UT}(\overline{\text{UT}})}{\text{Mean DT}\ (\overline{\text{DT}}) + \text{Mean UT}\ (\overline{\text{UT}})} \tag{34.45}$$

Now that we have established the definitions and computations for MDT, let's expand our discussion to Types of Maintenance Actions.

34.4.6 Types of Maintenance Actions

Principle 34.34

Types of Maintenance Principle

Maintenance consists of two types of actions: *preventive* and *corrective maintenance*.

Let's explore each of these topics in more detail.

34.4.6.1 Preventive Maintenance (PM)

Principle 34.35

Preventive Maintenance (PM) Principle

Preventive Maintenance (PM) consists of scheduled maintenance actions at regular intervals to *Inspect*, *replenish*, and *replace* (all italics) expendables and consumables such as lubricants, filters, and components to restore a System or Entity to perform as specified and return to active service.

PM represents to *periodic* or *scheduled* maintenance actions performed on a System, Product, Subsystem, etc., in accordance with the System Developer's or vendors instructions. Note the operative term "scheduled." The intent is to initiate *proactive* maintenance actions in risk areas *before* they start to have an impact and result in degraded System or Entity performance (Figure 34.25) or failure.

What is *PM* or *scheduled maintenance*? MIL-HDBK-470A defines it as:

- **Preventive Maintenance (PM)** All actions performed in an attempt to retain an item in specified condition by providing systematic inspection, detection, and prevention of incipient failures"(MIL-HDBK-470A, p. G-14).
- **Scheduled Maintenance** "Periodic prescribed inspection and/or servicing of products or items accomplished on a calendar, mileage or hours of operation basis … "(MIL-HDBK-470A, p. G-15).

Please note that *PM* is the standard term used. *Periodic* or *scheduled maintenance* are an alternative terms used by some Enterprises; however, it is not unique to PM. Scheduled maintenance *applies to both* Preventive and Corrective Maintenance, especially in cases where failures have not occurred and upgrades are *scheduled* in advance.

Automobile Preventive (Scheduled) Maintenance

Example 34.17 Automobile manufacturers recommend that owners change oil at operating intervals that do not exceed 3000 miles, 5000 miles, or Y months of use under specific Operating Environment conditions. Check the vehicle's owner's manual for specifics.

34.4.6.1.1 Types of PM Actions PM encompasses three types of actions: Periodic, Predictive, and Planned.

- **Periodic Maintenance:**
  a. Is performed on a predefined, short-term schedule based on elapsed operating time—hours, calendar time—weeks or months; or operating cycles such as landings and takeoffs.
  b. Generally consists of inspecting and performing short operating interval servicing of EQUIPMENT; replacing *expendable* products such as filters and tires; replacing/replenishing *expendable* fluids such as coolants and lubricants; calibrating and aligning; and performing any prescribed OH&S checks and follow-on diagnostic tests as necessary.
- **Predictive Maintenance** Employs concepts such as Prognostics and Health Management (PHM) addressed later to continuously:
  a. Measure the current operational condition of EQUIPMENT such as fluids, lubricants, metal conditions, mileage, elapsed hours.
  b. Predict capability performance based on the current condition of the SYSTEM or ENTITY in sufficient time to schedule preventive or corrective maintenance actions.
- **Planned Maintenance** consists of:
  a. Scheduled maintenance performed at longer operating intervals coincidental with the next Periodic Maintenance interval.
  b. Major inspections and corrective maintenance such as upgrades, retrofits, overhauls, refurbishment, or replacement of major items or LRUs, assuming they have not failed prematurely.

34.4.6.1.2 Preventive Maintenance Time (PMT) SEs often have to answer: *How much time is required to perform PM actions for a* SYSTEM *or* ENTITY*?* The solution resides in a Maintenance metric referred to a *PMT*. PMT represents the *actual* time required to perform routine maintenance such as inspections and replacement of fluids, filters, belts, and so forth.

The PMT for a specific type of maintenance action is represented by $\overline{M}_{PMT}$, which represents the *mean time* for all PM actions for the item— SYSTEM or ENTITY, is expressed as:

$$\overline{M}_{PMT} = \frac{\sum_{i=1}^{n}(f_{PMT(i)})(M_{PMT(i)})}{\sum f_{PMT(i)}} \tag{34.46}$$

where:

- $M_{PMT(i)}$ = Time required to perform the *i*th *PM* action.
- i = Sequential identifier for a specific type of *PM* action.
- $f_{PMT(i)}$ = Frequency of the *i*th *PM* action.

34.4.6.2 Corrective Maintenance

Corrective Maintenance Principle

Principle 34.36 *Corrective maintenance* consists of scheduled or unscheduled maintenance actions required to *restore* or *retrofit* a system, product, or service to a specified condition and return it to *active service*.

Corrective maintenance is referred to as *unscheduled maintenance*. It consists of those maintenance actions required to *restore* the SYSTEM or ENTITY performance back to the manufacturer's performance specifications after failure. In general, corrective maintenance includes maintenance actions such as Removal and Replacement (R&R) of LRUs, replenishment of fluids, recalibration, realignment, and refurbishment. *What is corrective maintenance*? MIL-STD-470A and MIL-STD-3034 state the following definitions:

- **Corrective Maintenance** "All actions performed as a result of failure, to restore an item to a specified condition. Corrective maintenance can include any or all of the following steps: Localization, Isolation, Disassembly, Interchange, Reassembly, Alignment, and Checkout" (MIL-HDBK-470A, p. G-3).
- **Corrective Maintenance** "Maintenance task performed to identify, isolate, and rectify a fault so that the failed equipment, machine, or system can be restored to an operational condition within the tolerances or limits established for in-service operations" (MIL-STD-3034, 2011, p. 4).

Corrective maintenance actions consist of both *scheduled* and *unscheduled* maintenance actions after a failure event or condition to make corrective action. For *repair*able items, corrective actions includes making *repairs* to an ENTITY to

restore it to satisfactory compliance with its specification or in the case of an Emergency (Figure 19.5) to a *failsafe* condition. Therefore, *what constitutes a repair*? The DAU (2012) defines a *repair* as follows:

- **Repair** "The restoration or replacement of parts or components of real property or equipment as necessitated by wear and tear, damage, failure of parts, or the like in order to maintain it in efficient operating condition" (DAU, 2012, p. B-190).

Former MIL-STD-480B makes a very important point concerning perceptions of repairs. "The purpose of *repair* is to reduce the effect of the nonconformance. Repair is distinguished from *rework* in that the characteristic after repair still does not completely conform to the applicable specifications, drawings, or contract requirements" (MIL-STD-480B, p. 13).

Since time is a critical factor in restoring a system to active service, a new term, *Repair Time*, emerges. *What is Repair Time*? MIL-HDBK-470A defines *Repair Time* as follows:

- **Repair Time** "The time spent replacing, repairing, or adjusting all items suspected to have been the cause of the malfunction, except those subsequently shown by interim test of the system not to have been the cause" MIL-HDBK-470A (p. G-15).

Repair Time is dependent on having the competency - knowledge, experience, and expertise; tools; and SYSTEM resources to be able to isolate and determine the *root, contributory,* or *probable cause* of a failure (Chapter 24). A *failure* is a condition that materialized from a *fault* such as a latent defect, weak component, workmanship, or component problem. The challenge for SEs and R&M Engineers is being able to innovate and develop *cost-effective* and *robust* System Design Solutions that can:

1. Detect potential *fault* conditions.
2. Quickly diagnose and isolate *faults* before they become problems.
3. Contain those *faults* when they occur to prevent impacts to other components (Principle 26.16).
4. Tolerate faults—robust, *fault* tolerant System Architecture design (Principle 26.17)—to complete the mission and achieve success.

Each of the four preceding points has implications in developing the System Design Solution via the System Maintainability metrics discussed above.

Given an understanding of *PM*, let's shift our focus to understanding its counterpart, *corrective maintenance,* which is initiated by failures. This brings us to a common linkage parameter between System Reliability and System Maintainability, the Failure Rate, $\lambda(t)$.

34.4.6.2.1 Failure Rate, λ(t), or Corrective Maintenance Frequency The Corrective Maintenance Frequency is a function of the SYSTEM or ENTITY's Failure Rate, $\lambda(t)$, which is expressed as:

$$\lambda(t) = \frac{1}{\text{MTTF}} \tag{34.47}$$

where:

- MTTF = Mean Time to Failure for *repairable* and *non-repairable* items.

34.4.6.2.2 Mean Time to Repair (MTTR) Users often need to know more details than provided by MMT. This leads to a new metric, Mean Time to Repair (MTTR). MIL-HDBK-470A defines MTTR as follows:

- **MTTR** "A basic measure of maintainability. The sum of corrective maintenance times at any specific level of repair, divided by the total number of failures within an item repaired at that level, during a particular interval under stated conditions" MIL-HDBK-470A (p. G-11).

One of the fallacies of the MTTR metric is that it focuses on the Time to Repair, not Return to Service (RTS) (Figure 34.22 – lower right). The FAA-HDBK-006A (2008) fills this void with a Mean Time to Restore Service (MTTRS) metric.

- **Mean Time to Restore Service (MTTRS)** "represents the time needed to manually restore service following an unscheduled service failure requiring manual intervention" (FAA-HDBK-006A, 2008, p. 13).

FAA-HDBK-006A (2008) also notes that the MTTRS:

- "… includes only *unscheduled* downtime and assumes an ideal support environment.
- "… includes not only the time for hardware replacements, but also times for software reloading and system restart times.
- *Does not include* the times for the successful operation of automatic fault detection and recovery mechanisms that may be part of the system design." (FAA-HDBK-006A, 2008, p. 13).

34.4.6.2.3 Mean Corrective Maintenance Time (MCMT) or $\overline{M}_{CMT}$ The Mean Corrective Maintenance Time (MCMT) or ($\overline{M}_{CMT}$), which is equivalent to MTTR, represents the *actual* time required to perform a repair action on a SYSTEM or ENTITY to *replace* a failed or damaged LRU. MIL-HDBK-470A, for example, uses the term *Mean Active Corrective Maintenance Time (MACMT)* and defines it as follows:

- MACMT "The average time associated with active corrective maintenance actions. Time includes only actual repair time associated with a repair person performing corrective maintenance steps (i.e., Localization, Isolation, Disassembly, Interchange, Reassembly, Alignment, and Checkout)" (MIL-HDBK-470A, p. G-10).

$\overline{M}_{CMT}$ represents a *weighted arithmetic mean* of repair times for *all* corrective maintenance actions within the SYSTEM or ENTITY. Mathematically, we express $\overline{M}_{CMT}$ as follows:

$$\overline{M}_{CMT} = \frac{\sum_{i=1}^{n}(\lambda_{(i)})(M_{CMT(i)})}{\sum\lambda_{(i)}} \tag{34.48}$$

where:

- $M_{CMT(i)}$ = Time required to perform the *i*th *corrective maintenance* action.
- i = Sequential identifier for a specific type of *corrective maintenance* action.
- $\lambda_{(i)}$ = Frequency of the *i*th *corrective maintenance* action.

Depending on the Lifetime Data of the SYSTEM OR ENTITY, $\overline{M}_{CMT}$ might be characterized by an Exponential, Gaussian (Normal), or Log-Normal distribution. Examples of $\overline{M}_{CMT}$ distributions include:

- **Exponential $\overline{M}_{CMT}$ Distribution**—Maintenance tasks performed on a SYSTEM or ENTITY that exhibits a constant failure (Hazard) Rate Figure 34.5).
- **Gaussian (Normal) $\overline{M}_{CMT}$ Distribution**—Maintenance tasks that exhibit a standard repair time, such as changing lubricants, filters, or lights in a vehicle.
- **Log-Normal $\overline{M}_{CMT}$ Distribution**—Maintenance tasks performed for a diverse range of SYSTEM or ENTITY corrective actions that require fewer minutes to several hours to complete.

A plot of the frequency distribution of corrective maintenance actions is often assumed to follow a Log-Normal Distribution as illustrated in Figure 34.3 Panel B. That is, a few of the failures can be isolated and repaired rather quickly. Most of the corrective actions will require longer time periods and approach the mean. The remainder of the corrective actions may be fewer in number but require a considerable number of hours to correct.

34.4.6.2.4 Repair Rate, μ(t) Some Enterprises track the Repair Rate, $\mu(t)$, which is computed as the reciprocal of the MCMT ($\overline{M}_{CMT}$).

$$\mu(t) = \frac{1}{\overline{M}_{CMT}} \tag{34.49}$$

34.4.6.2.5 Mean Time Between Maintenance (MTBM) MTBM is the mean operating interval since the last maintenance—*preventive* or *corrective*—action. MIL-HDBK-470A defines MTBM as follows:

- **Mean Time Between Maintenance (MTBM)** "A measure of reliability that represents the average time between *all* maintenance actions, both corrective and preventive" (DAU, 2012, p. B-139.).

MTBM is expressed in terms of *elapsed* operating hours for a specific set of operating conditions for the SYSTEM or ENTITY based on the mean of all *preventive* ($\overline{M}_{PMT}$) and *corrective* ($\overline{M}_{CMT}$) maintenance actions. Mathematically, MTBM is expressed as:

$$MTBM = \frac{1}{\frac{1}{\overline{M}_{PMT}} + \frac{1}{\overline{M}_{CMT}}} = \frac{1}{f_{PMT} + \frac{1}{\lambda(t)}} \tag{34.50}$$

where:

- $\overline{M}_{PMT}$ = Mean PM Time (Eq. 34.46)
- $\overline{M}_{CMT}$= Mean Corrective Maintenance Time (Eq. 34.48)
- $\lambda(t)$ = Frequency of failure or Failure Rate (Eq. 34.47)

34.4.7 Sources of Maintainability Data

The validity of maintenance computations is dependent on having *valid* data sets derived from actual designs and field data records. Potential sources of *maintainability data* include:

- Historical data from similar components.
- Component design and/or manufacturing data.
- Data recorded during SYSTEM or ENTITY demonstrations.
- Field maintenance repair reports or FRACAS data.

34.4.8 Provisioning

Our discussion to this point focused on System R&M. Despite all of the theory and equations, sustainment of SYSTEM or ENTITY missions ultimately is determined by the User's ability to keep it maintained. That includes development of sustainment strategies to ensure that parts are readily available to support preventive and corrective maintenance activities in a reasonable amount of time. The ultimate question to be answered is:

- On the basis of: (1) the estimated failure rate of a SYSTEM or ENTITY; (2) the *maximum acceptable* MTTR; (3) the *shelf life* of ENTITIES or PARTS, and (4) the *cost* of stocking commonly required ENTITIES or PARTS; *what are the optimal levels of* ENTITIES OR PARTS*, especially RCIs to maintain in a local inventory*?

This brings us to *Provisioning*. The DAU (2012, p. B-215) defines Provisioning as:

- **Provisioning** "The process of determining and acquiring the range and quantity (depth) of spares and repair parts, and support and test equipment required to operate and maintain an end item of material for an initial period of service. Usually refers to first outfitting of a ship, unit, or system."

For many contracts, System Developers are often asked to deliver a Level of Repair Analysis (LORA). The DAU (2012, p. B-125) defines LORA as:

- **LORA** "An analytical methodology used to assist in developing maintenance concepts and establishing the maintenance level at which components will be replaced, repaired, or discarded based on economic/noneconomic constraints and operational readiness requirements. Also known as an Optimum Repair Level Analysis (ORLA)."

Heading 34.3

At this juncture, we have addressed the fundamentals of R&M. The critical question is: *Will the system be operationally available to conduct its missions on-demand by the User*? Although a system, product, or service may be reliable and well maintained, will it be *unavailable* on demand when required to conduct missions. This leads us to our next topic, System Availability.

34.5 SYSTEM AVAILABILITY

Principle 34.37

Availability Principle

Availability is the probability that a system, product, or service will be available *on-demand* to perform a mission.

Availability focuses on a SYSTEM's state of *operational readiness* (Chapter 7) to perform missions "on-demand" *when* tasked or conduct an additional missions. We define *operational readiness* as consisting of:

1. Completion of *preventive* and *corrective* maintenance actions within a specified time frame.
2. The ability to reliably start and operate "on-demand".

The DoD *RAM Guide* (2006) and the FAA-HDBK-006A (2008) define *availability* as follows:

- **Availability** "A measure of the degree to which an item is in an operable state and can be committed at the start of a mission when the mission is called for at an unknown (random) point in time" (DoD *RAM Guide*, 2006 p. 1–1).
- **Availability** "The probability that a system or constituent piece may be operational during any randomly selected instant of time or, alternatively, the fraction of the total available operating time that the system or constituent piece is operational" (FAA- HDBK-006A, p. 11).

A SYSTEM or ENTITY's Availability is a function of its *R&M* established by its System Architecture and System Design Solution. Since Availability is a mathematical estimate based on the R&M of the SYSTEM OR ENTITY, It can drive the System Developer in a decision-making circle as they attempt to achieve a specific outcome while balancing R&M objectives. Consider the following two cases at opposite extremes:

- **Case #1**—SYSTEM A is a highly reliable system. During development, the System Acquirer had a choice between improving Reliability or improving Maintainability. The Acquirer chooses to focus design resources on Reliability. The rationale is: "*If the system is highly reliable, we don't need a lot of repairs, and therefore won't need a large maintenance organization.*" End result: When the SYSTEM does need maintenance, it requires weeks for corrective maintenance actions due to the delay time awaiting maintenance repair. Clearly, the highly reliable system is *unavailable* for missions for long periods of time when maintenance is required.
- **Case #2**—SYSTEM B has less reliability because the Acquirer had to make a choice similar to that for SYSTEM A. The Acquirer lessens the Reliability requirement and focuses on improving the Maintainability. The rationale is: "*We have an outstanding maintenance organization that can repair anything*." As a result, SYSTEM B *fails continuously* and is prone to numerous corrective maintenance actions, thereby making it *unavailable* to perform most missions.

The point of this discussion is: SE&D must include a practical Availability requirement that enables the achievement of an *optimal balance* between a SYSTEM's R&M and still meet mission and Enterprise objectives. Close collaboration between the System Acquirer and User is important during the

System Procurement Phase and with the System Developer after Contract Award throughout the System Development Phase (Figure 12.2).

34.5.1 Types of Availability

Availability Metrics Principle

Availability is characterized by three types of metrics:

Principle 34.38

1. Operational Availability (A_o)
2. Achieved Availability (A_a)
3. Inherent Availability (A_i)

Let's explore each of these concepts briefly.

34.5.2 Operational Availability (A_o)

Operational availability, which is symbolized by A_o, represents the probability that the an item—SYSTEM or ENTITY—will operate in accordance with its specified performance requirements and prescribed OPERATING ENVIRONMENT conditions when tasked to perform its mission. A_o includes maintenance delays such as ADT and LDT and other factors independent of its design.

Mathematically, a SYSTEM or ENTITY's Operational Availability, A_o, is expressed:

$$A_O = \frac{MTBM}{MTBM + MDT} \tag{34.51}$$

where:

- MDT = Mean Downtime (Eq. 34.42).
- MTBM = Mean Time Between Maintenance (*repairable items*) (Eq. 34.50)

Observe that the MDT parameter (1) includes both *preventive* (scheduled) and *corrective* (unscheduled) maintenance time and (2) comprises MAMT, MADT, and MLDT.

34.5.3 Achieved Availability (A_a)

System Developers typically have no control over the User's ENABLING SYSTEMS factors such as LDT and ADT. Therefore, *Achieved Availability* (A_a) represents the level of availability under the control of the System Developer as constrained by the SPS.

Mathematically, an item's Achieved Availability, A_a, is the ratio of the Mean Time Between Maintenance (MTBM) to the summation of MTBM and MAMT, (M):

$$A_a = \frac{MTBM}{MTBM + MAMT} \tag{34.52}$$

where:

- MAMT = Mean Active Maintenance Time (Eq. 34.43).
- MTBM = Mean Time Between Maintenance (*repairable items*) (Eq. 34.50).

Whereas Operational Availability, A_o includes MAMT (M), MADT (ADT), and MLDT (LDT), Achieved Availability, A_a *excludes* ADT and LDT.

34.5.4 Inherent Availability (A_i)

Finally, the third type of availability is *Inherent Availability*, A_i. MIL-HDBK-470A defines *inherent availability* as follows:

- **Inherent Availability** (MIL-HDBK-470A) "A measure of availability that includes only the effects of an item design and its application, and does not account for effects of the operational and support environment" (MIL-HDBK-470A, 1997, p. G-7).
- **Inherent Availability (A_i)** (FAA) "The maximum availability theoretically within the capabilities of the system or constituent piece" (FAA-HDBK-006A, p. 11).

Regarding the FAA-HDBK-006A definition, the FAA notes:

1. "Computations of this construct consider only hardware elements and they assume perfect failure coverage, an ideal support environment, and no software or power failures.
2. Scheduled downtime is *not included* in the Inherent Availability measure.
3. A_i is an inherent design characteristic of a system that is independent of how the system is actually operated and maintained in a real-world environment" (FAA-HDBK-006A, p. 11).

The FAA SEM (2006) makes two key points about *Inherent Availability.*

- First, "This (inherent) availability is based solely on the MTBF and MTTR characteristics of the system or constituent piece and the level of redundancy, if any, provided. For systems or constituent pieces employing redundant elements, perfect recovery is assumed. *Downtime* occurs *only if* multiple failures within a common time frame result in outages of the system or one or more of its pieces to the extent that the need for redundant resources exceeds the level of redundancy provided. *Inherent availability* represents the *maximum* availability that the system or constituent piece is theoretically capable of achieving … if *automatic recovery* (Figure 10.17) is 100 percent effective."

- Secondly, *inherent availability* "does not include the effects of scheduled downtime, shortages of spares, unavailability of service personnel, or poorly trained service personnel" (FAA SEM, 2006, p. 4.8–28).

Inherent Availability, A_i, is expressed mathematically as the ratio of the MTBF to the summation of MTBF and MCMT ($\overline{M}_{CMT}$). Mathematically, A_i is defined as follows:

$$A_i = \frac{\text{MTBF}}{\text{MTBF} + \overline{M}_{CMT}} \tag{34.53}$$

where:

- $\overline{M}_{CMT}$ = MCMT or MTTR (Eq. 34.48).
- MTBF = Mean Time Between Failure (Repairable Items).

Observe that *PM* actions, LDT, and ADT are excluded. Note the switch from MTBM used to compute A_o and A_a to MTBF for A_i. Figure 34.23 provides a graphical view of Inherent Availability (A_i) values as a function of MTBF and $\overline{M}_{CMT}$ or MTTR. For example:

- If a SYSTEM or ENTITY is required to achieve an A_i of 0.90 and the design is estimated to have an MTBF of 700 hours, the derived MTTR requirement is 77.8 hours.

Finally, performance of various System Operational, Inherent, and Achieved Availability trade-offs between MTBF, MTBM, MDT, and $\overline{M}_{CMT}$ can be facilitated by creating asymptotic System Availability curves. MIL-HDBK-470A (Figure 1, p. 2–3) provides an illustrative example.

34.6 OPTIMIZING RMA TRADE-OFFS

At this juncture, you should recognize the interdependencies between System R&M and their contributions to System Availability. The challenge is selecting the *optimal* values for RMA based on constraints imposed as SPS requirements by the User or System Acquirer. Let's investigate how SEs might approach the *optimal* selection process.

34.6.1 Create the Initial Reliability and Availability Starting Points

As an initial starting point, you must identify a realistic System Reliability number: (1) that is achievable and (2) expresses the *probability* of mission success required of the item. You also need to select a System Availability number based on the criticality of the SYSTEM to support Enterprise missions. Finally, you need: (1) a Maintenance Concept

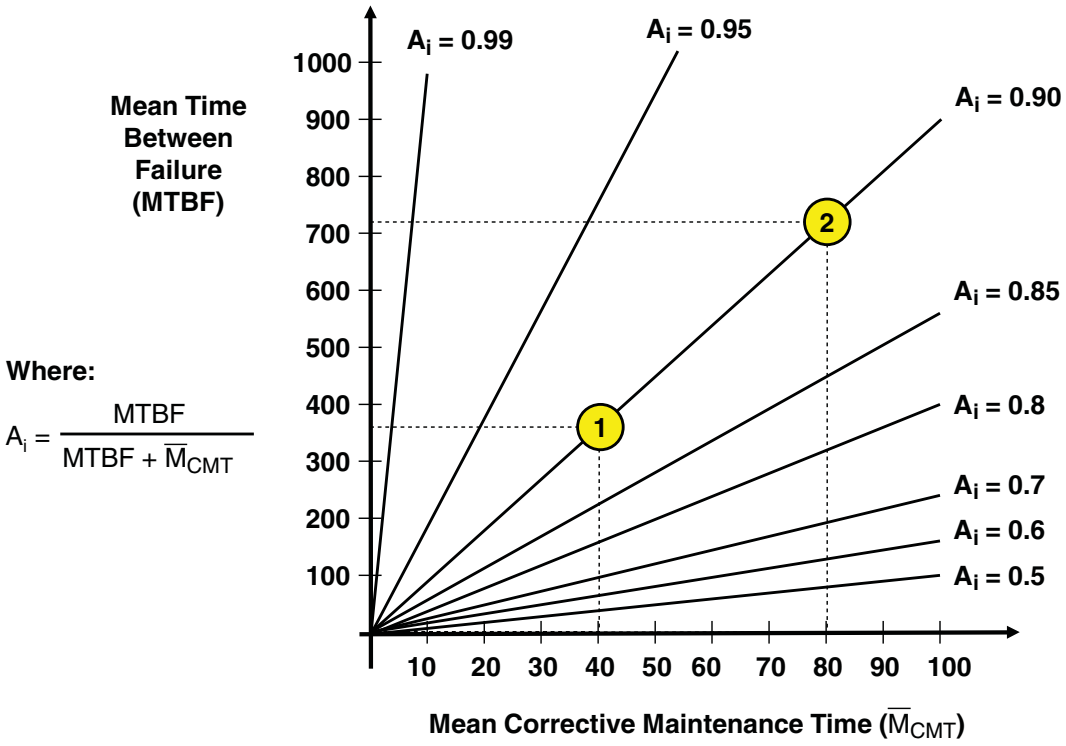


Figure 34.23 Inherent Availability (A_i) values as a Function of Mean Time Between Failure (MTBF) and Mean Time to Repair (MTTR)

that describes the philosophy of maintenance and (2) a System Maintainability metric.

34.6.2 Establish Development Objectives and Constraints

RMA Optimization Principle

Optimize RMA to achieve a System Design Solution that:

Principle 34.39

1. Complies with SPS requirements.
2. Minimizes the TCO.
3. Reduces *risk* to a level acceptable to the User.

One of the key User objectives is to *minimize* a SYSTEM or ENTITY's TCO. Inevitably, there are trade-offs at high levels between System Development costs and OM&S costs. Depending on the type of system, 60–70% (DAU, 1997, p. 13–6) of the *recurring* life cycle costs of many systems occur during the System OM&S Phase. Most of these costs are incurred by *corrective* and *PM* actions. Therefore, you may have to make trade-off decisions for every dollar you spend on System Design Reliability. *What is the cost avoidance in maintenance and support costs*? Let's explore this point further.

Using the illustrative example in Figure 34.24, let's increase the System Design Solution Reliability, which increases its System Development cost. We should then expect a responsive *decrease* in the System Maintenance costs. If we sum the *non-recurring* System Development cost and System Maintenance cost over the range of reliability, we emerge with a bowl-shaped curve representing total life cycle cost, or more appropriately the Total Cost of Ownership (TCO). Conceptually, there is a Target Reliability Objective, R_A, and a Target LC Cost Objective, TCO_A that represent the *minimum* TCO_A for the User. These objectives, in combination with the core RMA criterion of identifying the Reliability level, should form the basis for specifying RMA requirements.

Philosophically, the TCO *minimization* approach assumes that the User, by virtue of resources, has the flexibility to select the *optimal* level of Reliability. However, as is typically the case, the User has limited resources for how much Reliability they can afford "up front" within System Development costs. Hypothetically, depending on the operational need(s) and *level of urgency* to procure the SYSTEM, they may be confronted with selecting a more affordable reliability, R_B, based on System Development cost limitations that result in a higher-than-planned System Maintenance Cost ($MAINT_B$) and a higher TCO_B.

For innovative Enterprises, this presents both an *opportunity* and a *challenge*. For Enterprises that do not perform SE from the standpoint of an Analysis of Alternatives (AoA) solution and jump to a *point design solution*, (Figure 2.3) this

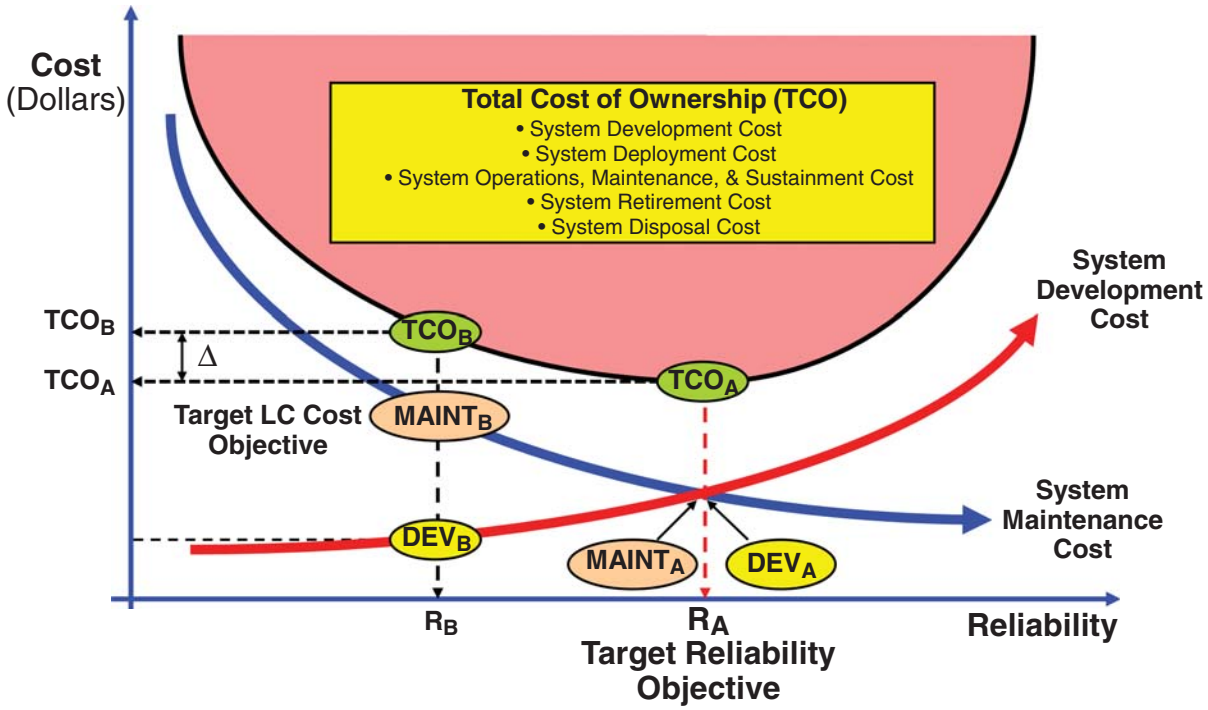


Figure 34.24 Achieving the Optimal, Cost-Effective Balance Between System Reliability and Maintainability for the Least Cost

may be a *missed* business opportunity. During the AoA activity, new approaches may be discovered that enable the User to *minimize* TCO within the constraints of System Development costs.

Author's Note 34.10

Selecting an Availability Number Target

For this first step, most Engineers and Users are reluctant to pick a *reasonable* number for fear of inferring they may not be able to achieve the performance requirements. From an Engineering perspective, you need a System Availability performance target as an *initial starting point* and that is all it is. Unfortunately, there are those who criticize these efforts without recognizing that it may not be realistic to achieve the "endgame" reliability number; it is only a starting point … nothing more, and nothing less.

Author's Note 34.11

Specifying Only an Availability Requirement

We began our discussion with Reliability followed by Maintainability and subsequently Availability, as a product or R&M. The objective of the sequencing of these topics was not to simply compute Availability. The point is that some System Acquirer RFP System Requirements Documents (SRDs) for a *reusable* system may only specify a performance requirement for *availability* and no performance requirements for R&M. The expectation is that the System Developer conducts an AoA to select an *optimal mix* of R&M values and costs with an aim of reducing the optimal TCO. This can be a difficult challenge, especially if you do not understand the User, their preferences, and priorities.

Now, contrast the preceding example of a *reusable* system with a *single-use* system such as a missile. *System Availability is a critical issue; System Reliability is equally or more critical*. Launching a missile is one aspect; however, in the end, it has to find, track, and destroy its target.

34.6.3 System Availability—Final Thoughts

In summary, we have briefly defined each of the three types of System Availability. From a contractual *SPS* perspective, Operational Availability, A_o obviously has a major significance to the User versus Achieved Availability, A_a and Inherent Availability, A_i. So, *how does a System Developer deal with the User's* ENABLING SYSTEM *LDT and ADT factors for which they have no control*?

- First, the System Acquirer should specify in the contract that the User by virtue of *ownership* has full *accountability* for supplying LDT and ADT data and/or assumptions.
- Second, the System Acquirer and System Developer should consider creating a Logistical Support Working Group (LSWG) that includes the User LDT and ADT Stakeholders with actions to supply the data and participate in making recommendations via the formal contract protocol.

Assuming the Logistics Support WG performs as expected, the User Stakeholders must assume *accountability* for their contributions to achieving the overall success of the SYSTEM and verification of the appropriate SPS requirements. If they: (1) specify an Operational Availability requirement that includes LDT and ADT (Eq. 34.42) for which they control and (2) require the System Developer to comply with the requirement, they should be *accountable* for providing that information or assumptions "up front" as part of the Request for Proposal (RFP) and contract data. Blanchard (1998, p. 121) observes that there is *limited value* in constraining the design of equipment to achieve a 30-minute $\overline{M}_{CT}$ if the ENABLING SYSTEM requires 3 months on average, consisting of LDT + ADT, to respond with parts. A_o is certainly not achieved in this manner.

Heading 34.4

The preceding discussions addressed the fundamentals of RMA. Given this foundation, we integrate these concepts into a discussion of the overarching theme of the chapter, RCM and CBM.

34.7 RELIABILITY-CENTERED MAINTENANCE (RCM)

Principle 34.40

Reliability-Centered Maintenance (RCM) Principle

Then objective of Reliability Centered Maintenance (RCM) is not to preserve the Equipment Element or to keep it from failing; it is to *avoid* or *reduce* the effects of failure consequences to ensure continuity of capabilities required to complete the mission without disruption.

Due to the expense of repairs and *unnecessary* PM, some complex systems embed Condition-Based Maintenance (CBM) capabilities into their systems. These systems *continuously monitor* and *analyze* the OH&S of engine lubricants, such as motor oil, to determine the level of metal particles and other conditions that may indicate premature wear or potential failure condition.

CBM systems are critical for systems such as aircraft engines applications and are part of an overall RCM activity by the User's Enterprise. *What is RCM*? MIL-HDBK-470A, the DoD *RMA Guide* (2006), NavAir (2013), and Weibull.com (2013) provide a diverse range of RCM definitions:

- **RCM** (MIL-STD-3034, 2011, p. 4) "A method for determining maintenance requirements based on the analysis of the likely functional failures of systems/equipment having a significant impact on safety, operations, and life cycle cost. RCM supports the failure-management strategy for any system based on its inherent reliability and operating context."
- **RCM** (MIL-HDBK-470A, p. G-15) "A disciplined logic or methodology used to identify preventive and corrective maintenance tasks to realize the inherent reliability of equipment at a minimum expenditure of resources"
- **RCM** (DoD *RMA Guide*, 2006, p. 4–49) "A logical, structured framework for determining the optimum mix of applicable and effective maintenance activities needed to sustain the desired level of operational reliability of systems and equipment while ensuring their safe and economical operation and support. RCM is focused on optimizing readiness, availability, and sustainment through effective and economical maintenance."
- **RCM** (NavAir, 2013, Slide 5) "An analytical process used to determine appropriate failure management strategies to ensure safe and cost-effective operations of a physical asset in a specific operating environment."
- **RCM** (Weibull.com, 2007) "… a structured framework for analyzing the functions and potential failures for a physical asset (such as an airplane, a manufacturing production line, etc.) with a focus on preserving system functions, rather than preserving equipment."

If you ask most people why we perform maintenance on EQUIPMENT, the typical response is "to ensure that the equipment does not breakdown." The reality is: EQUIPMENT does not exist for its own sake. It exists to provide a *capability* to enable a User to reliably perform missions and accomplish performance-based mission objectives. Bottom line: the objective of Maintenance is to ensure that EQUIPMENT capabilities support User objectives, *not* preserve the EQUIPMENT. *How is this accomplished*? Through a process referred to as RCM Analysis. *What is the purpose of RCM Analysis*?

NavAir (2013) makes two key points concerning the *Goal of RCM*:

1. "Avoid or reduce failure *CONSEQUENCES.*
2. Not necessarily to avoid failures." NavAir (2013, p. 5).

34.7.1 RCM Origins

JA1012 (2002, p. 1) notes that the origin of RCM traces to a report first documented by F. Stanley Nowlan and Howard F. Heap for United Airlines. The report was later published by the US DoD in 1978. Weibull.com (2007) adds that the report addressed the transformation of "aircraft maintenance as the Boeing 747 was being introduced."

Nowlan and Heap (DoD, 1978) state up front that:

- "A central problem addressed in this book is how to determine which types of scheduled maintenance tasks, if any, should be applied to an item and how frequently assigned tasks should be accomplished" Nowlan and Heap (DoD, 1978, p. ii).
- The net result is a structured, systematic blend of experience, judgment, and operational data/information to identify and analyze which type of maintenance is both applicable and effective for each significant item as it relates to a particular type of equipment" Nowlan and Heap (DoD, 1978, p. ii).

Over the years, various RCM processes and documents have been published and merged. SAE JA1012 (2002, p. 1) makes two key points concerning other RCM documents:

- "… key elements of the RCM process have been omitted or misinterpreted" essentially deviating from basic principles of Nowlan and Heap.
- "While most of these processes may achieve some of the goals of RCM, a few are actively counterproductive, and some are even dangerous."

So, for reference, *what were Nowlan and Heap's (1978) original precepts*?

"RCM is based on the following precepts (Author's emphasis):

- A failure is an unsatisfactory condition. There are two types of failures: *functional failures*, usually reported by operating crews and *potential failures*, usually discovered by maintenance crews.
- The consequences of a *functional failure* determine the priority of maintenance effort. These consequences fall into four categories:
  a. *Safety consequences*, which involve possible loss of the equipment and its occupants.
  b. *Operational consequences*, which involve an indirect economic loss as well as the direct cost of repair.
  c. *Nonoperational consequences*, which involve only the direct cost of repair.
  d. *Hidden-failure consequences*, which involve exposure to a possible multiple failure as a result of the undetected failure of a hidden function.
- Scheduled maintenance is required for any item whose loss of function or mode of failure could have *safety* consequences. If preventive tasks cannot reduce the risk of such failures to an acceptable level, the item must be redesigned to alter its failure consequences.

- Scheduled maintenance is required for any item whose *functional failure* will not be evident to the operating crew, and therefore reported for corrective action.
- In all other cases, the consequences of failure are economic, and maintenance tasks directed at preventing such failures must be justified on economic grounds.
- All failure consequences, including economic consequences, are established by the design characteristics of the equipment and can be altered only by basic changes in the design:
  a. Safety consequences can, in nearly all cases, be reduced to economic consequences by the use of redundancy.
  b. Hidden functions can usually be made evident by instrumentation or other design features.
  c. The feasibility and cost-effectiveness of scheduled maintenance depend on the inspectability of the item, and the cost of corrective maintenance depends on its failure modes and inherent reliability.
- The inherent reliability of the equipment is the level of reliability achieved with an effective maintenance program. This level is established by the design of each item and the manufacturing processes that produced it. Scheduled maintenance can ensure that the inherent reliability of each item is achieved, but no form of maintenance can yield a level of reliability beyond that inherent in the design." Nowlan and Heap (DoD, 1978, p. xvi–xviii).

RCM literature typically refers to *functional failures*. *What is a functional failure*? MIL-STD-3034 (2011) defines *functional failures* as follows:

- **Functional Failures** "A functional failure exists when a system or subsystem ceases to provide a required function; whether the function is active, passive, evident, or hidden." (MIL-STD-3034, 2011, p. 14).

34.7.2 Condition-Based Maintenance (CBM)

Central to RCM and FMEAs as well as Nowlan and Heap's RCM (1978) principles is the need to:

1. *Sense* the physical condition of Entities and Parts within a System or Entity.
2. *Detect* when maintenance is required.

As a result, CBM evolved. *What is CBM*?

NavAir (2013), for example, defines CBM as follows:

- **CBM** (NavAir) —"An equipment maintenance strategy based on measuring the condition of equipment in order to assess whether it will fail during some future period, and then taking appropriate action to avoid the consequences of that failure" (NavAir, 2013, Unit 1, Module 3, Slide 26).
- **Condition Monitoring** (NavAir)—"The use of specialized equipment to measure the condition of equipment" (NavAir, 2013, Unit 1, Module 3, Slide 25).

One of the challenges of traditional maintenance is to perform a prescribed set of maintenance actions on Equipment at prescribed intervals such as mileage, cycles, or actuations. The actions are performed mechanistically based on Lifetime Data Distribution models whether they are needed or not. The presumption is that all Equipment is subjected to the same Operating Environment conditions on a daily basis. Due to the expense of maintenance, especially in aircraft, heavy construction equipment, and plant machinery, some view those actions as *poorly timed*, *unnecessary*, and should be performed only when needed. This, in turn, leads to:

1. *How do you sense the current condition of a System or Entity?*
2. *Given knowledge of that condition, how do you estimate when Just-in-Time (JIT) maintenance will be required for scheduling purposes including the ordering of Parts?*

Answers to these questions require two complementary approaches: (1) traditional inspection and (2) advanced technology monitoring and test equipment. *Why two approaches*? Recall from our discussion in Chapter 24 some tasks are best performed by Personnel; in other cases Equipment performs best (Figure 24.14).

Sensing the Equipment's current condition can be accomplished with Built-In-Test (BIT) or BIT Equipment (BITE). BIT/BITE continuously monitors current conditions such as aircraft in flight. When the aircraft is on the ground, maintainers can perform visual inspections, take samples, and employ various types of sensing equipment to assess current conditions. On the basis of those conditions, the next step is to be able to *estimate* when maintenance will be required to be scheduled. Therefore, *what is the System Design strategy (Chapter 12) for developing a reliable System or Entity that can be easily maintained and failures mitigated at the right time*?

One method is to incorporate a Prognostics and Health Management (PHM) System that includes an embedded CBM System within the System or Entity in combination with a proactive Enabling System maintenance inspection and corrective action process. Assessment of that information requires a more advanced technology, *Predictive Maintenance (PdM),* or to a higher level *Prognosis and Health Management (PHM) System*. *What are prognostics*?

- **Prognostics** The science of analyzing past usage and trends, assessing the current state and condition of a

SYSTEM or ENTITY, and projecting its future condition to determine the *optimal* time for performing preventive and corrective maintenance actions.

Given this definition, the key question is: *where are prognostics implemented*? Core PHM knowledge originates from the Mission System. However, in terms of Enterprise effectiveness and efficiency in rapidly performing preventive or corrective maintenance requires:

1. Continuous Mission System Situational Awareness of its current condition and trending.
2. Enabling System Situational Awareness of the Mission System's current or projected condition to enable deployment of the right resources – Personnel, Equipment CSE and PSE (Chapter 8), Mission Resources - parts, Procedural Data – maintenance manuals, and Facilities – Just-in-Time (JIT) to perform maintenance actions when the System such as an aircraft arrives.

During missions, the Mission System via its Personnel and Equipment Elements Monitor, Command, & Control (MC2) their respective capabilities (Figures 24.12 and 26.6) to perform PHM and telemeter data directly with Enabling Systems, typically ground-based, in or near real-time. Historically, "black boxes" on aircraft, trains, medical pacemakers, and other types of systems may record the mission data. However, the raw data may not be reduced – summarized - or reviewable until the System arrives at its destination and is downloaded by the Enabling System User(s), which impacts Mission System turnaround performance.

Where revenue generation is paramount, the ENABLING SYSTEM must be able to perform preventive and corrective maintenance when the *Mission System*: (1) arrives at a maintenance facility or location or (2) make a service call to the MISSION SYSTEM at a remote location – heavy construction or farm equipment, especially if the MISSION SYSTEM is mobile. So, the MISSION SYSTEM PHM must communicate maintenance needs "ahead" to the ENABLING SYSTEM Just-in-Time (JIT). As a result, the PHM System requires the integration of both MISSION SYSTEM and ENABLING SYSTEM(s) (Figure 9.2).

New concept? Actually not. For example, military aircraft refueling systems, which consist of a MISSION SYSTEM aircraft requiring fuel and an ENABLING SYSTEM tanker aircraft illustrate the basic via preplanning and "radio ahead" communications. Another example includes medical pacemaker device patient event logs forwarded to a central hospital unit for analysis by a cardiologist and potential follow-up.

Please note that both of these examples have a potential disruption in time due to the need for the MISSION SYSTEM – aircraft or patient – to initiate communications with the ENABLING SYSTEM – tanker aircraft or cardiologist. The primary thrust of this section is direct *real-time* communications between the MISSION SYSTEM and the ENABLING SYSTEM. In the case of the aircraft refueling and patient pacemaker monitoring, in general, there is no need to communicate in real-time time, only when necessary.

Earlier in our Maintainability discussion, Eq. 34.42 illustrated how MDT is impacted by MAMT, LDT, and ADT. Time is money. Since MDT impacts the revenue stream, the ENABLING SYSTEM has to be geophysically positioned (Principle 34.31) with the "right" spare parts provisions to *eliminate* LDT and ADT and perform the JIT preventive or corrective maintenance actions immediately.

In the mix of various maintenance terms, NavAir (2013, Module 3, p. 25) notes that "Predictive Maintenance (PdM) and Prognosis and Health Management (PHM) are usually interchangeable with Condition Monitoring." DoD 4151.22-M (2011) employs the term On-Condition Task and defines it as follows:

- **On-Condition Task** "The purpose of an on-condition task is to identify when action is required based on the evidence of need. How often an on-condition task is performed depends on the potential to functional failure (P-F) interval." (DoD 4151.22-M, 2011, p. 9).

The context of this definition is not unique to EQUIPMENT capabilities; it applies to the ENABLING SYSTEM maintenance inspection and corrective action process as well. Observe the reference to the " … potential to functional failure (P-F) interval." So, w*hat is a P-F Interval*?

The P-F Interval refers to the condition-based trajectory referred to as the P-F Curve. Let's explore the topic to attain a better understanding of its relationship to CBM.

A Word of Caution 34.11

Despite being commonly used, the "potential to functional failure (P-F)" label is title mismatch for what is actually accomplished. As defined in Chapter 3, a *function* is a unitless *action to be performed*, not performance. More appropriately, a *capability* represents an action to be performed at a specific level of performance, the threshold definition of a failure. So, a SYSTEM or ENTITY has to do more than "perform a function." Therefore, the *caution* concerning the correctness of the P-F phrase!

An automobile brake pad with little remaining material performs a function, a *necessary* but *insufficient* condition for stopping a moving vehicle. The ability to safely stop the vehicle requires a *capability*, which is considered *necessary* and provides *sufficient* braking action with reserve to spare.

34.7.3 The P-F Curve Concept

The DoD 4151.22-M definition of On-Condition referenced the "potential to functional failure (P-F) interval," which is

often referred to as the P-F Curve. The P-F Curve, like the Bathtub Curve, is a concept applicable to EQUIPMENT.

The P-F Curve represents the deterioration of EQUIPMENT over time as a function of its OPERATING ENVIRONMENT stresses and maintenance or lack thereof. The reality is EQUIPMENT operational conditions are *conditional* and *diminish* over time unless it is maintained. Characterizing EQUIPMENT's operational condition as a *potential failure,* which is true, neglects the fact that the condition may be a symptom of a latent defect - *hazard* or *fault* - that may and will materialize over time *versus* stress-related degradation from normal "wear and tear."

From an SE perspective, what is important is to determine the appropriate maintenance schedule to *avoid* a SYSTEM or ENTITY *catastrophic failure* such as Reason's Accident Trajectory Model illustrated in Figure 24.1. Figure 34.25 provides an example profile of an EQUIPMENT Failure Condition Trajectory (Potential-Function) Curve.

When an ENTITY or PART is installed, it has an *inherent* operating condition, $Cond_1$, that is: (1) dependent on removal of *latent defects* via Design Verification and (2) workmanship, and component issues via Product Verification. Over time, the condition of mechanical PART Level components *deteriorate*. The onset of this condition, $Cond_2$, initiates what is referred to as the *physics of failure* and a chain of EQUIPMENT operational conditions begin to materialize. For example, here is a hypothetical Run-to-Failure (RTF) scenario of the chain of events illustrated in Figure 34.25 assuming no maintenance is performed.

- Event 3—Failure condition, $Cond_3$, becomes *detectable* via human inspections or supporting technologies. For example, ENTITIES and PARTS may begin to exhibit visible or audible evidence such as: cracks from fatigue; slight vibrations; lubricant viscosity, color, or odor changes; particulates such as metal fragments appear in lubricants; tires or brake pads become worn down; and leaks appear.
- Event 4—Operators report *continuous* or *intermittent* vibrations, $Cond_4$.
- Event 5—Metal fragments begin to appear as particulates in lubricants, $Cond_5$.
- Event 6—Non-specific audible noises become apparent, $Cond_6$.
- Event 7—PARTS exhibit excessive heat, $Cond_7$, detectable to the touch and via thermal imagery.
- Event 8—PARTS begin to loosen due to vibrations and lack of maintenance, $Cond_8$.
- Event 9—The SYSTEM or ENTITY experiences a *catastrophic or "hard" failure*, $Cond_9$

Events 1 – 9 illustrate a sampling of SYSTEMS OR ENTITIES that are allowed to RTF as shown in Figure 34.1. Using that information, an appropriate PM program can be developed to ensure failures are detected to ensure that System capabilities will be preserved throughout the mission. To aid human inspections, technologies such as acoustical sensors and vibration analytics, lubricant analysis, infrared thermography,

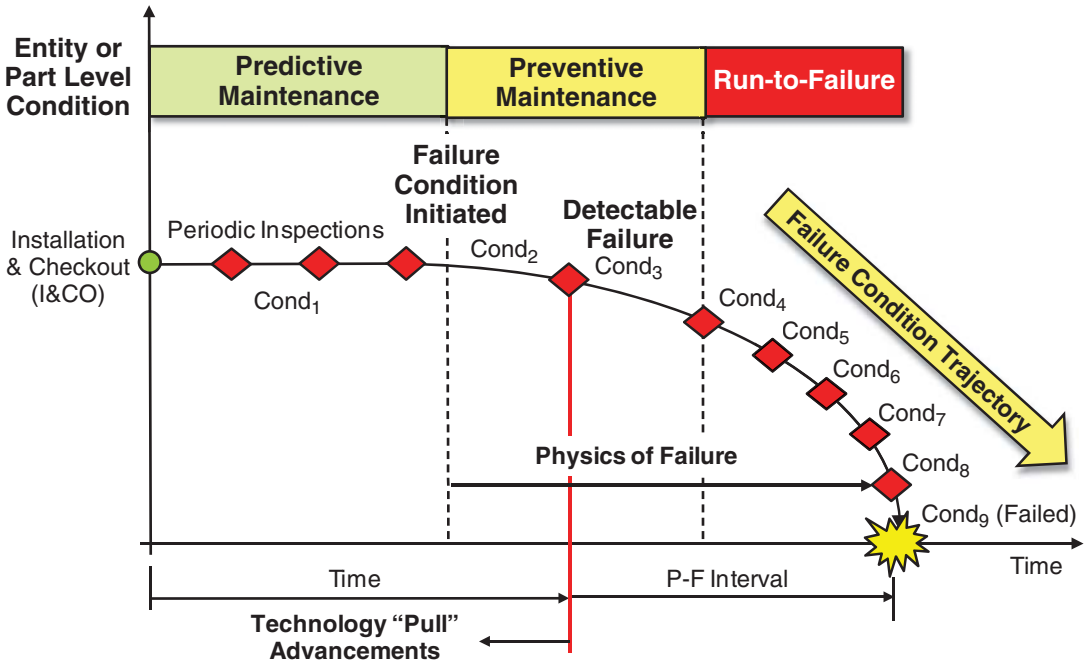


Figure 34.25 EQUIPMENT Failure Condition Trajectory Curve

X-ray radiography, and vibration analysis have been developed.

On the basis of today's technologies, *what analytical tools and devices can we employ to support PHM and CBM*? Examples include: Vibration Analysis, Acoustical Analysis—Ultrasonic Monitoring and Sonic Monitoring, Infrared Monitoring and Analysis, and Oil Analysis.[7]

34.8 SYSTEM RMA CHALLENGES

Implementation of System RMA practices poses a number of challenges. Let's explore some of the key challenges.

34.8.1 Challenge 1: Defining and Classifying Failures

Despite all the complexities of curve fitting and creating mathematical equations to model RMA, one of the most sensitive issues is *what constitutes a failure*. *Does it mean loss of a critical mission function*? Your task, as an SE, is to "maintain intellectual control of the problem-solution" (Principle 1.3). This requires developing a consensus definition of a failure that is shared by Acquirer and System Developer teams.

34.8.2 Challenge 2: Failure to Document Assumptions and Data Sources

The *validity* of Engineering data typically requires:

- Identifying *credible* data sources.
- Making assumptions about the MISSION SYSTEM or ENABLING SYSTEM User Stories, Use Cases (UCs), scenarios (Chapter 5), and OPERATING ENVIRONMENT conditions.
- Documenting those assumptions and observations.

Yet, few Enterprises instill *professional* discipline in their Engineers to document these decisions. Then, when the time comes to make critical *informed* decisions, the decision process is in a conundrum as to whether the R&M Engineer *did* or *did not* consider all relevant factors. Time exacerbates the problem. Therefore, educate and train Engineers in your Enterprise to document assumptions and data sources as part of an RMA analysis.

34.8.3 Challenge 3: Validating RMA Models

RMA analyses, assuming they are performed properly, are only as good as the models employed to generate data for the analyses. Referring to Principle 33.5, you should recall Box's (1979, p. 202) quote concerning the *validity* of models. From the beginning of a program, strive to validate the models used to generate decision-making data with actual vendor or field data.

[7]Refer to Wikipedia (2013b) for additional information concerning these tools.

34.8.4 Challenge 4: The Criticality of Scoping System Availability

Today, the government and private sectors employ contracting for services approach in which a contractor develops and provides systems and services on a *fee-per-service* basis. For example, contracts will often state that the EQUIPMENT and services must be *operationally available* from 8 a.m. to 5 p.m., Monday through Friday and at other times under specified conditions. The System Acquirer pays a fee for mission time for SYSTEM use within those time frames.

Author's Note 34.12

Understand the Contract

The challenge in developing the contract is to thoroughly understand all of the use cases and scenarios that may become *major showstoppers* for progress payments in performance of the contract and clarifying how all parties will accommodate them. Think about the implications of System Availability requirements *before* you write them into System Acquisition requirements of a proposal and contract.

34.8.5 Challenge 5: Quantifying Software RMAs

We live in the Information Age where seemingly all systems, especially complex systems, are software intensive. Despite HARDWARE RMA curve fitting and the mathematical equations discussed, *how do you measure and model the RMA of software*? Unlike HARDWARE components that can be tested in controlled laboratory conditions and maintained via repairs, *how do you model and estimate the RMA of* SOFTWARE?

This is perhaps one of the most perplexing areas of Engineering. *What about the RMA of software used in consumer products versus mission critical software for the International Space Station (ISS), NASA Space Shuttle, passenger aircraft, and medical equipment*? There are no easy or simple answers.

One solution may be to employ the services of Independent Verification and Validation (IV&V) (Chapter 13) contractors. IV&V contractors perform services to review and test software designs for identifying and eliminating design flaws, coding errors, and product defects and their services can be very costly, especially from a system development perspective. Depending on the legal and financial ramifications of SYSTEM or ENTITY abuse, misuse, or misapplication, the Return on Investment (ROI) for IV&V activities may be cost-effective. Consult with software SMEs

in your industry to gain insights into how to specify software RMA.

34.9 CHAPTER SUMMARY

Our discussion of RMA practices is intended to provide a basic understanding that will enable you to communicate with Reliability Engineers, logisticians, and others. Chapter 34's discussions are intended to introduce you the System RMA concepts, principles, and practices to enable you to interact with or employ RMA professionals, understand their computations and assumptions, and understand the types of questions to ask. Overall, there are several key points you should remember:

- RMA practices apply model-based mathematical and scientific principles to *estimate* reliability, availability, and maintainability to support SE design decision-making.
- RMA *estimates* are only as *valid* as the assumptions, inputs, and level of validation of models.
- RMA models require best-fit selection to provide an estimate of the probability related to SYSTEM or ENTITY success.
- RMA practices involve art, science, and sound judgment:
  a. *Art* from the standpoint of empirical knowledge, wisdom, and experience gleaned over an entire career.
  b. *Science* from the application of mathematical and scientific principles.
  c. *Sound judgment* to recognize and reconcile the differences between *art* and *science*.
- *Always* employ the services of a *qualified*, *competent*, R&M Engineer recognized as a SME, either as a staff member or as a credible consultant with integrity. Remember, RMA involves critical areas that involve ethical, legal, and safety issues and their associated ramifications.

34.10 CHAPTER EXERCISES

34.10.1 Level 1: Chapter Knowledge Exercises

1. What does the acronym RMA represent?
2. When should RMA be addressed on a project?
3. Why are RMA addressed as a set?
4. Who is accountable for RMA?
5. Differentiate System Reliability and Mission Reliability. What is the scope of each?
6. Do you *predict* or *estimate* a SYSTEM or ENTITY's Reliability?
7. When expressing a SYSTEM or Entity's Reliability, what *criteria* should be included? Where should those criteria be documented? Why?
8. Which Level of Abstraction—SYSTEM, PRODUCT, SUBSYSTEM, ASSEMBLY, SUBASSEMBLY, or PART—does System Reliability apply to?
9. If a SYSTEM or ENTITY's Failure Rate, $\lambda(t)$, and instant in time, $t(t)$, are known, how do you compute its Reliability?
10. What is the difference between a SYSTEM or ENTITY's Average Failure Rate, $\lambda(t)_{Avg}$, and Instantaneous Failure Rate, $\lambda(t)_{Inst}$?
11. If MTTF represents the Mean *Time* to Failure, what Greek letter is used to represent the *frequency* of MTTF ?
12. MTTF applies to what types of SYSTEMS or ENTITIES?
13. If MTBF represents the Mean *Time* Between Failure, what Greek letter is used to represent the *frequency* of MTBF?
14. MTBF applies to what types of SYSTEMS or ENTITIES?
15. What are Lifetime Data Distributions and how are they applied to RMA?
16. What are the seven Lifetime Data Functions?
17. What is a Normal (Gaussian) Distribution, what is its shape, and what are its characteristics?
18. What is a Log-Normal Distribution, what is its shape, and what are its characteristics?
19. What is an Exponential Data Distribution, what is its shape, and what are its characteristics?
20. What is a the Cumulative Distribution Function (CDF), what is its source of derivation, and what information does it communicate?
21. What is a Weibull Distribution, what is its shape, what are its key parameters, and what do the parameters control?
22. What Lifetime Data Distribution provides the most flexibility in modeling the failure characteristics of a SYSTEM or ENTITY?
23. What is the Failure Rate Function?
24. What is the Bathtub Curve? What are its origins? What is its shape? Define each of its regions and discuss their characteristics. Illustrate how the Bathtub Curve was composed?

25. What is the characteristic of the Failure Rate that is unique to the Exponential Distribution in the Bathtub Curve Stabilized Failure Regions (SFR)?

26. What is a Parts Count Estimate? What is its purpose and relevance?

27. Is the Bathtub Curve valid and does it apply to all SYSTEMS or PRODUCTS? If not, why?

28. What is the relevance and importance of RMA to System Architecting?

29. How are specification reliability requirements allocated from the SPS?

30. What types of network configurations are used to compute the reliability of a SYSTEM or ENTITY Architecture?

31. What is a FMEA, its purpose, and what outcomes does it expect to achieve?

32. What is System Maintainability?

33. What is the difference between Maintainability and Maintenance?

34. MTBF is composed and computed by the summation of two other metrics. What are they?

35. Within an SPS or EDS outline, where are the RMA requirements specified?

36. A specific System or Product is characterized by an Exponential Distribution PDF, $f(t)$. When it reaches its mean life point, what percentage of its population remain as survivors?

37. What parameters are used to characterize the Maintainability of a SYSTEM or ENTITY?

38. What is System Availability?

39. What are the three types of Availability, define what each includes/excludes, and develop a statement for each that expresses its computation?

40. How do you improve System Reliability?

41. How do you improve System Maintainability?

42. How do you improve System Availability?

43. Is the Bathtub Curve valid? Why? State the pros and cons of its validity.

44. As an SE, why should you be expected to understand the contents of this chapter?

34.10.2 Level 2: Chapter Knowledge Application Exercises

Refer to www.wiley.com/go/systemengineeringanalysis2e

34.11 REFERENCES

Bazovsky, Igor (1961), *Reliability Theory and Practice*, Englewood Cliffs, NJ: Prentice-Hall.

Barnard, R. W. A (2008), "What is Wrong with Reliability Engineering?", *Proceedings of the 18th Annual International Symposium of the International Council on Systems Engineering (INCOSE)*, Hatfield, South Africa: Lambda Consulting.

Billinton, Roy, and Allan, Ronald N. (1996), *Reliability Evaluation of Power Systems*, 2nd Edition, New York, NY: Springer.

Blanchard, Benjamin S. (1998), *System Engineering Management,* 2nd ed. New York: John Wiley.

Boehm, Barry W. (1981), *Software Engineering Economics*, Upper Saddle River, NJ: Prentice Hall PTR.

Box, George E. P. (1979), *Robustness is the Strategy of Scientific Model Building in Robustness in Statistics*, eds. R.L. Launer and G.N. Wilkinson, New York, NY: Academic Press.

Defense Acquisition University (DAU) (2012), *Glossary: Defense Acquisition Acronyms and Terms*, 15th ed, Ft Belvoir, VA: Defense Acquisition University (DAU) Press. July 2011. Accessed on 3/27/13 http://www.dau.mil/pubscats/PubsCats/Glossary%2014th%20edition%20July%202011.pdf.

Dhillon, B. S. (1999), *Engineering Maintainability: How to Design for Reliability and Easy Maintenance*, Houston, TX: Gulf Publishing Company.

DoD RMA Guide (2006), *Guide for Achieving Reliability, Availability, and Maintainability*, Washington, DC: Department of Defense (DoD). Retrieved on 11/18/13 from https://acc.dau.mil/adl/en-US/142103/file/27585/DoD-RAMGuide-April06%5B1%5D.pdf.

DoD 4151.22-M (2011), *Reliability Centered Maintenance (RCM) Manual*, Washington, DC: Department of Defense (DoD). Retrieved on 12/8/13 from http://www.dtic.mil/whs/directives/corres/pdf/415122m.pdf.

FAA SEM (2006), *National Air Space System - Systems Engineering Manual*, FAA Systems Engineering Council, Washington. DC: FAA, Retrieved on 3/11/13 from http://fast.faa.gov/SystemEngineering.cfm.

FAA-HDBK-006A (2008) *Reliability, Maintainability, and Availability (RAM) Handbook*, Washington, DC: Federal Aviation Administration (FAA).

Kaplan, Jeremy A. (2011), *45 Years Later, Does Moore's Law Still Hold True?*, New York: NY: Fox News, Inc. Retrieved on 12/7/13 from http://www.foxnews.com/tech/2011/01/04/years-later-does-moores-law-hold-true/.

Klutke, Georgia-Ann; Kiessler, Peter C.; and Wortman, M. A. (2003), "A Critical Look at the Bathtub Curve," *IEEE Transactions on Reliability*, Vol. 52, No. 1, New York, NY: IEEE.

Leitch, R. D. (1988), *BASIC Reliability Engineering Analysis*, London: Butterworth & Co, Ltd.

Machol, Robert E.; Tanner, Jr., Wilson P.; and Alexander, Samuel N. (1965), *System Engineering Handbook*, New York: McGraw-Hill.

MIL-HDBK-0217F (1991), *Reliability Prediction of Electronic Equipment*. Washington, DC: Department of Defense (DoD).

MIL-HDBK-338B (1998), *Electronic Reliability Handbook*, Washington, DC: Department of Defense (DoD).

MIL-HDBK-470A (1997), DoD Handbook: *Designing and Developing Maintainable Systems and Products,* Vol. I. Washington, DC: Department of Defense (DoD).

MIL-STD-480B (1988), Military Standard*: Configuration Control - Engineering Changes, Deviations, and Waivers,* Washington, DC: Department of Defense (DoD).

MIL-STD-882E (2012), *System Safety,* Washington DC: Department of Defense (DoD).

MIL-STD-1629A (1980), Military Standard *Procedures for Performing a Failure Mode, Effects, and Criticality Analysis (FMECA)*, Washington, DC: Department of Defense (DoD).

MIL-STD-3034 (2011), *Reliability-Centered Maintenance (RCM) Process*, DoD Standard Practice, Washington, DC: Department of Defense (DoD).

Moore, Gordon E. (1965), "Cramming More Components onto Integrated Circuits," *Electronics*, Vol. 38, No. 8, pp. 114–117, April 19, 1965, (City): Publisher.

NASA PD-EC-1101 (1995), *Best Reliability Practices - Environmental Factors*, Washington, DC: NASA. Retrieved on 5/30/14 from http://engineer.jpl.nasa.gov/practices/1101.pdf.

NavAir (2013), *Fundamentals of RCM Analysis Course*, Patuxent River, MD: Naval Air Systems Command. Retrieved on 12/4/13 from http://www.navair.navy.mil/logistics/rcm/.

NIST (2013) *Engineering Statistics Handbook*, Washington, DC: National Institute of Science and Technology (NIST). Retrieved on 11/23/13 from http://itl.nist.gov/div898/handbook/index.htm.

Nelson, Wayne (1990), *Accelerated Testing: Statistical Models, Test Plans, and Data Analyses*. New York: Wiley.

Nicholls, David (2007), "An Introduction to the RIAC 217PLUS™ Component Failure Rate Models," *The Journal of the Reliability Information Analysis Center*, 1st Qtr. 2007, Belcamp, MD: The Reliability Information Analysis Center (RIAC).

Nowlan, F. Stanley and Heap, Howard F. (1978), *Reliability-Centered Maintenance (RCM)*, DoD Report Number A066-579 Distribution of United Airlines Report, Washington, DC: Department of Defense (DoD). Retrieved on 12/8/13 from http://oai.dtic.mil/oai/oai?&verb=getRecord&metadataPrefix=html&identifier=ADA066579.

O'Connor, Patrick D. T. (1991), *Practical Reliability Engineering*, 3rd Edition, West Susses, UK: John Wiley, & Sons, Ltd.

RAC (2001), *Reliability Desk Reference*: "Redundancy Equations for Calculating Reliability" web page, Reliability Analysis Center (RAC) Rome, NY: Alion Science Corporation. Retrieved on 12/15/13 from http://src.alionscience.com/pdf/RedundancyEquations.pdf.

Radle, Byron, and Bradicich, Tom Eds. (2013), White Paper: *What is Reliability*? Austin, TV: National Instruments Corporation. Retrieved on 12/22/13 from http://www.ni.com/white-paper/14412/en/.

Reason, James (1990), *Human Error*, Cambridge, UK: Cambridge University Press.

ReliaSoft (2001), "*Limitations of the Exponential Distribution*" web page, Qtr 4 2001, Vol. 2 - Issue 3, Tucson, AZ: ReliaSoft Corporation. Retrieved on 12/4/13 from http://www.reliasoft.com/newsletter/4q2001/exponential.htm.

Reliawiki.org (2012), *Basic Statistics Background* web page, Tucson, AZ: ReliaSoft, Inc. Retrieved on 12/10/13 from http://reliawiki.org/index.php/Basic_Statistical_Background.

Reliawiki.org (2013a), *Chapter 8: The Weibull Distribution* web page, Tucson, AZ: ReliaSoft, Inc. Retrieved on 12/10/13 from http://reliawiki.org/index.php/The_Weibull_Distribution

Reliawiki.org (2013b), *Life Distributions* web page, Tucson, AZ: ReliaSoft, Inc. Retrieved on 12/10/13 from http://reliawiki.org/index.php/Life_Distributions.

Reliawiki.org (2013c), *The Normal Distribution* web page, Tucson, AZ: ReliaSoft, Inc. Retrieved on 12/10/13 from http://reliawiki.org/index.php/The_Normal_Distribution.

RIAC (2004), *Reliability Desk Reference: "Derating" web page*, Rome, NY: Alion Science Corporation.

SAE JA 1012 (2002), *A Guide to the Reliability-Centered Maintenance (RCM) Standard*, Warrendale, PA: Society of Automotive Engineers (SAE) (International), Inc.

Smith, Anthony M. (1992), *Reliability-Centered Maintenance,* New York: McGraw-Hill.

Speaks, Scott (2005), *Reliability and MTBF Overview*, Andover, MA: Vicor Corporation. Retrieved on 12/12/13 from http://www.vicorpower.com/documents/quality/Rel_MTBF.pdf.

TM 5-698-4 (2006), *Technical Manual: Failure Modes & Effects Criticality Analysis (FMECA) for Command, Control, Communications, Computer, Intelligence, Surveillance, and Reconnaissance (C4ISR) Facilities*, Washington, DC: HQ Department of the Army. Retrieved on 12/8/13 from http://armypubs.army.mil/eng/DR_pubs/DR_a/pdf/tm5_698_4.pdf.

Weibull.com (2004), "*Specifications and Product Failure Definitions*" web page, *Reliability Hotwire*, Issue 35 January 2004, Tucson, AZ: ReliaSoft. Retrieved on 11/19/13 from http://www.weibull.com/hotwire/issue35/relbasics35.htm#footnote1.

Weibull.com (2007), "*The RCM Perspective on Maintenance*" web page, *Reliability Hotwire*, Issue 71, January 2007, Tucson, AZ: ReliaSoft Corporation. Retrieved on 12/4/13 from http://www.weibull.com/hotwire/issue71/hottopics71.htm.

Weibull.com (2013), *Life Data Analysis (Weibull Analysis): An Overview of Basic Concepts* web page, Tucson, AZ: ReliaSoft Corporation. Retrieved on 11/19/13 from http://www.weibull.com/basics/lifedata.htm.

Weibull.com (2014), *Life Data Analysis (Weibull Analysis) Visual Demonstration of the Effect of Parameters on the Distribution*, Tucson, AZ: ReliaSoft Corporation. Retrieved on 5/30/14 from http://www.weibull.com/basics/parameters.htm.

Wikipedia (2013a), *Ernst Hjalmar Waloddi Weibull*, San Francisco, CA: Wikipedia. Retrieved on 12/4/13 from http://en.wikipedia.org/wiki/Waloddi_Weibull.

Wikipedia (2013b), *Predictive Maintenance*, San Francisco, CA: Wikipedia. Retrieved on 12/9/13 from http://en.wikipedia.org/wiki/Predictive_maintenance.

Wilkins, Dennis J. (2002), "*The Bathtub Curve and Product Failure Behavior Part Two - Normal Life and Wear-Out*," *Reliability Hotwire*, Issue 22, December 2002, Tucson, AZ: ReliaSoft Corporation. Retrieved on 11/19/13 from http://www.weibull.com/hotwire/issue22/hottopics22.htm.

EPILOG

Chapters 1–34 presented the key *concepts*, *principles*, and *practices* of *System Engineering Analysis, Design, and Development*. Our purpose has been to help *bridge the gap* between a User's *abstract vision* for a system, product, or service and its physical realization that will satisfy their Enterprise mission or personal needs and objectives. Equipped with this new knowledge of systems, you should be prepared to embark on applying *what you have learned* without having to take a *quantum leap* to *bridge the gap* (Figure 2.3).

Chapter 1 introduced the definition of a *System* repeated below.

> **System** An integrated set of interoperable elements or entities, each with specified and bounded capabilities, configured in various combinations that enable specific behaviors to emerge for Command & Control (C2) by Users to achieve performance-based mission outcomes in a prescribed operating environment with a probability of success.

Reread it frequently. When you read other generic definitions such as … *a system is a collection of things* … hopefully Chapters 1–34 provide a better understanding of *why* the definition requires the *robustness* of its wording.

The same is true with the definition of *Systems Engineering*:

> **System Engineering (SE)** The multi-disciplined application of analytical, mathematical, and scientific principles for formulating, selecting, and developing an optimal solution from a set of viable candidates that has acceptable risk, satisfies User operational need(s), and minimizes development and life-cycle costs while balancing Stakeholder interests.

Recognize that 21st Century System Engineering & Development (SE&D) encompasses more than the traditional, "Engineering the Box" mindset (Chapter 1). As an SE, if your perception of SE is *plugging* and *chugging* equations, you need to *reexamine* the traditional Engineering view of your role as an SE as well as your values and priorities in achieving mission success! Learn to *recognize* and *exercise caution* when Engineers and Enterprises claim to be performing SE when, in fact, it is nothing more than the *ad hoc, endless loop* Specify-Design-Build-Test-Fix (SDBTF) Engineering Paradigm with Archer's (1965) Design Process Model (DPM) embedded.

Chapter 2 addressed The Evolving State of SE Practice – Challenges and Opportunities from the author's Organizational Development (OD) perspective. During that discussion, we highlighted some of the perceptions, misperceptions, paradigms, and issues that limit achievement of Enterprise and project level application of Systems Engineering (SE). Based on Chapters 1–34, you should have acquired new levels of awareness about these issues and gained new insights that enable you to shift to a new *paradigm* for *thinking* about, *analyzing, organizing*, and *orchestrating* the development of systems, products, and services. Periodically reread Chapter 2 and use it as a frame of reference for improving Enterprise SE capabilities and project performance.

System Engineering Analysis, Design, and Development: Concepts, Principles, and Practices, Second Edition. Charles S. Wasson.

Companion Website: www.wiley.com/go/systemengineeringanalysis2e

Chapters 3–34 highlighted key multi-discipline topics and methodologies required for the "Engineering Systems." As a structured *problem solving/solution development* methodology, these *concepts, principles,* and *practices* enable us to develop systems, products, and services ranging from *simple* to *highly complex*. The scalability and flexibility of the methodology facilitates application to any type of system, product, or service regardless of business domain. These systems may range from institutional/Enterprise systems such as a transportation, energy, Aerospace and Defense (A&D), financial, education, medical, healthcare, and military to the Engineering of Enterprise Systems and their Engineered systems, products, or services."

During the writing of the First Edition, the author searched to find a highly successful project that exemplified all of the challenges of multi-discipline Systems Engineering. At that time, NASA's Jet Propulsion Laboratory (JPL) had successfully completed the Mars Pathfinder Project several years earlier in 1997. JPL's business model was confronted with the following challenges:

1. Limited windows of opportunity to launch every 26 months; you don't just decide to go to Mars when design teams finish the job.
2. Innovating and creating new technologies in a very short development window of time.
3. Innovating, designing, and developing systems in a short development cycle followed by Command and Control (C2) of the missions. Whereas Enterprises typically develop and deliver systems when completed, JPL has to C2 their own missions and live with the consequences of their SE system development decisions.
4. Bottom Line: Funding driven by accomplishments and successes.

Bottom line: JPL is a success-driven decision-making Enterprise with missions that involve major technical and technological challenges.

In 2001, I met with Brian Muirhead, Mars Pathfinder Project Manager, and members of his team to learn more about the factors that contributed to the project's success. Three key points emerged from our discussions:

- The team consisted of multi-discipline Engineers from around the JPL facility who were "*motivated* by the *excitement* of working on a mission that required major technological challenges".
- Members of the team were *Systems Thinkers* that understood the importance of proactive, interdisciplinary collaboration and communications.
- The team understood the *importance* and *criticality* of System Verification and Validation (V&V) through design risk mitigation testing and a robust validation test program the "working system" such as the flight vehicle, ground mission control, and communications.

During these intervening years, Brian and his team member comments have *resonated* and been *validated* by the author's own independent observations and experiences across business domains.

Brian is now the Chief Engineer at JPL and still actively engaged in the Engineering of their missions. The concepts and ideas that Brian and his team pioneered on the Mars Pathfinder Project in 1997 serve as a frame of reference for today's JPL successes.

In preparing this Second Edition, it seemed worthwhile to revisit the Pathfinder Team's comments and follow-up with Brian to discuss how his views of SE may have *evolved* over these years. During our conversation, he responded with the following points:

1. Start thinking about the test program "up front" at the beginning of a project, especially in developing specifications and how you will demonstrate and prove the "functionality and interfaces." Every requirement should be challenged leaving only *valid, essential* requirements and a plan for System Verification and Validation (V&V).
2. "Government and industry have become bogged down with processes almost excessively. We've replaced (System) thinking with process"—Paint-By-Number Engineering (Chapter 2).
3. The Mars Pathfinder Project Team made a decision "up front" to reserve 50% of the development schedule for designing the System and the other 50% for testing and validating the *working system*. Design risk mitigation testing, especially for new technologies, was an integral part of the design portion. The Team met the 50% design/50% test schedule goals.
4. (Unlike the SDBTF-DPM Engineering Paradigm that lacks a focused strategy) The Pathfinder Team conducted a significant amount of well-planned "build-test-break-learn" exercises and outcome-based corrective action activities that included *validation* of the analytical models with actual test data. At that time (1990's) some analytical Engineering tools such as a Cray computer system could not provide the answers they needed; *validation testing* was the only logical alternative.
5. System Verification has its place in terms of proving *compliance* to specification requirements. However, System Validation is where you get to find out if the system provides the *behaviors* and *interactions*

required by the "functionality and interface" requirements. You should recall from Chapter 2 that Warwick and Norris (2010) noted Dr. Michael Griffin's observations about "… understanding the dynamic behavior of those interactions."

In summary, Chapters 1–34 provided the SE concepts, principles, and practices required to perform SE&D. The next steps are up to you, your project, and Enterprise. The question you and your Enterprise need to answer is:

How do we preserve the integrity of the SE concepts, principles, and practices addressed in Chapter 1–34 while confronted with aggressive schedules, limited budgets, and challenging technologies?

That requires experience, sound engineering judgment and a *proactive willingness to perform* – tailor and scale SE practices - to achieve the required performance-based outcomes.

With these points in mind, I extend to you best wishes for *success* as an SE, System Analyst, Engineer, Project Manager (PM), functional manager, or executive applying *System Engineering Analysis, Design, and Development: Concepts, Principles, and Practices* to achieve System Engineering and Development (SE&D) excellence!

Charles S. Wasson

Wasson Strategics, LLC
www.wassonstrategics.com
August 2015

Appendix A

ACRONYMS AND ABBREVIATIONS

A	
A_o	Operational Availability
A_i	Inherent Availability
A_a	Achieved Availability
A&D	Aerospace & Defense
ABD	Architecture Block Diagram
ABET	Accreditation Board of Engineering & Technology
ABS	Anti-lock Braking System
ACA	After Contract Award
ACO	Acquirer Contracting Officer
AD	Architecture Description
ADT	Administrative Delay Time
$\overline{\text{ADT}}$	Mean Administrative Delay Time (MADT)
AFE	Acquirer Furnished Equipment
AFSCM (US)	Air Force Systems Command
AFP	Acquirer Furnished Property
AMSL	Above Mean Sea Level (See MSL)
ANSI	American National Standards Institute
AoA	Analysis of Alternatives
AR	As Required
ASEP (INCOSE)	Acquisition Systems Engineering Professional
ATC	Air Traffic Control
ATE	Automated Test Equipment
ATP	Acceptance Test Procedure
ATR	Acquirer's Test Representative
B	
BCD	Baseline Concept Description
BES	British Engineering System
BioMed	BioMedical Engineering
BIT	Built-In Test
BITE	Built-In Test Equipment
BOM	Bill of Materials (See EBOM)
C	
CA	Contract Award
CAD	Computer-Aided Design
CAIV	Cost as an Independent Variable
C2	Command and Control
C4I	Command, Control, Communications, & Computers Integration
CBM	Condition-Based Maintenance
CCB	Configuration Control Board
CCM	Counter-Counter Measures
CDD	Capabilities Development Document
CDF	Cumulative Distribution Function
CDR	Critical Design Review
CDRL	Contract Data Requirements List
CEP	Circular Error Probability

System Engineering Analysis, Design, and Development: Concepts, Principles, and Practices, Second Edition. Charles S. Wasson.

Companion Website: www.wiley.com/go/systemengineeringanalysis2e

CFD Computational Fluid Dynamics
CG Center of Gravity
CofC Certificate of Compliance
CoM Center of Mass
ChemE Chemical Engineering
CHI Computer-Human Interface
CI Configuration Item
CLIN Contract Line Item
CLS Contract Logistics Support
CM Configuration Management
CM (US Apollo) Command Module
CMM Capability Maturity Model
CMMI Capability Maturity Model Integration
CMMI-ACQ CMMI for Acquisition
CMMI-DEV CMMI for Development
CMMI-SVC CMMI for Services
CMP Configuration Management Plan
CMT Corrective Maintenance Time
COI Critical Operational Issue
COMSEC Communications Security
ConOps Concept of Operations (Document)
COS Conditions of Satisfaction
COTS Commercial-Off-the-Shelf
CPAT (DoD) Critical Process Assessment Tool
CPFF Cost Plus Fixed Fee (Contract)
CPIF Cost Plus Incentive Fee (Contract)
CR Change Request
CSA Configuration Status Accounting
CSC Computer Software Component
CSU Component Software Unit
CSCI Computer Software Configuration Item
CSE Common Support Equipment
CSEP (INCOSE) Certified Systems Engineering Professional
CSOW Contract Statement of Work
CTI Critical Technical Issue
CTO Certified Test Operator
CW Courseware
CWCI Courseware Configuration Item
CWBS Contract Work Breakdown Structure
C&UT Code and Unit Test

D

DAL Data Accession List
DAU Defense Acquisition University
DBDD Database Design Description
DCL Design Criteria List
DFR Decreasing Failure Region
DFSS Design for Six Sigma
DI Data Item
DID Data Item Description
DOA Dead on Arrival
DoD (US) Department of Defense
DOE (US) Department of Energy
DOF Degrees of Freedom
DORT Daily Operational & Readiness Test
DPM (Archer's) Design Process Model
DR Discrepancy Report
DRD Design Rationale Document
DSM Design Structure Matrix
DT Down Time
DT&E Developmental Test & Evaluation
DTC Design-to-Cost
DTV Design-to-Value

E

E3 Electromagnetic Environment Effects
EBOM Engineering Bill of Materials
ECEF Earth-Centered-Earth-Fixed
ECI Earth-Centered Inertial
ECP Engineering Change Proposal
ECR Engineering Change Request / Earth-Centered Rotating
EDS Entity Development Specification
EE Electrical/Electronic Engineering
EF Exploration Factor (Highsmith)
EFL Exponential Failure Law
EIA Electronic industries Association
EIS Environmental Impact Study
EMC Electromagnetic Compatibility
EMF Electromotive Force
EMI Electro-Magnetic Interference
EMP Engineering Management Plan
ENU East-North-Up (Local Reference)
E/QT Environmental/Qualification Test
ERs Entity Relationships
ERD Entity Relationship Diagram
ERR Engineering Release Record
ESD Electro-Static Discharge
ESEP (INCOSE) Expert Systems Engineering Professional
ES&OH Environmental, Safety, and Occupational Health
ET (US NASA Space Shuttle) External Tank

E($\overline{T}$) Expected Life
EVMS Earned Value Management System

F

FAA (US) Federal Aviation Administration
FAIT Fabrication, Assembly, Integration, and Test
FAR (US) Federal Acquisition Regulation
FCA Functional Configuration Audit
FFP Firm Fixed Price (Contract)
FMEA Failure Modes & Effects Analysis
FMECA Failure Modes & Effects Criticality Analysis
FFBD Functional Flow Block Diagram
FIS Facility Interface Specification
FOC Full Operational Capability
FOM Figure of Merit
FRACAS Failure Reporting and Corrective Action System
FSP Full Scale Production
FTA Fault Tree Analysis

G

GN&C Guidance, Navigation, and Control
GPS Global Positioning System
GSE Ground Support Equipment

H

HAZMAT Hazardous Material
HDBK Handbook
HDD Hardware Design Description
HDP Hardware Development Plan
HE Human Engineering
HF Human Factors
HFE Human Factors Engineering
HFES Human Factors and Ergonomics Society
HITL Hardware-in-the-Loop
HoQ (QFD) House of Quality
HRR Hazard Rate Region
HRS Hardware Requirements Specification
HSI Human-System Integration
HSR Hardware Specification Review
HTR Hardware Trouble Report
HW Hardware
HWCI Hardware Configuration Item

I

I/F Interface
I/O Input/Output
IBR Integrated Baseline Review
I&CO Installation & Checkout
IC Integrated Circuit
ICAO International Civil Aviation Organization
ICD Interface Control Document
ICWG Interface Control Working Group
IDD Interface Design Description
IDE Integrated Development Environment
IDEF Integration DEFinition (Modeling languages)
IE Industrial Engineering
IEC International Electrotechnical Commission
IEEE Institute of Electrical and Electronic Engineers
IFR Increasing Failure Region
IID Iterative and Incremental Development
IMP Integrated Master Plan
IMS Integrated Master Schedule
INCOSE International Council on Systems Engineering
INFOSEC Information Security
INS Inertial Navigation System
INU Inertial Navigation Unit
IOC Initial Operational Capability
IP Integration Point
IP Intellectual Property
IPPD Integrated Product and Process Development
IPR In-Process Review
IPT Integrated Product Team
IRI International Roughness Index
IRS Interface Requirements Specification
ISD Instruction System Development
ISO International Organization of Standards
ISS International Space Station
ITA Independent Test Agency
ITAR (US) International Traffic and Arms Regulations
ITT Independent Test Teams
IV&V Independent Verification & Validation

J

JIT Just-In-Time
JPL Jet Propulsion Laboratory

K

KE Kinetic Energy
KPP Key Performance Parameters
KPI Key Performance Indicator

KSA Knowledge, Skills, and Abilities
KSC (US NASA) Kennedy Space Center

L

LAN Local Area Network
LCC Life Cycle Cost
LCL Lower Control Limit
LDT Logistics Delay Time
$\overline{\text{LDT}}$ Mean Logistics Delay Time (MLDT)
LH Left-Hand
LM (US Apollo) Lunar Module
LOB Line of Business
LOE Level of Effort
LORA Level of Repair Analysis
LRIP Low Rate Initial Production
LRU Line Replaceable Unit
LSA Logistics Support Analysis
LSB Least Significant Bit
LSE Lead Systems Engineer
LSWG Logistics Support Working Group

M

$\overline{M}$ Mean Active Maintenance Time (MAMT)
M_{ct} Corrective Maintenance Time
$\overline{M}_{ct}$ Mean Corrective Maintenance Time
M_{pt} Preventive Maintenance Time
$\overline{M}_{pt}$ Mean Preventive Maintenance Time
M & S Modeling and Simulation
MAMT Mean Active Maintenance Time
MBSE Model-Based Systems Engineering
MC2 Monitor, Command, and Control
MDD Model-Driven Design
MDT Maintenance Down Time
ME Mechanical Engineering
MET Mission Event Timeline
MIL Military
MKS Meter-Kilogram-Second (System)
MLE Maximum Likelihood Estimation
MMI Man-Machine Interface
MNS Mission Needs Statement
MOE Measure of Effectiveness
MOP Measure of Performance
MOS Measure of Suitability
MPS Master Program Schedule
MRB Material Review Board
MRI Magnetic Resonance Imaging
MSB Most Significant Bit
MSDS Material Safety Data Sheet
MSL Mean Sea Level (See AMSL)
MTBF Mean-Time-Between-Failure
MTBM Mean-Time-Between-Maintenance
MTTF Mean-Time-to-Failure
MTTR Mean-Time-to-Repair
MTTRS Mean-Time-to-Restore-Service

N

NASA (US) National Aeronautics & Space Administration
NDA Non-Disclosure Agreement
NCR Non-Conformance Report
NDI Non-Developmental Item
NDIA National Defense Industrial Association
NED North-East-Down (Local Reference)
NEPA (US) National Environmental Policy Act
NGT Nominal Grouping Technique
NIH (US) National Institute of Health
NIST (US) National Institute of Standards and Technology
NOAA (US) National Oceanic and Atmospheric Agency

O

OM&S Operations, Maintenance, & Sustainment
OCD Operational Concept Description
OE Operating Environment
OEM Original Equipment Manufacturer
OH&S Operational Health & Status
OJT On-the-Job (Training)
OM&S Operations, Maintenance, and Sustainability
OMG Object Management Group
Ops Operations
OPSEC Operations Security
ORD Operational Requirements Document
OSE Open Systems Environment
OSHA (US) Occupational Safety & Health Administration
OSP Organizational Standard Process
OS&H (US) Occupational Safety and Health
OT&E Operational Test and Evaluation
OTW Out-the-Window (Display)
OV (US Space Shuttle) Orbiter Vehicle

P

PBS Product Breakdown Structure

PCA Physical Configuration Audit
PDF Probability Density Function
PDT Product Development Team
PE Professional Engineer (Registration)
PERT Program Evaluation and Review Technique
PdM Predictive Maintenance
PDR Preliminary Design Review
PGE Powered Ground Equipment
PHM Prognosis and Health Management (PHM)
PHS&T Packaging, Handling, Storage, & Transportation
PHYSEC Physical Security
PIA Proprietary Information Agreement
PM Project Management/Project Manager
PMB Performance Measurement Baseline
PMP Project Management Plan
PMT Preventive Maintenance Time
POC Point of Contact
PR Problem Report
PRR Production Readiness Review
PSE Peculiar Support Equipment
PWBS Project Work Breakdown Structure
PWO Project Work Order

Q

QA Quality Assurance
QAP Quality Assurance Plan
QAR Quality Assurance Representative
QFD Quality function Deployment
QMS Quality Management System
QT Qualification Test
QR (ISO) Quality Record

R

R&M Reliability and Maintainability
R&D Research & Development
RCM Reliability-Centered Maintenance
RE Requirements Engineering
REQ Requirement
Reqmt. Requirement
RFI Request for Information
RFS Request for Service
RFQ Request for Quotation
RH Right Hand
RFP Request for Proposal
RMA Reliability, Maintainability, & Availability
RMA Rate Monotonic Analysis (Software)
ROMP Risk and Opportunity Management Plan
RMP Risk Management Plan
RMT Requirements Management Tool
ROBP Requirements, Operations, Behavior, and Physical (SE Process)
ROC Required Operational Capability
ROI Return on Investment
ROIC Return on Invested Capital
ROM Rough Order of Magnitude
RPY Roll, Pitch, and Yaw
RRX Rank Regression on X
RRY Rank Regression on Y
RSO Range Safety Officer
RTA Requirements Traceability Audit
RTF Run-to-Failure (See TTF)
RTM Requirements Traceability Matrix
RTSR Ready-to-Ship Review
RVM Requirements Verification Matrix
RVTM Requirements Verification Traceability Matrix

S

6-DOF Six Degrees of Freedom
S&T Simulation and Training
S/N Signal-to-Noise/Serial Number (Context)
SA&M System Acquisition and Management
SBA Simulation-Based Acquisition
SBD System Block Diagram
SCCB Software Configuration Control Board
SCR Software Change Request
SDBTF Specify-Design-Build-Test-Fix (Paradigm)
SDD Software Design Description
SDN System Design Notebook
SDP Software Development Plan
SDR System Design Review
SDRL Subcontract Data Requirements List
SDT System Development Team
SE System Engineering/Systems Engineer
SE&D Systems Engineering and Development
SEA System Element Architecture
SEI Software Engineering Institute
SEIT System Engineering & Integration Team
SEMP Systems Engineering Management Plan
SETA Systems Engineering and Technical Assistance
SFP Single Failure Point
SFR Stabilized Failure Region
SI International System of Units

SI&T System Integration and Test
SITE System Integration, Test, & Evaluation
SIVP System Integration & Verification Plan
SM (US Apollo) Service Module
SMART-V Specific, Measurable, Achievable, Realistic, Testable (Doran); Verifiable and Traceable (Wasson) (Requirements)
SME Subject Matter Expert
SOI System of Interest
SOA Service-Oriented Architecture
SOO Statement of Objectives
SOPP Standard Operating Practices & Procedures
SOS System of Systems
SOW Statement of Work
SPC Statistical Process Control
SPR Software Problem Report
SPS System Performance Specification
SQA Software Quality Assurance
SRB (US NASA Space Shuttle) Solid Rocket Booster
SRD System Requirements Document
SSET Source Selection Evaluation Team
SRR System Requirements Review
SRS Software Requirements Specification
SSDD System/Segment Design Review
SSR Software Specification Review
SysML™ (OMG) Systems Modeling Language™
SVD Software Version Description (Document)
SVR System Verification Review
SVT System Verification Test
SW Software
SwE Software Engineering
SWOT Strengths, Weaknesses, Opportunities, and Threats

T

T&E Test & Evaluation
Ts&Cs (Contract) Terms and Conditions
TBD To be Determined
TBS To Be Supplied
TC Test Case
TCO Total Cost of Ownership
TD Test Discrepancy
TDD Test Driven Development
TEMP Test and Evaluation Master Plan
TEWG Test & Evaluation Working Group
TMDE Test, Measurement, and Diagnostics Equipment
TMP Technical Management Plan
TOO Target of Opportunity
TPM Technical Performance Measure
TPP Technical Performance Parameter
TRR Test Readiness Review
TSO Test Safety Officer
TSR Trade Study Report
TTF Time-to-Failure (See RTF)

U

UC Use Case
UCI User-Computer Interface
UCL Upper Control Limit
UCSD User-Centric System Design
UML™ Unified Modeling Language™
UT Up Time
UUT Unit-Under-Test
UAS Unmanned Aerial System

V

V&V Verification & Validation
VAL Validation
VDD Version Description Document
VOC Voice of the Customer
VTR Verification Test Report

W

WAN Wide Area Network
WBS (Project) Work Breakdown Structure
WCS World Coordinate System

Appendix B

INCOSE HANDBOOK TRACEABILITY

For those reading INCOSE SEHv4 (2015) or studying for one of the International Council on Systems Engineering (INCOSE) ACEP, CSEP, or ESEP Certifications, Appendix B provides a quick reference for suggested linkages between the INCOSE SEHv4 outline topics to this textbook – Systems Engineering Analysis, Design, and Development: Concepts, Principles, and Practices.

B.1 REFERENCE

INCOSE SEHv4 (2015), Systems Engineering Handbook: A Guide for System Life Cycle Process and Activities, 4th ed. D. D. Walden, G. J. Roedler, K. J. Forsberg, R. D. Hamelin, and, T. M. Shortell (Eds.). San Diego, CA: International Council on Systems Engineering.

System Engineering Analysis, Design, and Development: Concepts, Principles, and Practices, Second Edition. Charles S. Wasson.

Companion Website: www.wiley.com/go/systemengineeringanalysis2e

INCOSE SE Handbook	System Engineering Analysis, Design, and Development	
1.1.1 Chapter 1 Systems Engineering Handbook Scope		
Chapter 2. Systems Engineering Overview		
	Chapter 2	The Evolving State of SE Practice—Challenges and Opportunities
2.1 Introduction		
2.2 Definitions and Concepts of a System	Part 1 Chapters 1–34	Systems Engineering and Analysis Concepts Definitions of Key Terms
2.3 The Hierarchy Within A System	Chapter 8 Chapter 9	System Levels of Abstraction, Semantics, and Elements Architectural Frameworks of the SOI and Its Operating Environment
2.4 Definition of Systems-of-Systems	Chapter 1	Introduction to Systems Engineering, Analysis, Design, and Development
2.5 Enabling Systems	Chapter 4 Chapter 8 Chapter 9	User Enterprise Roles, Missions, and System Applications System Levels of Abstraction, Semantics, and Elements Architectural Frameworks of the SOI and Its Operating Environment
2.6 Definition of Systems Engineering	Chapter 1	Systems, Engineering, and Systems Engineering
2.7 Origins and Evolution of Systems Engineering	Chapter 1	Systems, Engineering, and Systems Engineering
2.8 Use and Value of Systems Engineering	Chapter 1 Chapter 2	Systems, Engineering, and Systems Engineering The Evolving State of SE Practice - Challenges and Opportunities
2.9 Systems Science and Systems Thinking	Chapter 1	Systems, Engineering, and Systems Engineering
2.10 Systems Engineering Leadership	Chapters 12–18	Introduction to System Development Strategies
2.11 Systems Engineering Professional Development	Chapter 2	The Evolving State of SE Practices: Challenges and Opportunities
Chapter 3. Generic Life Cycle Stages		
3.1 Introduction		
3.2 Life Cycle Characteristics	Chapter 3	System Attributes, Properties, and Characteristics
3.3 Life Cycle Stages	Chapter 3	System Attributes, Properties, and Characteristics
3.4 Life Cycle Approaches	Chapter 3	System Attributes, Properties, and Characteristics
3.5 What is Best for Your Organization, Project, or Team?	Chapter 1 Chapter 2 Chapter 11 Chapter 14 Chapter 15 Epilog	Systems, Engineering, and Systems Engineering The Evolving State of SE Practices: Challenges and Opportunities Analytical Problem-Solving and Solution Development Synthesis The Wasson Systems Engineering Process System Development Process Models Epilog

3.6 Introduction to Case Studies	Chapters 1, 2, 4, 5, 11, 20, 24–28, 33	
	Companion Website	Refer to Book's Companion Website at www.wiley.com/go/systemengineeringanalysis2e
Chapter 4. Technical Processes	Chapter 12	Introduction to System Development Strategies
	Chapter 13	System Verification and Validation (V&V) Process Strategy
	Chapter 14	The Wasson Systems Engineering Process
	Chapter 15	System Development Process Models Strategy
4.1 Business or Mission Analysis Process	Chapter 4	User Enterprise Roles, Missions, and System Applications
	Chapter 5	User Needs, Mission Analysis, Use Cases, and Scenarios
	Chapter 29	System Deployment, OM&S, Retirement, and Disposal
4.2 Stakeholder Needs and Requirements Definition Process	Chapter 3	System Attributes, Properties, and Characteristics
	Chapter 4	User Enterprise Roles, Missions, and System Applications
	Chapter 5	User Needs, Mission Analysis, Use Cases, and Scenarios
	Chapter 22	Requirement Statement Development
	Chapter 29	System Deployment, OM&S, and Retirement, and Disposal
4.3 Systems Requirements Definition Process	Chapter 4	User Enterprise Roles, Missions, and System Applications
	Chapter 5	User Needs, Mission Analysis, Use Cases, and Scenarios
	Chapter 6	System Concepts Formulation & Development
	Chapter 7	System Command & Control (C2)–Phases, Modes, and States of Operation
	Chapter 10	Modeling MISSION SYSTEM and ENABLING SYSTEM Operations
	Chapter 21	Requirements Derivation, Allocation, Flow Down, and Traceability
	Chapter 22	Requirements Statement Development
	Chapter 29	System Deployment, OM&S, and Retirement, and Disposal
4.4 Architecture Definition Process	Chapter 7	System Command and Control (C2)–Phases, Modes, and States of Operation
	Chapter 8	System Levels of Abstraction, Semantics, and Elements
	Chapter 9	Architectural Frameworks of the SOI and Its Operating Environment
	Chapter 12	Introduction to System Development Strategies
	Chapter 26	System and Entity Architecture Development
	Chapter 27	System Interface Definition, Analysis, Design, and Control
4.5 Design Definition Process	Chapter 12	Introduction to System Development Strategies
	Chapter 16	System Configuration Identification and Component Selection Strategy
	Chapter 24	User-Centered System Design (UCSD)
	Chapter 25	Engineering Standards of Units, Coordinate Systems, and Conventions
	Chapter 26	System and Entity Architecture Development
	Chapter 27	System Interface Definition, Analysis, Design, and Control

(*continued*)

INCOSE SE Handbook	System Engineering Analysis, Design, and Development	
4.6 System Analysis Process	PART 1	SYSTEMS ENGINEERING AND ANALYSIS CONCEPTS
	Chapter 30	Introduction to Analytical Decision Support
	Chapter 31	Performance Analysis, Budgets, and Safety Margins
	Chapter 34	System Reliability, Maintainability, and Availability (RMA)
4.7 Implementation Process	Chapter 12	Introduction to System Development Strategies
	Chapter 13	System Verification and Validation (V&V) Strategy
	Chapter 14	The Wasson Systems Engineering Process
	Chapter 15	System Development Process Models Strategy
	Chapter 16	System Configuration Identification and Component Selection Strategy
	Chapter 17	System Documentation Strategy
	Chapter 18	Technical Reviews Strategy
4.8 Integration Process	Chapter 8	System Levels of Abstraction, Semantics, and Elements
	Chapter 12	Introduction to System Development Strategies
	Chapter 16	System Configuration Identification and Component Selection Strategy
	Chapter 28	System Integration, Test, and Evaluation (SITE) Practices
4.9 Verification Process	Chapter 12	Introduction to System Development Strategies
	Chapter 13	System Verification and Validation (V&V) Strategy
	Chapter 14	The Wasson Systems Engineering Process Model
	Chapter 28	System Integration, Test, and Evaluation (SITE)
4.10 Transition Process	Chapter 6	System Concepts Formulation and Development
	Chapter 29	System Deployment, OM&S, and Retirement, and Disposal
4.11 Validation Process	Chapter 12	Introduction to System Development Strategies
	Chapter 13	System Verification and Validation (V&V) Strategy
4.12 Operation Process	Chapter 3	System Attributes, Properties, and Characteristics
	Chapter 6	System Concepts Formulation and Development
	Chapter 10	Modeling MISSION and ENABLING SYSTEM Operations
	Chapter 29	System Deployment, OM&S, Phase-Out, Retirement, and Disposal
4.13 Maintenance Process	Chapter 3	System Attributes, Properties, and Characteristics
	Chapter 6	System Concepts Formulation and Development
	Chapter 10	Modeling MISSION SYSTEM and ENABLING SYSTEM Operations
	Chapter 29	System Deployment, OM&S, Retirement, and Disposal
	Chapter 34	System Reliability, Maintainability, and Availability (RMA)
4.14 Disposal Process	Chapter 3	System Attributes, Properties, and Characteristics
	Chapter 6	System Concepts Formulation and Development
	Chapter 29	System Deployment, OM&S, Retirement, and Disposal

5.0 Technical Management Processes		
5.1 Project Planning Process	Chapter 12	Introduction to System Development Strategies
	Chapter 13	System Verification and Validation (V&V) Strategy
	Chapter 14	The Wasson Systems Engineering Process
	Chapter 15	System Development Process Models Strategy
	Chapter 16	System Configuration Identification and Component Selection Strategy
	Chapter 17	System Documentation Strategy
	Chapter 18	Technical Reviews Strategy
5.2 Project Assessment and Control Process	Chapter 12	Introduction to System Development Strategies
	Chapter 13	System Verification and Validation (V&V) Strategy
	Chapter 16 -	System Configuration Identification and Component Selection Strategy
	Chapter 18	Technical Reviews Strategy
5.3 Decision Management Process	Chapter 13	System Verification and Validation (V&V) Strategy
	Part 3	Analytical Decision Support Practices
	Chapter 30	Introduction to Analytical Decision Support Practices
	Chapter 31	System Performance Analysis, Budgets, and Safety Margins
	Chapter 32	Trade Study Analysis of Alternatives (AoA)
	Chapter 33	System Modeling and Simulation (M&S)
	Chapter 34	System Reliability, Maintainability, and Availability (RMA)
5.4 Risk Management Process	Chapter 12	Introduction to System Development Strategies
	Chapter 13	System Verification and Validation (V&V) Strategy
	Chapter 18	Technical Reviews Strategy
	Chapter 24	User-Centric System Design (UCSD)
	Chapter 32	Trade Study Analysis of Alternatives (AoA)
	Chapter 33	System Modeling and Simulation (M&S)
	Chapter 34	System Reliability, Maintainability, and Availability (RMA)
5.5 Configuration Management Process	Chapter 12	Introduction to System Development Strategies
	Chapter 16	System Configuration Identification and Component Selection Strategy
	Chapter 27	System Interface Definition, Analysis, Design, and Control
5.6 Information Management Process	Chapter 16	System Configuration Identification and Component Selection Strategy
	Chapter 25	Engineering Standards of Units, Coordinate Systems, and Frames of Reference
	Chapter 33	System Modeling and Simulation (M&S)
	Chapter 34	System Reliability, Maintainability, and Availability (RMA)
5.7 Measurement Process	Chapter 12	Introduction to System Development Strategies
	Chapter 13	System Verification and Validation (V&V) Strategy
	Chapter 15	System Development Process Models Strategy
	Chapter 21	Requirements Derivation, Allocation, Flow Down, and Traceability
	Chapter 29	System Deployment, OM&S, Retirement, and Disposal

(*continued*)

INCOSE SE Handbook	System Engineering Analysis, Design, and Development	
	Chapter 31	System Performance Analysis, Budgets, and Safety Margins
	Chapter 34	System Reliability, Maintainability, and Availability (RMA)
5.8 Quality Assurance Process	Chapter 12	Introduction to System Development Strategies
	Chapter 13	System Verification and Validation (V&V) Strategy
	Chapter 14	The Wasson Systems Engineering Process
	Chapter 15	System Development Process Models Strategy
	Chapter 18	Technical Reviews Strategy
	Chapter 20	Specification Development Approaches
	Chapter 23	Specification Analysis
	Chapter 28	System Integration, Test, and Evaluation (SITE) Practices
Chapter 6. Agreement Processes		
6.1 Acquisition Process	Chapter 3	System Attributes, Properties, and Characteristics
	Chapter 16	System Configuration Identification and Component Selection Strategy
	Chapter 18	Technical Reviews Strategy
6.2 Supply Process	Chapter 4	User Enterprise Roles, Missions, and System Applications
	Chapter 29	System Deployment, OM&S, Retirement, and Disposal
	Chapter 34	System Reliability, Maintainability, and Availability (RMA)
Chapter 7. Organizational Project-Enabling Processes		
7.1 Life Cycle Model Management Process		
7.2 Infrastructure Management Process		
7.3 Portfolio Management Process		
7.4 Human Resource Management Process		
7.5 Quality Management Process		
7.6 Knowledge Management Process	Chapter 4	User Enterprise Roles, Missions, and System Applications
	Chapter 5	User Needs, Mission Analysis, Use Cases, and Scenarios
	Chapter 14	The Wasson Systems Engineering Process
	Chapter 16	System Configuration Identification and Component Selection Strategy
	Chapter 22	Requirements Statement Development
	Chapter 29	System Deployment, OM&S, Retirement, and Disposal Practices
	Chapter 33	System Modeling and Simulation (M&S)
	Chapter 34	System Reliability, Maintainability, and Availability (RMA)

Chapter 8. Tailoring Process and Application of Systems Engineering		
8.1 Tailoring Process	Chapter 12	Introduction to System Development Strategies
	Chapter 19	System Specification Concepts
	Epilog	
8.2 Tailoring for Specific Product Sector or Domain Application	Chapter 5	User Needs, Mission Analysis, Use Cases, and Scenarios
8.3 Application of Systems Engineering for Product Line Management	Chapter 3	System Attributes, Properties, and Characteristics
	Chapter 5	User Needs, Mission Analysis, Use Cases, and Scenarios
	Chapters 12 - 18	System Development Strategy Practices
8.4 Application of Systems Engineering for Services	Chapters 1–34	
	Chapter 14	The Wasson Systems Engineering Process
8.5 Application of Systems Engineering for Enterprises	Chapters 1–34	
	Chapter 5	User Needs, Mission Analysis, Use Cases, and Scenarios
	Chapter 12	Introduction to System Development Strategies
	Chapter 13	System Verification and Validation (V&V) Strategy
	Chapter 14	The Wasson Systems Engineering Process
	Chapter 15	System Development Process Models Strategy
	Chapter 16	System Configuration Identification and Component Selection Strategy
	Chapter 17	System Documentation Strategy
	Chapter 18	Technical Reviews Strategy
8.6 Application of Systems Engineering for Very Small and Micro Enterprises (VSME)	Chapters 1–34	
	Chapter 12	Introduction to System Development Strategies
	Chapter 13	System Verification and Validation (V&V) Strategy
	Chapter 14	The Wasson Systems Engineering Process
	Chapter 15	System Development Process Models
	Chapter 16	System Configuration Identification and Component Selection Strategy
	Chapter 17	System Documentation Strategy
	Chapter 18	Technical Reviews Strategy
Chapter 9. Cross-Cutting Systems Engineering Methods	Chapters 1–34	
9.1 Modeling and Simulation	Chapter 10	Modeling MISSION and ENABLING SYSTEM Operations Behavior, and Physical Interactions
	Chapter 33	System Modeling and Simulation (M&S)

(*continued*)

INCOSE SE Handbook		System Engineering Analysis, Design, and Development
9.2 Model-Based Systems Engineering	Chapter 2	The Evolving State of SE Practices: Challenges and Opportunities
	Chapter 10	Modeling MISSION and ENABLING SYSTEM Operations
	Chapter 33	System Modeling and Simulation (M&S)
9.3 Functions-Based Systems Engineering Method	Chapter 2	The Evolving State of SE Practices: Challenges and Opportunities
	Chapter 14	The Wasson Systems Engineering Process Model
9.4 Object-Oriented Systems Engineering Method	Chapter 5	User Needs, Mission Analysis, Use Cases, and Scenarios
	Chapter 7	System Command and Control (C2) - Phases, Modes, and States of Operation
	Chapter 8	System Levels of Abstraction, Semantics, and Elements
	Chapter 9	Architectural Frameworks of the SOI and Its Operating Environment
	Chapter 10	Modeling MISSION and ENABLING SYSTEM Operations, Behavior, and Physical Interactions
	Chapter 28	System Integration, Test, and Evaluation (SITE) Practices
	Appendix C	Systems Modeling Language (SysML™) Constructs
9.5 Prototyping	Chapter 7	System Command and Control (C2)–Phases, Modes, and States of Operation
	Chapter 16	Configuration Identification and Component Selection Strategy
	Chapter 20	Specification Development Approaches
	Chapter 24	User-Centered System Design (UCSD)
	Chapter 33	System Modeling and Simulation (M&S)
9.6 Interface Management	Chapter 12	Introduction to System Development Strategies
	Chapter 16	Configuration Identification and Component Selection Strategy
	Chapter 27	System Interface Definition, Analysis, Design, and Control
9.7 Integrated Product and Process Development	Chapter 2	The Evolving State of SE Practices: Challenges and Opportunities
	Chapter 12	Introduction to System Development Strategies
	Chapter 14	The Wasson Systems Engineering Process
	Chapter 15	System Development Process Models Strategy
	Chapter 16	Configuration Identification and Component Selection Strategy
9.8 Lean Systems Engineering	Chapter 2	The State of SE: Challenges and Opportunities
9.9 Agile Systems Engineering	Chapter 15	System Development Process Models Strategy
Chapter 10. Specialty Engineering Activities		
10.1 Affordability/Cost Effectiveness/Life-Cycle Cost Analysis	Chapter 3	System Attributes, Properties, and Characteristics
	Chapter 5	User Needs, Mission Analysis, Use Cases, and Scenarios
	Chapter 21	Requirements Derivation, Allocation, Flow Down, and Traceability
	Chapter 30	Introduction to Analytical Decision Support Practices
	Chapter 34	System Reliability, Maintainability, and Availability (RMA)

10.2 Electromagnetic Compatibility	Chapter 8	System Levels of Abstraction, Semantics, and Elements
	Chapter 9	Architectural Frameworks of the SOI and Its Operating Environment
	Chapter 30	Introduction to Analytical Decision Support
10.3 Environmental Engineering/Impact Analysis	Chapter 9	Architectural Frameworks of the SOI and Its Operating Environment
	Chapter 28	System Integration, Test, and Evaluation (SITE)
	Chapter 30	Introduction to Analytical Decision Support
10.4 Interoperability Analysis	Chapter 3	System Attributes, Properties, and Characteristics
	Chapter 5	User Needs, Mission Analysis, Use Cases, and Scenarios
	Chapter 10	Modeling MISSION and ENABLING SYSTEMS Operations
	Chapter 26	System and Entity Architecture Development
	Chapter 27	System Interface Definition, Analysis, Design, and Control
	Chapter 30	Introduction to Analytical Decision Support
10.5 Logistics Engineering	Chapter 5	User Needs, Mission Analysis, Use Cases, and Scenarios
	Chapter 29	System Deployment, OM&S, and Retirement, and Disposal Practices
	Chapter 30	Introduction to Analytical Decision Support
	Chapter 34	System Reliability, Maintainability, and Availability (RMA)
10.6 Manufacturing and Producibility Analysis	Chapter 30	Introduction to Analytical Decision Support
10.7 Mass Properties Engineering	Chapter 3	System Attributes, Properties, and Characteristics
	Chapter 30	Introduction to Analytical Decision Support
	Chapter 31	System Performance Analysis, Budgets, and Safety Margins
	Chapter 34	System Reliability, Maintainability, and Availability (RMA)
10.8 Reliability, Availability, and Maintainability	Chapter 26	System and Entity Architecture Development
	Chapter 27	System Interface Definition, Analysis, Design, and Control
	Chapter 34	System Reliability, Maintainability, and Availability (RMA)
10.9 Resilience Engineering	Chapter 26	System and Entity Architecture Development
	Chapter 27	System Interface Definition, Analysis, Design, and Control
	Chapter 34	System Reliability, Maintainability, and Availability (RMA)
10.10 System Safety Engineering	Chapter 4	User Enterprise Roles, Missions, and System Applications
	Chapter 5	User Needs, Mission Analysis, Use Cases, and Scenarios
	Chapter 7	System Command and Control (C2)–Phases, Modes, and States of Operation

(*continued*)

INCOSE SE Handbook		System Engineering Analysis, Design, and Development
	Chapter 10	Modeling MISSION and ENABLING SYSTEMS Operations
	Chapter 18	Technical Reviews Strategy
	Chapter 20	Specification Development Approaches
	Chapter 23	Specification Analysis
	Chapter 24	User-Centered System Design (UCSD)
	Chapter 26	System and Entity Architecture Development
	Chapter 27	System Interface Definition, Analysis, Design, and Control
	Chapter 30	Introduction to Analytical Decision Support
	Chapter 31	Performance Analysis, Budgets, and Safety Margins
	Chapter 32	Trade Study–Analysis of Alternatives (AoA)
	Chapter 33	System Modeling and Simulation (M&S)
	Chapter 34	System Reliability, Maintainability, and Availability (RMA)
10.11 System Security Engineering	Chapter 4	User Enterprise Roles, Missions, and System Applications
	Chapter 5	User Needs, Mission Analysis, Use Cases, and Scenarios
	Chapter 23	Specification Analysis
	Chapter 26	System and Entity Architecture Development
	Chapter 27	System Interface Definition, Analysis, Design, and Control
	Chapter 30	Introduction to Analytical Decision Support
10.12 Training Needs Analysis	Chapter 4	User Enterprise Roles, Missions, and System Applications
	Chapter 5	User Needs, Mission Analysis, Use Cases, and Scenarios
	Chapter 30	Introduction to Analytical Decision Support
	Chapter 33	System Modeling and Simulation (M&S)
10.13 Usability Analysis/Human Systems Integration	Chapter 5	User Needs, Mission Analysis, Use Cases, and Scenarios
	Chapter 24	User-Centered System Design (UCSD)
	Chapter 30	Introduction to Analytical Decision Support
10.14 Value Engineering	Chapter 32	Analysis of Alternatives (AoA) Trade Studies
Appendix A References	Chapters 1–34	End of Chapter References
Appendix B Acronyms	Appendix A	Acronyms and Abbreviations
Appendix C Terms and Definitions	Chapters 1–34	Definitions of Key Terms in each Chapter
Appendix D: N^2 Diagram of Systems Engineering Processes		
Appendix E: Input/Output Descriptions		
Appendix F: Acknowledgements		
Appendix G: Comment Form		
	Appendix C	Systems Modeling Language (SysML™) Constructs

Appendix C

SYSTEM MODELING LANGUAGE (SYSML™) CONSTRUCTS

C.1 INTRODUCTION

Systems Engineering (SE), as a multi-discipline *problem-solving* and *solution development* methodology is supported by methods such as Model-Based Systems Engineering (MBSE), Model-Driven Design (MDD) and tools to conceptualize, express, and elaborate systems in terms of their operational, behavioral, and physical relationships, properties, and characteristics. Modeling a system, product, or service requires establishing an analytical framework that enables us to represent the time-dependent, *sequential* and *concurrent* process flows and tasks with embedded *transfer functions* that enable us to transform a set of *acceptable/unacceptable* inputs into a set of *acceptable/unacceptable* behavioral outputs (Figure 3.2). *How do we create and characterize these process flows*? The solution begins with a graphical description language such the Systems Modeling Language (SysML™). SysML™ established by the Object Management Group (OMG™). SysML™ constructs provide the basis for some of the figures used in *System Engineering Analysis, Design, and Development: Concepts, Principles, and Practices*.

A Word of Caution C.1

SE Analysis, Design, and Development focuses on establishing a foundation in SE concepts, principles, and practices, not SysML™. Since SysML™ is an integral part of what many SEs and System Analysts do, this text employs a few the SysML™ diagrams-constructs-to communicate … *concepts, principles,* and *practices*. Therefore, Appendix C should not be interpreted as a tutorial on SysML™ or its application. Only specific SysML™ diagrams related to discussions in the text are used. For more detailed descriptions of SysML™ refer to Delgatti (2013), Friedenthal, et al (2014), and OMG (2012).

Due to space restrictions and the need to maintain minimum font sizes, figures in this text do not graphically portray all of the attributes of a SysML™ construct such as model elements, diagram frames, guillemets—≪ ≫, namespaces, compartments, ports and flows, and semantics. For the current, official specification and description of SysML™, its diagrams, and their applications, always refer to the Object Management Group (OMG™) web site for SysML™ at www.omgsysml.org.

C.2 ENTITY RELATIONSHIPS (ERs)

Systems, by definition, are composed of multiple levels of *abstraction* and *entities* (Figure 8.4) within each level that may or may not have direct relationships. Similarly, a system may have direct or associative Entity Relationships (ERs) to external systems that are time dependent. Analytically, we refer to the systems and their internal and external interactions as ERs. ERs are depicted graphically in an Entity Relationship Diagram (ERD) such as Figure 26.2.

ERs graphically express *compositional* and associative relationships and linkages typically between *vertical* levels of abstraction or external systems. *Compositional* ERs are characterized by two concepts, *Generalization* and *Aggregation*, as shown in Figure C.1.

System Engineering Analysis, Design, and Development: Concepts, Principles, and Practices, Second Edition. Charles S. Wasson.

Companion Website: www.wiley.com/go/systemengineeringanalysis2e

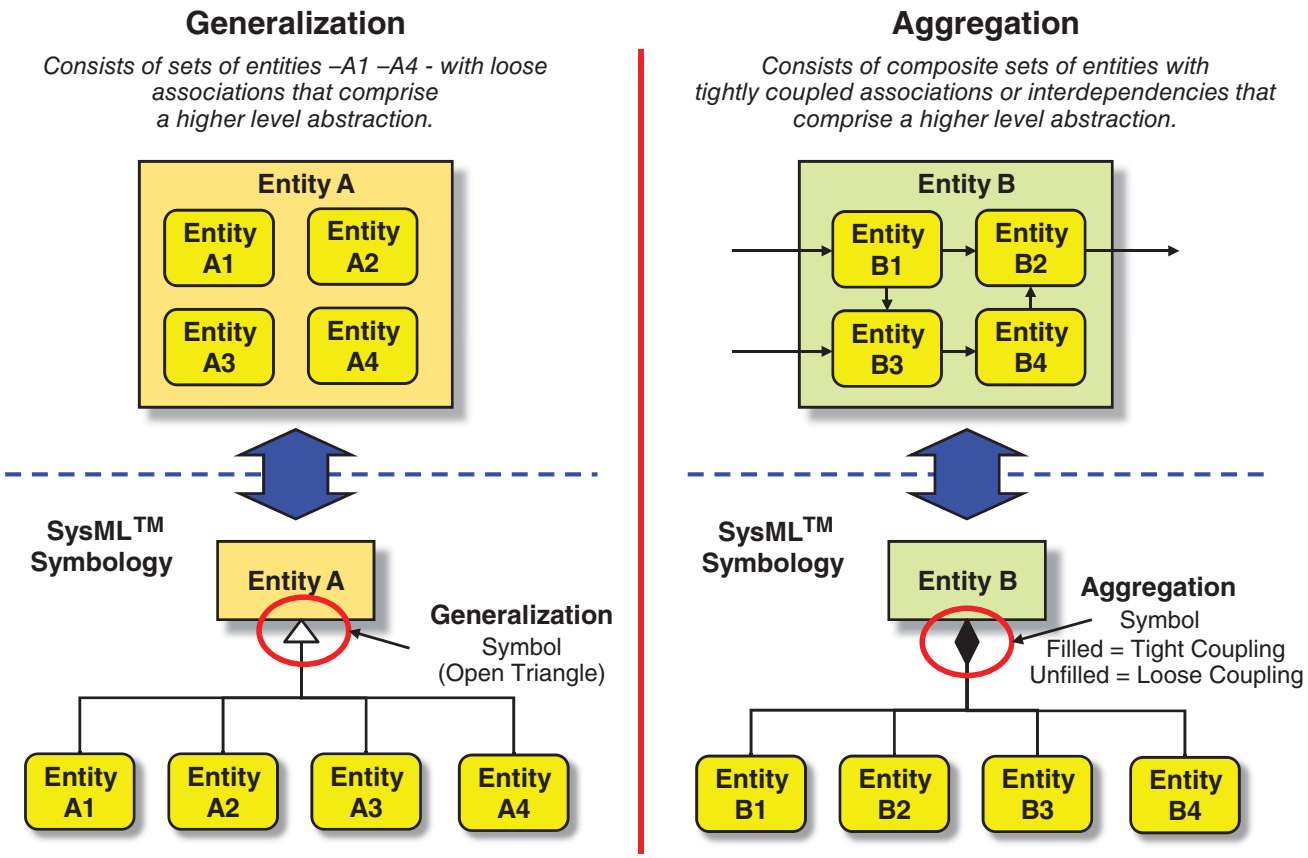


Figure C.1 SysML™ Generalization versus Aggregation Concepts

Let's explore each of these concepts.

C.2.1 Composition by Generalization

Reference: OMG (2012, p. 35)

The concept of *Generalization* simply means that a SYSTEM or one or more of its multi-level ENTITIES are characterized by an associative ER with lower level entities (Figure 21.2). Referring to the upper left side of Figure C.1, a SYSTEM is composed of a *loose* association of four entities, A1–A4. For example, a Mounting Kit is a *generalization* of a bag or box of parts that may or may not have *physical* associations (Figure 8.9) other than being in the bag together.

From an ER perspective, the relationships are *hierarchical* – bag/box (parent) consists of parts (children) as shown in Figures 21.2 and C.1. *Generalizations* are symbolized by an open triangle attached to the higher level *class* or *parent*. Consider the following example.

Example C.1

Division of Motor Vehicles (DMV) Database

A government's Division of Motor Vehicles (DMV) issues licenses and vehicle tags authorizing the vehicle to be used on roadways. A database is used to track vehicle attributes—trucks/cars, models, and so forth—and vehicle tag numbers.

Within the database, *classes* of *vehicles*—trucks and cars by manufacturer—are hierarchically structured into multiple levels of *subclasses* representing unique models. As a result, we can state that *classes* of vehicles such as trucks are *generalizations* of Manufacturer X's truck models, Manufacturer Y's truck models, and so on. Since the vehicles have no physical ERs other than being created by a given manufacturer or set of designs, we refer to these as associative relationships as generalizations.

Graphically, however, the open triangle symbol in a *generalization* ER diagram expresses the "existence" of lower level ERs but nothing about quantity; it is *unitless*. Analytically, we need a mathematical relationship to express quantity information for each ER. This is accomplished by annotating each ER link as being One-to-One, One-to-Many, or Many-to-One ER. Therefore, the end of the ER opposite to the *Generalization* open triangle—and later Aggregation diamond—is annotated with:

- 0..*—Represents a Zero to Many ER. Where the existence of lower level entities may or may not be present; a dashed line is sometimes used in ER link on the end where the condition occurs.
- 1..*—Represents a *One-to-Many* ER.

Figure 26.2 provides an illustration that expresses these relationships.

The preceding discussion illustrates ERs (Figure 8.9) based on logical *associations*, not physical ERs. This brings us to our next topic, Composition by Aggregation.

C.2.1.1 Composition by Aggregation Some systems or products are composed of components that are physically connected—mechanically, electrically, wirelessly, optically, and so forth. For example, an automobile engine, as an abstract entity, represents the physical integration—*composition*—of the engine block, pistons, cam shaft, and other components. Where direct, compositional relationships exist, we refer to it as *aggregation* as shown on the right side of Figure C.1.

Aggregation represents the manifestation of the SYSTEM definition provided in Chapter 1. It is symbolized by a *diamond* attached to the higher-level *class* or *parent* of an ER. The *Aggregation* diamond occurs in two forms as (1) a *filled* diamond and (2) an *unfilled* diamond. Let's define these:

- **Filled Diamond**—Represents a "Part Association" (OMG, 2012, p. 35) consisting of a *tightly bound coupling*—physical connections—between the parent and its children—"Aggregation = Composite" (OMG, 2012, p. 47). Where there are two or more Part Associations, the linkages are referred to as a Multi-branch Parts Association (OMG, 2012, p. 35).
- **Open (Unfilled) Diamond**—Represents a "Shared Aggregation" (OMG, 2012, p. 35) or *loose coupling*. Again, where there are two or more Part Associations, the lines are referred to as a Multi-branch Shared Association (OMG, 2012, p. 35).

As in the case of *Generalization*, *Aggregation* is annotated to express ER quantity attributes such as One-to-One (1), One-to-Many (1..*), or Many-to-One.

C.3 SysML™ DIAGRAMS

As a system description language, SysML™ provides a toolkit of diagrammatic structures—constructs—that enable SEs, System Analysts, and Engineers to model various aspects of a system, product, or service. SysML™ diagrams consist of three types as shown in Figure C.2: Behavior, Requirements, and Structure.

- **Behavior Diagrams**—Enable us to model the external and internal, stimulus–response, behavioral performance interactions between a system, product, or service and its Users, external systems, and its OPERATING ENVIRONMENT.
- **Requirements Diagrams**—Enable us to characterize the hierarchical structure of system, product, or service requirements.
- **Structure Diagrams**—Enable us to establish the architectural framework and compositional ERs of the systems, products, or services we choose to model.

Let's briefly describe each of these types of diagrams.

C.3.1 Behavior Diagrams

Behavior Diagrams enable us to model the behavioral interactions between two or more entities including their ERs. SysML™ Behavioral Diagrams used in this text include Use Case Diagrams (UCDs), Sequence Diagrams, Activity Diagrams, and State Diagrams.

C.3.1.1 Use Case Diagrams (UCDs) **References:** OMG (2012, p. 123–125)

One of the first steps in SE is understanding (1) *who* a system, product, or service's Stakeholders are—Users and End Users, (2) *what* they expect the system, product, or service or accomplish— performance-based outcomes, and (3) *how well* the outcome is to be achieved in terms of performance. This is a critical step that forms the basis for deriving system, product, or services capabilities from which specification requirements can be derived.

SysML™ Use Case Diagrams (UCDs) such as the one shown in Figure C.3 enable us to analytically represent Users and their UCs—stimulus–response interactions—with a system, product, or service.

A few of the key attributes of a UCD include the following:

- **Actors**—Users are represented as *stick figures*. Actors can be any entity that interacts with a system, product, or service such as (1) humans and human roles—pilot, navigator, and so forth, (2) places—environmental conditions and so forth, and (3) things—external systems and so forth. Different Actors may share the same UCs. For example, UCs #1 and #2 are common to Actors #1 and #2.
- **System Boundary**—Is depicted by a rectangular box.
- **Use Cases (UCs)**—Are represented by ovals and assigned titles with a specific syntax: (1) an *active verb* followed by (2) an *outcome-based result* expected from the system. For example, UC #X—Print Document. The *outcome* represents *what* the Users (Actors) expect the system, product, or service to accomplish (Principle 5.14); *not* the capability-based action the system performs to produce the outcome—Printing.

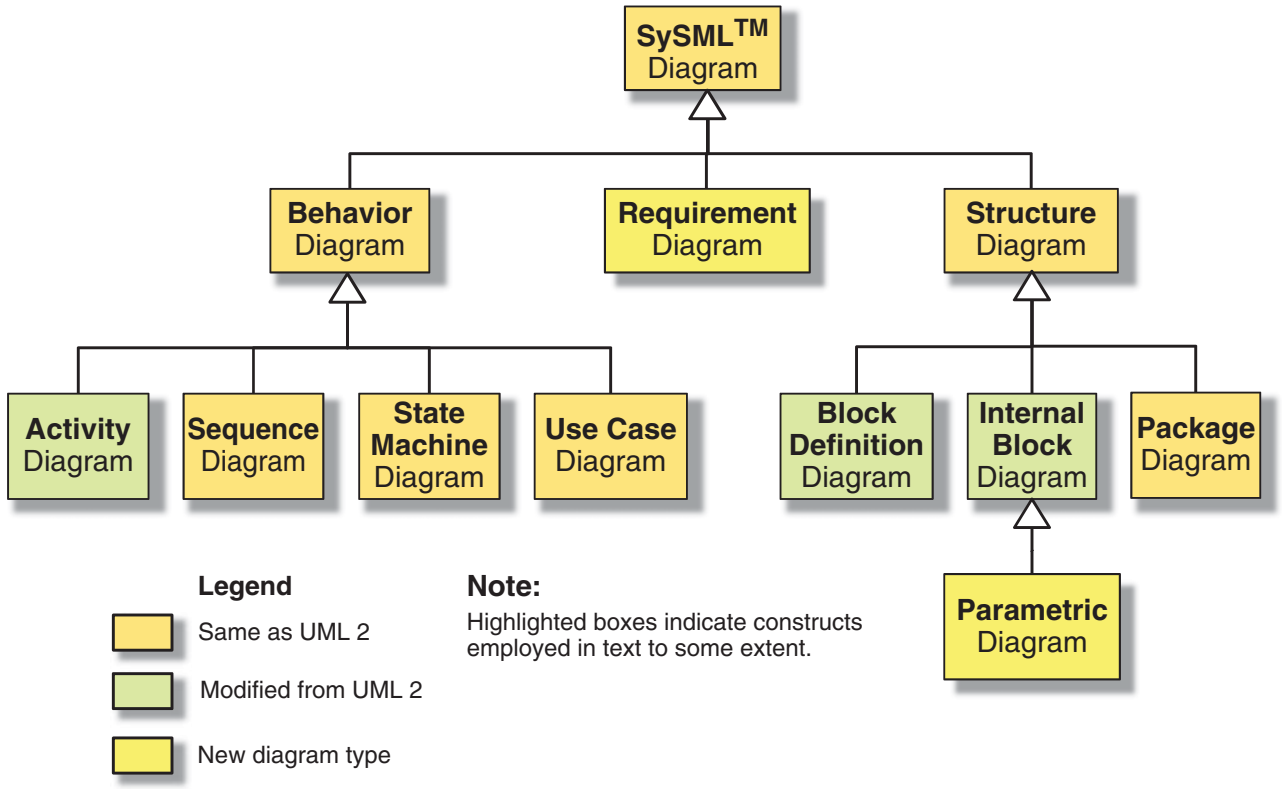


Figure C.2 SysML™ Taxonomy of Diagram Types (OMG, 2012, p. 167)

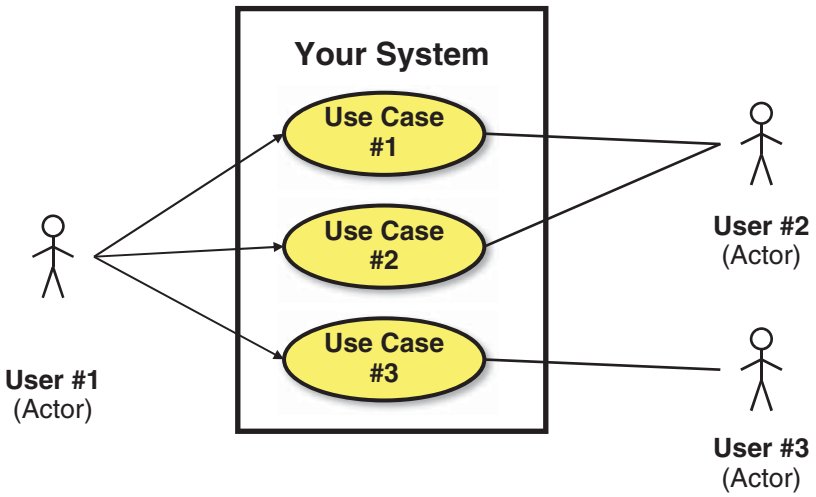


Figure C.3 SysML™ Use Case Diagram (UCD) Construct

- **Extension UC**—A refinement of a base UC. For example, automobile UC #X Drive Vehicle as a base UC can be elaborated into extension UCs such as Start Vehicle, Accelerate Vehicle, Decelerate Vehicle, and Stop Vehicle (OMG, 2012, p. 186).

A Word of Caution C.2

Pay close attention to the above-mentioned UCs point. Most SEs and Systems Analysts *default* to assigning UC titles based their perspective of what *action* (function) the System or Product is performing internally. Remember—you have to understand *what*

performance-based outcomes the User expects the System or Product to accomplish operationally *before* you can define *how* to accomplish it—implementation. Recognize the difference!

C.3.1.2 Sequence Diagrams

References: OMG (2012, p. 113–118)

Once we have established: (1) *who* the Users (Actors) are and (2) *what* they expect the system to accomplish, the next step is to expand the UCs into a level of detail that represents the stimulus–response behaviors of the system, product, or service. This enables us to collaborate with Users (Actors) to characterize *how* they expect the system, product, or service to respond behaviorally to their stimulation, excitations, or cues.

SysML™ Sequence Diagrams such as the one shown in Figure C.4 enable us to depict two types of interactions: (1) external interactions with the User and (2) internal interactions that depict how it processes the User's inputs to produce behavioral responses.

A few of the key attributes of a Sequence Diagram include the following:

- **Convention**—Begin with the generalized Engineering practice of a left-to-right flow.
- **Actors**—Represent Users—humans and roles, places, and things—relevant to a Level of Abstraction and system boundary.
- **Lifeline**—A vertical line extending below each Actor representing time (top-down).
- **Activation Box**—A vertical rectangular box along each Lifeline representing the Input/Output (I/O) stimulus–response processing performed by the Actor.
- **Events**—Represent triggers-stimuli, excitations, or cues—that initiate processing by another Actor.

Author's Note C.1

Observe:

1. That the Activation Boxes do not extend the full length of the Lifelines.
2. The operative term "Activation."

When a system, product, or service is not actively performing stimulus–response processing, it is in a Wait State—idle—awaiting the next input or in a Standby Mode (Chapter 7) of operation. In most cases, when an Actor is activated and is designed to conserve energy, it may enter a Standby Mode—dormant or sleep—requiring the operator to "Activate" it for normal operation. An office copier,

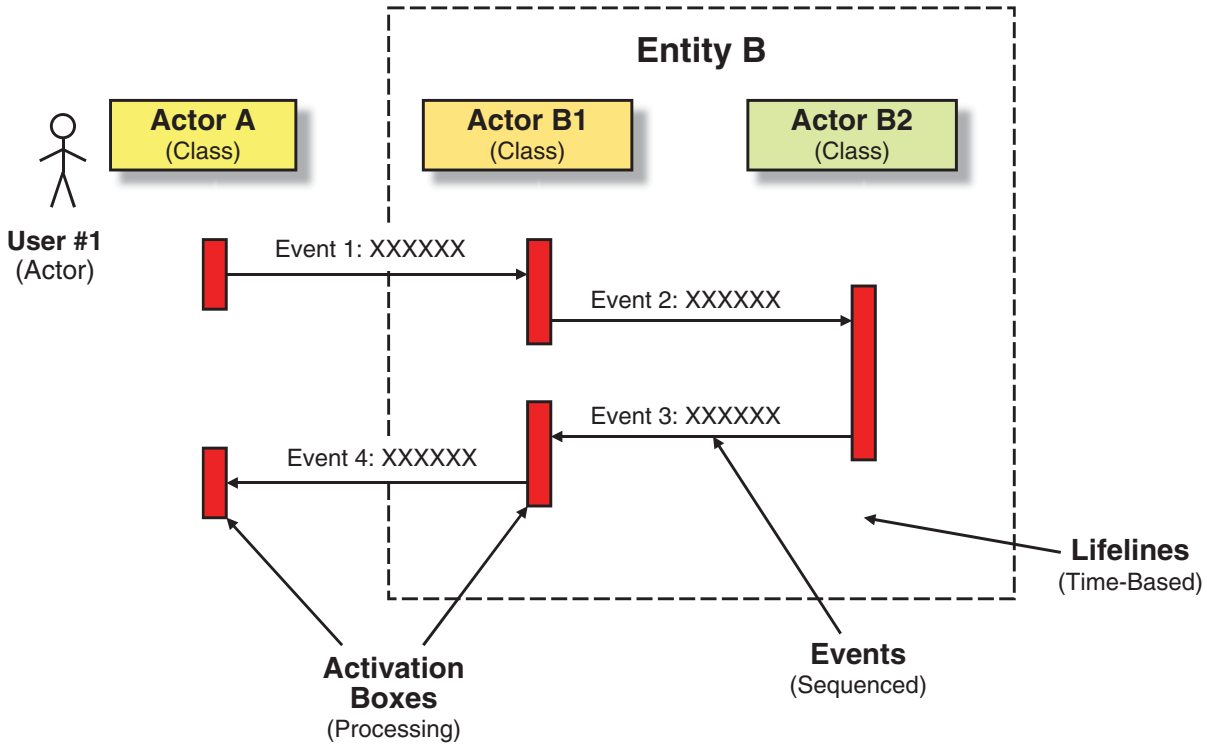


Figure C.4 SysML™ Sequence Diagram Construct

(Table 5.1) for example, might enter the Standby Mode after X minutes of inactivity by a User and remain there until reactivated for normal operation.

C.3.1.3 Activity Diagrams

References: OMG (2012, p. 92–111)

Once we establish the Sequence Diagrams for a system, product, or service, we need to expand the Activation Boxes into actionable activities – operational tasks. We do this via Activity Diagrams such as the construct shown in Figure C.5. Activity Diagrams elaborate in more detail *how* Sequence Diagram Activation Boxes process information acquired via the interactions between Actors to produce the expected UC outcomes. Depiction of these interactions are represented by Control Flow (vertically) and Data Flow (horizontally) in the Activity diagrams as shown in Figure 10.9.

Within the top-down sequential Control Flow of an Activity Diagram, Decision Blocks may direct processing along specific paths depending on conditions. Figure C.6 introduces two additional symbols, a Fork Node and a Join Node.

Using Figure C.5 as a reference, a few of the key attributes of an Activity Diagram include the following:

- **SwimLanes**—Expansions of Actor Lifelines into vertical zones for depicting the sequence of Control Flow and Data Flow activities (Figure 10.9) with internal peer Actors or external System Actors.
- **Initial Node**—Each Actor's processing activities begin with an Initial Node (OMG, 2012, p. 94).
- **Inputs**—Generally represent external stimuli, excitations, cues, or resources each with its own unique identifier such as Input 10.
- **Activities**—Symbolized by rectangular boxes with rounded corners to represent Input–Output (I/O) processing that transforms inputs via mathematical *transfer functions* to produce a specific performance-based outcome—Output XX.
- **Fork Node** (Figure C.6)—Represent *concurrent* processing paths (OMG, 2012, p. 94).
- **Join Node** (Figure C.6)—Represents synchronization - reunification-of one or more Fork Control Flows into a single Control Flow (OMG, 2012, p. 94).
- **Decision Node**—Symbolized by diamonds that represent decisions to be made within a sequential flow of activities. While traditional decision blocks included a question to be answered, SysML™ decisions typically do not contain text. Instead, the decision paths are annotated (OMG, 2012, p. 93). Whereas a traditional process flow decision box (diamond) states a question with conditional "Yes" or "No" branches, SysML™ simply places a "?" inside the decision box and annotates the decision branches with a title.

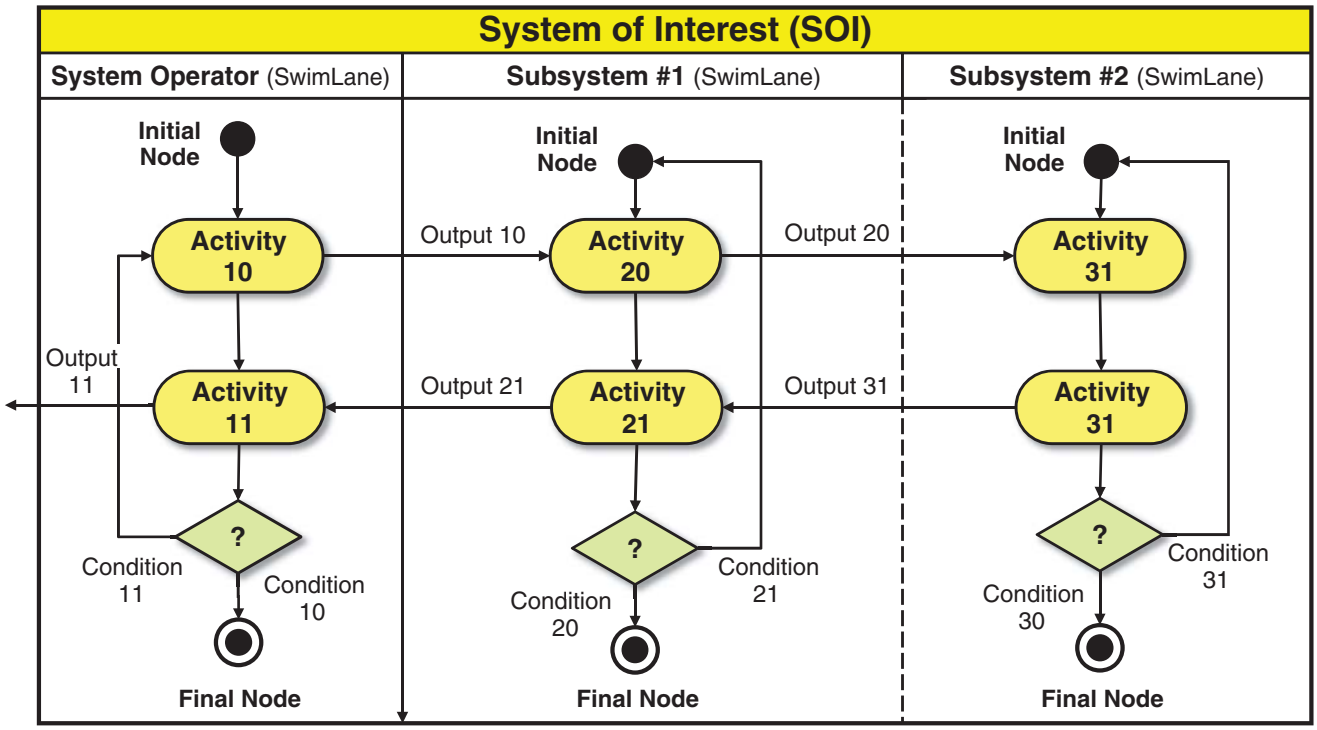


Figure C.5 SysML™ Activity Diagram Construct

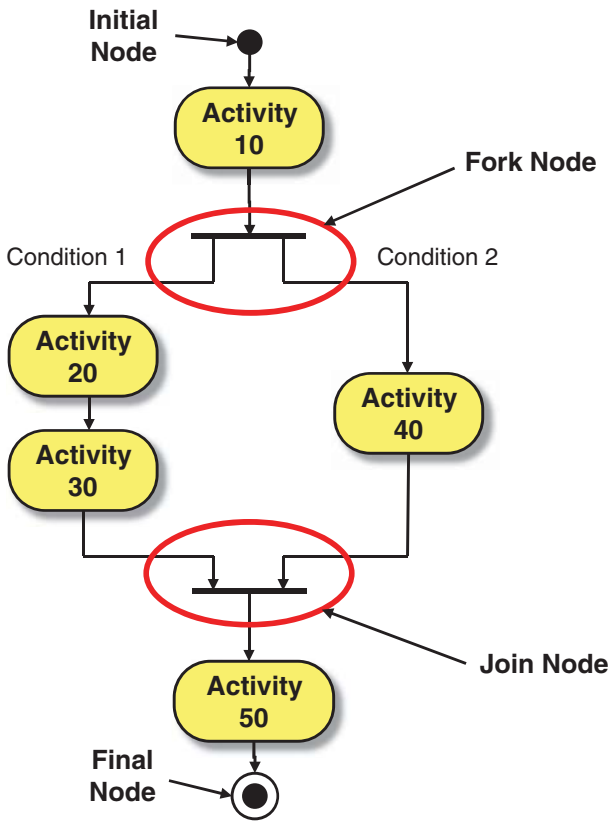


Figure C.6 Activity Diagram Illustrating Fork and Join Nodes

- **Outcomes**—Represent response results produced as an output, each with its own unique identifier such as Output 21.
- **Final Node**—Each Actor's processing activities terminate with a Final Node (OMG, 2012, p. 93).

Author's Note C.2

Inputs and Outputs can have any type of designation you choose. Brevity, however, should be the rule. The generic Output 21 used previously was simply only for reference. Recognize, however, that the output parameter names should be *explicit* such as XYZ Trigger; Location XYZ Coordinates; and so forth.

C.3.1.4 State Machine Diagrams **References:** OMG (2012, p. 119–122)

State machine concepts enable us to characterize processing to achieve a specific set of objective-based outcomes until a *change of state* occurs based on conditions or receipt of an external trigger. Examples include (1) Modes of Operation (Figure 7.7) and (2) States of Operation (Figure 7.5).

A few of the key attributes of a State Machine Diagram include the following:

- **Initial Pseudo State**—The condition on *entry* into a state or operation.
- **Transition Trigger**—A *condition* or *event* that causes a transition from one state to another. Triggers are annotated with the name of the trigger or condition.
- **Final State**—The condition on *completion* of a state of operation.

Since Modes and States remain in the *current* state until triggered to exit to the *next* state, traditionally this condition has been symbolized by a 270° loop with an arrow attached externally to one of the corners of the Mode or State box (Figure 7.5).

C.3.2 Requirement Diagrams

References: OMG (2012, p. 139–153)

SysML™ Requirement Diagrams provide a construct for defining, allocating, and flowing down system, product, or service requirements. In general, Requirements Diagrams are presented in this text using hierarchal structures such as the one shown in Figure 14.3. The collection of Requirements Diagrams form the basis for the Requirements Architecture of a system, product, or service's Requirements Domain Solution addressed in:

- Figure 11.2—Four Domain Solutions.
- Figures 14.1 and 14.2—Develop the Requirements Domain Solution.

C.3.3 Structure Diagrams

The architectural framework of a system, product, or service's and its internal and external interactions drives the need for a construct to depict the Composition by Aggregation (Figure C.1) ERs. This includes interactions among various components at (1) various levels of abstraction – SYSTEM, SUBSYSTEM, ASSEMBLY, SUBASSEMBLY, and PART - and (2) within each level. SysML™ Structure Diagrams provide the construct.

System Engineering *problem-solving* and *solution-development* requires analytical partitioning - decomposition or refinement - of abstract entities into lower level components that are manageable in terms of risk (Principle 4.17). Similarly, we need to represent the hierarchical *partitioning* and *integration*—Product Structure—of a SYSTEM or PRODUCT. So, *how do we depict analytical decomposition and physical integration*? Figure 8.7 is an SE application example.

SysML™ Structure Diagrams include two types used in this text: Block Definition Diagrams (BDDs) and Internal Block Diagrams (IBDs).

C.3.3.1 Block Definition Diagrams (BDDs) **References:** OMG (2012, pp. 32–36, 38–40, 59–61, et al.)

The *composition* concepts of *generalization* and *aggregation* enable us to derive and refine - decompose - a system, product, or service into some meaningful entities that have manageable risk. SysML™ Block Definition Diagrams (BDDs) serve this purpose. Figure C.7 serves as an example of a BDD.

Systems, products, or services represent a System of Interest (SOI) that is composed of at least (1) one or more Mission Systems and (2) one or more Enabling Systems (Figure 9.1). Each Mission System and Enabling System is composed of Systems Elements that include Personnel, Equipment, Mission Resources, Procedural Data, System Responses, and Facilities (Figure 9.2). We illustrate these ERs using the BDD shown in Figure C.7.

Observe the *dashed line* underneath the Mission System in Figure C.7. The dashed—interrupted—line symbolizes that fact that Personnel *may or may not* be part of the Mission System. For example, an intravenous medical device for dispensing prescription drugs as a Mission System obviously does not include Personnel "out of the box" from the manufacturer. However, the device is integrated with the medical staff to form a higher level Intravenous Drug Delivery System that does include Personnel—doctors, nurses, and so forth—and Equipment—medical device and so forth.

If we elaborate a BDD entity into a lower level of detail, we can create another instance such as the one shown in Figure C.8. Observe the block's attributes include the ≪block≫ notation and a title—Power Supply; performance values; operations performed; and inputs and outputs.

C.3.3.2 Internal Block Diagrams **References:** OMG (2012, pp. 37–38, 40–42, 61–62, et al.)

Now that we have established an understanding of the BDD construct, we need to characterize the interactions between its entities. SysML™ provides Internal Block Diagrams (IBDs) for this purpose.

Every system, product, or service as a Mission System interacts with external systems in its Operating Environment. The Operating Environment is composed of the

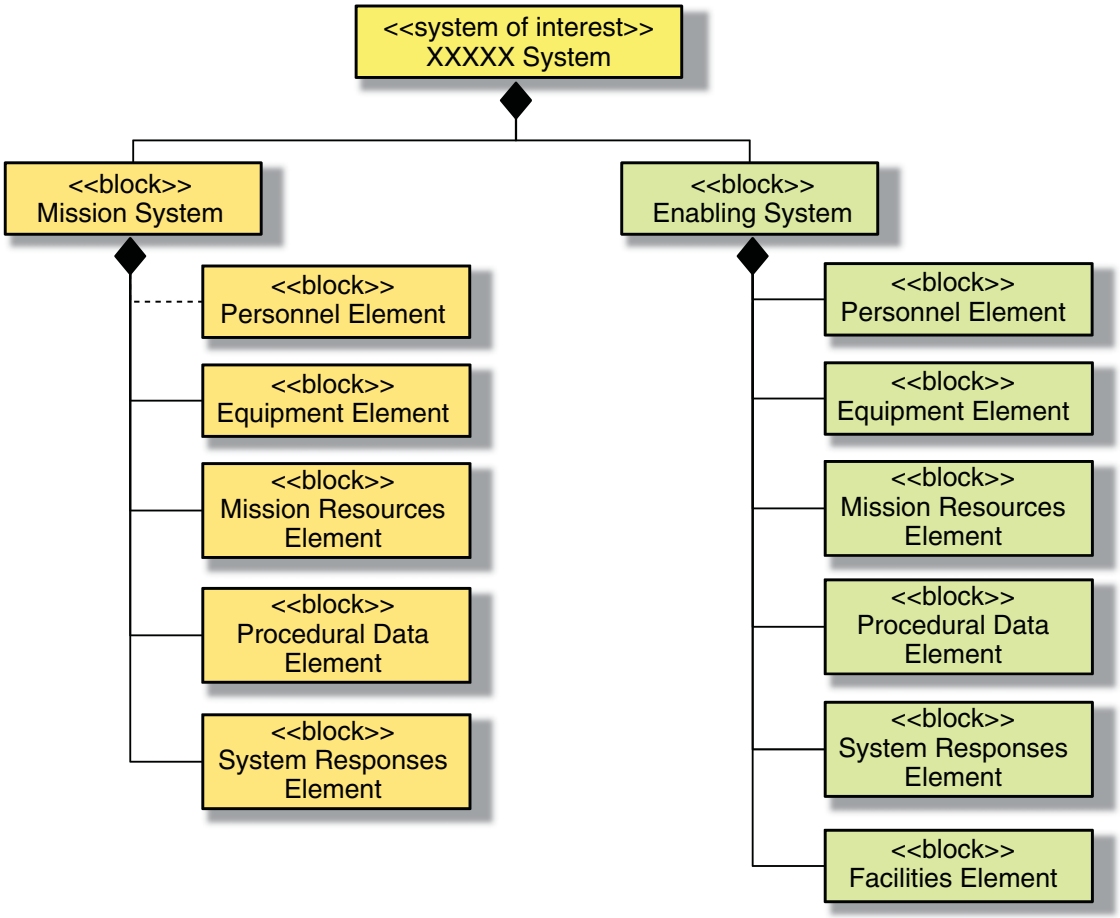


Figure C.7 Block Definition Diagram (BDD) Construct

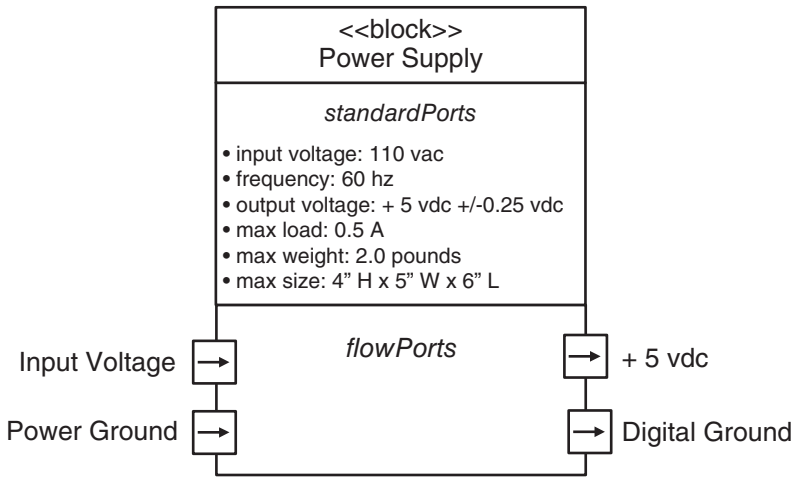


Figure C.8 Detailed Block Definition Diagram (BDD) Construct

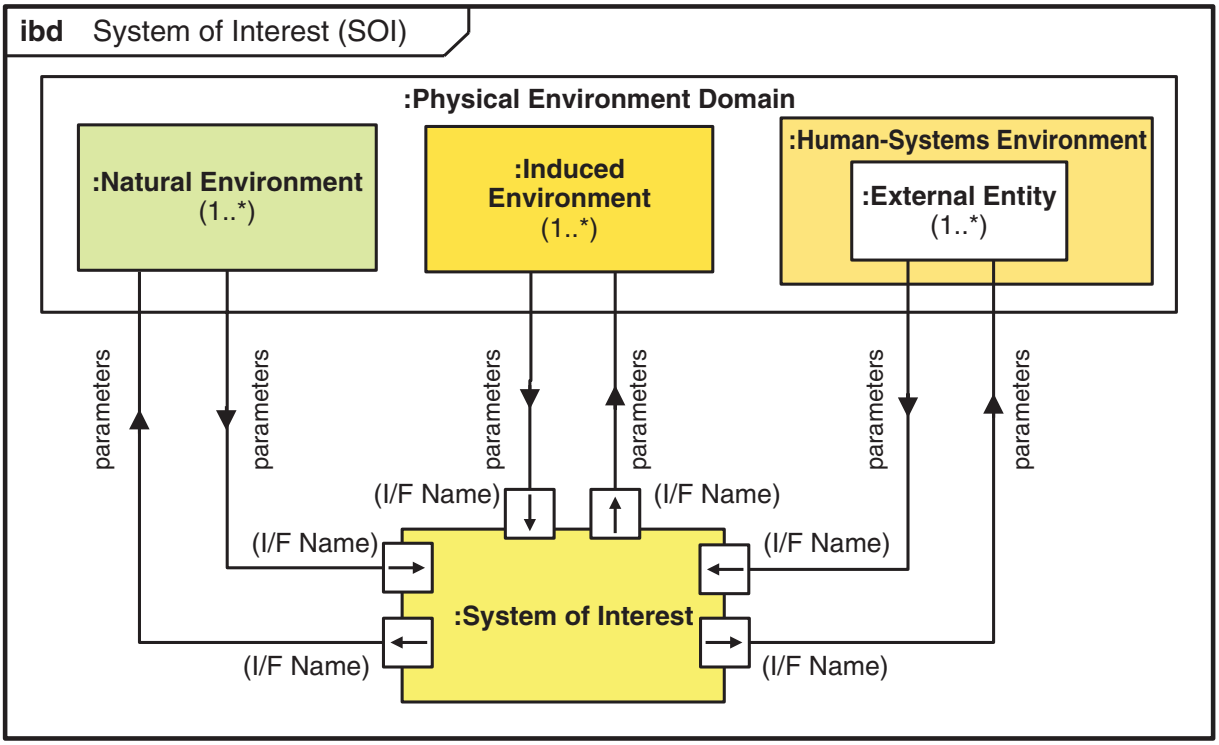


Figure C.9 Internal Block Diagram (IBD) Construct

Physical Environment, which consists of Natural, Induced, and Human-Systems Environments. Figure C.9 provides an example of an IBD depicting these interactions.

A few of the key attributes of an Internal Block Diagram include the following:

- **ibd**—Positioned in the upper left corner identifies the type of diagram.
- **Syntax**—Block titles within the ibd (lower case) consist of a colon followed by the name of the block referred to as a :(Namespace).
- **Range**—Observe that depending on the circumstances, the Natural, Induced, and Human-Systems Environments may consist of *Zero* or *One-to-Many*—0..* or 1..*—entities.

- **Ports**—Include arrows indicating the direction flow of a force, data, or energy.
- **Port Names**—Listed adjacent to the port on the dependency—interaction—line.
- **Arrows**—Along the interactions indicate directional flow and include annotations for parameters being passed.

C.3.4 Diagram Usage

OMG (2012) also introduces a concept for diagram usage. This includes usage of a specific type of diagram such as a Context Diagram (Figure 8.1) as a form of usage of BDDs, IBDs, and UCDs (OMG, 2012, p. 170).

C.4 REFERENCES

Delgatti, Lenny (2013), *SysML Distilled: A Brief Guide to the Systems Modeling Language*, Boston, MA: Addison-Wesley Professional.

Friedenthal, Sanford; Moore, Allan; and Steiner, Rick (2014), *A Practical Guide to SysML, Third Edition: The Systems Modeling Language*, Burlington, MA: The MK/OMG Press - Morgan Kaufmann.

OMG SysML™ (2012), OMG Systems Modeling Language (SysML™), Version 1.3, Document Number: formal/2012-06-01, Needham, MA: Object Management Group (OMG®). Retrieved on 2/23/14 from http://www.omgsysml.org.

INDEX

Note: Page numbers with "*f*" and "*t*" denote figures and tables, respectively, and page numbers in **bold** represent key terms or definitions.

System Engineering Analysis, Design, and Development: Concepts, Principles, and Practices, Second Edition. Charles S. Wasson.

Companion Website: www.wiley.com/go/systemengineeringanalysis2e

WILEY SERIES IN SYSTEMS ENGINEERING AND MANAGEMENT

Andrew P. Sage, Editor

ANDREW P. SAGE and JAMES D. PALMER
Software Systems Engineering

WILLIAM B. ROUSE
Design for Success: A Human-Centered Approach to Designing Successful Products and Systems

LEONARD ADELMAN
Evaluating Decision Support and Expert System Technology

ANDREW P. SAGE
Decision Support Systems Engineering

YEFIM FASSER and DONALD BRETTNER
Process Improvement in the Electronics Industry, Second Edition

WILLIAM B. ROUSE
Strategies for Innovation

ANDREW P. SAGE
Systems Engineering

HORST TEMPELMEIER and HEINRICH KUHN
Flexible Manufacturing Systems: Decision Support for Design and Operation

WILLIAM B. ROUSE
Catalysts for Change: Concepts and Principles for Enabling Innovation

LIPING FANG, KEITH W. HIPEL, and D. MARC KILGOUR
Interactive Decision Making: The Graph Model for Conflict Resolution

DAVID A. SCHUM
Evidential Foundations of Probabilistic Reasoning

JENS RASMUSSEN, ANNELISE MARK PEJTERSEN, and LEONARD P. GOODSTEIN
Cognitive Systems Engineering

ANDREW P. SAGE
Systems Management for Information Technology and Software Engineering

ALPHONSE CHAPANIS
Human Factors in Systems Engineering

YACOV Y. HAIMES
Risk Modeling, Assessment, and Management, Third Edition

DENNIS M. BUEDE
The Engineering Design of Systems: Models and Methods, Second Edition

ANDREW P. SAGE and JAMES E. ARMSTRONG, Jr.
Introduction to Systems Engineering

WILLIAM B. ROUSE
Essential Challenges of Strategic Management

YEFIM FASSER and DONALD BRETTNER
Management for Quality in High-Technology Enterprises

THOMAS B. SHERIDAN
Humans and Automation: System Design and Research Issues

ALEXANDER KOSSIAKOFF and WILLIAM N. SWEET
Systems Engineering Principles and Practice

HAROLD R. BOOHER
Handbook of Human Systems Integration

JEFFREY T. POLLOCK and RALPH HODGSON
Adaptive Information: Improving Business Through Semantic Interoperability, Grid Computing, and Enterprise Integration

ALAN L. PORTER and SCOTT W. CUNNINGHAM
Tech Mining: Exploiting New Technologies for Competitive Advantage

REX BROWN
Rational Choice and Judgment: Decision Analysis for the Decider

WILLIAM B. ROUSE and KENNETH R. BOFF (editors)
Organizational Simulation

HOWARD EISNER
Managing Complex Systems: Thinking Outside the Box

STEVE BELL
Lean Enterprise Systems: Using IT for Continuous Improvement

J. JERRY KAUFMAN and ROY WOODHEAD
Stimulating Innovation in Products and Services: With Function Analysis and Mapping

WILLIAM B. ROUSE
Enterprise Tranformation: Understanding and Enabling Fundamental Change

JOHN E. GIBSON, WILLIAM T. SCHERER, and WILLAM F. GIBSON
How to Do Systems Analysis

WILLIAM F. CHRISTOPHER

Holistic Management: Managing What Matters for Company Success

WILLIAM B. ROUSE

People and Organizations: Explorations of Human-Centered Design

MO JAMSHIDI

System of Systems Engineering: Innovations for the Twenty-First Century

ANDREW P. SAGE and WILLIAM B. ROUSE

Handbook of Systems Engineering and Management, Second Edition

JOHN R. CLYMER

Simulation-Based Engineering of Complex Systems, Second Edition

KRAG BROTBY

Information Security Governance: A Practical Development and Implementation Approach

JULIAN TALBOT and MILES JAKEMAN

Security Risk Management Body of Knowledge

SCOTT JACKSON

Architecting Resilient Systems: Accident Avoidance and Survival and Recovery from Disruptions

JAMES A. GEORGE and JAMES A. RODGER

Smart Data: Enterprise Performance Optimization Strategy

YORAM KOREN

The Global Manufacturing Revolution: Product-Process-Business Integration and Reconfigurable Systems

AVNER ENGEL

Verification, Validation, and Testing of Engineered Systems

WILLIAM B. ROUSE (editor)

The Economics of Human Systems Integration: Valuation of Investments in People's Training and Education, Safety and Health, and Work Productivity

ALEXANDER KOSSIAKOFF, WILLIAM N. SWEET, SAM SEYMOUR, and STEVEN M. BIEMER

Systems Engineering Principles and Practice, Second Edition

GREGORY S. PARNELL, PATRICK J. DRISCOLL, and DALE L. HENDERSON (editors)

Decision Making in Systems Engineering and Management, Second Edition

ANDREW P. SAGE and WILLIAM B. ROUSE

Economic Systems Analysis and Assessment: Intensive Systems, Organizations, and Enterprises

BOHDAN W. OPPENHEIM

Lean for Systems Engineering with Lean Enablers for Systems Engineering

LEV M. KLYATIS

Accelerated Reliability and Durability Testing Technology

BJOERN BARTELS , ULRICH ERMEL, MICHAEL PECHT, and PETER SANDBORN

Strategies to the Prediction, Mitigation, and Management of Product Obsolescence

LEVANT YILMAS and TUNCER ÖREN

Agent-Directed Simulation and Systems Engineering

ELSAYED A. ELSAYED

Reliability Engineering, Second Edition

BEHNAM MALAKOOTI

Operations and Production Systems with Multiple Objectives

MENG-LI SHIU, JUI-CHIN JIANG, and MAO-HSIUNG TU

Quality Strategy for Systems Engineering and Management

ANDREAS OPELT, BORIS GLOGER, WOLFGANG PFARL, and RALF MITTERMAYR

Agile Contracts: Creating and Managing Successful Projects with Scrum

KINJI MORI

Concept-Oriented Research and Development in Information Technology

KAILASH C. KAPUR and MICHAEL PECHT

Reliability Engineering

MICHAEL TORTORELLA

Reliability, Maintainability, and Supportability

YACOV Y. HAIMES

Risk Modeling, Assessment, and Management, Fourth Edition

CHARLES S. WASSON

System Engineering Analysis, Design, and Development: Concepts, Principles, and Practices, Second Edition

WILEY END USER LICENSE AGREEMENT

Go to www.wiley.com/go/eula to access Wiley’s ebook EULA.