INTERNATIONAL COUNCIL ON SYSTEMS ENGINEERING

SYSTEMS ENGINEERING HANDBOOK

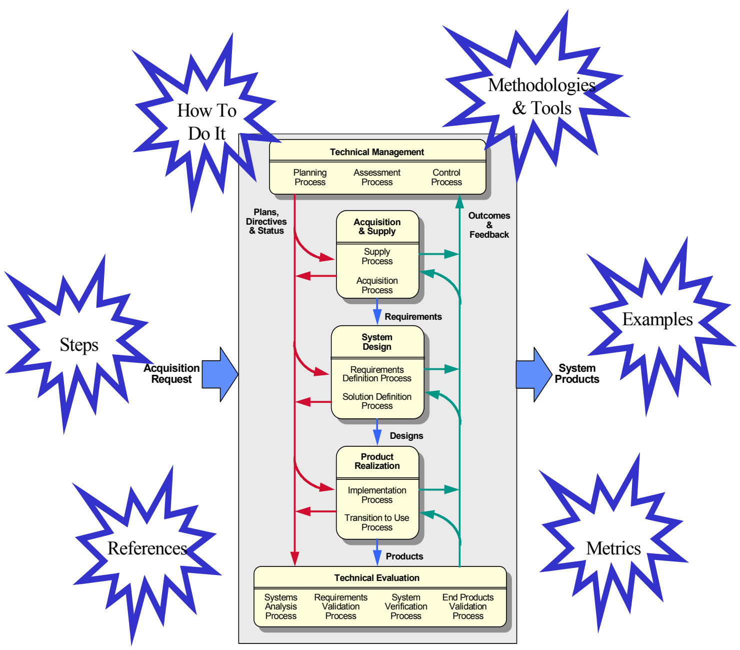
A “WHAT TO” GUIDE FOR ALL SE PRACTITIONERS

INCOSE-TP-2003-016-02, Version 2a, 1 June 2004

Released by:

Technical Board

International Council on Systems Engineering (INCOSE)



This INCOSE Technical Product was prepared by the Systems Engineering Handbook Working Group of the International Council on Systems Engineering (INCOSE). It is approved by the INCOSE Technical Board for release as an INCOSE Technical Product.

Preface

This document has been prepared and produced by a volunteer group of contributors within the International Council on Systems Engineering (INCOSE): the Systems Engineering (SE) Handbook Working Group (SEH WG). The original document was based on inputs from numerous INCOSE contributors, was edited by Dorothy McKinney and published in draft form in June 1994 for internal INCOSE review. Version 1 incorporated INCOSE comments, was edited by Tim Robertson, and released and published in January 1998 to the INCOSE web site. The next update (Version 2.0) incorporated changes to conform to ANSI/EIA-632 and EIA-731 as well as to include new contributed material. This document was edited by Jim Whalen, Richard Wray, and Dorothy McKinney. It was completed in July 2000 and released for distribution in early 2002.

This current version (Version 2.0a) includes grammatical corrections and a restructuring of the document into smaller sections to facilitate readability and use of the handbook as a study guide for INCOSE SE certification testing. No new material has been inserted for this release.

Approved:

Terje Fossnes
Terje Fossnes, Chair, SEH WG

Kevin Forsberg
Kevin Forsberg, Co-Chair, SEH WG

Richard Wray
Richard Wray, Co-Chair for Products Development

Gerard Fisher
Gerard Fisher, Chair, Process & Improvement Technical Committee

William Mackey
Chair, INCOSE Technical Board

THIS PAGE
INTENTIONALLY
LEFT BLANK

TABLE OF CONTENT

LIST OF FIGURES AND TABLES

1 SCOPE

1.1 OBJECTIVE FOR THIS DOCUMENT

The objective for this document is to provide a description of the key process activities performed by Systems Engineers. The purpose for each process activity, what needs to be done, and how it can be done is described in some detail. The intended audience is primarily the new Systems Engineer, an engineer in another discipline that needs to perform some Systems Engineering functions or a more-experienced Systems Engineer who needs a convenient reference. The intent is to provide enough information for the user to determine whether a given process activity is appropriate in supporting the objective(s) of the program or project they support, and how to go about implementing the process activity.

The process activities which are described are applicable to most engineering projects. The appropriate resources, including manpower and schedule time, devoted to any process activity should be based on cost/benefit considerations. Anecdotal and "lessons learned" experience from some large programs indicates that serious problems were caused by insufficient Systems Engineering. However, Systems Engineering is not advocated as a universal solution to all program problems. Rather, this handbook attempts to describe the purpose and value of specific Systems Engineering process activities, together with some guidance to help determine when each activity is complete.

The intent of the descriptions in this handbook is to show what each Systems Engineering process activity entails, including the need to design for affordability as well as performance. On some projects, a given activity may be performed very informally (e.g., on the back of an envelope, or in an engineer's notebook), or very formally, with interim products under formal baseline control. This document is not intended to advocate any level of formality as necessary or appropriate in all situations. On each program or project, the appropriate degree of formality in the execution of any Systems Engineering process activity is determined by:

a. the need for communication of what is being done (across members of a project team, across organizations, and/or over time to support future activities), and

b. the level of risk that is acceptable.

On smaller programs/projects, where the span of required communications is small (few people and short project/product life cycle) and the cost of redesign is low, Systems Engineering activities can be conducted very informally (and thus at low cost). On larger programs, where the cost of failure or redesign is high, increased formality and depth in the Systems Engineering activities can significantly mitigate program risk.

The reader may encounter difficulty in understanding some terminology. We have attempted to use the most standard terminology. However, there is no accepted universal terminology standard. One of the principal areas of terminology difference is program phase. A comparison of US Department of Defense (DoD), other government, and commercial program phases is given in Section 3. This should be helpful when phase terminology is encountered which is one of the principal areas of terminology differences.

1.2 MOTIVATION FOR CREATING VERSION 2A

Version 2.0 was successful in that it has received wide use and acceptance, and a number of corporations have built their internal Systems Engineering processes around its framework. INCOSE structured its Certified Systems Engineering Professional examination around Version 2.0. There are recognized shortcomings, however. Many readers felt it is too United States Department of Defense (DoD) centric, and it showed little awareness of international issues. Also, the document treated topics with inconsistent degrees of depth, and it has been criticized for being too lengthy in several sections.

The development of a totally new handbook, SE Handbook version 3.0, is in its early stages, with a schedule leading to release in the latter half of 2005. The goal is to create a focused treatment of the SE process in a document of 100 to 150 pages. Appendices will be used to elaborate on significant topics. The appendices will be integrated into the INCOSE Guide to the Systems Engineering Body of Knowledge (G2SEBOK). It is the intent of the SE Handbook Version 3.0 Development Team to follow closely the ISO 15288 Standard where it is prudent and reasonable.

In the fall of 2002 the INCOSE Board of Directors authorized the Certification Working Group (CWG) to develop a plan for certifying Systems Engineering professionals. Certification will be based on four primary considerations: 1- work experience doing Systems Engineering, 2- technical education (with the path of added experience compensating for lack of a technical degree), 3- references from professionals in the field who know the applicant's Systems Engineering performance, and 4- evidence of understanding of the fundamentals of Systems Engineering as demonstrated by passing a written exam. With approval of the Board, the CWG contracted with Prometric, Inc., to assist in developing the program. Prometric is an international corporation with extensive experience in helping technical societies as well as major corporations establish certification programs, and they have over 6,000 computerized test sites world-wide to facilitate applicants taking certification exams.

The CWG decided to use the SE Handbook version 2.0 (SEHv2) as the source document in the creation of the questions that constituted the 30 June 2003 Beta Test for the certification exam. Other documents were considered, including the INCOSE Guide to the System Engineering Body of Knowledge. However, only the SEHv2 met the Prometric guidelines for a complete, definitive reference which would be suitable for a legally defensible certification reference document.

After processing the Beta Test results, Prometric, together with the certification working group in September 2003, selected the final set of questions that will be used in the certification exam. The tests are now ready for distribution throughout the world-wide Prometric system of test centers, as soon as the INCOSE Board of Directors gives its final approval to the balance of the program. Those who have passed the Beta Test are in the process of submitting the balance of their applications in June 2004. The open enrollment is scheduled to begin in the fall of 2004.

Although the exam questions were written based on the content of Version 2.0, many felt that a better study guide had to be published for the benefit of certification candidates. It was quickly determined that accelerating the development of Version 3.0 was not reasonable, and if attempted it could have a serious impact on quality. Thus the Board of Directors accepted an offer from the SEHWG for a fast-track effort to create an interim handbook, version 2a, with the following four constraints:

- Shorten the main body of text from 308 pages to 200 pages or less,
- Retain all material that supports each individual Certification exam question,
- Reduce DoD-centric material where-ever possible, and
- Introduce no new material.

The effort was started in November 2003, with the first draft completed at the end of January, and final reviews of Version 2.0a were completed on 10 May. Many thanks to the reviewers for their rapid turn-around of the final draft. We are sorry that we could not accept a number of requests to update figures and nomenclature to the most recent release of NASA, DoD, or EIA standards, but in most instances the reviewer's request violated bullets 2 and 4 above. In all instances the basic concepts remain sound as presented, even though a term might have been changed in recent usage (e.g., it was suggested that the acronym for the System Engineering Management Plan, or SEMP, be changed to SEP, per recent DoD usage). The SEMP remains as it was in Version 2.0.

This interim handbook, Version 2a, will be replaced by the next generation, Version 3.0, in the fall of 2005. The objective of Version 3 is to create a solid handbook from which corporations can develop SE processes. It will also serve as a study guide for the fundamentals of Systems Engineering certification and the source for new certification exam questions as the exam is updated in the future. It is not the intent to create a Version 3.0 which is written to support the existing exam questions, however. On the contrary, the certification exam will evolve to match the new handbook and other available information sources. As the new handbook is written, new questions will be proposed such that replacement questions can be pre-tested and considered ready for use upon approval by the Certification team in the fall of 2005.

1.3 RELATIONSHIP TO SE STANDARDS

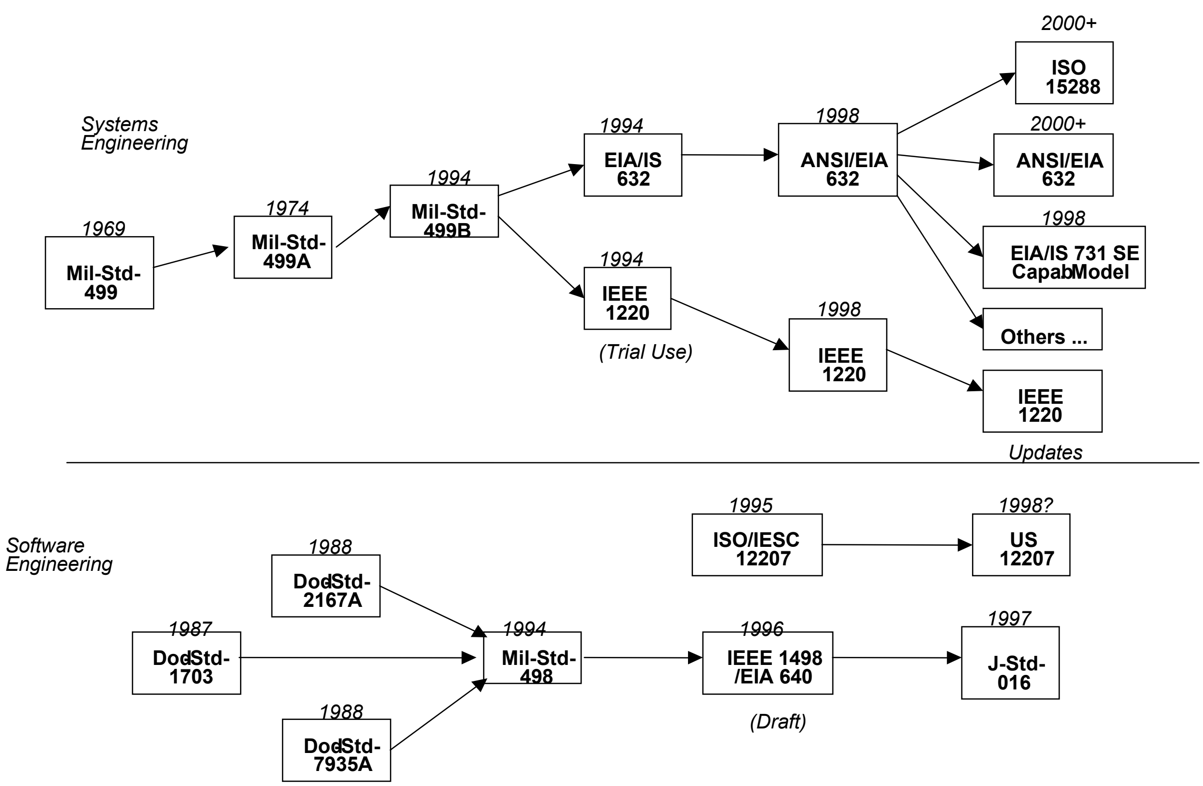


Figure 1-1. Heritage of Systems Engineering Standards

Since the late 1960s, the need for Systems Engineering Process Standards has been recognized. Figure 1-1 shows the heritage of Systems Engineering Standards and their closely related Software Engineering Standards. The effort to upgrade Systems Engineering military standards (MIL-STD) and US DoD standards (DoD-STD), such as MIL-STD-499 culminated with a draft of MIL-STD-499B, which was never issued, because of changes in US DoD procurement policies. The American National Standards Association (ANSI), Electronic Industries Alliance (EIA) and International Electronic and Electrical Engineers (IEEE) standards groups picked up the effort with an ANSI/EIA-632 Processes for Engineering a System issued in 1998. This handbook has been written to serve as a stand-alone reference for Systems Engineering processes in conformance with ANSI/EIA-632 and the related EIA/Interim Standard (IS) 731 Systems Engineering Capability Model. There is no intent to specify what "should" be done on a program. Rather, the focus is on what "needs" to be done and how to do it, in order to successfully implement each Systems Engineering activity. With this focus, this INCOSE handbook should remain a useful reference and complement the Systems Engineering standards.

1.4 ORGANIZATION OF THE HANDBOOK

This document is organized into twelve sections. These sections, and the primary purpose of each, are:

- **Section 1 – Scope:** (this section) describes the background, purpose, and organization of the handbook, and lists the published reference documents for this material.

- **Section 2 – Systems Engineering Overview:** gives a short overview of Systems Engineering, including its evolution.

- **Section 3 – Mapping the SE Process into Product Development Cycles:** describes the relationship of the Systems Engineering activities that are the focus of this document to the larger context in which these activities are performed, including the program and project life cycle. Several government and commercial project life cycles are compared.

- **Section 4 – Process Activities:** contains the primary content of this document: the description of each Systems Engineering process activity. Many of the activities have been hierarchically decomposed into lower-level activities. Where appropriate, a summary description is given of the nature and purpose of the higher-level activity, but the detailed descriptions are given for the lowest level activities.

- **Section 5 – SE Technical Management:** describes how process activities can be tailored to meet the needs of different programs and projects.

- **Section 6 – Risk Management:** describes steps that should be taken to identify potential sources of risk; quantify risks; determine sensitivity of risks; determine and evaluate alternative approaches; ensure risk is factored into decisions; and take actions to avoid, control, assume, or transfer each risk.

- **Section 7 – Organizational Practices:** describes organizational issues, such as Concurrent Engineering and Integrated Product Teams

- **Section 8 – Requirements Definition Process:** describes approach to identifying requirements from user requirements to system requirements.

- **Section 9 – Functional Analysis:** establishes what the system must do.

- **Section 10 – System Architecture Synthesis:** establishes alternative concepts and applies criteria to select the “best” among a number of competing candidates.

- **Section 11 – Manufacturing and Producibility Analysis:** focuses on issues related to system effectiveness.

- **Section 12 – Integration, Verification, and Validation:** discusses why these topics must not be relegated to the end of the project, but rather how vital it is that Systems Engineers consider Integration , Verification, and Validation from the earliest concept development activities through to system delivery.

- Appendix A contains a discussion of Quality Function Deployment (QFD) and its application to Systems Engineering.
- Appendix B contains a detailed discussion of Human Systems Engineering.
- Appendix C provides an annotated outline for a Systems Engineering Management Plan (SEMP), including Systems Engineering Schedules.
- Appendix D provides details on Methods That Support Functional Analysis and Allocation
- Appendix E contains a Glossary and Definitions of key terms used throughout this volume.
- Appendix F lists (or spells out) the Acronyms used in this volume.
- The last page provides a comments form for use with this handbook.

The authors had originally considered providing an appendix containing vendor self-assessments of their requirements management tools. However, since these assessments quickly become obsolete, the reader is instead referred to the INCOSE World Wide Web site, which attempts to maintain current information. The site can be reached at URL: http://www.incose.org/

1.5 REFERENCE DOCUMENTS

1. ANSI/EIA-632, "Processes for Engineering a System." January 7, 1999.

2. ANSI/EIA-731, “Systems Engineering Capability.” September 1998.

3. *AT&T Engineering Guides for Managing Risk*, McGraw-Hill Professional Book Group (1-800-842-3075):
   - “Design to Reduce Technical Risk”
   - “Design's Impact on Logistics”
   - “Moving a Design into Production”
   - “Testing to Verify Design and Manufacturing Readiness”

4. Bicknell, Barbara A. and Bicknell, Kris D., *The Road Map to Repeatable Success: Using QFD to Implement Change.* CRC Press, Boca Raton, Florida, 1995.

5. Blanchard, B.S. and Fabrycky, W. J., *Systems Engineering and Analysis* (2nd ed.). Prentice-Hall, Englewood Cliffs, N.J., 1990

6. Brassard, M., "The Memory Jogger Plus +." Goal/QPC, 1989.

7. Defense Systems Management College (DSMC), "Systems Engineering Fundamentals." Fort Belvoir, Virginia, December 1999.

8. Eisner, Howard, *Essentials of Project and Systems Engineering Management*, 2nd Edition, John Wiley & Sons, New York, NY, 2002

9. European Cooperation For Space Standardization (ECSS), “Space Engineering Systems Engineering Standard.” ECSS–E–10A, ECSS Secretariat, ESA–ESTEC Requirements & Standards Division, Noordwijk, The Netherlands, 19 April 1996

10. Grady, Jeffry O., *Systems Requirements Analysis*, McGraw Hill, New York, NY, 1993.

11. Hall, Arthur D., *Methodology for Systems Engineering*, Van Nostrand, Princeton, N.J., 1962.

12. IEEE Std 1220-1994, "IEEE Trial-Use Standard for Application and Management of the Systems Engineering Process." February 1995 (Trial period extends until February 28, 1997). ISBN 1-55937-496-9. To order call 1-800-678-IEEE (in USA and Canada); others: 1-908-981-1393. FAX orders 1-908-981-9667. (Inserted alphabetically, by author)

13. Lacy, J. A., *Systems Engineering Management* McGraw Hill, 1992.

14. Mar, B. W., "Systems Engineering Basics," The Journal of NCOSE, July-Sept. 1994.

15. NASA SP-6105, "NASA Systems Engineering Handbook.," June 1995.

16. NAVSO P6071, “Best Practices: How to Avoid Surprises in the World's Most Complicated Technical Process.” March 1986. (Inserted alphabetically, by author)

17. Oliver, David W.; Kelliher, Timothy P.; and Keegan, James G.; *Engineering Complex Systems with Models and Objects,* McGraw Hill. 1997.

18. Rechtin, E., *System Architecting* Prentice-Hall, Englewood Cliffs, N. J. 1991

19. Sage, A. P., *Systems Engineering* John Wiley & Sons, Inc., Somerset, N.J., 1992.

20. Shaw, T. E. and Lake, J. G., Ph.D., “Systems Engineering: The Critical Product Development Enabler.” APICS Conference, April 1993.

21. Stevens, Richard; Brook, Peter; Jackson, Ken; and Arnold, Stuart; *Systems Engineering: Coping with Complexity*, Prentiss Hall – Europe. 1998.

22. US DoD 4245.7-M, “Transition from Development to Production.” September 1985.

23. US DoD Directive 5000.2R, “”Mandatory Procedures for Major Defense Acquisition Programs (MDAPs) and Major Automated Information System (MAIS) Acquisition Programs,” January 4, 2001.

Additional references follow some sections.

1.6 ACKNOWLEDGMENTS

The development of this material has been significantly aided by the generous permission of several corporations to share material that was developed for in-house use. These companies include Ascent Logic, CSM, GTE (now part of General Dynamics), Lockheed Martin, Raytheon Systems, TRW, and Loral (now part of Lockheed Martin). Although no material has been taken unchanged from any company’s documents, the company information was a very useful starting point for many of the sections in this document.

The real credit for the development of this volume goes to a small band of patient, persistent, and enthusiastic volunteers in the San Francisco Bay Area Chapter of INCOSE who gave up many lunch hours, and put in many evening and weekend hours leading up to the June 1994 draft. Their willingness to take the risk of being incomplete or wrong in the eyes of their peers from other companies, in describing what they all do for a living, was tremendous. And the willingness of the entire team to bring many divergent points of view to the review of each section, to participate in lively debates, and yet to reach consensus eventually, enabled the June 1994 effort to come to closure. Release 1 was issued in January 1998 and was a substantial extension and revision of the June 1994 draft. This work was done in large part by Tim Robertson. This update provides compatibility with the latest standards, and incorporates contributed additional material and updates to the existing material. Much of the work in the second update has been done by Jim Whalen and Richard Wray.

It would be difficult to accurately characterize the specific contributions of each of the approximately forty-seven volunteers--writers, reviewers, and rewriters. Many served multiple roles. We extend sincere appreciation to, in alphabetical order: Mack Alford, Ed Chan, John Cox, Bill Cutler, Ted Dolton, Melissa Dugger, David Ehlers, Patty Ehlers, Joseph Fiskel, Jack Fisher, Dick Folkerth, Kevin Forsberg, Renee Freedman, Tom Harley, Rick Harwell, Lawrence Hauben, Jack Houle, Randy Jackson, Scott Jackson, Tom Jackson, Tom Keaton, Ed Kujawski, Lew Lee, William Mackey, Brian Mar, Fred Martin, Burt Masnick, Dick Mayer, Dorothy McKinney, Barney Morais, Ron Morris, Ron Olson, Chris Parker, Kevin Patrick, Dave Preklas, Al Reichner, Jeff Riggs, Terry Robar, Tim Robertson, Gary Roedler, Don Rucker, Doug Sailor, Rick Schinella, Tom Shaw, Don Sifferman, Susan Shreve, George Vlay, Rhonda Wentzel, Jim Whalen, E. Richard Widmann, John Winters and Richard Wray. Our apologies if we accidentally left anyone out. Our special thanks go to Dr. Donna Rhodes, INCOSE Past President and past Chair of the Technical Board, for her many useful contributions during the INCOSE review and publication process.

Grateful appreciation is extended to the following for permitting us to excerpt from their materials: Mack Alford, Alford Enterprises, for Functional Thread Analysis; Jonathan Hodapp and Scott Hyer, Johns Hopkins APL, for Modeling & Simulation; Yngvar Tronstad, Ascent Logic Corp., for Object-Oriented Systems Engineering; Joseph Fiskel, Decision Focus, for integrating the excellent June 1994 version of Risk Management; and John Winters, Basic Commerce & Industries Inc, for Human Systems Engineering.

We want to thank the 28 reviewers of Version 2a for their generous support and thoughtful comments. We extend sincere appreciation to two teams of reviewers: John Clark and his team at the Hampton Roads Area Chapter of INCOSE (William Cochran, Dan Fontaine, Richard Kitterman, Matthew Needy, Bill Olson, and Kevin Topp), and John Muehlbauer and his AIAA committee on Systems Engineering (Jimmy Allison, Michelle Bailey, Jerry Baldwin, Lauren Crews, John Day, John Eiler, Curt Farrell, Mike McCoy, Brooks Nolan, William Shields, and Jim Sturges). In addition we extend sincere appreciation to individual reviewers, in alphabetical order: Gerard Fisher, David Hall, Cecilia Haskins, Michael Krueger, William Mackey, Paul Robitaille, William Schoening, Jonette Stecklein and Richard Wray. We apologize in advance if we omitted anyone from this list in the final minutes before going to the printer.

The editors of Version 2a assume responsibility for any errors or misstatements of material drawn from previous releases and new submittals. Any errors introduced in the process are ours, not theirs.

THIS PAGE
INTENTIONALLY
LEFT BLANK

2 SYSTEMS ENGINEERING OVERVIEW

This section traces some of the key developments and rationale that led to Systems Engineering as it is today - a powerful approach to organizing and conducting complex programs. Systems Engineering is still evolving toward stronger commercial and team-based engineering organizations. This section gives a brief historical overview of Systems Engineering; defines key systems and Systems Engineering terms; discusses key Systems Engineering functions across the project life cycle; outlines the basic tasks performed by Systems Engineers; and also discusses the benefits of applying the Systems Engineering approach to a program.

2.1 ORIGIN AND EVOLUTION OF SYSTEMS ENGINEERING

Prior to World War (WW) II, architects and civil engineers were, in effect, the Systems Engineers of their time, on large, primarily civil engineering projects such as: the Egyptian pyramids, Roman aqueducts, Hoover Dam, the Golden Gate Bridge, and the Empire State Building. Other architects designed trains and large ships. Despite their successes, these early Systems Engineers operated without any documented theory or science of Systems Engineering or any defined and consistently-applied processes or practices.

During WW II a project manager and chief engineer could oversee the development of an aircraft program if assisted by leaders for key subsystems, such as propulsion, controls, structure, support systems, etc. Some additional elements of Systems Engineering, such as operations research and decision analysis, gained prominence during and after WW II. Today, with more complex requirements and systems, the chief engineer uses a Systems Engineering team to help develop requirements and to interact with all the project teams.

Systems Engineering began to evolve as a branch of engineering during the late 1950's. During this time both the race to space and the race to develop missiles with nuclear warheads were considered absolutely essential for national survival. Extreme pressure was placed on the military services and their civilian contractor teams to develop, test, and place in operation nuclear tipped missiles and orbiting satellites. Tools and techniques were developed to support system performance (mission success), and project management (technical performance, delivery schedule, and cost control).

Systems Engineering was also evolving in parallel in the commercial sector. Arthur Hall, with an AT&T communications background, published an early book on Systems Engineering in 1962.

Engineering management evolved and standardized the use of specifications, interface control documents, design reviews, and formal change control. The advent of the computer permitted extensive simulation and evaluation of systems, subsystems, and components; thus accurate synthesis of system elements and design trade-offs became possible.

Many lessons were learned from difficulties and failures. These lessons led to innovations in practices in all phases of high technology product development, including all phases of engineering, procurement, manufacturing, testing, and quality control. A driving force for these innovations was attainment of high system reliability. Some examples of changes introduced during the period are:

1. Change control. Designs, manufacturing, and testing processes were sometimes informally changed to "improve" the product, without updating drawings or process descriptions or fully disclosing the changes. When failures occurred, it was difficult to trace the causes. This led to more careful procedures, by all affected groups, to document, review, and approve changes in advance. In most organizations, formal change control boards were established.

2. Improved product accountability. Mass production techniques, with each worker focusing on only a few items, left no one responsible and accountable for individual, high value-added products. Although the proper reports may have been issued, action may not have been taken. Often critical parts, software, or tests were not available on schedule, and costly delays resulted. This led to the establishment of product managers to ensure that all parts were available when needed and that all tests were conducted properly.

3. Formal interface control. Without early definition and strict control of interfaces between components, subsystems, and system elements, the individual elements were delivered which, while performing their task, would not operate in the overall system. While some programs recognized this from the outset, others did not. This resulted in chaos during integration tests, as teams worked round-the-clock to fix the incompatibilities. At times, it was too late, resulting in major program delays or outright cancellations.

The Systems Engineering processes, which have evolved since the 1950's, encompass techniques to address potential problems represented by the five above examples plus many hundreds of others.

In its present (and still evolving) form, Systems Engineering integrates elements of many disciplines such as system modeling and simulation, decision analysis, project management and control, requirements development, software engineering, specialty engineering, industrial engineering, specification writing, risk management, interpersonal relations, liaison engineering, operations analysis, and cost estimation. Any one Systems Engineer is not expected to be expert in all of the above disciplines. However, over the years, a competent Systems Engineer gains experience in most of them.

Systems Engineering is an overarching discipline, providing the tradeoff analyses and integration between system elements to achieve the best overall product and/or service. Although there are some important aspects of project management in the Systems Engineering process, it is still much more of an engineering focus than a management discipline. Systems Engineering has a very quantitative approach, involving tradeoff, optimization, selection, and integration of the products of many engineering disciplines.

2.2 WHAT IS A SYSTEM?

A system can be broadly defined as an integrated set of elements that accomplish a defined objective. People from different engineering disciplines have different perspectives of what a "system" is. For example, software engineers often refer to an integrated set of computer programs as a "system." Electrical engineers might refer to complex integrated circuits or an integrated set of electrical units as a "system." As can be seen, "system" depends on one's perspective, and the "integrated set of elements that accomplish a defined objective" is an appropriate definition.

Some examples of large-scale systems from a Systems Engineer's perspective are:

1. The international air traffic control system is comprised of electronics on-board each aircraft designed to communicate with ground stations at all international air traffic control centers. The system also includes training of flight and ground crews to use specific commands for taxi, take-off, flight path adjustment, and landing instructions in English, design of runways and runway markings at international airports to meet a common international standard, implementation of airport security systems to meet international standards, provision of ground support equipment to service all aircraft that are planned to use each airport.

2. NASA's Apollo lunar landing system was comprised of the launch vehicles, various upper stage modules to accomplish lunar orbit rendezvous, descent and return from the lunar surface, earth return, reentry, and recovery. The system also includes mission and support crews, launch vehicle assembly and checkout equipment, crew training and many support organizations and their facilities (which might be shared with other systems), such as downrange tracking and communications relay stations, and mission control.

It should be self-evident that on large systems, such as the above, methodologies and techniques would need to be used to help all the elements and subsystems work closely together. Flawless performance was and is required of both systems. So the projects evolved a Systems Engineering and management philosophy that maximized their chances of success. But what about smaller systems, can they profit from the use of the same methodologies and techniques? First, some examples of smaller systems:

1. A computer system network, including multiple servers, terminals, printers, network links, software, users, and support systems, including maintenance and repair, training, and spare parts. All these elements are essential for the computer network system to function.

2. A typical 35 mm camera system, consisting of interchangeable lenses and filters, the lens focusing mechanism, camera body, view finder/range finder, flash subsystem, film advance/rewind (or memory status for digital systems), electrical subsystem and power source(s), light meter with shutter/exposure controls, carrying case, film or digital memory cards, and support elements, including photographic paper, film or digital processing materials and equipment, repair and parts suppliers.

Even on smaller systems, such as the last two examples, Systems Engineering techniques will prove useful in rapidly developing and deploying low cost, reliable, high performance, maintainable systems which meet user (customer) needs.

It is sometimes confusing as to which elements comprise a system. This depends entirely upon the focus of the one defining the objective or function of the system. For example, if one's objective is to print out some data, a printer (and its supporting elements) could be defined as "the system." If the objective is expanded to include the processing of data before displaying the results, “the system of interest” would include other components such as a typical home computer with terminal, keyboard and printer. Further expanding the objective to include worldwide data connections and storage would introduce communications equipment and databases into our definition of “the system.”

Aircraft, automobiles, and telephones are also examples of systems that on one level, which can be considered elements or subsystems within a large context,, for example as key elements of transportation, or communications systems. This recognizes their critical dependence on other support elements such as fuel, electric power, human operators, maintenance and repair to accomplish their defined functions.

2.3 SOME BASIC SYSTEMS ENGINEERING DEFINITIONS

While there is general recognition that "Systems Engineering" plays an important role in the development and operation of large-scale systems, there is also a great deal of variety in the way that "Systems Engineering" is described and applied.

Here are some basic definitions relating to Systems Engineering, as they will be used in this handbook:

System An integrated set of elements that accomplish a defined objective. These elements include products (hardware, software, firmware), processes, people, information, techniques, facilities, services, and other support elements.

Systems Engineering[1] An interdisciplinary approach and means to enable the realization of successful systems.

Systems Engineer An engineer trained or experienced in the field of Systems Engineering.

Systems Engineering Process A predefined set of activities selectively used to accomplish Systems Engineering tasks.

System Architecture The arrangement of elements and subsystems and the allocation of functions to them to meet system requirements.

The name of the discipline is termed "Systems Engineering" in that the practice of the discipline can be applied to many varied types systems (e.g. natural, physical, organic, people, hardware, etc.) operating with respect to its environment (open, closed, etc.) The term "System Engineering" is only used with respect to the act of engineering a specific system.

2.4 THE HIERARCHY WITHIN A SYSTEM

One of the Systems Engineer's first jobs on a project is to establish nomenclature and terminology that support clear, unambiguous communication and definition of the system, its functions, components, operations, and associated processes. (See Appendix E *Glossary and Definitions* and Appendix F *Acronym List*.)

It is essential to the advancement of the field of Systems Engineering that common definitions and understandings be established regarding general methods and terminology. In many well-established industries, there is historical precedent and good reason not to change terminology. This is certainly acceptable. What is not acceptable is an undefined or inconsistent system terminology. As more Systems Engineers accept and use a common terminology, we will experience improvements in communications, understanding, and ultimately, productivity. Toward that end, the following definitions of succeeding levels of the system hierarchy are useful for the discussions in this handbook.

System An integrated set of elements, segments and/or subsystems that accomplish a defined objective, such as an air transportation system.

Element or Segment A major product, service, or facility of the system, e.g., the aircraft element of an air transportation system (commonly used, but subsystems can be used instead of element/segments).

Subsystem An integrated set of assemblies, components, and parts which performs a cleanly and clearly separated function, involving similar technical skills, or a separate supplier. Examples are an aircraft on-board communications subsystem or an airport control tower as a subsystem of the air transportation system.

Assembly An integrated set of components and/or subassemblies that comprise a defined part of a subsystem, e.g., the pilot's radar display console or the fuel injection assembly of the aircraft propulsion subsystem.

1 The above definition of Systems Engineering is recommended by the INCOSE.

Subassembly An integrated set of components and/or parts that comprise a well-defined portion of an assembly, e.g., a video display with its related integrated circuitry or a pilot's radio headset.

Component Comprised of multiple parts; a cleanly identified item, e.g., a cathode ray tube or the ear-piece of the pilot's radio headset.

Part The lowest level of separately identifiable items, e.g., a bolt to hold a console in place.

An example of a common hierarchy is shown in Figure 2-1.

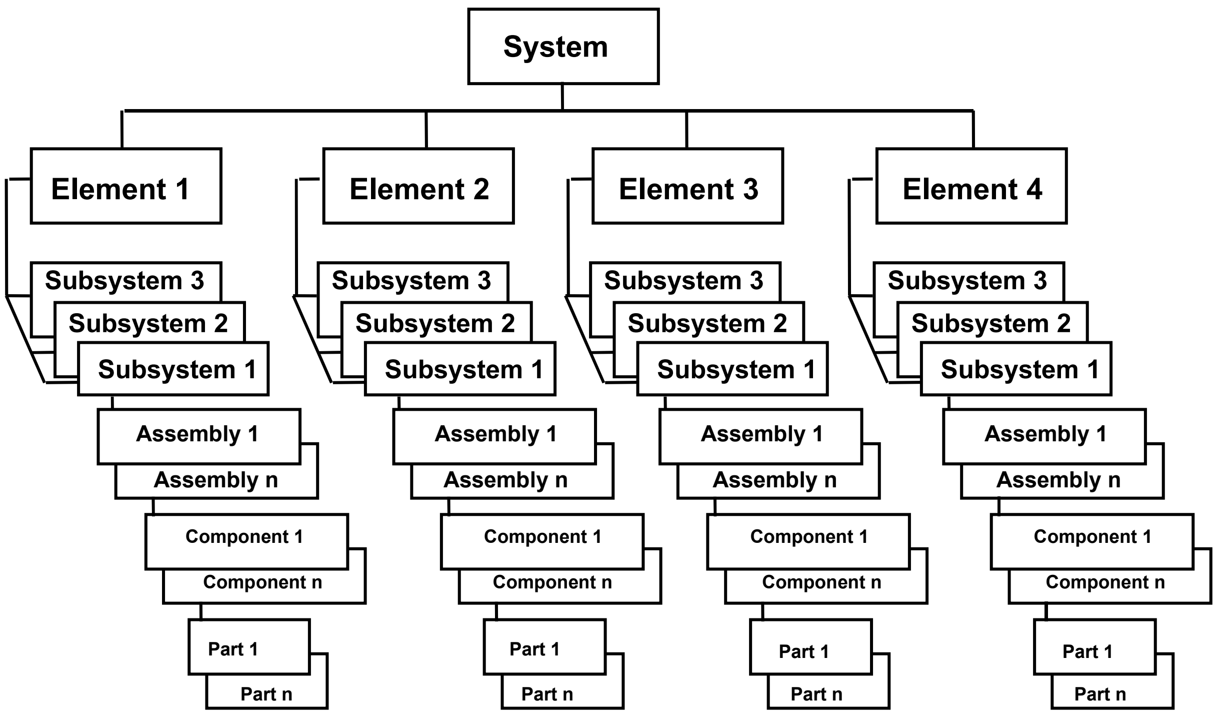


Figure 2-1. Hierarchy within a System

The depth of the hierarchy can be adjusted to fit the complexity of the system. For example, in the complex Apollo program, NASA added a "Module Level" in the hierarchy to breakout the Command Module, Lunar Module, etc. of the Space Vehicle Element. Simple systems may have fewer levels in the hierarchy than complex systems. Some examples of the hierarchy of system terminology are shown in Figure 2-2.

SYSTEM	AIR LOGISTICS	AIRCRAFT	INFORMATION	ELECTRIC CAR
ELEMENTS	AIRCRAFT PKG. PROCESSING SUPT. EQUIP. AIR & GRND. CREWS HUB, BASE, FACILITY		COMPUTERS NETWORK PRINTERS DATA STORAGE PERSONNEL	
SUB-SYSTEMS		PROPULSION STRUCTURE CONTROLS	DATA PROCESSOR OPERATING SYS. SOFTWARE	PWR. TRAIN BODY CHASSIS
COM-PONENTS		INTAKES COMPRESSOR INJECTORS CONTROLS	INPUT DEVICES OUTPUT DEVICES PROCESSING MEMORY	BATTERY MOTOR(S) GENERATOR CONTROLLER

Figure 2-2. Examples of System Hierarchy

2.5 WHAT ARE SYSTEMS ENGINEERS AND WHY ARE THEY NEEDED?

This handbook expands upon the relatively simple definitions of Systems Engineering, Systems Engineers, and the Systems Engineering Process that were given above in Section 2.3. The Systems Engineering process should be studied and used by all engineers -- just as scientists apply the scientific method. There is a distinction between one who simply understands and can apply the Systems Engineering process to their discipline and a trained, experienced Systems Engineer. One can learn the process in a few hours of reading and possibly several months of experience in applying it. However, it usually takes a good Systems Engineer five years or more to gain the experience, knowledge, and acceptance by his/her peers that is required to make the critical tradeoffs and decisions between subsystems on a large system. In addition, if he/she changes industries -- for example from aerospace to automotive -- experience in the new field must be acquired to achieve equal professional effectiveness.

The need for Systems Engineers is most apparent on large, complex systems. But systems engineering contributions are also important in the development, production, deployment, and support of much smaller systems, such as cameras and printers (note that some "systems" can also be subsystems of larger systems). The growing complexity in all areas of systems development has increased the need for Systems Engineers. For example, 25 years ago in the semiconductor industry a single chip contained no more complexity than a series of a few gates, or at most, a four-stage register. Intel's Pentium processor demands far more sophisticated analysis and immensely expands the application horizon. Today, systems engineering is being applied to social and human systems.

Systems engineers perform many useful tasks during a project's lifetime, but most managers consider their role during the development phase as the most important. During this phase Systems Engineers define the overall requirements and help evolve the system architecture (its key elements and their configuration). Systems engineers help allocate and "balance" the requirements to lower level system elements. (See Appendix D *Methods for Functional Analysis and Allocation,* with key supporting methodologies).

Requirements "balancing" is usually important in deciding how much, if any, technology development risk each element should undertake, as well as allocating "budgets" for such things as weight, power, size (physical envelope) and performance requirements. The Systems Engineer is intended to be the unbiased arbitrator of these natural internal conflicts. Each product team's primary objective is developing its subsystem or component to deliver specified performance on schedule, within their allocated costs (development cost, production cost, life cycle cost). Any systems responsibilities these teams assume are often overlooked or forgotten due to the press of their primary priorities; sometimes with disastrous consequences.

It is common for product development teams to announce that they "do not require any Systems Engineering support." "Fire the Systems Engineers and give us the funds ... we need the funds for hardware and software item development ... and we'll do our own Systems Engineering coordination." Program managers who give in to these demands often find themselves with components that do not meet performance or interface requirements and therefore their systems do not work - requiring costly redesigns.

The following sections further discuss basic Systems Engineering activities.

2.6 THE ROLE OF SYSTEMS ENGINEERS

During the past thirty years, managers have found it advantageous, on projects of moderate size and above, to designate individuals whose <u>primary</u> responsibilities are system oriented. These people are the "glue" that binds all the sometimes diverse system elements together. They provide the decentralized leadership and paths for the up/down communications that must occur in 1) flowing "down" the system level and project perspectives and 2) flowing "up" the component and subsystem perspectives of problems and difficulties with implementing designs and the associated necessities for changes.

Systems Engineering <u>represents</u> the program manager in controlling the overall program technical effort. Systems Engineer’s responsibilities include requirements analysis, functional analysis and allocation, architecture/design, systems analysis and control, and verification and validation of the system. They report, in this capacity, to the program manager as do the design/development elements. The program manager thus maintains direct, two-way communications with all elements of his team (not relayed through Systems Engineering). Systems Engineers are responsible for day to day management of overall system functionality and performance and for internal interfaces between system elements as well as all external system interfaces.

Roles that might be defined on a program include a Chief Engineer or Deputy Program Manager - Technical, who is responsible for directing the program technical effort. This person may also be the Systems Engineering manager and/or leader of the Systems Engineering team.

The Systems Engineering team designates various individuals to maintain tight liaison with all technical areas of the program, including: analysis, design, manufacturing, development testing, and verification. These Systems Engineers <u>must</u> be experienced enough to be "hands-on" participants in the process - not just observers/messengers (whom other engineers would resent).

The Systems Engineer's job includes defining, clarifying, and documenting requirements; performing (or insuring his or her team performs) the necessary parametric analysis and tradeoffs; recognizing when interface impacts might occur and taking early action to avoid problems. The SE should have a good overall perspective of the system to help interpret and explain motivations for requirements to project team members and thereby gain their acceptance and commitment to objectives.

A Systems Engineer may need to explain (and justify) to a subsystem team why it is necessary to cut excess "fat" from a design - in the form of weight, power usage, envelope, operating time, or cost-to-produce. A Systems Engineer may encourage a subsystem team to pursue a more-risky (or less risky) development approach which promises overall system payoffs. And, a Systems Engineer may help explain to management why his system team requires more resources.

Basically, the Systems Engineer, at any stage of a project cycle, works with and between the project's systems engineering team(s) and the other teams at equal, lower, and higher system levels to ensure a smooth technical program, with no surprises or adverse consequences.

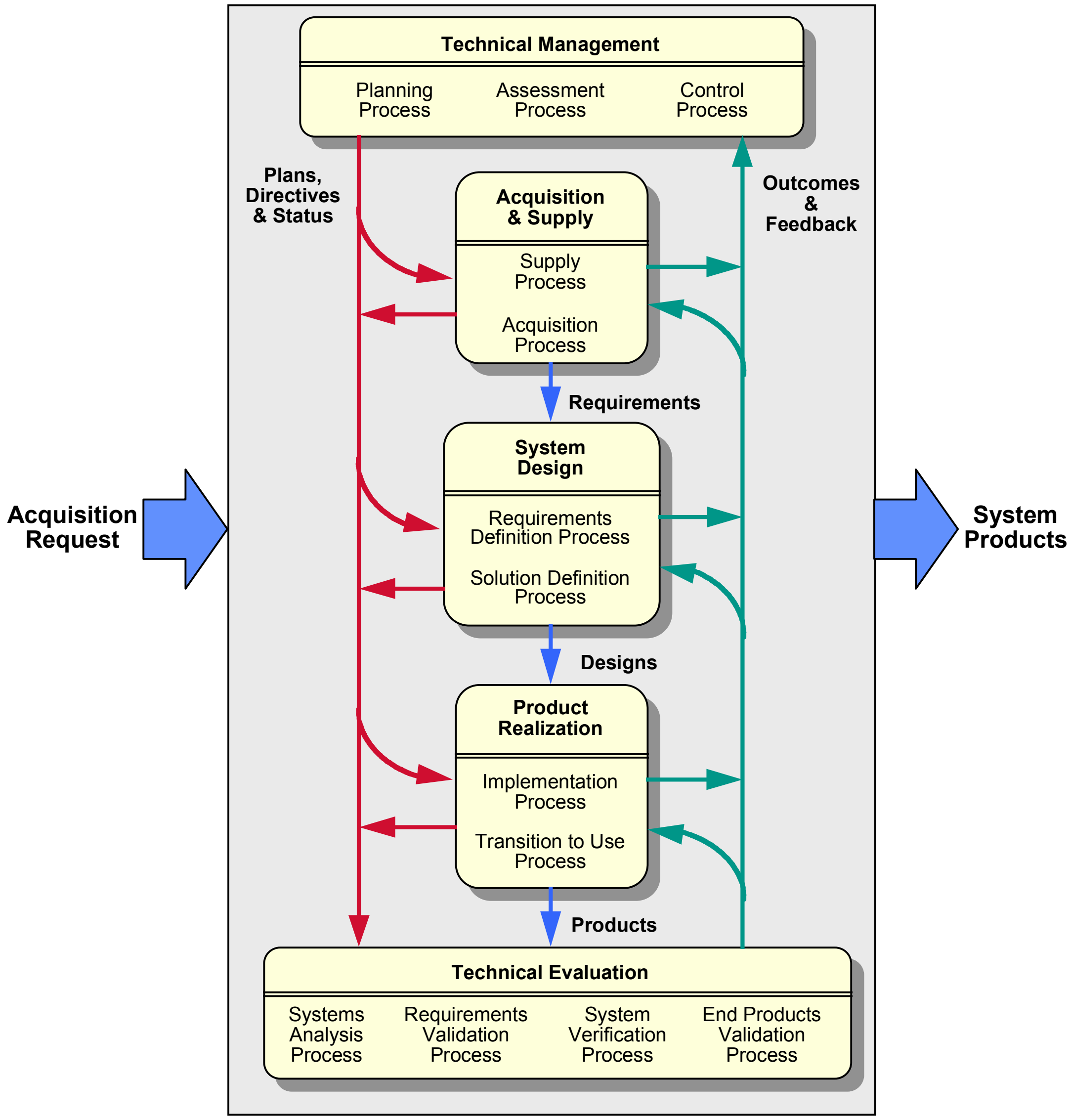
During project development phases, it has been found that expenditures of twenty to thirty percent of the total engineering budget for Systems Engineering activities are worthwhile. The higher figure is appropriate if Systems Engineering is also responsible for the internal subsystem integration (as opposed to a development engineering integration team).

If no formal Systems Engineering effort is included, projects run the risk of fifty to one hundred percent development cost overruns to fix major downstream integration problems (costs can be very high due to the necessity of keeping a major portion of the project team around until all problems are solved).

Section 7 will discuss some methods of team organization and how Systems Engineers are essential in the integrated product and process development team environment (for the same aforementioned reasons).

2.7 THE SYSTEMS ENGINEERING PROCESS

What is known as the *Systems Engineering Process* is basically an iterative approach to technical management, acquisition and supply, system design, product realization, and technical evaluation at each level of the system, beginning at the top (the system level) and propagating those processes through a series of steps which eventually lead to a preferred system solution. The System Engineering Process must include life-cycle considerations of development, deployment, operations and maintenance, and system disposal. Environmental impact of manufacturing processes, operational expendables, and ultimately system decommissioning may define system design drivers. At each successive level there are supporting, lower-level design iterations which are necessary to gain confidence for the decisions taken.



(Source: ANSI/EIA-632)

Figure 2-3. Systems Engineering Process Overview

Note: Acquisition & Supply process Requirements consist of stakeholder needs and expectations, as opposed to system technical requirements that result from the System Design process.

During each iteration, many concept alternatives are postulated, analyzed, and evaluated in trade-off studies. There are many integration factors, where decisions on one subsystem affect other subsystems. These factors must also be evaluated. An overview of the steps in the Systems Engineering process is shown in Figure 2-3. It should be observed that this process view is predominantly focused on the product development and deployment and does not cover the full lifecycle as will be discussed in section 3.

Systems Engineering is involved in all steps in Figure 2-3 and *leads* during System Design down into the subsystem level, and *integrates* many other activities including design, design changes and upgrades, customer feedback, and operational support. The Integrated Product Development Teams are primarily responsible for internal integration within the areas of their team's responsibility during preliminary design, detail design, and development. Systems Engineers closely monitor these development activities and help to resolve interface issues between teams. The processes involved will be discussed in detail in subsequent sections.

2.8 THE SYSTEMS ENGINEERING PROCESS ACROSS PROJECT LIFE CYCLE

Systems Engineering activity spans the entire program life cycle from systems analysis, requirements definition and conceptual design at the outset of a program through production, operational support, planning for replacement, and eventual retirement and disposal at the end of a program.

An example of program phases, as recommended for United States Department of Defense (US DoD) programs in MIL 499B, is summarized in Figure 2-4. This figure also shows twenty-two key program tasks, which are conducted during a typical program life cycle. These tasks are shown here for perspective on the Systems Engineering process. They will be discussed in more detail in Section 3.

The program phases for commercial firms generally cover the same spectrum of activities, but with different definitions. As an example, a car manufacturer may conduct "man-in-the-street" surveys to collect desirable features for their next generation product. On the other hand, both a military and commercial aircraft company might use similar program phases.

NEED

PHASE O	I	II	III
CONCEPT EXPLORATION	PROG. DEFINITION & RISK REDUCTION	ENGINEERING & MANUFACTURING DEVELOPMENT	PRODUCTION, FIELD-ING/DEPLOY, & OPNL. SUPPORT
1. SYSTEM ANALYSIS 2. REQTS. DEFINITION 3. CONCEPTUAL DESIGNS 4. TECHNOLOGY & RISK ASSESSMENT 5. PRELIM. COST, SCHED. & PERF. OF RECOMMENDED CONCEPT	6. CONCEPT DESIGN UPDATE 7. SUBSYS. TRADEOFFS 8. PRELIMINARY DESIGN 9. PROTOTYPING, TEST, & EVALUATION 10. INTEGRATION OF MANUFACTURING & SUPPORTABILITY CONSIDERATIONS INTO DESIGN EFFORT	11. DETAIL DESIGN 12. DEVELOPMENT 13. RISK MANAGEMENT 14. DEVELOPMENT TEST & EVALUATION 15. SYSTEM INTEGRATION, TEST, & EVALUATION 16. MANUFACTURING PROCESS VERIFICATION	17. PRODUCTION RATE VERIFICATION 18. OPERATIONAL TEST & EVALUATION 19. DEPLOYMENT 20. OPERATIONAL SUPPORT & UPGRADE 21. RETIREMENT 22. REPLACEMENT PLANNING

DISPOSAL

EVOLUTIONARY REQUIREMENTS DEFINITION

Figure 2-4. Program Life Cycle

The Systems Engineering process applies across all program phases and primary functions conducted during a program. This breadth is represented in Figure 2-4, which gives an overview of the applicability of the Systems Engineering process. However, the Systems Engineering process has evolved primarily to support the initial phases of a program -- through design, development, and verification testing. The need for a well-integrated approach for system design and development can be better appreciated when it is realized that approximately eighty to ninety percent of the development cost of a large system is predetermined by the time only five to ten percent of the development effort has been completed. Therefore, an efficient, orderly process for defining and developing large systems is essential.

2.9 TAILORING THE SYSTEMS ENGINEERING PROCESS

It is recommended that each organization tailor this Systems Engineering process to its own terminology, development, and support approaches. When appropriate, the basic tenets of this guide can serve as the backbone of organizational practices and procedures developed to implement sound Systems Engineering principles. Tailoring the Systems Engineering process is discussed in more detail in Section 5. (See Appendix C *Systems Engineering Management Plan*, and Appendix E *Glossary and Definitions*)

THIS PAGE
INTENTIONALLY
LEFT BLANK

3 MAPPING THE SYSTEMS ENGINEERING PROCESS ONTO SYSTEM LIFE CYCLES

The Systems Engineering process can be described in different contexts and levels. An example of these levels is shown in Figure 3-1, below. The remaining sections of this handbook focus on the third level Systems Engineering process activities. This section provides the context for these third level processes by briefly discussing level one and two activities. These top levels may vary by industry, organization and product.

Many of the phases/steps in these top two levels use some or all of the Systems Engineering process activities of the third level. The fourth level is addressed in this document only for selected disciplines at a summary level (describing what needs to be done and why, but not how to do it). These areas, such as software and hardware engineering, are the subject of well-defined fields of study, with textbooks, standards, and guidelines, and are beyond the scope of this handbook.

Level	Description	Examples
1	Life Cycle Phase	Concept Definition, Development, Production
2	Program Activity	Mission Analysis, Prelim. Design, Detail Design
3	SE Process	Reqts. Analysis, Architecture Definition, System Design
4	Eng. Specialty Area	Software, Human Factors, Mechanical Design

Figure 3-1. Descriptive Levels for the Systems Engineering Process

3.1 HOW SYSTEMS ENGINEERING RELATES TO SYSTEM DEVELOPMENT PHASES

Figure 3-2 illustrates a mapping of Level 1, the Program Life Cycle, taken from Figure 2.4 onto key program activities (level 2). This provides an overview of typical Systems Engineering activities performed during representative life cycle phases of a product. The Concept Exploration phase is usually preceded by a Pre-Concept phase, which is not shown in the figure.

3.1.1 PRE-CONCEPT PHASE

This phase begins when an organization begins to perceive the need for a new or modified system or service. It might perform "in-house" systems analysis studies to quantify the need and determine how best to meet it, or try to interest a customer in funding low-level, sole-source or competitive studies.

In other words, this phase often begins with an idea or a short vision statement. There may be no source of technical guidance or requirements, i.e., starting with the proverbial "clean sheet of paper." Some of the key Systems Engineering activities during this phase are: definition of project or product objectives; mission definition; definition of functional requirements; definition of candidate architectures; allocation of requirements to one or more selected architectures and concepts; tradeoffs and conceptual design synthesis; and selection of a preferred design concept. An important part of this phase is assessment of concept performance and technology demands and initiation of risk management process.

Efforts during the Pre-Concept phase are sometimes called feasibility studies, where a small team of experts attempts to synthesize a new concept. Many organizations invest in Research & Development groups that are tasked with generating innovations and new ideas for future products.

NEED

PHASE O	I	II	III
CONCEPT EXPLORATION (CE)	PROG. DEFINITION & RISK REDUCTION (PD&RR)	ENGINEERING & MANUFACTURING DEVELOPMENT (EMD)	PRODUCTION, FIELD-ING/DEPLOY, & OPNL. SUPPORT (PFD&OS)
1. SYSTEM ANALYSIS 2. REQTS. DEFINITION 3. CONCEPTUAL DESIGN 4. TECHNOLOGY & RISK ASSESSMENT 5. PRELM. COST, SCHED. & PERF. OF RECOMMENDED CONCEPT	6. CONCEPT DESIGN UPDATE 7. SUBSYS. TRADEOFFS 8. PRELIMINARY DESIGN 9. PROTOTYPING, TEST, & EVALUATION 10. INTEGRATION OF MANUFACTURING & SUPPORTABILITY CONSIDERATIONS INTO DESIGN EFFORT	11. DETAIL DESIGN 12. DEVELOPMENT 13. RISK MANAGEMENT 14. DEVELOPMENT TEST & EVALUATION 15. SYSTEM INTEGRATION, TEST, & EVALUATION 16. MANUFACTURING PROCESS VERIFICATION	17. PRODUCTION RATE VERIFICATION 18. OPERATIONAL TEST & EVALUATION 19. DEPLOYMENT 20. OPERATIONAL SUPPORT & UPGRADE 21. RETIREMENT 22. REPLACEMENT PLANNING

DISPOSAL

EVOLUTIONARY REQUIREMENTS DEFINITION

KEY TASKS

OBJECTIVES MISSION FUNCTIONAL REQTS CANDIDATE CONCEPTS & ARCHITECTURES REQTS. ALLOCATION TRADEOFFS / SYNTH. DEFINE CONCEPT SCHED. & LCC EST.	REVIEW CONCEPT HIGH FIDELITY MOD-ELING & SIMULATION INTEGRATE SUBSYSTEM DESIGNS / TRADEOFFS WRITE TOP-LEVEL SPECIFICATIONS DEVELOPMENT PLAN COST & RISK ANALYSIS RISK MGT. PLANNING	TECHNICAL PROGRAM INTEGRATION TECHNICAL PERF. MEAS. DESIGN REVIEWS REQTS. REALLOCATE (AS NECESSARY) DOCUMENT SYSTEM INTERFACE CONTROL CHANGE CONTROL IPDT PARTICIPATION	CHANGE CONTROL MFG. LIASON TEST ANALYSIS DESIGN VERIFICATION TROUBLESHOOTING ENGINEERING SUPPORT MAINTAINENCE TRAINING SUPPORT MOD DEVELOPMENT EVOLUTIONARY PLAN

Figure 3-2. Key Phases and Tasks in a System Life Cycle

3.1.2 CONCEPT EXPLORATION

In some instances there is no distinction between the Concept Exploration Phase and the Pre-Concept Phase. Otherwise, this phase begins where the Pre-Concept Phase ends. Usually there is a feasibility study report, prepared during the previous phase, with recommendations as to what should be done next. This report can serve as a starting point. Or, higher authorities may direct a somewhat different approach, based on competitive, political, or emerging market data, intelligence data, and/or other sources.

Generally, passing into this phase signals higher interest and commitment in the project. The Pre-Concept Phase results were promising enough to warrant further work, usually with a larger team. Therefore, the same types of Systems Engineering activities performed during the Pre-Concept Phase are prominent again during CE, except they receive even more emphasis. This is indicated in Figure 3-3, where an abbreviated set of Systems Engineering processes are shown versus program phases. Note that most Pre-Concept SE processes are performed again during CE.

During the CE phase, additional effort is applied to define all aspects of the system. This includes improved fidelity simulations of the system and its environment; improved analysis, tradeoffs, and definition of system elements and their subsystems; and improved modeling and analysis of subsystem performance. Requirements are developed during CE. These are developed in a top-down/bottom-up fashion, with Systems Engineers defining what they want in a top-down fashion and subsystems engineers describing what they can provide in a bottom-up fashion. Alternate system and subsystem tradeoff studies are performed until the system and subsystems engineers converge on workable solutions. The subsystem development engineers usually need to create good models of their proposed subsystems to better understand potential capabilities and limitations.

The most important product of the CE phase is the final report and its conclusions. Is there a viable concept? If not, what are the problems? If yes, what is the concept, its performance, its effectiveness, cost, and development time? What critical issues remain to be addressed? What are the risks and mitigation plans?

Program Phases →		0	I	II	III	
Legend: ● = Major Use; o = Minor Use System Engineering Processes	PRE CON	CE	PD & RR	EMD	PFD & OS	DISPOSAL
1. Pre-Proposal Activities	●	●	●	●	●	●
Mission, SRD, SOW, RFP, CDRL						
2. Requirements Analysis						
Capture Source Requirements	●	●	●	●		●
Develop Operational Concept	●	●	●	●		
Functional Performance Reqts.	o	●	●	●		
Design Constraint Reqts.	o	●	●	●		
Requirements Allocation	o	●	●	●		
3. Functional Analysis	o	●	●	●		
4. Sys. Architecture Synthesis						
Synthesize Multiple Arch's	o	●				
System Element Reqts.	o	●	●	●		
Eval/Select Pfd. Architecture	o	●	●	●		
Integrated Sys. Physical Config.	o	●	●	●		
Define/Refine Interfaces	o	●	●	●		
Develop Spec. Tree & Specs.	o	●	●	●		
5. Systems Analysis						
Tradeoff Studies	o	●	●	●		
System Modeling & Simulation	o	●	●	●	o	
Risk Management	o	●	●	●	●	●
Life Cycle Cost Analysis	●	●	●	●	o	
Cost & Effectiveness Analysis	●	●	●	●	o	●
6. SE Product Control		●	●	●	●	
7. SE Process Control						
SEMP, SEMS/SEDS, TPM, Audits		●	●	●	●	
Mfg. Involvement in SE Process	o	o	●	●	●	
8. Sys. Implementation Support						
System Integration	o	●	●	●	o	●
System Verification		●	●	●	o	●
Baseline Maintenance		●	●	●	●	
Sustaining Engineering				●	●	

Figure 3-3. Mapping of SE Processes into Program Phases

During CE other Systems Engineering activities become important as the size and depth of detail of the project grows. These include SE product and process control and system implementation support. As the concept begins to take shape, it is important to bring in stakeholders with interests in the downstream evolution of the system. Stakeholders include manufacturing engineering, operations, maintenance, and supportability personnel. These engineers can suggest changes, which are relatively easy to incorporate into the system at this early stage, but can result in major cost savings and utility enhancements during production, deployment, operations, and support. Systems engineers must screen these change requests for favorable benefits vs. costs.

3.1.3 PROGRAM DEFINITION AND RISK REDUCTION

When conducted, this phase exists as part of the risk management strategy to prove that the system will work prior to committing large amounts of resources to its full-scale engineering and manufacturing development (EMD). For very complex systems, such a demonstration can be conducted at perhaps twenty percent of EMD cost. When the total investment is modest, prototypes and proof-of-concept models may be constructed during this phase.

Integrated Product Development Teams (IPDTs) become very important during this program phase. This is the first phase in the development cycle where significant effort is allocated to develop a tangible product. In the prior two phases, the product was mostly intellectual, i.e., reports. Systems Engineers become involved in key tasks such as preparing or upgrading top-level specifications, decomposing and allocating system requirements and design constraints to lower levels, supporting preliminary design, monitoring integration of subsystems tradeoffs and designs, and detailed project plans, including scheduling and the "water fall" of ever more inclusive system design reviews.

Most analysis efforts also continue and are intensified. Higher fidelity models and simulations of system elements and the entire system are developed and thoroughly exercised. Through the use of Design of Experiments techniques, a modest number of simulation runs and tests can characterize system/element performance under a wide variety of conditions. The Risk Management process continues with updates to all activities. Thus, in Figure 3-3, most of the Systems Engineering process activities receive major emphasis.

3.1.4 ENGINEERING AND MANUFACTURING DEVELOPMENT

Moving into the EMD Phase signifies a successful PD&RR phase, justifying the major commitment of resources required to fully develop, produce and deploy the system. This phase may also be called the product development phase in some organizations. This phase exercises Systems Engineering processes (except for conceptual design) more fully than any other program phase.

During EMD, detail design and test of all components and the integrated system are accomplished. This may involve fabrication and test of engineering models and prototypes to check that the design is correct. The hardware and software design for EMD may differ from those of the PD&RR phase. This may seem inefficient, but it is usually justified to minimize PD&RR phase costs and to take advantage of lessons learned during PD&RR to improve the EMD design. Thus, most of the analysis, modeling, simulation, tradeoff, and synthesis tasks performed during CE and PD&RR are repeated at higher fidelity. This, in turn, leads to continued application of most of the Systems Engineering processes listed in Figure 3-3.

Before the EMD hardware and software is produced, a requirements validation process is conducted to ensure that the entire system will function as conceptualized. During this phase all hardware and software development is closely monitored by Systems Engineering and program management to ensure the developments remain on schedule. Systems Engineers are usually in lead roles in establishing and maintaining external interfaces, documenting the system (descriptive materials, not design drawings), strongly supporting design reviews and change control, and coordination between Integrated Product Development Teams.

Systems Engineers also perform in-process Technical Performance Measurement to assess the aggregate effect of all changes on system performance requirements and continue to maintain the risk management process. When deviations occur, Systems Engineering supports the determination of program adjustments to minimize or avoid program impact. After the system is built, its performance within specifications is verified through a planned series of analysis, inspection, demonstration and test.

3.1.5 PRODUCTION, DEPLOYMENT/FIELDING, AND OPERATIONS & SUPPORT

During production, deployment, and operational use there are many activities requiring the attention of Systems Engineers. The system is already developed, so the focus is on solving problems that arise during manufacturing, assembly, integration, and verification into its deployed configuration, and on customer orientation, validation, and acceptance testing. Systems Engineering activities include troubleshooting, risk management, risk mitigation, problem solving, maintaining the technical baseline, processing change requests to the technical baseline, design change control, manufacturing liaison, and product sell-off to the customer.

During Operations and Support many systems are under the control of the purchasers and operators, which requires a turnover from very-experienced developers to less-experienced operators. This leads to a strong operations and support presence by the developers to train and initially help operate the system. During this period, there may be upgrades to the system to achieve higher performance. This triggers a restart of the entire Systems Engineering cycle to develop the upgraded product. This type of effort is purposely omitted from Figure 3-3, but from this perspective, it becomes more apparent that most of the Systems Engineering processes are used in some way throughout all program phases of the system life cycle.

3.1.6 DISPOSAL

The safe and efficient disposal of many systems and products is a very complicated process, requiring specialized support systems, facilities, and trained personnel. These should be anticipated and accommodated from the very outset, even during the Concept Exploration phase. Design for disposal must be accommodated during EMD and/or Production, Deployment/Fielding and Operations & Support, but it is actually implemented during the disposal phase. As in every other phase, it is prudent for Systems Engineers and Systems Engineering processes to also be present and in use during the planning and conduct of disposal operations.

3.2 COMPARISON BETWEEN COMMERCIAL AND GOVERNMENT PROGRAM PHASES

A comparison of program phases for five different organizations is shown in Figure 3-4. This figure emphasizes that all programs are fundamentally similar in that they move from requirements definition through deployment, operations and support, to deactivation; but they differ in the vocabulary used and nuances within the sequential process. The two shaded areas on the figure represent zero time intervals used to facilitate comparisons among the programs and milestones. The milestones will occur at different time intervals for different programs.

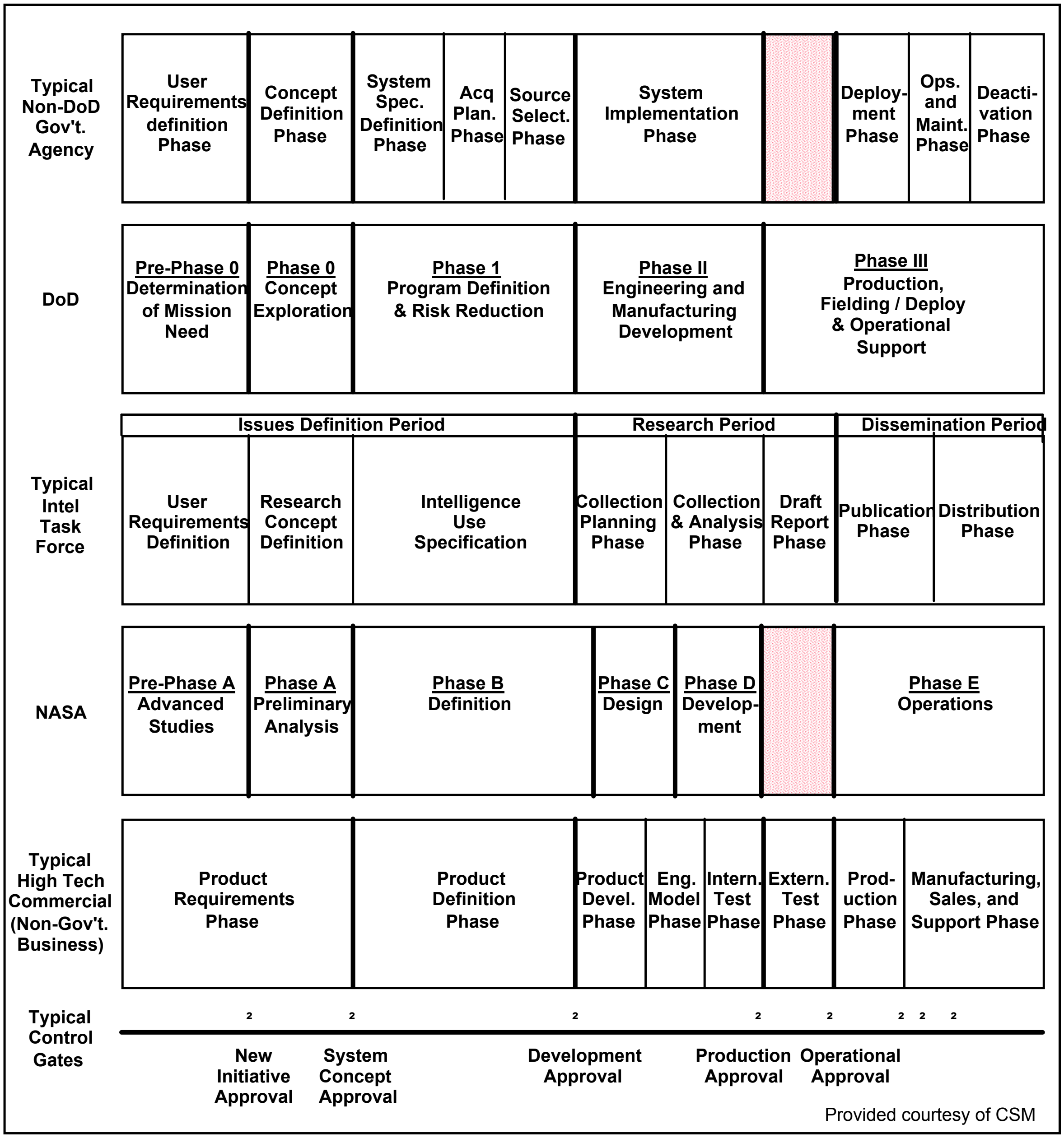


Figure 3-4. Comparison of System Life Cycle Phases

4 SYSTEMS ENGINEERING PROCESS

The Systems Engineering process is based on good engineering practice and experience. As engineered systems became more complex to include software and personnel interactions, engineering disciplines and organizations sometimes became fragmented and specialized in their attempt to cope with this increasing complexity. Organizations focused on the optimization of their primary products and often lost sight of the overall system. Each organization perceived that their part must be optimal, using their own disciplinary criteria, and failed to recognize that all parts of a system do not have to be optimal for the system to perform optimally. *The need to recognize that system requirements can differ from disciplinary requirements is a constant challenge in systems development***.** The Systems Engineering process should be viewed as a major effort in communication and the management of teams of experts that lack a common paradigm and a common language but must work together to achieve a common goal.

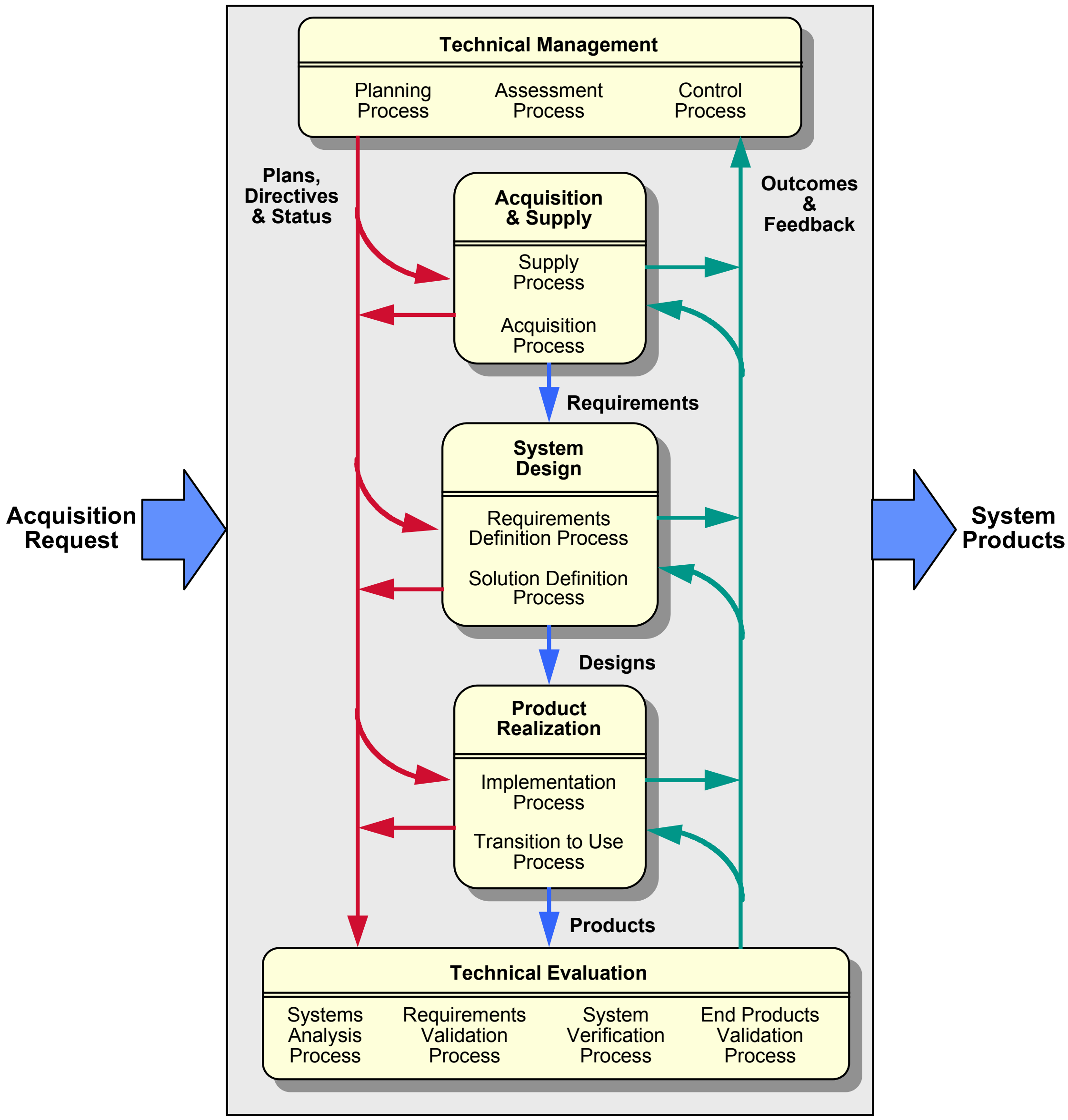
The basic engine for Systems Engineering is an iterative process that expands on the common sense strategy of (1) understanding a problem before you attempt to solve it, (2) examining alternative solutions (do not jump to a point design), and (3) verifying that the selected solution is correct before continuing the definition activities or proceeding to the next problem.

The basic Systems Engineering process tasks are:

(1) Define the System Objectives (User's Needs)
(2) Establish the Functionality (Functional Analysis)
(3) Establish Performance Requirements (Requirements Analysis)
(4) Evolve Design and Operations Concepts (Architecture Synthesis)
(5) Select a Baseline (Cost/Benefit Analysis)
(6) Verify that the Baseline Meets Requirements (User's Needs)
(7) Validate that the Baseline Satisfies the User (User's Needs)
(8) Iterate the Process through Lower Level Analysis (Decomposition)

These tasks are implemented through the process shown in Figure 4-1 for the system level. The basic tasks listed above are discussed in the System Design process block. The process involves a Requirements Definition Process that establishes both performance (quantified) requirements and functional area (architecture) requirements, and a Solution Definition Process that translates those requirements into design and operational solutions. Overarching Technical Management Processes and Technical Evaluation Processes are performed continuously throughout the development processes.

The Systems Engineering process is used iteratively at each phase of the development cycle to generate more detailed descriptions of the system under development. These descriptions are maintained in the "Decision Database." The database includes *what* needs to be achieved (functional requirements and concept of operations), *how well* it must be achieved (performance requirements and technical performance measures), *how* it is to be achieved (design), and the results of the analysis and tests of the latter's capability to actually satisfy requirements (verification) and satisfy the user (validation). As an example, hardware engineers traditionally developed physical views of the systems, while software engineers traditionally provided functional views of their code. Effective Systems Engineers communicate a "shared vision" of the systems being developed and help avoid omissions or confusion that often results from a lack of integration.



(Source: ANSI/EIA-632)

Figure 4-1. Systems Engineering Process Overview

Note: Acquisition & Supply Process Requirements consist of stakeholder needs and expectations, as opposed to system technical requirements, which result from the System Design process.

The Systems Engineering process leads to the evolution of a more detailed decision database for each subsequent phase in the development cycle (through the final/critical design review in a full-scale development program). Each phase starts from an upper level decision database of the system and first decomposes the upper level functional descriptions, and then the upper level requirements into lower level requirements for each child function. The decomposed function/requirement sets are then documented into specifications. Trade studies can then be conducted to search for the most cost/beneficial solutions, and the capability verified through test and analysis.

4.1 ACQUISITION AND SUPPLY (Defining Needs)

The initiation of a project begins with user need. Once a need is perceived and resources are committed to establish a project it is possible to define the parameters of an acquisition and supply relationship. This acquisition/supply relationship exists whenever the person or organization with a need does not have the ability to satisfy the need without the assistance of a supplier. As individuals our experience of the acquisition process is typified by the purchase of telephones or automobiles. To facilitate the purchase of more complex services and products is a primary Systems Engineering responsibility. The start of an acquisition/supply process begins with the determination of and agreement on user needs.

A. Stakeholders/Participants

Determining user need is normally designated the "mission definition" or "objectives definition" phase of the development life cycle, since the "big picture" is developed and authorization to fund the development and select the developers/suppliers are its primary functions. If Systems Engineering is employed as the implementation process, the key to this activity is to establish a decision database containing objectives, top-level quantified mission requirements, potential design and operational concepts, and a substantiation (verification) that the database is a valid interpretation of user needs. All parties involved in this process (stakeholders: users, developing agencies, builders, support organizations, etc.) should maintain and contribute to this database. In many cases each stakeholder has their own process to evaluate and introduce their needs, and the integration process is accomplished by “committee action”. This can lead to confusion, political actions, and unsatisfactory decision making. Under some circumstances, for example if the product is a refrigerator, a telephone, an automobile or other consumer product, it may not be practical to elicit needs from the "user" but rather from the marketing organization or other surrogate.

As the Systems Engineering process is applied, a common paradigm for examining available information and determining the value of added information can be created. Each of the stakeholders’ views of the needed systems can be translated to a common description that is understood by all participants, and all decision making activities recorded for future examination. The top-level program and system descriptions can be established.

B. Recommended Activities

Systems Engineering should support acquisition program management in defining what must be done and gathering the information, personnel, and analysis tools to define the mission or program objectives. This includes gathering customer inputs on "needs" and "wants", system constraints (costs, technology limitations, and applicable specifications/legal requirements), and system "drivers" (such as capabilities of the competition, military threats, and critical environments). The set of recommended activities that follow are written for a complex project that meets a stated mission or goal, but the word “product” can be substituted to apply these steps to commercial products, for example.

1. Identify stakeholders and understand their needs. Develop and document the new mission needs of all user organizations through user surveys.

2. Perform mission analysis to establish the operational environment, requirements, functionality, architecture, and verify capability as shown in Figure 4-1.

3. Document the inadequacies or cost of existing systems to perform new mission needs.

4. If mission success is technology driven, develop concepts and document the new capabilities that are made possible by the introduction of new or upgraded technology. Document the tradeoffs in mission performance vs. technology steps.

5. Prepare a justification for the need for this mission compared to alternative missions competing for the same resources.

6. Prepare the necessary documents to request funding for the first program phase.

7. If system procurement is involved, develop the information needed to release a request for proposal, establish the selection criteria and perform a source selection.

The inputs for acquisition & supply analysis depend on the market and the sellers. For products, where the first to market gains a larger market share and developers bear the burden for development costs, incremental or evolutionary development techniques are more common and the solicitation of user needs is conducted with stepwise refinements. For complex systems, the analysis of the need is ongoing.

The output of mission level activities should be sufficient definition of the operational need or concept of operations to gain authorization and funding for program initiation and to generate a request for proposal if the system is to be acquired through a contract acquisition process, or to gain authorization to develop and market the system if market driven. These outputs can be documented in a mission needs statement, a system requirements document, a statement of work, and a request for proposal.

C. End Results

Contributing stakeholders have well defined completion criteria:

- User organizations have gained authorization for new system acquisition.

- Program development organizations have prepared a SOW, SRD, and gained approval for new system acquisition, issued an RFP, and selected a contractor.

- Potential contractors have influenced the acquisition needs, submitted a proposal, and have been selected to develop and deliver the system.

- If the system is market driven, the marketing group has learned what consumers want to buy. For expensive items (aircraft) they have obtained orders for the new systems.

- If the system is market and technology driven, the development team has obtained approval to develop the new system from the corporation.

The metrics used to evaluate the acquisition & supply process are usually based on completion criteria. For example, appropriate metrics could be the estimated percent completion of the analysis and documentation of each of the required outputs, such as:

1. System Requirements Document, Percent Completion
2. Requirements Stability and Growth Metrics, such as, Number of Requirements Added, Modified, or Deleted during the preceding time interval (month, quarter, etc.).
3. Percent Completion of contract requirements documentation: SOW, CDRL, etc. (each)

D. Methods / Techniques

There are many techniques that can be used for eliciting user requirements such as, marketing and technical questionnaires or surveys, focus groups, prototypes, and beta release of a product.

The Systems Engineering "Decision Database" is discussed further in section 8.

Trade-off analysis and simulation tools can be used to evaluate mission operational alternatives and select the desired mission alternative.

Organizations with mature Systems Engineering practices will identify and manage risks associated with an acquisition program and establish the controls and actions to reduce such risks to an acceptable level as discussed in section 6.

4.2 SYSTEMS ENGINEERING TECHNICAL MANAGEMENT

Systems Engineers should perform technical management activities commensurate with project objectives. Experience indicates that good results are achieved within an Integrated Product and Process Development (IPPD) Team environment focused on designing for affordability as well as performance. Technical management includes planning, scheduling, reviewing, and auditing the Systems Engineering process. Technical program planning should include the Systems Engineering Management Plan (SEMP), the Systems Engineering Master Schedule (SEMS), and any other technical plans identified as contract deliverables or in conformance with company procedures. **In all cases, activities performed should result in a lower overall life cycle cost within acceptable risk levels.** (See Appendix C *Systems Engineering Management Plan*, including Systems Engineering Schedules.)

The participants, recommended activities and end results of technical management are closely linked to the techniques used.

4.3 SYSTEM DESIGN

System design is an iterative process performed by Systems Engineers, in cooperation with others, to transfer the customer/user's needs into a cost effective solution. "Cost effective" implies finding a solution that appropriately balances the solution for a user's needs with an acceptable amount of risk.

A clear understanding of the difference between defining what must be done and how well it must be done is mandatory for effective Systems Engineering. Unless requirements are expressed in measurable terms, it is difficult to determine when a job is done or when a product is acceptable. In addition, a requirement is not effective unless it can be verified.

A. Stakeholders/Participants

The system design activity progressively defines solution elements. However, arriving at the optimized set of elements is a multi-faceted activity requiring process expertise, application experience, solution domain expertise, and the best efforts of the extended Systems Engineering team including customer/users, solution providers, specialty engineers, integration and test engineers, and maintenance personnel.

B. Recommended Activities

System design involves iterative definition of requirements and decomposition into solution elements. Figure 4-2 shows the relationship between these activities.

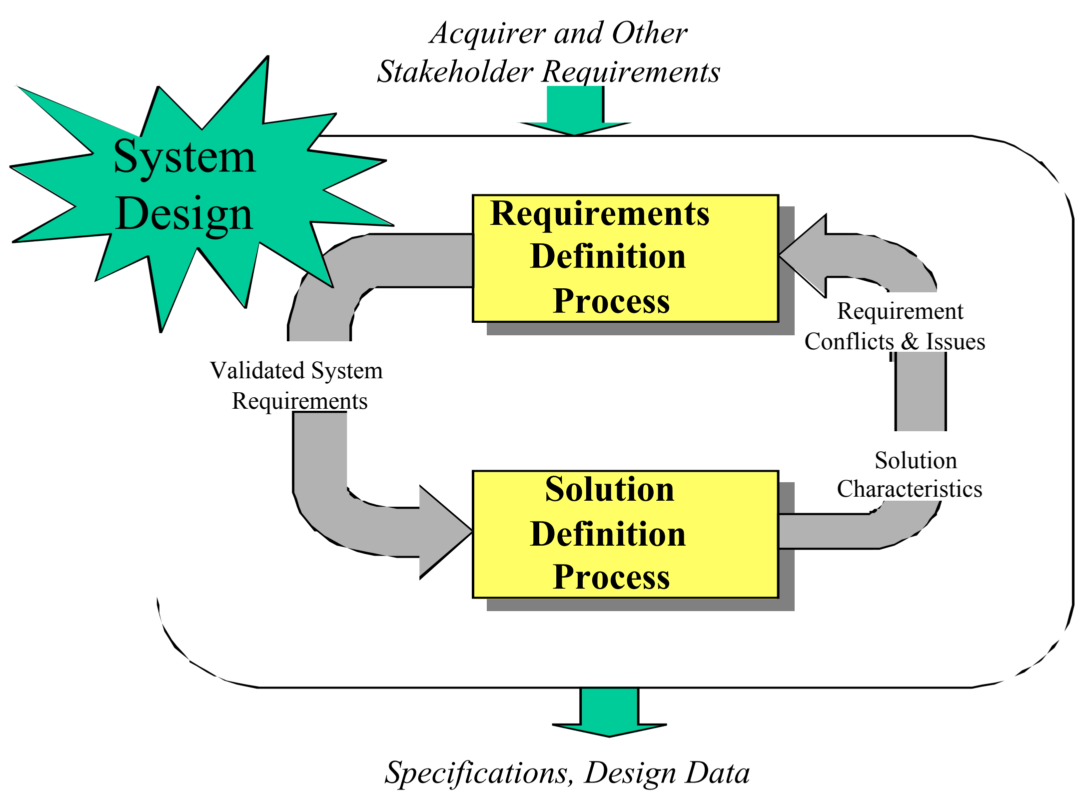


Figure 4-2. System Design Process

At each level in the system hierarchy the system design process is performed. The input to the process is requirements from the acquirer and other stakeholders. At each level in the hierarchy the higher level element is the "acquirer". With system design the first process performed is the requirements definition process. Once established, these requirements provide input to the solution definition process (also known as system design or system architecting). Requirement conflicts and errors/omissions and product characteristic issues (difficulty in meeting requirements or ease of providing additional functionality) are fed back to the requirement definition process. The output of the process at each level is specifications for lower level solution elements or design data for implementation at the lowest level.

The iterative and hierarchical nature of the process is shown in Figure 4-3. The depth and width of the hierarchy is dependent on the complexity of the specific system. A small system has a small simple hierarchy while a large system has a large hierarchy. (See Appendix D *Methods for Functional Analysis and Allocation* with Key Supporting Methodologies.)

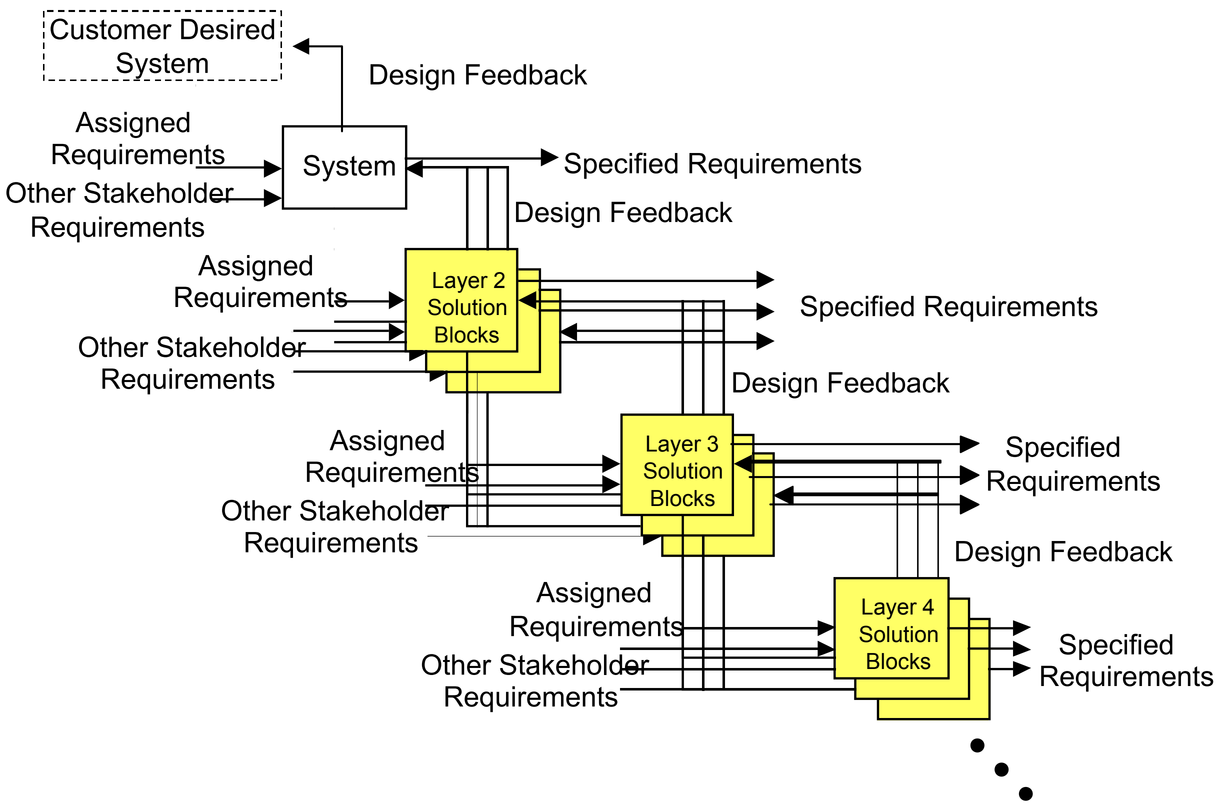


Figure 4-3. System Design Hierarchy

4.4 PRODUCT REALIZATION

Product realization is concerned with the implementation and transition into use of the designed solution. Involvement of Systems Engineers in this phase include activities such as baseline maintenance, requirements and design loops, verification, system integration, and deployment. Effective systems engineering of products requires close coordination with manufacturing engineering during development to ensure a product can be produced which will be affordable by the customer.

The discussion of these practices is limited to the activities within the discipline of Systems Engineering; i.e., it is not a general description of how to manage or execute the overall program during these phases.

4.5 TECHNICAL SUPPORT

Systems Engineers perform many different analysis and evaluation activities throughout the development lifecycle as indicated in the sections above. The participants, recommended activities and end results of technical support activities are closely linked to the techniques used.

5 SYSTEMS ENGINEERING TECHNICAL MANAGEMENT

Section 5 covers diverse topics under the heading of Techncial management. Section 5.1 discusses Systems Engineering process control. Section 5.2 discusses product control. Section 5.3 addresses tailoring criteria for applying these recommendations to projects.

5.1 SYSTEMS ENGINEERING PROCESS CONTROL

Systems Engineers should execute process control activities commensurate with project objectives. It is also adviseable to establish processes that meet the requirements of capability maturity as identified in EIA-731, SE Capability Model and the Capability Maturity Model Integration[SM] (CMMI[SM]). Processes must be established as standard practice, with effective review, assessment, audit, and change implementation. An effective feedback control process is an essential element to enable the improvement of the SE process as implemented. SE process planning control for projects should include the Systems Engineering Management Plan (SEMP), the Systems Engineering Master Schedule (SEMS), and any other technical plans identified as contract deliverables or company best practices.

5.1.1 SYSTEMS ENGINEERING MANAGEMENT PLAN (SEMP)

The Systems Engineering Management Plan is the top-level plan for managing the Systems Engineering effort. The SEMP defines how the project will be organized, structured, and conducted and how the total engineering process will be controlled to provide a product that satisfies customer requirements. A SEMP should be prepared early in the project, submitted to the customer (or to management for in-house projects), and used in technical management for the study and development periods of the project, or the equivalent in commercial practice. The format of the SEMP can be tailored to fit project, customer, or company standards. The SEMP outline included in Appendix C can be used as a guide. An example of a SEMP is available at URL = http://www.sparc.airtime.co.uk/users/wysywig/semp.html.

To maximize reuse of the SEMP for multiple projects, many commercial organizations maintain a standard company SEMP focused on Systems Engineering processes. Project-specific appendices are often used to capture detailed and dynamic information such as the Systems Engineering Detailed Schedule (SEDS), decision database, and deliverables schedule.

A. Stakeholders

Participants in the creation of the SEMP include senior Systems Engineers, representative subject matter experts, the project management, and often the customer.

B. Recommended activities

Creation of the SEMP involves defining the SE processes, functional analysis approaches, schedule and organization to name a few of the more important sections. A few summary activities are provided here, but appendix C contains a more comprehensive discussion of the work that should be done to create a useful SEMP.

The Systems Engineering Master Schedule (SEMS) is a tool for project control. The SEMS is an essential part of the SEMP. It should include a description of the critical path analysis used to derive the SEMS and the supporting Systems Engineering Detailed Schedule (SEDS) and their structure. As critical elements of the SEMP, a description of how to create these schedules is also included in Appendix C.

Section 7 of this handbook discusses integrated project team organizations in some detail. The SEMP should report the results of the effort undertaken to form a project team.

A technical objectives document should be developed. This may be one of the source documents for the decision database described below. The document may be part of a Concept of Operations for the system.

Include in the SEMP the approach and methods to arrive at a balanced set of requirements and a balanced functional and physical architecture to satisfy those requirements. The SEMP must identify the source material to be used in developing deliverables such as the Systems Specification and Technical Requirements Document.

The approach and methods used to define the performance and functional requirements for the following areas of Specialty Engineering should also be documented:

a. Reliability and Availability
b. Maintainability, Supportability, and Integrated Logistics Support (ILS)
c. Survivability including Nuclear, Biological, and Chemical
d. Electromagnetic Compatibility, Radio Frequency Management, and Electrostatic Discharge
e. Human Engineering and Human Systems Integration (See Appendix B Human Systems Engineering)
f. Safety, Health Hazards, and Environmental Impact
g. System Security
h. Producibility
i. Test and Evaluation
j. Testability and Integrated Diagnostics
k. Computer Resources
l. Transportability
m. Infrastructure Support
n. Other Engineering Specialties bearing on the determination of performance and functional requirements

The SEMP should indicate what trade studies will be included in the project.

Technical reviews are essential to insure that the system being developed will meet requirements, and that the requirements are understood by the development team. The SEMP should list what technical reviews will be conducted and the methodology to be used in solving problems uncovered in reviews.

The schedule in Figure 5-1 illustrates the appropriate time for major reviews. They may or may not be right for your project. You may need more or fewer reviews. Remember that formal, documented reviews, with the customer in attendance can have a significant cost, so also use more-frequent informal, in-house reviews to resolve most issues; and strive to exit the major, formal reviews with no major customer-imposed Action Items, i.e., be prepared.

Transitioning critical technologies should be done as a part of the risk management. A discussion of risk management is contained in section 6 of this handbook. It is called out separately here for special emphasis. Identify what technologies are critical and follow the steps outlined for risk management. Reference the work done (or to be done) explicitly in the SEMP.

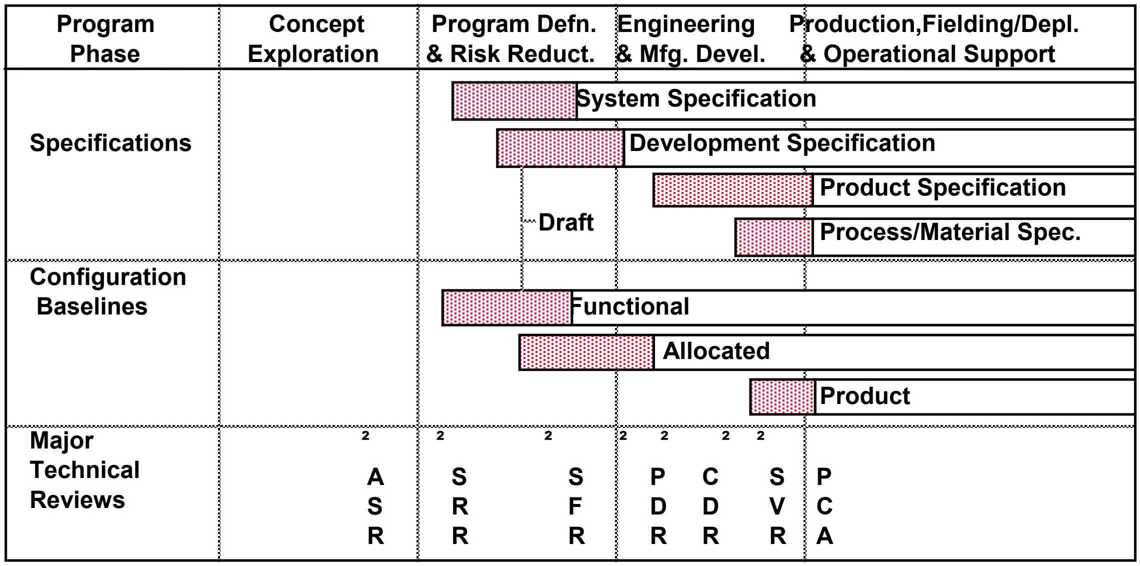


Figure 5-1. Typical Schedules for Specifications, Baselines, and Reviews

The system being proposed may be complex enough that the customer will require training in order to use it. During the project it may be necessary to train those who will develop, manufacture, verify, deploy, operate, support, do training, or dispose of the system. A plan for this is required in the SEMP. Include in the training section:

a. Analysis of performance
b. Behavior deficiencies or shortfalls
c. Required training to remedy the above
d. Schedules to achieve required proficiencies

Verification planning is usually done following a verification matrix which lists all the requirements. The possible methods of verification include inspection, analysis, demonstration, and test. The SEMP should state that a verification plan will be written to define the items to be verified and which methods will be used to verify performance. Detailed procedures are usually not written for inspection, analysis, and demonstration methods. Simulations may be used for testing when quantifiable results are needed or demonstration when qualititaive results are satisfactory.

The plan should define, at least in preliminary general terms, which performance items will be verified by which of the above methods. The plan should also define who is to perform and witness the verification of each item. This should also relate to the SEMS or SEDS for time phasing of the verification process.

C. End Results

A well-written SEMP provides guidance to a project and helps the organization avoid unnecessary discussions about how to perform Systems Engineering. In addition, a schedule and organization are defined that help the project procure the personnel needed throughout the development lifecycle and assess progress. The SEMP outlines the major deliverables of the project including a decision database, specifications and baselines.

The Process Inputs paragraph of the SEMP identifies the source material to be used for these deliverables, such as the Statement of Work, the technical requirements document, and the specification from the request for proposal. It also may include previously developed specifications for similar systems and company procedures affecting performance specifications.

The Project (or Contract) Work Breakdown Structure (WBS) document defines and outlines the project/program task hierarchy. Work authorization is the process by which the project is baselined and financially controlled. A description of your company's procedures for starting work on the detailed parts of the WBS should be defined in the SEMP.

The SEMP should also address Design to Cost and Value Engineering which would provide insight into system/cost effectiveness. For example, “Can the project be engineered to have significantly more value with minimal additional cost?” If so, does the customer have the resources for even the modest cost increase for the improvement? The intent is to assure the customer that no obvious cost effective alternatives have been overlooked.

D. Methods/Techniques

The best SEMP template to use to help you prepare a SEMP is the last one prepared by your company. However, be careful to remove everything that does not apply and remove the name of the previous project in the new SEMP!

Technical performance measurement (TPM) is a tool for project control and the extent to which TPM will be employed should be defined in the SEMP.

Technical Performance Measurement

Without a Technical Performance Measurement (TPM) program it is very easy for the inexperienced project manager to fall into the trap of relying only on cost and schedule status and perhaps the assurances from technical leaders to assess project progress. This can lead to a product developed on schedule and within cost that does not meet some key requirements.

It’s very easy, under the various pressures of development, for a responsible product manager to suggest "minor" changes that let him develop his product faster and cheaper. These arguments can become persuasive unless someone is keeping track of the integrated impact on the overall system of all these "minor" changes. This is the role of TPM.

TPM is the continuing verification of the degree of anticipated and actual achievement of technical parameters. Measured values that fall outside an established tolerance band will alert management to take corrective action. As an example, TPM is used to identify the importance of a design deficiency that might jeopardize meeting a critical system level requirement. Relevant terms and relationships are illustrated in Figure 5-2.

a. **Achievement to Date -** measured progress or estimate of progress plotted and compared with the planned progress at designated milestone dates.

b. **Current Estimate -** the value of a technical parameter that is predicted to be achieved with existing resources by the End of Contract (EOC).

c. **Milestone -** time period when a TPM evaluation is accomplished. Typically, evaluations are made to support technical reviews, during significant test events, and may also occur at cost reporting intervals.

d. **Planned Value -** predicted value of the technical parameter for the time of measurement based on the planned profile.

e. **Planned Profile -** profile representing the projected time-phased demonstration of a technical parameter requirement

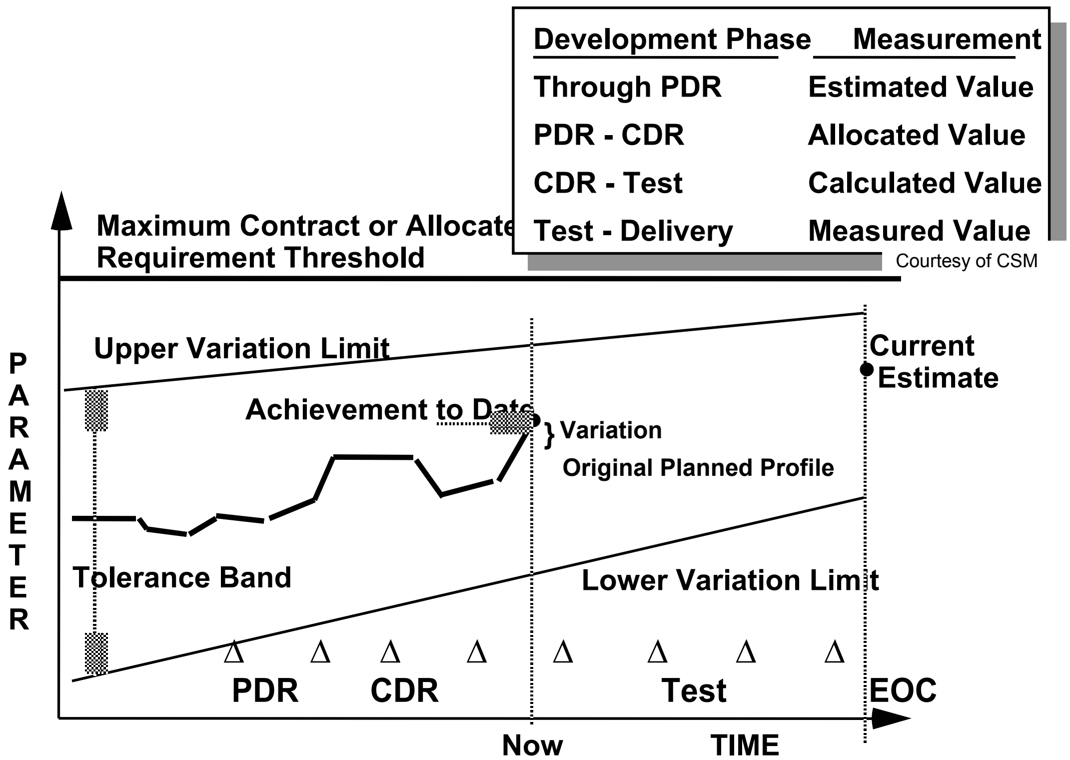


Figure 5-2. Technical Performance Measurement Profile Illustration

f. **Tolerance Band –** management alert limits placed each side of the planned profile to indicate the degree of variation allowed. The tolerance band represents the projected level of estimating error.

g. **Threshold -** the limiting acceptable value of a technical parameter; usually a contractual performance requirement.

h. **Variation(s) - Two** variations are essential to TPM: demonstrated technical variance – the difference between the planned value and the demonstrated/measured value at a specific point in time; and predicted technical variance – the difference between the specification or objective value and the current estimate of the parameter.

TPM should be established on all projects complex enough such that the status of technical performance is not readily apparent. It should be used to identify deficiencies that jeopardize the ability of the system to meet a performance requirement. Critical requirements and objectives should be selected for tracking. These might include performance parameters; hardware items such as interface definition and compatibility; and design or manufacturing process errors. The level of detail and documentation should be commensurate with the potential impact on cost, schedule, and performance of the technical project.

Designate a senior Systems Engineer to devise and integrate the TPM program.

The following items should be addressed in the TPM activity:

a. Identification of critical parameters
b. Parameter relationships to SEMS/SEDS
c. TPM parameter planning data
d. TPM tracking and control

5.1.2 STANDARD SYSTEMS ENGINEERING PROCESS AND PRACTICES

Stakeholders/Participants

An organization engaged in Systems Engineering should identify the standard process and the project tailored process. It must provide the requirements for establishing, maintaining, and improving the standard SE process. It must define a process for tailoring the standard SE process for use on projects, and define improvements to the tailored project SE processes. It is applicable to every engineering capability maturity focus area (EIA/IS-731) or process area (CMMI^{SM}).

The organization should establish standard policies; SE processes, SE practices, and supporting functional processes (see Figure 5-3). Organizational management must review and approve the standard SE process and changes to it. Organizations should establish a SE process group (SYSPG) to oversee SE process definition and implementation.

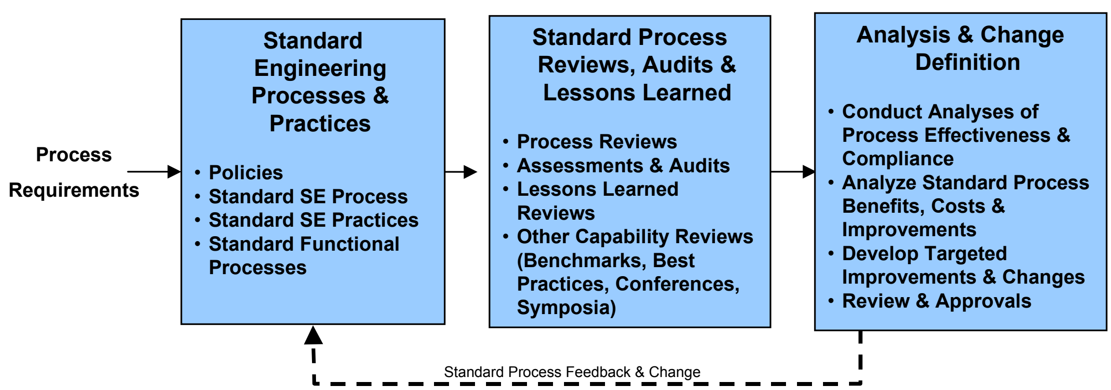


Figure 5-3. Standard SE Process Flow

Recommended Activities

An organization establishes a standard SE process using a reference SE process model, which is tailored by projects to meet specific customer and stakeholder needs. The reference model should tailor industry, government or other agency “best practices” based on multiple government, industry and organization reference SE process documents. The reference SE model must include a SE improvement process. Projects are expected to follow this process, as tailored to meet project-specific SE process needs. The standard process must be tailorable, extensible, and scalable to meet a diverse range of projects, from small study contracts to large projects requiring thousands of participants.

The standard SE process model is established by selection of specific processes and practices from this handbook, industry SE process references (such as ANSI/EIA-632 and ISO 15288), and government SE process references as appropriate.

A high performing organization also must conduct reviews of the process (as well as work products), conduct assessments and audits (such as CMMI assessments and ISO audits of SE), retain its corporate memory through the understanding of lessons learned, and establish how benchmarked processes and practices of related organizations can affect the organization. Successful organizations should analyze their process performance, its effectiveness, compliance to organizational and higher directed standards, benefits and costs, and develop targeted improvements.

End Results

The basic requirements for standard and tailored project SE process control, based on EIA/IS-731 and CMMI^{SM}, are:

a. SE processes shall be identified for use on projects.
b. Implementation and maintenance of SE processes shall be documented.
c. Inputs and outputs shall be defined for SE subprocesses.
d. Entrance and exit criteria shall be defined for SE process major activity.
e. Projects shall use a defined set of standard methods or techniques in SE process.
f. Tailoring guidelines shall be used to permit the standard process to meet project-specific needs.
g. Project management shall identify what parts of the standard SE process have been tailored to meet project-specific needs.
h. Strengths and weaknesses in the SE process shall be assessed.
i. The SE process shall be periodically assessed.
j. The SE process shall be compared to benchmark processes used by other organizations.

In addition, basic requirements specifically for SE process improvement control from these standards are:

a. Organization best practices shall be identified and communicated to projects.
b. The standard SE process shall identify areas for future improvement.
c. SE process users shall be able to identify proposed improvements.
d. Compliance with improvement processes, plans and practices shall be verified.
e. The project tailored SE improvement process shall include a means for evaluating its effectiveness.
f. The project tailored SE improvement process shall include a means for making needed improvements.
g. The standard SE process work products shall be reviewed and results used to improve this process.
h. The standard SE process compliance shall be reviewed and results used to improve this process.

Methods/Techniques

Appendix C contains suggestions for deriving the SE process for a project.

5.1.3 REVIEWS, AUDITS AND LESSONS LEARNED

The standard SE process must meet requirements for review, assessment, and audit; and for establishment of lessons learned and best practices.

5.1.3.1 PROCESS COMPLIANCE REVIEWS

The standard SE process must include a periodic process compliance review (PCR) for assessing key process element implementation effectiveness. PCRs must be conducted on a recurring basis determined by the SE organization with management involvement. If the organization conducts other assessments or audits on a recurring basis (such as for self assessment or ISO 9000), they can be combined into one assessment review to reduce the perceived burden. A standard SE process checklist should be used as the basis for this PCR. It may be augmented by additional issues and questions. Each review may address a subset of the standard SE checklist. The questions asked and results from this review should be recorded and stored. The review should address defects in the SE process in use at the time of the review. The review should address the improvement process, tailoring of the SE process, and tailoring of the improvement process (if applicable).

The PCR must be organized by a PCR Coordinator who will notify responsible personnel of the specific dates, formats and requirements for the reviews, define the lists of required attendees and invitees, and set the agenda. The data presented in these reviews should be archived. Key results from PCRs must be provided for management consideration.

The PCR should cover at least the following:

- Identify strengths and weaknesses in the SE process and its improvement process.
- Identify key process elements which need to be followed in large and/or small projects
- Identify areas for future improvement
- Address the effectiveness of the tailored improvement process
- Address the conduct of, defects in, and improvements to the SE improvement process
- Review SE work products to identify potential trends indicating possible systemic issues
- Review the results of PCRs to identify potential trends indicating possible systemic issues
- Review a sampling of in-process reviews to identify potential trends indicating possible systemic issues
- Review the definition and use of metrics in SE process measurement.

5.1.3.2 ASSESSMENTS AND AUDITS

Assessments and audits should be conducted, which include internal and external assessments of capability maturity, and internal and external audits of key SE processes and those personnel which implement them.

Internal assessments of capability maturity should be conducted to improve the organization's SE process, and to prepare for external assessments. The assessment team should consist of at least one external, qualified lead assessor. The standard for use in capability assessments will be an external, industry formal standard such as EIA/IS-731 Systems Engineering Capability Model (SECM), the Software Engineering Institute (SEI)'s CMM Based Assessment for Internal Process Improvement (CBA IPI), the CMMI^{SM} or equivalent.

External assessments of capability maturity should be conducted. They should be led by an external, qualified lead assessor, with a significant part of the assessment team consisting of external, qualified assessors. The standard for use in capability assessments should be the external, industry formal standard required by organization or customer, such as EIA/IS-731 SECM, the SEI's CBA IPI, the CMMI^{SM} or equivalent.

Internal audits of organizations using SE processes should be conducted to prepare for an external audit of the organization. A standard SE process activity checklist should be used as the basis for this audit. It may be augmented by additional issues and questions. Each audit may address a subset of the standard checklist. The questions asked and results from this audit must be recorded and stored. The audit should investigate defects (i.e., process errors) in the SE process in use at the time of the audit to understand why the defects or errors occurred.

5.1.3.3 LESSONS LEARNED

Lessons learned are needed to recognize an organization's good and best practices, understand the lessons of the organization's history and avoid repeating the mistakes of the past. They must address technical SE management and engineering, specialty management and engineering, and any other project or organization activities affecting SE processes.

The SE organization should review lessons learned to gather information supporting analysis and improvement of SE processes and practices. It should establish best practices and capture them in an easy-to-retrieve form.

5.1.3.4 OTHER CAPABILITY REVIEWS

Reviews of other types of SE process capability should be conducted.

Benchmarks from other organizations can be useful. Reference processes, practices and other capabilities must be accessed through either direct contact or an intermediary's compilations of benchmarked processes, practices and other capabilities.

Related industry conferences, symposium and workshops are a source of good ideas. Industry or organization-specific conferences that are held annually bring together key practitioners in various fields and disciplines. The International Council on Systems Engineering has an International Symposium and an International Workshop each year. The INCOSE symposium brings together in July-August typically 100 to 200 papers, about 20 panels of experts, and about 50 working group sessions covering all aspects of SE and technical management processes and cases studies. The INCOSE workshop in January typically supports about 50 working group sessions covering advances in processes related to SE management, processes and methods, measurements and assessments, systems modeling and tools, system standards, special systems applications, systems research and education. The Electronic Industries Association (EIA) conducts an annual conference covering high technology industry products, processes and tools, including SE. Other similar conferences and symposia provide access to the latest advances in process technology and best practices from many companies. The organization must encourage attendees to report to the appropriate SE organization their experiences from these conferences and symposia.

Best practices and lessons learned can be obtained from the internet also. When personnel identify advanced process-related material in the course of their jobs from the internet, they are encouraged to report it to the SYSPG, who will review it for applicability to the organization's SE process.

5.1.4 ANALYSIS AND CHANGE DEFINITION

The standard SE process must meet requirements for analysis and change of the standard SE process.

5.1.4.1 ANALYSIS OF PROCESSES

The SYSPG should sample and monitor project implementation of tailored SE processes to identify potential systemic problems in the standard SE process. Feedback, minutes, and reports from project assessments, audits, formal reviews, in-process reviews, and PCRs should be sampled and analyzed. Results of training evaluations and action item compliance reports should be analyzed. Reports of lessons learned and best practices should be analyzed. These analyses should identify and define potential process strengths, weaknesses, deficiencies, and problem areas.

The SYSPG should analyze results from reviews, assessments and audits to assess their utility in improving SE process performance. The SYSPG should assess reported work product defects to determine if systemic flaws in the standard SE process are revealed or changes needed are identified. The SYSPG should identify the root causes of these defects to determine whether changes to the standard SE processes are required to prevent future occurrences.

The SYSPG should analyze suggestions, best practices and other general or ad hoc inputs addressing the standard SE process, its value and its costs. It should also analyze reviews of and material from lessons learned; benchmarks; and related industry conferences, symposium and workshops. It should analyze organization business practices, and other company practices and standards for application to the standard SE process. These analyses should identify and define potential process strengths, weaknesses, deficiencies, and problem areas.

The SYSPG should assess activities providing insight into project SE process effectiveness and compliance. The assessments should address project SE process implementation to understand project issues and concerns with the SE process, and to identify potential systemic problems in the standard SE process. Assessments should identify potential strengths, weaknesses, deficiencies or problem areas in the standard SE process that are revealed in the project process assessments. These assessments will not evaluate or judge project performance; they will focus on internal standard SE process improvement.

Assessments should address at least the following issues:

1. Is the SE process effective and useful (e.g., are we getting what we need from it)?
2. Can the SE process be improved (e.g., (1) are there process elements which were a "waste of time" or (2) are there process elements which should have been done or could have been done better)?
3. What can we change in the SE process to make it better (e.g., what could we do to eliminate the recorded action items or defects)?
4. What is the productivity of the standard major SE process elements?
5. Are the SE support tools and facilities effective and useful?
6. Is information being collected on the effectiveness and usefulness of the SE process?
7. Is information being used to improve the effectiveness and usefulness of the SE process?

These analyses and assessments should establish for the standard SE process, its:

- Effectiveness
- Utility
- Information collection utility
- Support tool utility
- Issues and concerns
- Compliance in the organization
- Understanding of implementation impacts
- Potential systemic problems
- Potential for improvement

Rationales for and results from decisions should be recorded and stored.

5.1.4.2 DEFINING IMPROVEMENTS NEEDED

The analyses and assessments should determine whether changes are needed in the standard SE process and its documentation. Improvements needed in tailoring guidance to better meet project-specific needs should be identified. The SYSPG should document and store process compliance in process compliance and/or exception reports.

The organization should improve the SE process based in large part on usage, experience, and feedback from the programs as noted above. The standard SE process improvement should be managed and improved with the participation and support of key stakeholders in the process activities.

The SYSPG should evaluate the use of the standard SE process, and its tailoring for projects, to determine the basic utility of the standard process' elements for the organization. It should identify the benefits and associated costs (positive and negative factors) associated with or implied by project implementation of the standard SE process.

The SYSPG should evaluate the results of lessons learned analysis and best practice reviews. Changes needed in the standard SE process should be identified and prioritized. The rationale for these changes should be documented and stored.

The SYSPG must identify strengths of and areas for improvement in the standard SE process. Assessments must consider trends in process technology such as changes in capability maturity assessment practices wanted by organizational stakeholders. Areas of improvement should be developed from the results of project work product reviews, management reviews, and SE process compliance reviews. Areas of improvement should be prioritized and alternatives assessed. The requested or recommended areas for improvement and the impact of consequential changes should be identified. If the improvements or changes involve errors or problems with the existing process, these are identified to determine the actions needed to prevent future occurrences.

The SYSPG should identify and refine the criteria being used in analyses and assessments to improve their focus on essential organization business and process needs. Criteria should be recorded and stored.

5.1.4.3 SE PROCESS CHANGES

The SYSPG should prioritize the requested or recommended areas for improvement for the standard SE process. Management should approve the prioritized areas for improvement. Management should decide on what changes will be made, and adjusts budgets and labor estimates as needed to enable the changes to be accomplished. Changes may be required, requested, or recommended based on prioritized areas for improvement, process compliance requirements and/or exception reports. The SYSPG should study the priority areas for improvement, identify the specific changes needed, and recommend adjustments. The SYSPG should determine which changes can be made in the standard SE process to implement the priority improvements within budget and schedule.

The SYSPG should also assess the areas for improvement and related analyses to determine if additional tailoring guidelines are needed. If so, they should identify the tailoring changes needed, fit them into the overall improvement priority scheme, and recommend which changes should be made. A SE Process Improvement Plan should be developed and updated at least annually based in part on targeted improvements and results from reviews. After the SYSPG have prepared standard SE process process changes, they will be submitted to management for approval, with the coordination of the project managers.

5.2 SYSTEMS ENGINEERING PRODUCT CONTROL

Techniques for Systems Engineering process control are discussed as follows:

Technique	Paragraph
Configuration Management	5.2.1
Data Management	5.2.2

5.2.1 CONFIGURATION MANAGEMENT

The primary function of Configuration Management is to establish and maintain control over requirements, specifications, configuration definition documentation, and design changes. This activity supports the audit of the functional and physical configuration.

Objective

The primary objective of Configuration Management (CM) is to ensure effective management of the evolving configuration of a system, both hardware and software, during its lifecycle. Fundamental to this objective is the establishment, control, and maintenance of software and hardware functional, allocated, development, test, and product baselines. Baselines are reference points for maintaining development and control. These baselines, or reference points, are established by review and acceptance of requirements, design, and product specification documents. They may reflect the end of one phase or segment of a phase as well as the beginning of a new phase. The baseline may reflect the standard project milestone event, and the event reviews. As each segment of a phase is approved the software and/or hardware baseline is placed under configuration control.

5.2.1.1 BASELINE OVERVIEW

The *functional baseline* is established following a successful review with the customer of a system specification. The system specification is then decomposed and allocated to lower level elements of the system, normally assigned to software and hardware configuration item specifications. These lower level elements comprise the *allocated baseline.* The decomposition of these to lower level requirements is a function of the Systems Engineering process. The *developmental baseline,* established at top-level design, is specific to software development. This baseline is maintained to ensure control of the internal development cycle and to incrementally achieve interim milestones. Finally the *product baseline* is established. Items included in the hardware product baseline include engineering drawings and their complementary Configuration Item Lists (reflects the as-built hardware configuration). The software product baseline includes source code, compilers, and tools that assist in creating the software product baseline. Table 5-1 summarizes the four types of baselines and the events that identify their initial establishment.

Table 5-1. Baseline Types

Baseline	Approval Authority	Baseline Event
Functional	Customer	Established at System Specification approval
Allocated	Project Management (PMO)	Established at completion of specific reviews covering documentation items . i.e., Software Specification Review (SSR)
Developmental Configuration	PMO and software manager	Established by start of Preliminary Design Review (PDR)
Test/Product	PMO and customer	Established at completion of formal acceptance testing and the physical and functional configuration audits.

5.2.1.2 IMPLEMENTATION

There will always be a need to make changes; however, Systems Engineering must ensure that the change is (1) necessary, and (2) that the most cost-effective solution has been proposed.Configuration management is a discipline applying technical and administrative direction, surveillance, and services to:

- Identify and document the functional and physical characteristics of individual configuration items making them unique and accessible in some form;
- Establish controls to changes in those characteristics; affect product release, and ensure consistent products via the creation of baseline products;
- Record, track, and report change processing and implementation status and collect metrics pertaining to change requests or problems with the product baseline.

The initial planning efforts for CM should be defined in a Configuration Management Plan. The configuration management program is implemented at the onset of the project. It establishes the resources and skill level required, defines the tasks to be performed, identifies CM tools and processes, as well as methodology, standards and procedures to be used.
The configuration management process to accomplish this is shown in Figure 5-4.

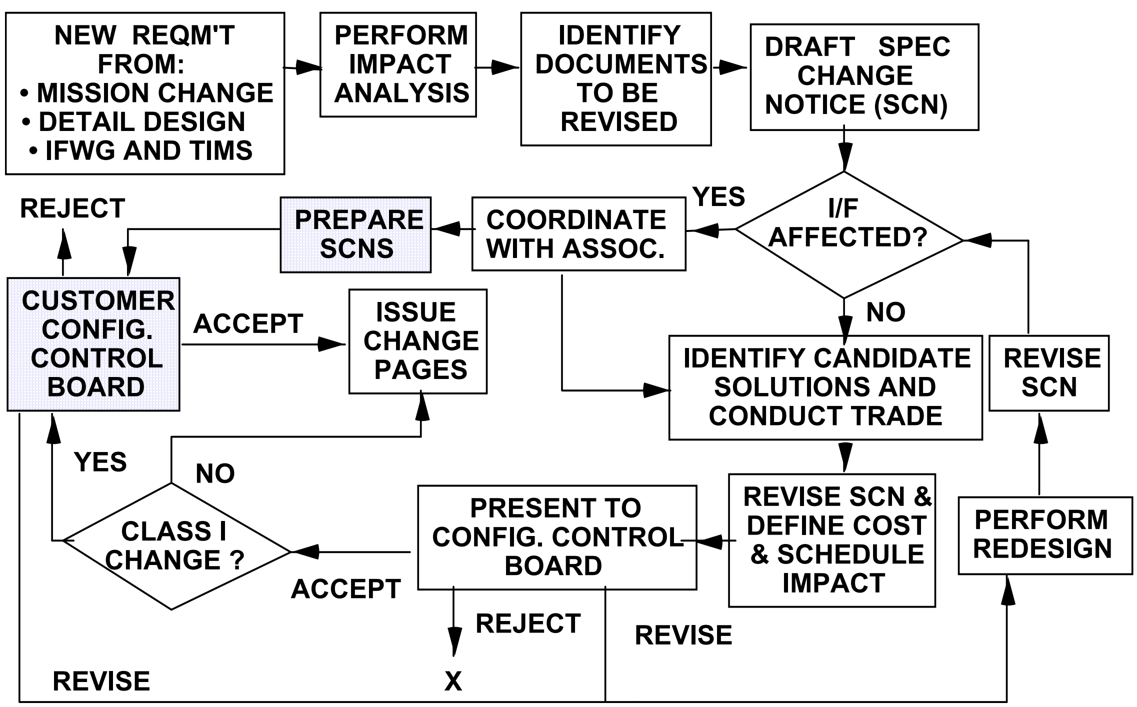


Figure 5-4. Configuration Management Process

<u>Typical steps in a project CM process with a CCB</u>

- Prepare a Configuration Management Plan
- Organize a Configuration Control Board (CCB)

- Identify the need for changes through Technical Interchange Meetings (TIMS) with the customer and users, design reviews, Interface Working Group (IFWG) Meetings, detailed design, test, engineering specialty, logistics inputs, etc. Systems Engineering should be an active participant in this process.

- Identify the documents (specifications) requiring change, and redline them appropriately. Prepare a preliminary Specification Change Notice (SCN).

- Manage changes to all baselined specifications as directed by the CCB

- Where interfaces are affected, coordinate changes with the other contractors or developers.

- Identify (with Engineering) potential design solutions, and conduct a trade study to identify the most cost effective solution.

- Prepare an Engineering Change Proposal (ECP) that includes the SCN and the cost of making the change.

- Have the Engineering organizations prepare a change package for the CCB containing the results of the analyses conducted, the proposed change description, and the impact of making the change.

- Upon approval by the project CCB, the ECP with the SCN is forwarded to the customer for review and approval or rejection.

The CM process consists of four subprocesses: configuration identification, control, status accounting, and audits (validation and distribution). The CM program is implemented at the onset of the project

5.2.1.3 CONFIGURATION IDENTIFICATION

The configuration identification process uniquely identifies the elements within a baseline configuration. This unique identification promotes the ability to create and maintain master inventory lists of baselines. As part of the Systems Engineering effort the system will be decomposed into Configuration Items. This Configuration Item list reflects items that may be developed, vendor produced, or provided by the customer for integration into the final system. These items may be deliverable items under the contract or used to produce the deliverable items.

5.2.1.4 CONFIGURATION CONTROL

Managing the collection of the items to be baselined is another aspect of configuration management. Configuration control maintains the integrity of the configuration items identified by facilitating approved changes and preventing the incorporation of unapproved changes into the baseline. Change control should be in effect beginning at project initiation.

Change Classification

Effective configuration control requires that the extent of analysis and approval action for a proposed engineering change be in concert with the nature of the change. The problem statement includes a description of the proposed change, the reason for the proposed change, the impacts on cost and schedule, and identifies all affected documentation. Change classification is a primary basis of configuration control. All changes to baselined documents are classified as outside of the scope of the requirements or within the scope of the requirements. A change outside the scope of project requirements is a change to a project baseline document that affects the form, fit, specification, function, reliability, or safety. The coordinating review board determines if this proposed change requires a change notice for review and approval.

Changes are sometimes categorized into two main classes: Class I and Class II. A Class I change is a major or significant change that may affect cost, schedule, or technical issues. Normally Class I changes require customer approval prior to being implemented. A Class II change is a minor change that often affects documentation errors or internal design details. Generally, Class II changes do not require customer approval.

Configuration Control Board

An overall configuration control review board is implemented at the time of project initiation and is established to provide a central point to coordinate, review, evaluate, and approve all proposed changes to baselined documentation and proposed changes to baselined configurations including hardware, software, and firmware. The review board is composed of members from the functional disciplines including Systems Engineering, software and hardware engineering, project management, product assurance, and configuration management. The chairperson is delegated the necessary authority to act on behalf of the project manager in all matters falling within the review board responsibilities. CM is delegated responsibility for maintaining status of all proposed changes. *Satellite* or *subordinate* boards are established for reviewing software or hardware proposed changes. If those changes require a higher approval review they are forwarded to the overall review board for adjudication.

Changes that fall within the review board jurisdiction should be evaluated for technical necessity, compliance with project requirements, compatibility with associated documents, and project impact.

As changes are written while the hardware and/or software products are in various stages of manufacture or test, the review board should require specific instructions for identifying the effectivity or impact of the proposed software or hardware change and disposition of the in-process or completed hardware and/or software product. The types of impacts the review board should assess typically include that:

- All parts, materials, and processes are specifically approved for use on the project;
- The design depicted can be fabricated using the methods indicated;
- Project quality and reliability assurance requirements are met; and
- The design is consistent with interfacing designs

Change Requests

Problem Reports or Change Requests are written to identify the occurrence of a problem. The problem should be documented in either electronic or hardcopy. The problem report or change request will identify time, date, location of the occurrence, and is reviewed by the review board. The problem statement should provide accurate and clear information of the problem. The review board validates the problem statement, assigns a responsible engineer to implement the change. When implementation of the change has been made, feedback of the resolution is provided to CM and the review board members.

Methods/Techniques

Change control forms provide a standard method of reporting problems and enhancements that lead to changes in formal baselines and internally controlled items. The following forms provide an organized approach to changing hardware, software or documentation:

- Software Problem/Change Reports can be used for documenting problems and recommending enhancements to software or its complementary documentation. These forms can be used to identify problems during software design, code, integration, and test.

- Specification Change Notice is used to propose, transmit, and record changes to baselined specifications.

- Engineering Change Proposals are used to propose Class I changes to the customer. These proposals describe the advantages of the proposed change, available alternatives, and identify funding needed to proceed.

- Request for Deviation/Waiver is used to request and document temporary deviations from configuration identification requirements when permanent changes to provide conformity to an established baseline are not acceptable.

5.2.1.5 CONFIGURATION STATUS ACCOUNTING

Status accounting is performed by CM to record and report information to management. CM maintains a status of approved documentation that identifies and defines the functional and physical characteristics, status of proposed changes, and status of approved changes. This subprocess synthesizes the output of the identification & control subprocesses. All changes authorized by the configuration review boards (overall and subordinate) culminate in a comprehensive traceability of all transactions.

Such activities as check-in and check-out of source code, builds of configuration items, deviations of manufactured items, waiver status are part of the status tracking.

By statusing and tracking project changes, a gradual change from the *build-to* to the *as-built* configuration is captured.

Metrics

Suggested metrics for consideration are: number of changes processed, adopted, rejected, and open; status of open change requests; classification of change requests summary; number of deviations or waivers by Configuration. Item; number of problem reports open, closed, and in-process; complexity of problem reports and root cause; labor associated with problem resolution, and test phase when problem was identified; processing times and effort for: deviations, waivers, ECPs, SCNs, Change Requests, and Problem Reports; activities causing a significant number of Change Requests; and rate of baseline changes.

5.2.1.6 CONFIGURATION AUDITS

Configuration audits are performed independently by CM and product assurance to evaluate the evolution of a product to ensure compliance to specifications, policies, and contractual agreements. Formal audits are performed at the completion of a product development cycle. They are the Functional and Physical configuration audits.

The functional configuration audit is intended to validate that the development of a configuration item has been completed and it has achieved the performance and functional characteristics specified in the System Specification (functional baseline).

The physical configuration audit is a technical review of the configuration item to verify the as-built maps to the technical documentation.

Finally, CM performs periodic in-process audits to ensure that the configuration management process is followed.

5.2.2 Data Management

Data Management includes all data generated by the project. Generally specifications, hardware drawings, and software code are covered by configuration management as described above. However, test plans, test procedures, test results, engineering analysis reports, and similar documentation are of equal importance and should also be maintained under configuration control since they can directly affect system design and verification. Other documentation such as Project Directives, Project Procedures, Design-to-Cost reports, Project Schedules, Risk Analysis Reports, etc. have a major role in controlling project costs and should also be controlled data.

Recommended Activities

1. Prepare a Data Management Plan (DMP) at the beginning of the project identifying all data to be produced, who is responsible for reviews and approval, and how it is to be controlled and cross-referenced.

2. Enter all data into electronic databases as it is produced, identified by reference number, category, and keywords.

3. From the SOW, identify all items on the Contract Data Requirements List (CDRL) and their delivery dates, assign responsible individuals, and publish internal preparation schedules with set-backs to permit adequate time for internal reviews.

4. Process those items identified for CCB action as described previously in Section 4.6.1.

5. Transmit Contract Data Requirements Lists CDRLs and other items (or changes) to individuals for review and approval as noted in the DMP.

5.2.3 Baseline Maintenance

A configuration baseline is the configuration documentation formally designated at a specific time during a system's or subsystem's life cycle. Configuration baselines, plus approved changes from the baselines, constitute the current configuration documentation. Typically, three formally-designated, configuration baselines are useful: the functional, allocated, and product baselines. These baselines are progressive so that lower level baselines MUST conform to higher-level baselines unless there is a formal change to the higher level baseline.

The functional baseline establishes the functional, performance, interoperability/interface, and verification top-level requirements for the system and major subsystems. This baseline is also referred to as System Definition, and the primary documentation is the System Specification(s).

For example, the allocated baseline is the initially approved documentation describing a subsystem's or components functional, performance, interoperability and interface requirements that are allocated from those of the system or a higher level subsystem or components; interface requirements with interfacing subsystems or components; design constraints; derived requirements (functional and performance); and verification requirements and methods to demonstrate the achievement of those requirements and constraints. There is an allocated baseline for each subsystem or component to be developed. The Allocated baseline is also referred to as Preliminary Design, and the primary documentation is the subsystem and component performance level specifications.

The product baseline is the detailed baseline that defines the physical composition of the system. Also referred to as the Detailed Design, the primary documentation includes subsystem and component detail, material, and process specifications by the time of Critical Design Review to complete a technical data package once production stability has been achieved.

Stakeholders/Participation

The key participants in baseline maintenance are Systems Engineering; design engineering, and configuration management. The Systems Engineering task is to ensure and manage the technical integrity of the baseline, continually updating it as various changes are imposed on it during the life of the project. The configuration management task in Baseline Maintenance is to maintain control of the baseline documentation and integrate with project management.

One Systems Engineering group should be in charge of establishing and maintaining the technical description of the baseline. That group will be the focal point for the changes to the baseline. The Systems Engineering function (whether a separate department or scattered Systems Engineers within various product teams) works with the applicable organizations both to establish and maintain the baseline, whether it is a requirements or a design baseline.

Recommended Activities

The task is to establish and maintain a baseline for the project that can be referred to and used by all elements of the project at any point in the project.

The baseline referred to here is both the requirements and design baseline. These baselines are frozen at various points in the project to allow the project to work to an established, common baseline. A common baseline is needed to allow a uniform approach to the design, test, manufacturing, and all other project technical disciplines. A common baseline is also needed at project reviews.

The following steps describe Systems Engineering activities for Baseline Maintenance.

1. Determine the nature of the change. There are several categories of change possible. They may be divided into the type and the source of change. The type is a function of the degree of control imposed. The degree of control imposed is in turn a function of the maturity of the project. Early in the project, the change will be informal. There may be no change control, or there may be an informal engineering control, which is a precursor to a Change Control Board (CCB). The engineering control may be under an Engineering Review Board (ERB), or it may be informally under a designee within the Systems Engineering function.

   Such a board would not have the project manager nor configuration management participation. As the project matures, particularly after PDR, there will be formal change control under a CCB.

   The changes to the Baselines are from a variety of sources during the life of the project. Some examples of the various sources of the changes are:

   a. The customer, who may impose new requirements or an Engineering Change Proposal (ECP);

b. Inputs, due to new relevant technological changes which can be incorporated in a cost-effective manner;
c. Internally-generated changes due to configuration management or design processes (e.g., part no. change); and
d. Internally-generated changes due to derived requirements.

2. Establish the process for Systems Engineering/configuration management. Once the nature of the changes is understood, a Baseline Maintenance or change process can be established. At this point one of the approaches discussed in Step 1 will be implemented, and a plan established for future changes as the project matures. This plan should be a part of the SEMP. The Configuration Management Plan is fully discussed in the section above.

5.2.4 REQUIREMENTS AND DESIGN LOOPS

Design is the process of defining, selecting, and describing solutions to requirements in terms of products and processes. A design describes the solution (conceptual, preliminary or detailed) to the requirements of the system.

Synthesis is the translation of input requirements (including performance, function and interface) into possible solutions satisfying those inputs. Synthesis defines a physical architecture of people, product and process solutions for logical groupings of requirements and then designs those solutions.

Stakeholders Participation

The key participants in carrying out the Requirements and Design feedback loops are Systems Engineering, with design, manufacturing, specialty engineering and materials and processes engineering. The function of Systems Engineering is to ensure that the proper inputs and feedback to hardware and software are occurring at the system, subsystem and component levels.

The result of the Systems Engineering process in these loops is system and subsystem designs that are properly allocated to hardware and software and thoroughly audited to ensure that they meet requirements and are concurrent with established manufacturing practices.

Recommended Activities

The following steps describe the Systems Engineering activities in achieving Requirements and Design feedback loops.

Determine how SE process is tailored for different levels of the project. This is a Systems Engineering task, and is performed in conjunction with project management. It determines the amount and detail of Systems Engineering to be performed at each level. This should be established early in the project and is covered in the SEMP.

Audit the system design. To provide feedback to the requirements and design functions, an audit of the design is performed to ensure that requirements are met. Audits occur at various levels, from drawing reviews against requirements in specifications, to design reviews, both informal and formal. The results of the audits serve as feedback to previous SE activities. These results may cause changes in requirements at any level, or may cause design changes. It will also include assessment of the SE role to manage the functions; how do they differ at the different levels; how are they the same. How does the SE process change, including the amount of conceptual system analysis; risk analysis/assessment; spec tree formulation, differences in reviews; verification?

Iterate between systems (hardware and software), design, and manufacturing functions. Systems Engineering should ensure that concurrent engineering is taking place. This may be by chairing a Product Development Team, or by being a member of one. In either case, it is the responsibility of Systems Engineering to ensure that all necessary disciplines in the project are participating in any phase. Systems Engineering consults on all phases of the project to provide the traceability and flow of the customer's needs and requirements. As necessary, Systems Engineering will conduct producibility meetings (determine production methods and materials) and will conduct producibility trade studies.

Audit the design and manufacturing process. After CDR, Systems Engineering will perform audits on the design (hardware and software) and manufacturing process to demonstrate compliance with requirements.

Iterate with other parts of the Systems Engineering engine. As stated above, Systems Engineering will ensure that all the elements of the Systems Engineering engine are executed.

Interface with specialty engineering groups and subcontractors to ensure common understanding across disciplines. This is part of the Systems Engineering role in ensuring that concurrent engineering is being performed on the project.

Update models as better data becomes available. Systems Engineering should always ensure that models are updated within the discipline. The models will be databases for traceability, trades, and verification.

Input

- Results of Requirements Analysis and Functional Analysis steps; i.e., requirements flowed down to lowest levels.
- Project baseline, proposed changes (initiated by customer or internally from requirements and design analyses, new technology or test results).
- Between project phases, an input to the new phase is the Process Output from the previous phase. The process outputs include an audit trail of requirements, designs, and decisions.

Output

- New project baseline, which consists of requirements, specifications and designs which comply with requirements.
- A design, which has been audited to ensure compliance with requirements, and which can be produced.

Criteria for Successful Completion

- Successfully pass ERB and/or CCB.
- Completion of design audits

Methods/Techniques

Performance of standard configuration management processes will document a concurrent baseline that is consistent with the output of the project. Alternatively, create a baseline document, which contains drawings, specifications, published analyses, and deliverable documents that show the current baseline. Also, ensure that all internal and external interfaces and interactions are included.

Tools

Functional analysis tools (e.g., N^2 charts, functional flow diagrams, IDEF0/1 diagrams); Concurrent Engineering tools; and Traceability database; and Systems Engineering Management Plan (SEMP)

5.3 TAILORING THE PROCESS

Tailoring the Systems Engineering process consists of identifying the specific process activities and interim products appropriate to the specific project being planned or re-planned. Tailoring focuses on the ***products*** to be produced, the ***reviews*** to be conducted, the ***depth of detail*** of analysis anticipated in each Systems Engineering process activity, the ***formality*** of the conduct of the process, and ***the number of iterations planned*** through process activities which are closely related.

The Systems Engineering process contains key process activities, which experienced Systems Engineers agree should ***always*** be performed. However, the mapping of these process activities to the project/product life cycle can vary substantially (see Section 3). The time, energy, and effort devoted to each should reflect the economics and risks of the project being addressed. A trade study on one project might take several people months and require many reports to document, while on another project all trade studies might be completed in an afternoon, and be minimally documented.

Purpose of Tailoring

The purpose of tailoring the Systems Engineering process for a specific project is to ensure that the appropriate amount of Systems Engineering effort is devoted to the appropriate activities to ***reduce project risk to an acceptable level*** while at the same time ***making most cost-effective use of engineering resources***. It is often difficult to determine exactly how much Systems Engineering is "enough" on a given project (except in hindsight). A general guideline, however, is that enough Systems Engineering should be performed on a project to ensure that the system, its requirements, configuration, and performance are well-defined and verified; that all engineering risks have been identified and assessed; and that engineering resources in appropriate engineering disciplines (including Systems Engineering) are allocated throughout the project to deliver the required products and keep schedule, cost, and technical risks at an acceptable and cost-effective level.

Stakeholders/Participants

Since determining of how much Systems Engineering is "enough" requires judgment about the technical complexity of the project and how much risk and cost are acceptable, both management and Systems Engineering should participate in tailoring decisions. In general, management determines the level of risk and cost acceptable. Often, significant communication and/or negotiation efforts between management and Systems Engineering are required to clarify the complex tradeoffs between manning, tasks, risks, and cost impact. Many projects begin with unrealistically low levels of Systems Engineering effort, not realizing the adverse impact potential on risk and final cost.

Unrealistic expectations can lead to:

- Too little Systems Engineering effort expended near the beginning of the project, resulting in poorly-defined requirements, interfaces, and subsystem tradeoffs, leading to delays in prime item specifications and poor subsystem designs.

- Non-recognition of excessive risks in some subsystem developments and inadequate contingency planning and integration, leading to redesigns.

- Distribution of resources between different engineering disciplines is not consistent with their development tasks. By the time needs are better understood the resources have mostly been expended.

- Serious technical problems encountered during system integration and test or during the transition from design to production and support, leading to costly rework and schedule delays.

It is the responsibility of Systems Engineering personnel to translate Systems Engineering concerns about the project into terms that can be used as the basis for making good business decisions.

It is the responsibility of management and Systems Engineering to do enough analysis and planning to ensure that project costs and risks are well-enough understood, and to ensure that informed decisions are made.

Recommended Activities

The steps in tailoring the Systems Engineering process for a project are:

1. Identify the Systems Engineering process baseline from which tailoring is done. For organizations with a high level of Systems Engineering maturity (see Section 7), this is the documented Systems Engineering process on which the organization has standardized. It has been refined based on lessons learned from previous projects. Organizations that do not have a documented, standard Systems Engineering process must define one. This process handbook and appropriate Systems Engineering standards are a good place to start. Some recent draft/interim Systems Engineering standards have been published by the Electronic Industries Association and the IEEE. See Section 1.3 for more information.

2. Determine the cost targets and risk tolerance of the project. If the project goals are unachievable at an acceptable cost and risk level, the acceptable combination of project goals, costs and risk level must be negotiated until it is acceptable to management and seen as achievable by Systems Engineers.

3. Characterize what other engineering disciplines on the project will need from Systems Engineering. This, together with the size of the total engineering team, will determine the type and content of the products which Systems Engineering needs to produce for the engineering effort to be a success.

4. Identify the deliverable documents for which Systems Engineering is responsible. Also identify any other products that are in the baseline Systems Engineering process (see Step 1 above), which cannot be tailored out per any tailoring guidelines of the organization.

5. For each Systems Engineering product identified in Steps 3 and 4, identify the ***form*** the product should take, and the level of detail necessary to achieve the purpose of each product. This can often be done by citing examples of products from previous projects to give team members a common understanding of both the format and the level of detail planned. On US DoD projects, a Data Item Description can provide the format of a document; the level of detail is typically best described by examples of the content for each section.

6. Assess whether any products, or their forms, or their level of detail (as determined in Step 5) are unaffordable given the project goals, cost targets, and level of tolerable risk (as determined in Step 2). In other words, look at what products are needed to enable the process to work well, given the

circumstances (the project team, their familiarity with engineering processes involved in the project, their familiarity with the applications and product area technology, suspected staff turnover, etc.). In general, the less experienced the team or the more likely personnel turnover is, the more explicit/formal the Systems Engineering products should be for the process to be successful. The basic purpose of most of the products of the Systems Engineering process is communication between engineering project team members about what to do and how to do it.

7. Identify the life-cycle process planned for the project. This gives guidance on the number of iterations through related processes that should be planned. If the project is part of a larger project, this may also clarify which Systems Engineering process activities may have been partially completed. The decisions about the number of iterations appropriate for the project depend on the goals and constraints of the project. For a project that has a design-to -cost goal, you may choose to iterate through process activities Section 4 several times to assure that all requirements that drive a design above cost targets are identified and modified.

8. Identify and assess the risks on the project. For each risk that Systems Engineering can affect, determine cost-effective actions required to bring the risk levels after mitigation to acceptable levels.

9. Identify the level of detail needed for each process activity. One way to do this is to use this handbook, and for each subject of Section 4 that describes a process activity, identify which sub-paragraphs apply. Another approach is to write down the purpose of the activity and the risks to the project if it is not done adequately, and then derive the level of detail needed to serve this purpose and avoid these risks. If this level of detail process activity is not affordable determine in which areas risks can be allowed to rise.

10. Document the tailoring planned to the baseline Systems Engineering process, and obtain approval. If no formal authorization is required by your organization, request an informal review of the proposed tailoring from a senior Systems Engineer who has experience with the same customer.

11. Document the planned Systems Engineering processes, products, and reviews (see the description of outputs below).Describe the completion criteria for each process.

Inputs

The key inputs to Systems Engineering process tailoring are:

1. the goals and constraints of the project,
2. organizational and contractual requirements for Systems Engineering process or products,
3. the baseline Systems Engineering process for the organization and any tailoring guidelines, and
4. any cost targets and the acceptable level of risk for the project.

Outputs

The primary output of the Systems Engineering tailoring process is a documented description of the Systems Engineering activities planned for the project. The form of this output will vary depending on the size, complexity, and acceptable cost/risk level of the project. Examples of acceptable output from the tailoring activity are shown in Table 5-2.

Table 5-2. Acceptable Outputs from the SE Tailoring Process

Project Characteristics	Sys Eng Process Tailoring Plan
Small, simple, low cost w/high risk tolerance	A simple, 2 - 4 page plan
Small, simple, low cost w/ moderate risk tolerance	More detailed, 5 - 10 page plan
Medium, not very complex, moderate cost w/ moderate risk tolerance	Tailored SEMP, including schedules of all engineering activities, in a 20 - 30 page plan
Large, complex, high cost, low to moderate risk tolerance	Full Sys. Eng. Mgt. Plan (SEMP) including processes, products, & reviews. 50+ pg.

Whatever the ***form*** of the description of the Systems Engineering process tailoring, the description needs to include:

1. Specific products to be produced by the Systems Engineering effort, including interim products for the use by the Systems Engineering team and products for the use by other engineering disciplines, management, and/or the customer;

2. Reviews to be conducted or supported by Systems Engineering, including the role of Systems Engineering in each review;

3. A description of the depth of analysis anticipated in each Systems Engineering process activity;

4. The number of iterations planned through process activities that are closely related. (For instance, is it anticipated that requirements will be re-examined once the initial system synthesis and design has been done? This is common when doing a design for a product to be manufactured in large quantity, where low per-unit product cost may be more important than specific requirements);

5. How formal the conduct of the Systems Engineering process should be. (That is, who needs to know about the completion of each Systems Engineering process activity, and how much of the process needs to be able to be reconstructed after the fact from the written/electronic record of process activities?);

6. The completion criteria for each process activity; and

7. An integrated schedule for all the above.

End Results

The most obvious criterion for successful completion of Systems Engineering process tailoring is that management and the lead Systems Engineer reach agreement on the amount and distribution (over time or life-cycle phases) of the resources to be allocated to Systems Engineering. Subsidiary criteria can include:

a. Systems Engineering has completed a project risk analysis, and all significant risks have been addressed (either by mitigation, or by agreement between management and Systems Engineering that living with the risk is acceptable because the cost of effective mitigation is too high);

b. All products, both interim and final, to be produced by Systems Engineering have been identified, and the level of detail in each has been agreed upon;

c. All reviews to be conducted or supported by Systems Engineering have been identified, and the content and level of detail expected in the reviews has been defined; and

d. A plan and schedule for Systems Engineering activities which includes the sequence of process activities and the resources to be applied to each, has been agreed upon.

Methods/Techniques

Two categories of metrics are appropriate for Systems Engineering process tailoring:

1. Measure progress through developing the initial tailoring approach/plan
2. Measure the appropriateness of the tailoring as the project progresses. This can signal when it is time to re-assess the tailoring and consider changing it to meet changing project circumstances or needs.

The criteria for successful completion above can be adapted into metrics measuring the initial tailoring. Additionally, some of the same metrics used for overall Systems Engineering process monitoring can be used to assess on-going appropriateness of tailoring.

Tools that can be used are scheduling programs, including those which implement PERT/CPM techniques, risk analysis tools (see Section 4.5.3), and process description tools (including those which implement IDEF models/descriptive techniques).

5.4 REFLECTING THE TAILORED PROCESS IN THE SEMP

Simply describe the tailored products that will be provided in the SEMP along with material normally included in the SEMP. To avoid confusion in responses to a formal SOW, include "(tailored response to SOW___)" after each tailored item. Items to be tailored should be agreed upon by the customer either in advance of RFP release or during negotiations. Care must be taken to not *surprise* the customer with a tailored response that might be considered ***non-responsive***.

THIS PAGE
INTENTIONALLY
LEFT BLANK

6 RISK MANAGEMENT

Risk management, in the context of Systems Engineering, is the recognition, assessment, and control of uncertainties that may result in schedule delays, cost overruns, performance problems, adverse environmental impacts, or other undesired consequences.

There are two main branches of risk management:

- Project risk management (PRM): the management of technical risks and task performance uncertainties associated with any development or production project, in order to meet performance, cost, and schedule objectives.

- Environmental risk management (ERM): the management of environmental, health and safety risks associated with the production, operation and disposal of systems, in order to minimize adverse impacts and assure sustainability of these systems.

These two types of risk management have different objectives, involve different skills, and require different methodologies. They are related to the extent that ERM should be considered as an integral part of system development, and therefore is incorporated into PRM. This section is focused primarily upon PRM, while many aspects of ERM are addressed in Section 11, under the sub-topics of Safety and Environmental Impact Analysis.

Function

Risk management must be an integral component of overall project management, and must be proactive in nature to ensure that undesirable consequences are anticipated as early as possible in the life of the project.

The functions of the risk management activities are to:

1. Identify the potential sources of risk and identify risk drivers;

2. Quantify risks, including both the probability of occurrence and seriousness of impact, and assess their impacts on cost (including life-cycle costs), schedule, and performance;

3. Determine the sensitivity of these risks to project, product, and process assumptions, and the degree of correlation among the risks;

4. Determine and evaluate alternative approaches to mitigate moderate and high risks;

5. Ensure that risk is factored into decisions on selection of specification requirements and design and solution alternatives; and

6. Take actions to avoid, control, assume, or transfer each risk, and adjust the Systems Engineering Management Plan (SEMP) and Systems Engineering Management Schedule (SEMS) as appropriate;

Objective

The objective of risk management is to ensure the delivery of a system and its associated processes that meet the customer's need on time and within budget. Risk management must address uncertainties both in products and processes, as well as their interrelationships.

The challenge of risk management is to achieve the proper balance between risk and reward. A reasonable level of risk can be accepted when the payoff is to achieve a valuable goal; the athletic motto "no pain, no gain" applies here as well. Thus, risk management in Systems Engineering should not attempt to avoid all risk.

Result

Effective risk management requires a rigorous framework, supported by a formal model such as probabilistic decision theory. Even qualitative judgments of likelihood should be meshed with this framework. The result of applying a risk management framework is improved insight into the uncertainties that may inhibit successful project completion, and improved capability to deal with these uncertainties.

Organizational Participation

Risk management is usually performed by a risk management organization or team with specific responsibility for carrying out the process. However, it is important that consciousness of risk management not be confined to that team. Risk management cannot be successful unless the proper environment is first established by the most senior project management; personnel on the project must be free (indeed encouraged) to identify risk, assess risk, and mitigate risk as they find it. At all costs, management must avoid creating a risk-denial atmosphere where "messengers are shot" for surfacing risks. It is imperative that everyone on the project feel free to openly discuss risk; risks which are hidden tend to result in unexpected consequences with the potential to impede and even damage the project at a later time.

Recommended Activities

Risk management involves five processes -- planning, identification, assessment, analysis, and mitigation. These are further elaborated in Section 6.1.2.

- Risk planning is the process of deciding in advance how risk will be managed, including the specification of the risk management process and organizational responsibilities.
- Risk identification is the process of recognizing potential risks and their root causes as early as possible and setting priorities for more detailed risk assessment.
- Risk assessment is the process of characterizing or quantifying those risks which merit attention.
- Risk analysis is the process of evaluating alternatives for handling the assessed risks. This includes performing "what if" studies.
- Risk handling is the process of dealing with a risk by choosing a specific course of action. Risk can be mitigated by choosing to avoid the risk (perhaps with a change in the design), to control the risk (perhaps with additional development resources), to assume the risk (expecting that the event can be overcome in normal development), or to transfer the risk (for example, with special contract provisions).

Risk management should be part of the project manager's toolbox during all project phases, including pre-project activities. The above steps should be carried out in a flexible, iterative manner, with resources focused on the risks of greatest concern. Finally, risk management should be considered an integral part of the concurrent engineering process, since resolution of risks in an early phase has the "leverage of time" and can be achieved at lower cost than during a later phase.

Input
The inputs to the risk management process include the Systems Engineering Management Plan and Schedule; the system requirements, functional flows, design approach, and system elements; ongoing status reports for all project activities; and lessons learned from prior projects.

Output
Appropriate documentation must be created to record the results of each of the five steps in the risk management process. This documentation will support management of the risk management activities at the time it is developed and will provide historical records and risk traceability for future project participants.

Metrics
In practice, a variety of semi-quantitative and qualitative metrics, including technical performance indicators, are used to support risk management (see Section 6.2.).

6.1 RISK CONCEPTS
Risk can be defined as “A measure of the uncertainty of attaining a goal, objective, or requirement pertaining to technical performance, cost, and schedule.”

Risk always is present in the life cycle of systems. The system may be intended for technical accomplishments near the limits of the state of the art, creating technical risk. System development may be rushed to deploy the system as soon as possible to meet an imminent threat, leading to schedule risk. All systems are funding-limited so that cost risk is present. Risk can be introduced by external constraints or can develop from within the project, since technical risk can create schedule risk, which in turn can create cost risk.

There is no alternative to the presence of risk in system development. The only way to remove risk is to set technical goals very low, to stretch the schedule, and to supply unlimited funds. None of these events happen in the real world. No realistic project can be planned without risk. The challenge is to define the system and the project which best meet overall requirements, which allow for risk, and which achieve the highest chances of project success.

Fundamentals

Risk has two components -- the likelihood that an undesirable event will occur and the consequence of the event if it does occur. The likelihood that an undesirable event will occur often is expressed as a probability. The consequence of the event is expressed in terms that depend on the nature of the event (e.g., dollars, performance loss). These two components are illustrated in Figure 6-1. The combination of low likelihood and benign consequences gives low risk, while high risk is produced by high likelihood and severe consequences.

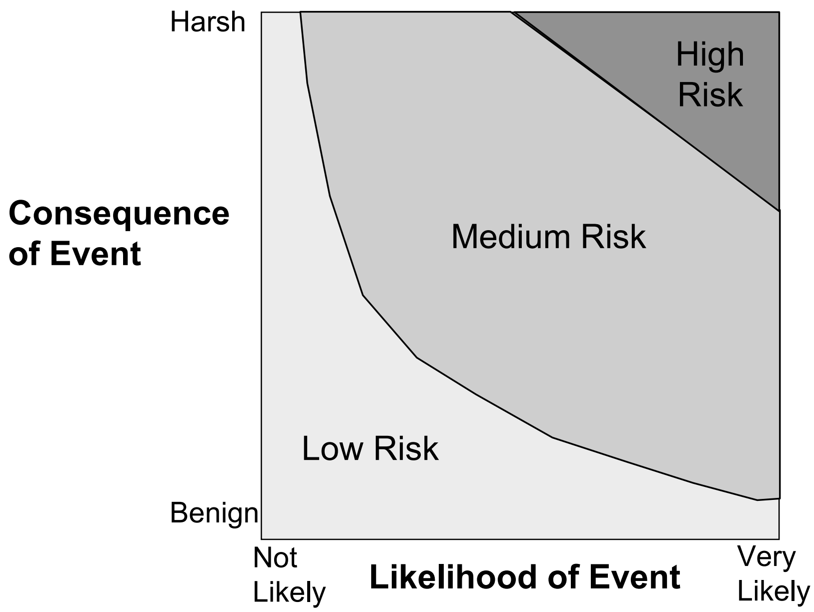


Figure 6-1. Level of Risk Depends Upon both Likelihood and Consequences

Air transport provides two examples of events and their consequences – the event of arriving at the destination 15 minutes late usually has benign consequences, while the event of an airplane crash has harsh consequences and possible loss and injury. Most people would judge both of these events to have low risk; the likelihood of arriving 15 minutes late is high but the consequences are not serious. On the other hand, the consequences of a crash are very serious but are offset by the low likelihood that such an event will occur.

Risk Categories

There are at least four categories of risk that can be distinguished:
1. technical
2. cost
3. schedule and
4. programmatic

Supportability is often included as a separate additional category.

Technical risk is the possibility that a technical requirement of the system may not be achieved in the system life cycle. Technical risk exists if the system may fail to achieve performance requirements. These performance requirements can be expressed in terms of distance, velocity, time, throughput, signal-to-noise ratio, mean-time-to-failure, required processor memory, or whatever parameters appropriately describe the performance and effectiveness of the specific system. Technical risk also exists if the system may fail to meet operability or producibility requirements or if it may fail to meet testability requirements. Technical risk also exists if the system may fail to meet integration requirements or environmental protection requirements. A potential failure to meet any requirement which can be expressed in technical terms is a source of technical risk.

Cost risk is the possibility that available budget will be exceeded. Cost risk exists if the project must devote more resources than planned to achieve technical requirements or if the project must add resources to support slipped schedules due to any reason. Cost risk exists if changes must be made to the number of items to be produced or if changes occur in the national economy. Cost risk can be predicted at the total project level or for some element of the work breakdown structure. The collective effects of lower-level cost risks can produce cost risk for the total project.

Schedule risk is the possibility that the project will fail to meet scheduled milestones. Schedule risk exists if there is inadequate allowance for piece-part procurement times. Schedule risk exists if difficulty is experienced in achieving scheduled technical accomplishments, such as the development of software. Schedule risk can be incurred at the total project level for milestones such as deployment of the first unit, or can be incurred at a lower element of the work breakdown structure. The cascading effects of lower-level schedule risks can produce schedule risk for the total project.

Programmatic risk is produced by events which are beyond the control of the project manager. These events often are produced by decisions made by personnel at higher levels of authority. Programmatic risks can be produced by reductions in project priority, by delays in receiving authorization to proceed with a project, by reduced or delayed funding, by changes in national objectives, etc. Programmatic risk can be a source of risk in any of the other three risk categories.

Figure 6-2 illustrates major relations among the four risk categories. The arrow names indicate typical risk relationships; others certainly are possible.

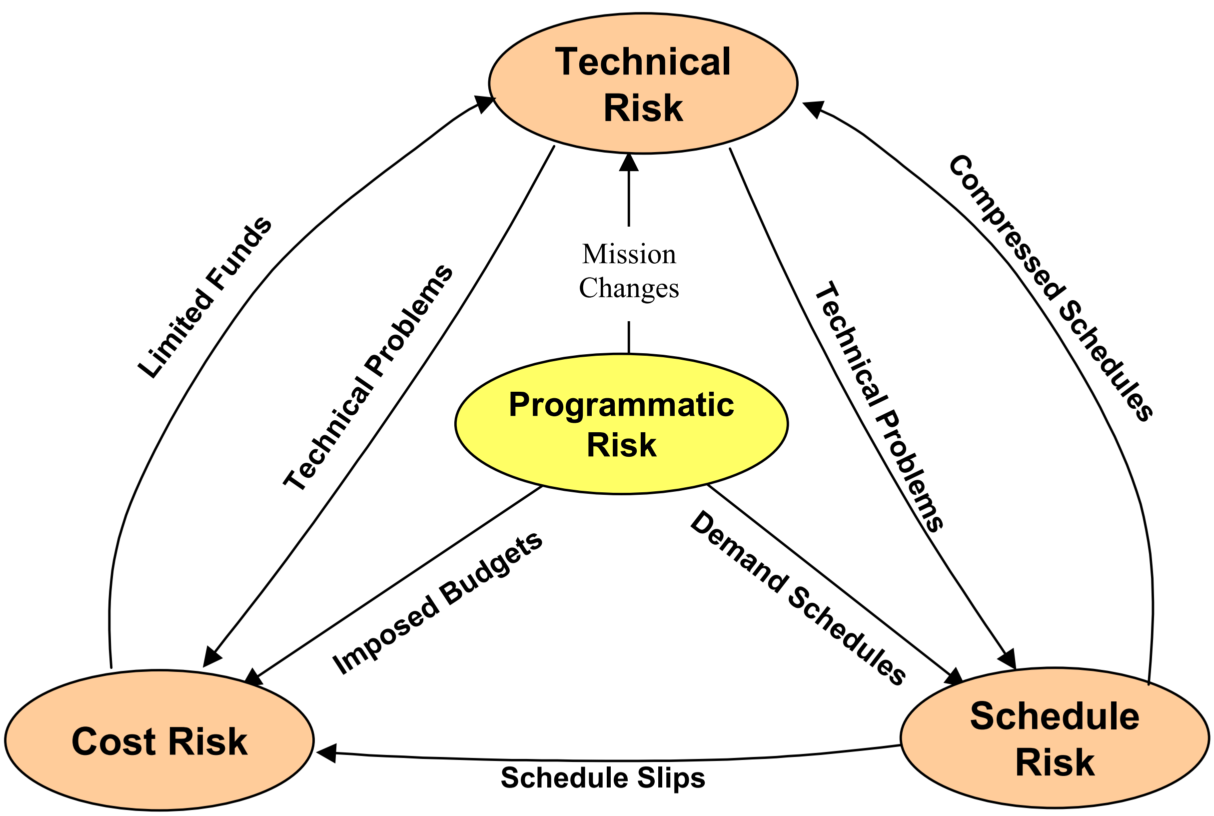


Figure 6-2. Typical Relationship among the Risk Categories

6.2 THE RISK MANAGEMENT PROCESS

As is depicted in Figure 6-3, risk management is an iterative process consisting of five primary activities: Planning, Risk Identification, Risk Assessment, Risk Analysis and Risk Handling.

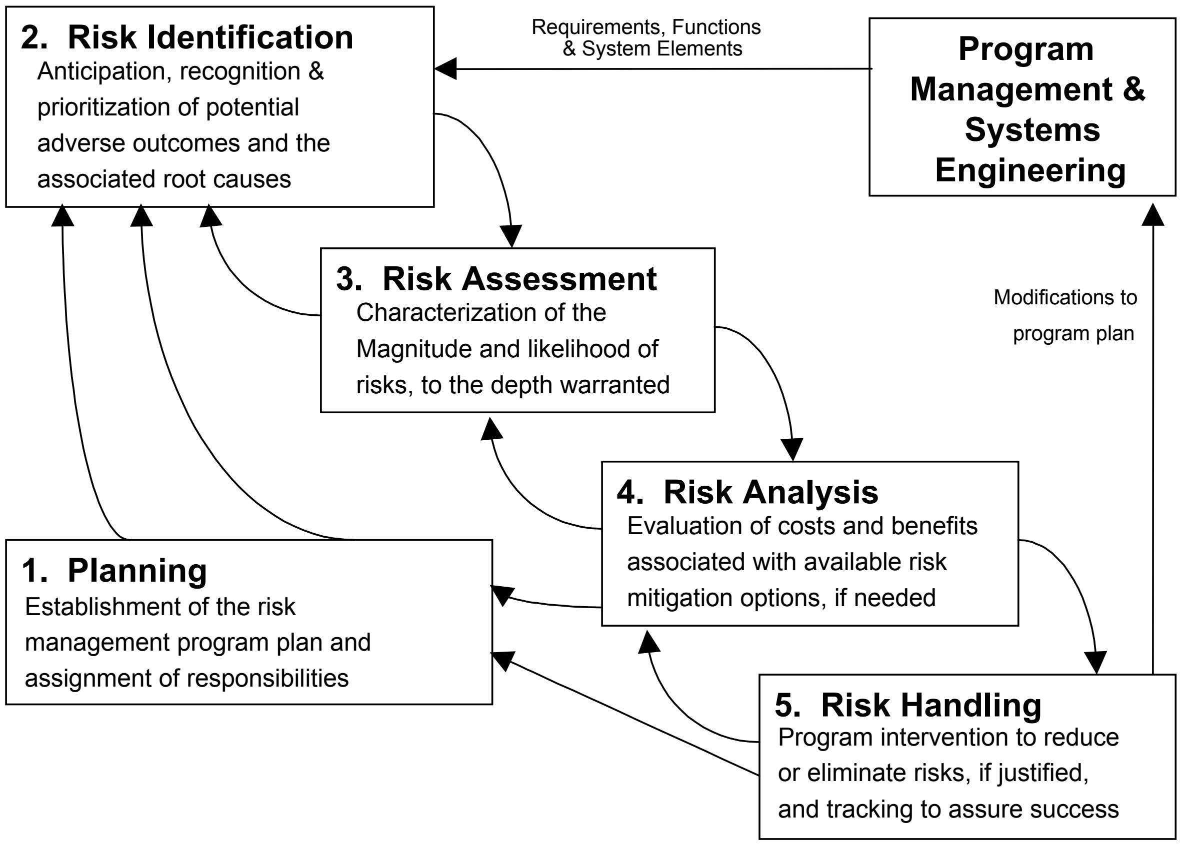


Figure 6-3. Five Steps in the Risk Management Process

The five steps in the risk management process must be practiced iteratively, with a continual exchange of information between project management and Systems Engineering.

Planning. The purpose is to establish a plan for risk management that encompasses the entire project so that the project can achieve and maintain an acceptable level of risk. The results should be included in the Systems Engineering Management Plan and risk should be considered in defining the Systems Engineering Management Schedule. Risk plans and reports are discussed in Section 6.2.4.

Risk Identification. The purpose is to screen the architecture and requirements and identify the medium and high-risk elements. The basis for this evaluation may be qualitative or quantitative using evaluation criteria to determine technical, cost, and schedule risk factors of the candidate technologies and design options. The process is further detailed in Section 6.2.1.

Risk Assessment. The purpose is to determine the probability of failure of the possible outcomes and the consequences of failure for those risks considered sufficiently important. Risk metrics can then computed from these factors. The result of this analysis is a collection of elements that are deemed to be at risk. The process is detailed in Section 6.2.2.

Risk Analysis - The goal is to determine how to best reduce the risk for each of the identified moderate and high risk elements such that, given any number of outcomes, the product and its associated processes will converge to a low-risk solution by the time the product enters production. For each risk associated with technical maturity, a mitigation plan must be derived that will apply a series of tests and/or analyses to reduce that risk. Similarly for cost and schedule risks, plans are established to consider possible options that may affect the selection of alternative architectures. Trades will have to be made regarding the technical performance, cost, and schedule objectives to achieve the proper balance. This process is further detailed in Section 6.2.3.

Risk Handling - The Risk Handling activities include monitoring the risk processes and assigning resources to the mitigation activities. Close coupling between the risk management activities and the test and evaluation activities have to be made such that engineering results from test and analyses can be used to update risk reduction estimates. The job of the Risk Manager/Organization is to ensure that there is closure in the process of reducing risk and that a clear course of action can be taken to steer the design toward a low risk solution. Recovery planning may need to be considered that will allow prudent decisions to be made. This process is further detailed in Section 6.2.3.

6.2.1 RISK IDENTIFICATION METHODS

The process of identifying risk should include all or most of a project's personnel. The project work breakdown structure provides a framework that can be used to help ensure that all risk areas are considered. Critical parameters and requirements with large variances and

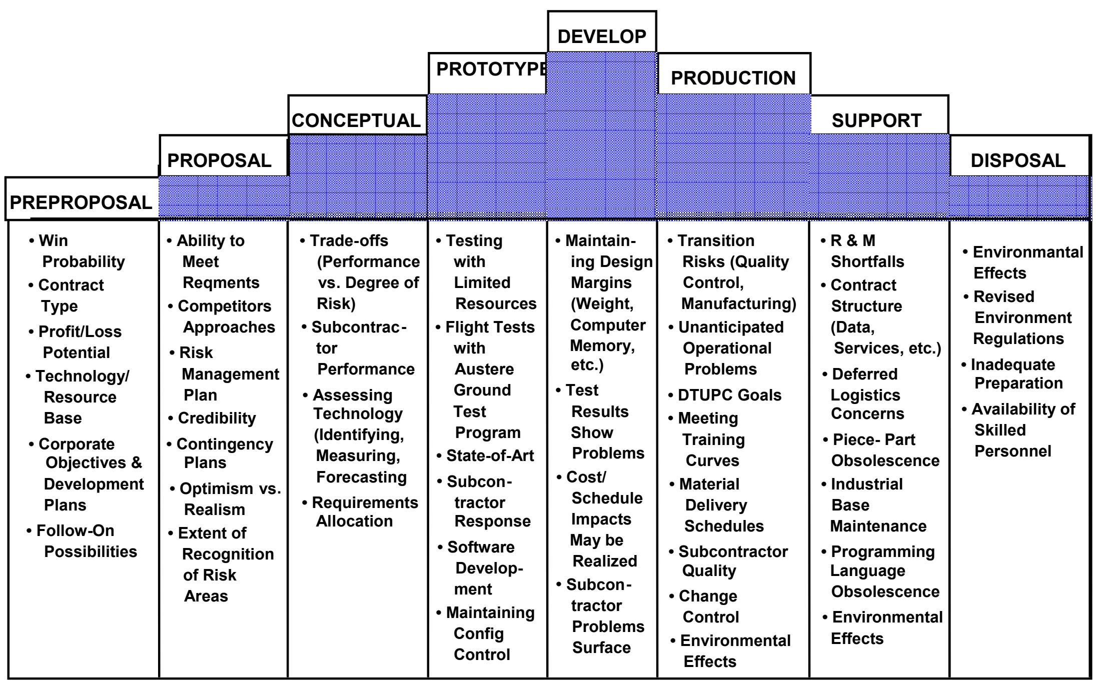


Figure 6-4. Risk Considerations By Project Phase
Analogy Comparison/Lessons Learned Technique

potentially serious consequences should be tracked via Technical Performance Measures (TPM). Engineers and management will then be able to set priorities and focus on the areas of risk that deserve the most attention. Figure 6-4 illustrates typical risk considerations that can be encountered as a project progresses from earliest planning through disposal.

The analogy comparison or lessons learned technique for risk identification and assessment is based on the idea that no new project is entirely unique. Many new projects simply represent a new combination of existing components or subsystems. Others evolve from existing projects with incremental advances in technology. This means that key insights can be gained concerning a current project's risk by examining the successes, failures, problems, and solutions of similar prior projects. The experience and knowledge gained, or lessons learned, can be applied to identify potential risk in a new project and to develop a strategy for risk management.

The first step is to determine the information needs in this phase of risk management. This could vary from assessing the risk in development of a custom computer chip to identifying the risks associated with a major system development. The second step is to define the basic characteristics of the new system. This is necessary to identify past projects that are similar in technology, function, design, etc. Then, based on the availability of data, analogous systems or subsystems are selected and data gathered. Often the data collection process and initial assessment lead to a further definition of the system for the purposes of comparison. After this has been accomplished, the last step in the process is the analysis and normalization of the historic data. Comparisons to prior systems may not be exact or the data may need to be adjusted to be used as a basis for estimating the future. The desired output is insight into cost, schedule, and technical risks of a project based on observations of similar past projects.

References

AT&T Engineering Guides for Managing Risk: Design to Reduce Technical Risk; Design's Impact on Logistics; Moving a Design into Production; Testing to Verify Design and Manufacturing Readiness, from the McGraw-Hill Professional Book Group (1-800-842-3075).

NAVSO P6071, *Best Practices: How to Avoid Surprises in the World's Most Complicated Technical Process,* March 1986.

Shaw, T.E. and Lake, J. G., Ph.D., "Systems Engineering: The Critical Product Development Enabler," APICS Conference, April 1993.

USAF Material Command, "Risk Management Pamphlet", Integrated Performance-Based Business Environment Guide. January 1997.

6.2.2 RISK ASSESSMENT METHODS

Risk involves both the probability and consequences of the possible outcomes. Although risk is intuitively familiar to most people, it is a complex and difficult concept to assess. Risk is associated with uncertainty, which is characterized by a distribution of outcomes based on likelihood of occurrence and severity of consequences. In its most general form, risk assessment should capture the spectrum of outcomes relative to the desired project technical performance, cost, and schedule requirements. Risk generally needs to be assessed subjectively because adequate statistical data are rarely available.

Expert Interviews

Efficient acquisition of expert judgments is extremely important to the overall accuracy of the risk management effort. The expert interview technique consists of identifying the appropriate experts, questioning them about the risks in their area of expertise, and quantifying these subjective judgments. Expert interviews nearly always result in information that can be used in the formulation of a watchlist. In fact, watchlists frequently evolve from the input of each expert functional manager on a project. Another useful output is the formulation of a range of uncertainty or a probability density function (with respect to cost, schedule, or performance) for use in any of several risk analysis tools.

Since expert interviews result in a collection of subjective judgments, the only real "error" can be in the methodology for collecting the data. If it can be shown that the techniques for collecting the data are not adequate, then the entire risk assessment can become questionable. For this reason, the methodology used to collect the data must be thoroughly documented and defensible. Experience and skill are required to encourage the expert to divulge information in the right format. Typical problems encountered include identification of the wrong expert, obtaining poor quality information, unwillingness of the expert to share information, changing opinions, getting biased viewpoints, obtaining only one perspective, and conflicting judgments. When conducted properly, the expert interviews provide very reliable qualitative information. However, the transformation of that qualitative information into quantitative distributions or other measures depends on the skill of the analyst.

Estimating Relationships

The estimating relationship method enables project office personnel to evaluate a project and then use an equation to determine an appropriate management reserve or risk funds budget. The management reserve funds represent the amount of funding required for work associated with unanticipated risks. The management reserve funds requirement is usually expressed as a percentage of the Baseline Cost Estimate (BCE). This technique is called an estimating relationship method because it uses the same principles associated with Cost Estimating Relationships (CERs), used in parametric cost estimating. The method is based on the observation that costs of systems correlate with design or performance variables. The independent variables, often called explanatory variables, are analyzed using regression analysis. The analysis characterizes the underlying mechanism relating such variables to cost.

Life Cycle Cost Analysis

Life cycle cost analysis encompasses the defined life of a given system. Life cycle cost analyses provide a basis for examining implicit risks associated with various programmatic decisions--for example, low initial funding increasing design risk; low funding for research, development, test, evaluation and production translating to higher maintenance costs; or expensive maintenance diagnostic equipment resulting in low maintenance personnel costs. Life cycle cost analysis is discussed in more detail under Risk Avoidance.

Risk Models

Risk is often expressed only in qualitative terms or by a single value. However, it is very important to quantify risk in some methodical way to assure a good allocation of resources for risk reduction. Ideally, risk would be characterized by using cumulative probability curves with the probability of failure and the consequences expressed quantitatively in measurable terms, but given the inherent lack of data and limited analysis, this is usually impractical. Several methods exist for quantifying and ordering relatively subjective assessments, three are described below. It is very important to properly quantify risk because an invalid assessment could lead to an improper conclusion with misapplication of resources.

Expected Value Model - A somewhat subjective, relative rating of risk is developed, where risk is expressed as:

Expected consequence = Probability of failure (P_f)* Consequences of failure (C_f).

For illustration purposes, consider a proposal to develop a new light-weight and compact power supply with an operating life of 8,000 hours. The consequences of failing to meet at least 6,000 hours are assessed to be catastrophic/critical, so the consequence of failure is assigned a value of 0.8. Given the present state of technology, cost and schedule, the probability of failing to achieve an operating life of 6,000 hours is judged to be relatively low and is estimated as 30% (0.3).

Applying the equation to the above example yields

Risk = 0.3*0.8 = 0.24,

This would suggest a relatively low risk situation. Intuitively, the described scenario represents a low/moderate risk (subjective judgment); therefore this approach appears to yield a valid relative ranking of risk.

Risk Assessment Matrix Model - For communication purposes, it is often preferable to limit oneself to qualitative estimates of P_f and C_f rather than the arbitrary scales employed above. The Risk Assessment Matrix in Table 6-1 is recommended. Risk is determined by its location in the matrix, which is established by the intersection of the row and column corresponding to the severity of the consequences and the associated probability.

Applying the Risk Assessment Matrix to the previous example (Consequences = critical, Probability = improbable) yields

Risk = 0.30 (low), which is consistent with the result above.

The Risk Assessment Matrix has several attributes of merit:

- Risk increases with increasing probability of failure and the severity of the consequence of failure.
- Low probability of failure or low consequences of failure result in low to medium risk.
- It avoids the use of an arbitrary numerical scale.

Table 6-1. Risk Assessment Matrix

CONSEQUENCES	FREQUENT (HIGH)* 0.7<P<1.0	PROBABLE (MEDIUM)* 0.4<P<0.7	IMPROBABLE (LOW)* 0<P<0.4	IMPOSSIBLE P = 0
CATASTROPHIC 1.0 - 0.9	0.9 HIGH	0.7	0.4	0.0
CRITICAL 0.8 - 0.7	0.8	0.6 MEDIUM	0.3	0.0 NONE
MARGINAL 0.6 - 0.4	0.6	0.4	0.2 LOW	0.0
NEGLIGABLE 0.3 - 0.0	0.3	0.2	0.1	0.0

* Additional terminology, not in US Air Force Guide on Software Risk Abatement
Note: Risk rating is consistent with R = P*C

Risk Profiles - The Expected Value and Risk Assessment Matrix models, however, are limited by the fact that a single number fails to fully capture the notion of risk. The single rating equates a low probability, high adverse outcome event with a high probability, low adverse outcome event. They do not allow consideration of multiple outcomes with different probabilities. Whenever possible, risk should be characterized by its risk profile, which is defined as the probability that the magnitude of consequences will exceed a given value.

The generation of a risk profile involves quantifying the various outcomes and the associated probabilities. A major problem is the difficulty in obtaining adequate data. To illustrate the generation of a risk profile, consider again the project to develop a light-weight, low-cost, and long-life power supply. The experts' assessment of the achievable performance given the technological options is summarized in Table 6-2. The corresponding performance profile is shown in Figure 6-5.

Table 6-2. Achievable Performance vs. Technology Option

Operating Life (Hours)	Probability of not achieving	Technology Option
5,000	0.0	Off-The-Shelf
6,000	0.3	Minor Redesign
8,000*	0.6	Moderate/Significant Complexity Increase
10,000	0.8	State-of-the-Art
12,000	0.9	Major Technology Breakthrough Required

* Requirement

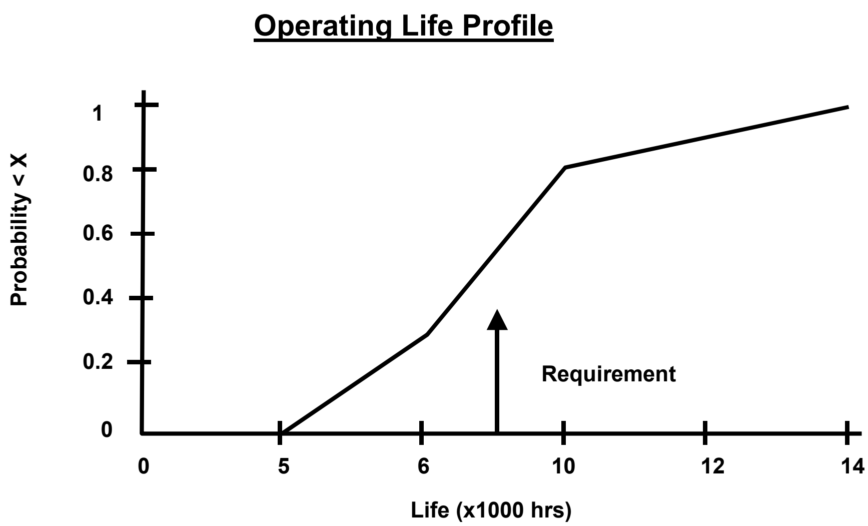


Figure 6-5. Performance Profile

Since the power supply requirement is 8000 hours and the probability of getting less than 5000 hours goes to zero, the possible shortages are between 0 and 3000 hours. A risk profile can be generated from the Performance Profile showing the probability density function (though not normalized) of the shortage. See Figure 6-6.

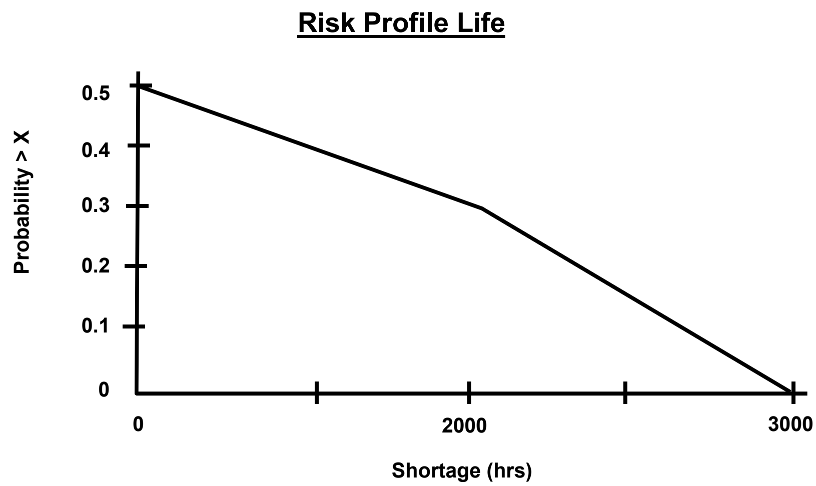


Figure 6-6. Risk Profile

6.2.3 Risk Analysis and Handling Methods

Risk handling approaches need to be established for the moderate and high-risk items identified in the risk assessment effort. Each risk item should be analyzed to select the most effective approaches for resolving them. These activities are formalized in the Risk Management Project Plan (RMPP, see Section 6.2.4.1). Each project needs to develop a project-specific risk plan and establish a strong Risk Management Organization to ensure its successful implementation.

Table 6-3 shows an example of a risk-resolution checklist for a hardware/software project. This table is project specific; but it illustrates sources of project risk and appropriate approaches for resolving them. Projects often generate a "top ten list"; however, the list of risk items/tasks may contain more or fewer items depending on the complexity of the project and the WBS level down to which the risk elements are being tracked.

Table 6-3. Sample List of Most Sensitive Tasks

TASK[1]	REASON FOR RISK	RISK HANDLING [2]
Control System Launch Weight	• Low Weight Margin for Phase 1 • Potential for Weight Growth	• Establish Weight Allocation and Control Board (1) • Select Lightweight Structure (2) • Contingency Plans: Ultra-Light Structure; Reduced Sensor Telescope Length (3) • Revisit Requirements for Peak Power with Customer (2)
Develop RF Power Source	• Klystrons too Heavy • Solid-state Devices Require Significant Scaling	• Proceed with Parallel Developments (3) • Incentives for Suppliers (1)
Auto-Operation and Control	• Only Partially Accomplished on Ground • Software & Hardware Design Not Mature	• Early Development Test Unit (DTU) & Software Prototype (3) • Maximize Technology Transfer from Laboratory (2) • Extensive Modeling and Sim. (3) • Provide Manual Control with Override as Backups (3)
Deliver Qualification Unit on Schedule	• Long-Lead Design and Procurement Required • Little Schedule Slack to Precede Flight Unit	• Request Long-Lead Funding and Procurement (3) • Initiate Some Development prior to Phase 1 (3) • Upgrade Unit as Late Parts are Delivered (1) • Carry Forward Alternate Designs until PDR (3)

Notes: 1. This table is program specific.
2. The risk handling approaches are: (1) Risk Tracking and Control; (2) Risk Avoidance; and (3) Risk Mitigation/Abatement

6.2.3.1 RISK TRACKING AND CONTROL

The Risk Management Organization should have the power and tools to ensure that the risk activities are properly implemented and resources appropriately allocated. The Risk Management Organization draws upon all project technical/management personnel and resources to perform the risk tracking and control of performance against specified requirements and cost/schedule against budget. These activities need to be correlated with the SEMP, the WBS, and cost and schedule.

Project management needs metrics to simplify and illuminate the risk management process. Each risk category has certain indicators, which may be used to monitor project progress for signs of risk. Tracking the progress of key system technical parameters can be used as an indicator of technical risk (see Figure 6-7).

The typical format in tracking technical performance is a graph of a planned value of a key parameter plotted against calendar time. A second contour showing actual value achieved is included in the same graph. Cost and schedule risk are monitored using the products of the Cost/Schedule Control System or some equivalent technique. Normally cost and schedule variances are used, along with a comparison of tasks planned to tasks accomplished.

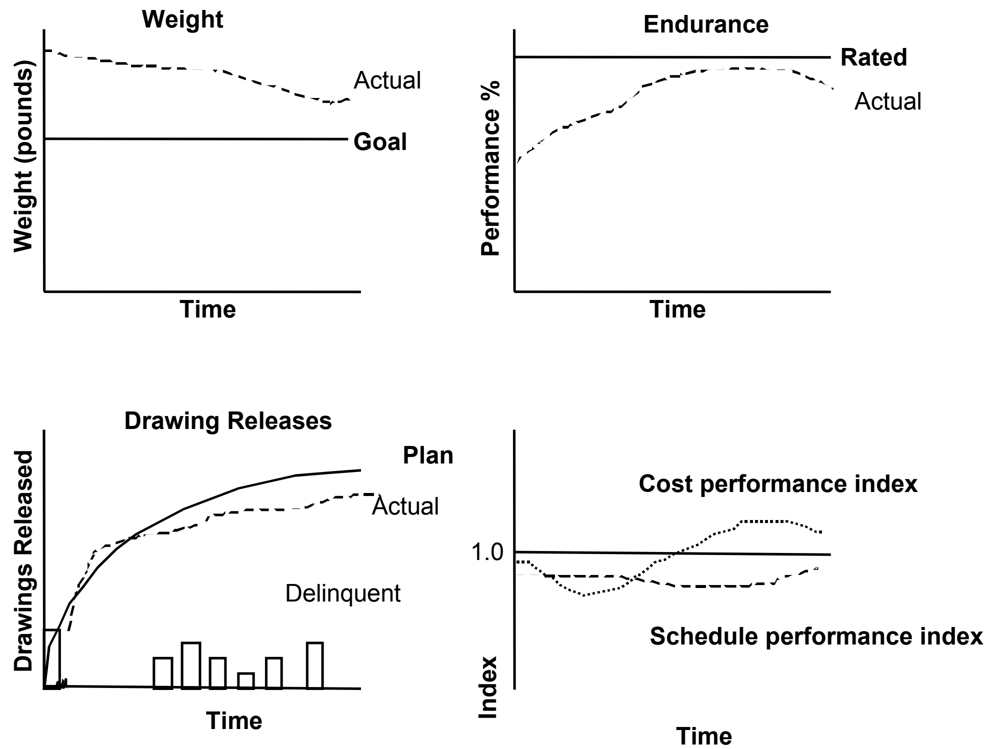


Figure 6-7. Risk Management Uses Metrics To Track Project Evolution

Design Engineering implements the hardware and software development projects for risk reduction. Systems Engineering continuously monitors risk items and performance parameters. Cost and schedule are monitored by Cost/Schedule Project Control. Appropriate surveillance and control of subcontractors and suppliers is provided by Subcontract Management. All these activities need to be integrated to ensure compatibility with the overall risk reduction effort. Results of the activities are documented and reported on a regular basis (weekly status reporting is recommended) to management and at scheduled design reviews.

6.2.3.2 RISK AVOIDANCE

Risk is inherent in every challenging project; the key is to select or set realistic performance, cost and schedule requirements. To quote General Patton: “Take calculated risks. That is quite different from being brash.” The “no-risk approach” option is rarely available. Instead, there are many situations where major risks can be avoided through the techniques summarized below.

Requirements scrubbing. The requirements should be analyzed to identify requirements of marginal value and these should be eliminated or scrubbed if they significantly complicate the hardware or software.

Selection of most promising options. In most situations several options are available, and a trade study should be performed to select the most promising one.

Staffing and team building. A proactive approach should be taken to avoid the risk of personnel shortfalls.

6.2.3.3 RISK MITIGATION/ABATEMENT

For high-risk technical tasks, control and avoidance often need to be supplemented by the following approaches:

- Early initiation of development activities
- Initiation of parallel developments
- Implementation of extensive analysis and testing
- Contingency planning

The high-risk technical tasks also involve high schedule and cost risks. Cost and schedule are impacted if technical difficulties arise and the tasks are not achieved as planned. Schedule risk is controlled by early development and procurement of long-lead items and provisions for parallel-path developments. However, these activities also result in increased early costs. Testing and analysis can provide useful data in support of key decision points. Finally, contingency planning involves weighing alternative risk mitigation options.

Decision analysis, as detailed in Appendix D-10, provides a simple and systematic framework for quantifying the potential outcomes and associated risks in support of contingency planning. To illustrate, consider the previously discussed project to develop a power supply. Two options are available – Option A and Option B. To mitigate risk, the Project Manager decides to initiate early development activities to support the selection process. The probability of success, cost, and schedule risks associated with technical risk can be quantified using decision trees. The resulting decision tree is shown in Figure 6-8; the associated cost and schedule cumulative probability curves or profiles are shown in Figures 6-9.

Given the budgeted cost and schedule the standard cost and schedule risk profiles can readily be generated from these curves. It should be noted that this approach using decision trees supplements is not a substitute for probabilistic risk network tools (RISNET, VERT, etc.).

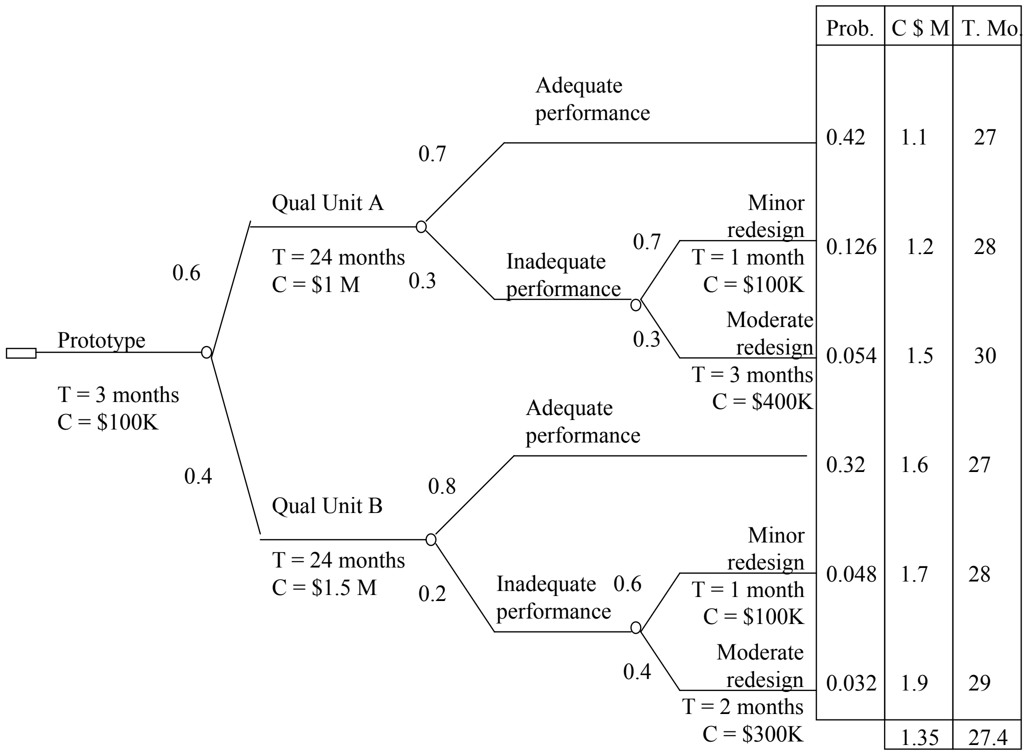


Figure 6-8. Decision Tree for Contingency Planning

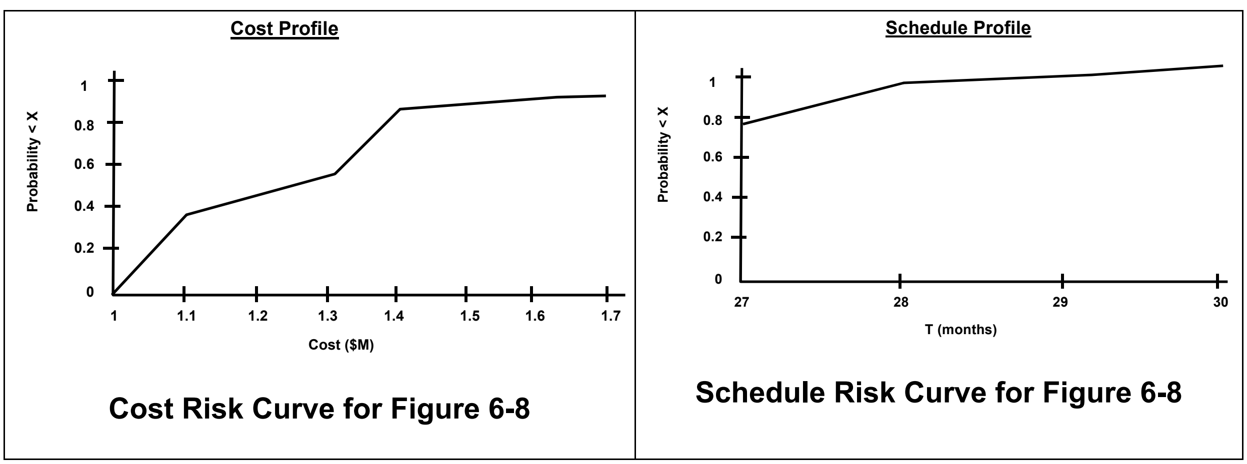


Figure 6-9. Cost and Schedule Risk curves for Figure 6-8

6.2.4 RISK PLANS AND REPORTS

The project risk documentation requirements depend on the contract.

6.2.4.1 RISK MANAGEMENT PROGRAM PLAN

An initial Risk Management Program Plan (RMPP) is often submitted as part of the proposal; it should be updated as required as the project evolves. Since Risk Management is an iterative process the intent of the RMPP is to define and establish the risk management of the project. To be of value, the RMPP should be uniquely tailored to the project and reflect the project concerns and management structure.

6.2.4.2 RISK HANDLING PLANS

It is generally good practice to prepare a Risk Handling Plan for each high-risk item. The suggested content is reproduced below because it captures most if not all of the ideas already discussed. Suggested plan contents include:

a. Statement and assessment of risk
b. Consequences of failure
c. Alternatives/options considered with risk and cost of each
d. Recommended risk reduction/abatement method
e. Implementation impact statement (cost/schedule/technical)
f. Criteria for closure of this risk
g. Decision points

Risk Handling Plans should also be considered for moderate-risk items. Because of the additional work required to ensure an effective risk management program, it is recommended that the total effort be limited to approximately the top ten items.

6.2.4.3 RISK REDUCTION REPORTS

Risk Reduction Reports should be submitted for each Risk Handling Plan. They describe the status of the risk reduction initiatives. Comprehensive reports should be included in the data package submitted for the major design reviews. Abbreviated status reports should be submitted monthly.

6.2.4.4 RISK SENSITIVITY ANALYSIS

The Risk Sensitivity Analysis explicitly presents the project's sensitivity to risk in terms of schedule and cost. It quantifies the impact on cost and schedule of potential risk reduction/abatement actions and addresses the benefits of alternatives. This ensures that the most effective approach is selected.

THIS PAGE
INTENTIONALLY
LEFT BLANK

7 ORGANIZATIONAL PRACTICES

This section discusses the concurrent engineering, Integrated Product & Process Development and the Integrated Product Development Team.

Introduction

In the 1990s the U.S. automobile industry strove to understand the differences between its products and processes and those of the Japanese automobile industry. Over time, some great distinctions were uncovered. One key distinction was the inordinate amount of planning and consensus-seeking done by the Japanese at the outset of a new project. The Japanese attempted to anticipate and resolve design, manufacturing, test, reliability, and other quality issues to the greatest extent possible at the outset of a program. They sought to eliminate costly downstream design changes. In contrast, American industry focused initially on design, with producibility and reliability issues deferred until later in the development and production process. While this leads to an earlier start (and completion) of prototype design, it usually leads to more downstream design changes after the initial automobiles are tested. Redesign and retooling introduces delays, extra cost, and a longer total time to market in comparison to the Japanese. Further, it was finally realized that quality could not be "tested in" during production. This results in high scrap rate and further redesign. Rather, quality must be "designed in" from the outset, as the Japanese did.

Concurrent Engineering Background

Many aspects of concurrent engineering have been practiced in the U.S. where innovative management techniques such as concurrent engineering and Program Evaluation & Review Technique (PERT) for program control have evolved.

In order to facilitate system development at maximum speed on all fronts, tight interfaces were defined and maintained between all subsystems. Then, development of all subsystems proceeded in parallel. As the subsystems were developed, they were connected in conformance with the earlier-established interface definitions. This approach had great risks when most of the key technologies were all in immature development status. Intense negotiation sessions ensued when suppliers could not meet their interface constraints. Some were too big, exceeding the physical envelope constraint. Project management adapted as best possible to minimize schedule and performance impacts. These activities can be described as concurrent engineering, but fall short of the intent of modern concurrent engineering programs.

Concurrent engineering is "... a systematic approach to the integrated, concurrent design of products and their related processes, including manufacturing and support." The stated rationale for this definition of concurrent engineering is to "... cause developers, from the outset, to consider all elements of the product life cycle from conception through disposal, including quality, cost, schedule, and user requirements." As can be seen from the above two quotes, the present definitions of concurrent engineering involve more than just engineering; they involve the whole project, including manufacture and support. Therefore, some U.S. companies adopted the terminology *Integrated Product Development* as more descriptive of this concurrency. Integrated product development implies the continuous integration of the entire product team, including engineering, manufacturing, test, and support, throughout the product life cycle. Later, as the importance of *process* was recognized, the terminology was modified to *Integrated Product and Process Development,* or IPPD.

A comparison of a concurrent/integrated product development program with a traditional or series development program is shown in Figure 7.1, Concurrent Development. Historically, traditional development took place in series with one activity starting as the preceding one was completed. This is a very lengthy process and the product could become obsolete before it is completed. With good interface definition and control, integrated product development, involving the entire team, can speed up the development process, as shown in the figure. Integrated or concurrent development could introduce *more* risk into a development program because downstream activities are initiated on the *assumption* that upstream activities will meet their design and interface requirements. However, the introduction of a hierarchy of cross-functional product teams, each developing and delivering a product has been found to actually reduce risks and provide better products faster - as will be discussed.

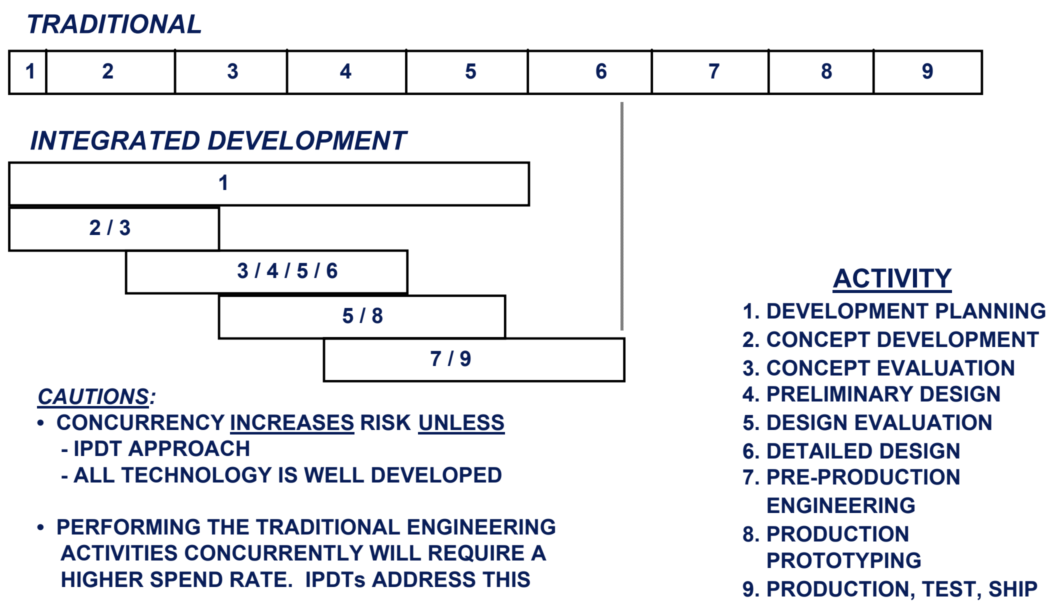


Figure 7-1. Concurrent Development vs. Traditional

7.1 OVERVIEW OF INTEGRATED PRODUCT & PROCESS DEVELOPMENT

This section will introduce the IPPD concepts and why to use them.

7.1.1 WHAT IS AN INTEGRATED PRODUCT DEVELOPMENT TEAMS (IPDT)?

An IPDT is a process-oriented, integrated set of cross-functional teams (i.e., an overall team comprised of many smaller teams) given the appropriate resources and charged with the responsibility and authority to define, develop, produce, and support a product (and/or service). Process orientation means they are staffed with all the skills necessary to complete their assigned processes -- which may include all or some of the development and production steps.

Although the teams are organized on a process basis, the organizational structure of the team of teams may approach a hierarchical structure for the product, depending upon the way the product is assembled and integrated.

Different members of a cross-functional team may have primary, secondary, or minor support roles during different phases of the project cycle. For example, the manufacturing and test representatives may have minor, part-time advisory roles during the early product definition phase, but will have primary roles later, during manufacture and test. The idea is to have them participate to the degree necessary from the outset to insure their needs and requirements are reflected in overall project requirements and planning to avoid costly changes later.

The team must be given both responsibility and authority to get the job done. If no one is in charge, things do not get done. The team should be empowered with authority to do the job. It should not be looking to higher management for its key decisions. It should, however, be required to justify its actions to others, including interfacing teams, the system integration team, and project management.

7.1.2 Why employ IPDTs?

Fierce global competition in the marketplace is driving companies in four major areas:

- Lower cost products and services
- Leaner corporate staffs (The horizontal corporation)
- Higher quality products and services
- Shorter development and production times (Time to market)

The tight schedule pressure essentially forces concurrent (overlapping) development, where components are developed almost in parallel, not in series. Concurrent development usually ***increases*** risks of development problems, because tight interfaces must be negotiated between components ***before they are developed.*** If problems are encountered with one component, it could affect others, resulting in redesign, schedule slips, and extra development costs.

To reduce the risks inherent in concurrent development, industry has learned that IPDTs, using best practices and continuous improvement, have been achieving significant ***process*** improvements, resulting in:

- Seamless interfaces within the teams
- Reduced engineering design time
- Fewer problems in transition from engineering to manufacturing
- Reduced development time and cost

In the early 1990s, companies began to discover that they really could be more productive if they moved away from the hierarchical management structure and organized into product development teams. These teams each mimic a small, independent project to produce its product. Some of the greatest productivity gains have come in three areas:

- Unleashing the team's ingenuity through decentralized processes
- Avoidance of previous problems through new, creative approaches
- Better integration between engineering and manufacturing

The above have led to improved product features, performance, quality, and customer satisfaction.

7.1.3 Some examples of IPDT success

In early use of IPDT techniques, Boeing said they have reduced the delivery time for a small satellite from 36 months to 18 to 24 months, and that costs could be halved relative to previous government formula pricing estimates.

Lockheed Martin created a small process action team of about 12 core members to review how they developed communication satellites and to define process changes they could introduce for major improvements in cost, quality, and competitiveness. This team identified major changes in standardization, integration practices, supplier teamwork, and modular hardware and software that reduced their long lead items, schedules, and costs. Some examples are: a focus on low cost, lightweight, common products; a standard 15-pin connector for all electronics - same harness for every box. Although each of their satellites is usually somewhat different (in order to meet unique requirements), designers are limited to one of ten design templates. The result, every satellite is always 85 percent designed (vs. a new start).

7.2 THE IPDT PROCESS

A basic principle of IPDT is to get all disciplines involved at the beginning of the development process to ensure that requirements are completely stated and understood for the full life cycle of the product. This up-front activity is considered part of the Systems Engineering process. Historically, the initial development of requirements has been led by Systems Engineers. In an IPDT, the Systems Engineers still lead the requirements development process, but now more (all) disciplines participate in it.

Requirements are developed initially at the system level, then successively at lower levels as the requirements are flowed down. Teams, led by Systems Engineers, perform the up-front Systems Engineering functions at each level. *This is different* from the previous, classical development approach where Systems Engineers did the up-front work and passed the requirements along to development engineers who passed their designs on to manufacturing, thence to test, without the continuous involvement of the initial engineers. This resulted in a loss of understanding caused by asynchronous communications.

The general approach is to form cross-functional product/process teams for all products and services, plus a Systems Engineering & Integration Team (SEIT) to cover systems issues, balance requirements between product teams, and help integrate the teams. This process is illustrated in Figure 7-2. Each of the teams may have members representing the different areas indicated on the left side of the chart.

These team members' participation will vary throughout the product cycle, as the effort transitions from requirements development to conceptual design, through preliminary design and detail design, to manufacturing, assembly and test, to delivery, operational support, and finally retirement (and possibly replacement). It is good for at least some of the team to remain throughout the product cycle in order to retain the team's "project memory."

The product teams do their own internal integration. A SEIT representative belongs to each product team (perhaps several); with both internal and external team responsibilities. There is extensive iteration between the product teams and the SEIT to converge on requirements and design concepts. This effort should slow down appreciably after the preliminary design review, as the design firms up.

There are typically three types of IPDT. These are:

1. Systems Engineering & Integration Team (SEIT)
2. Product Integration Team (PIT)
3. Product Development Team (PDT)

The focus areas for the three types of IPDT teams and their general responsibilities are summarized in Figure 7-3. This arrangement is often applicable to large, multi-element, multiple subsystem programs. It must obviously be adapted to the specific project. For example, on smaller programs, the number of PIT teams can be reduced or eliminated. In service-oriented projects, the system hierarchy, focus, and responsibilities of the teams must be adapted to the appropriate services.

Note that the teams are **process** oriented, focusing on components or their integration into more-complex subsystems and elements. The SEIT is used to focus on the integrated system, system processes, external and system issues, which, by their nature, the other teams would possibly relegate to a lower priority.

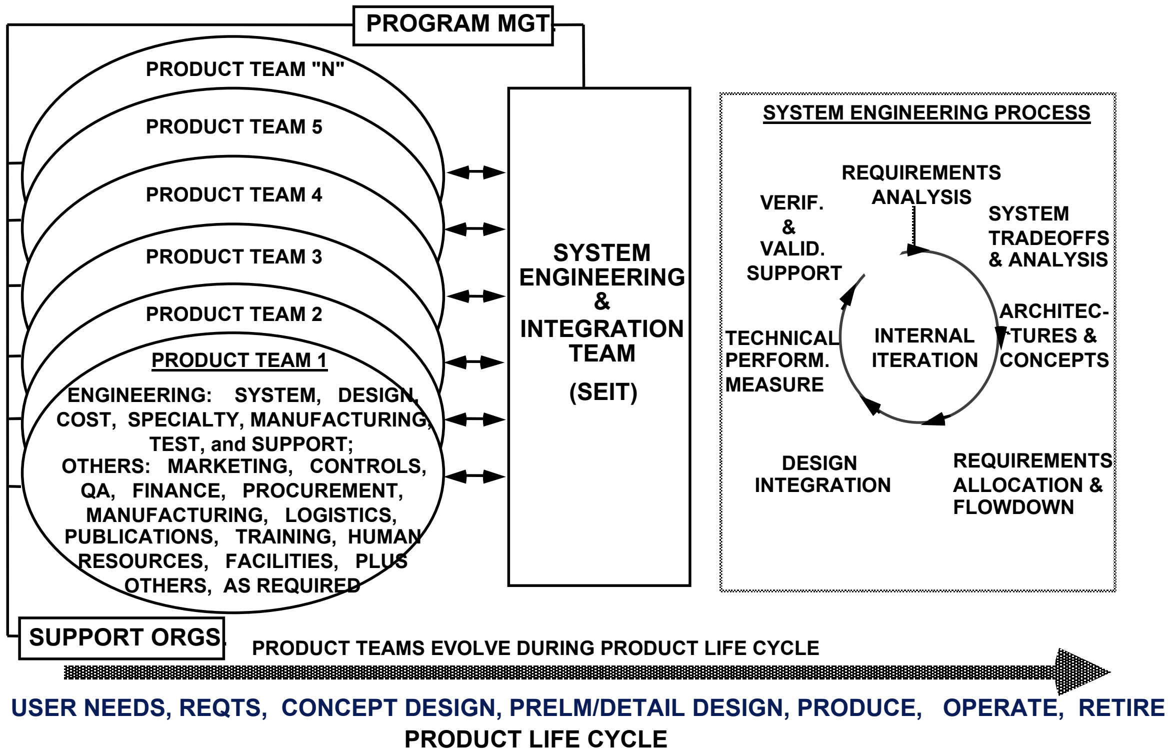


Figure 7-2. IPDT Process Overview

SYSTEM HIERARCHY	TEAM TYPE + FOCUS & RESPONSIBILITIES
EXTERNAL INTERFACE & SYSTEM	**SYSTEM ENGINEERING & INTEGRATION TEAM (SEIT)** • INTEGRATED SYSTEM AND PROCESSES • EXTERNAL & PROGRAM ISSUES • SYSTEM ISSUES & INTEGRITY • INTEGRATION & AUDITS OF TEAMS
ELEMENT & SUBSYSTEM	**PRODUCT INTEGRATION TEAMS (PITs)** • INTEGRATED H/W AND S/W • DELIVERABLE ITEM ISSUES & INTEGRITY • SUPPORT TO OTHER TEAMS (SE&IT and PDTs)
COMPONENTS, ASSEMBLIES, & PARTS	**PRODUCT DEVELOPMENT TEAMS (PDTs)** • HARDWARE AND SOFTWARE • PRODUCT ISSUES & INTEGRITY • PRIMARY PARTICIPANTS (DESIGN and MFG.) • SUPPORT TO OTHER TEAMS (SE&IT and PITs)

THESE MULTI-FUNCTIONAL TEAMS HAVE LIFE CYCLE (CONCEPT-TO-DISPOSAL) RESPONSIBILITY FOR THEIR PRODUCTS and THE SYSTEM

Figure 7-3. Types of IPDTs, their Focus And Responsibilities

Systems engineers participate heavily in the SEIT and PIT and to a much lesser extent in the PDT. The Systems Engineering processes described in this handbook are just as applicable to all teams in the IPPD environment as they were in previous styles of organization. The iterative Systems Engineering process is still used. In fact, it is easier to apply the process throughout the program because of the day-to-day presence of Systems Engineers on all teams.

All product teams have many roles. Their integration roles overlap, based on the type of product team and the integration level. Some examples are shown in Figure 7-4 for various program processes and system functions. In this figure, Program Processes covers just about anything required on the program. The three bars on the left side show the roles of the three types of product teams at different levels of the system. Note for example that the SEIT **leads** and **audits** in external integration and in system integration activities, as indicated by the shaded bar. But, for those program processes involving components, subassemblies, or parts, the appropriate PDT are the active participants, **supported** by the SEIT and the PIT.

Basic system functions include system requirements derivation, system functional analysis, requirements allocation and flowdown, system trade-off analysis, system synthesis, system integration, technical performance measurement, and system verification. The bars for functions 1, 2, and 3 in the chart show that the SEIT **leads** and **audits** activities on different system functions while the component and subsystem teams actively participate. The lower level part and subassembly teams support, if requested.

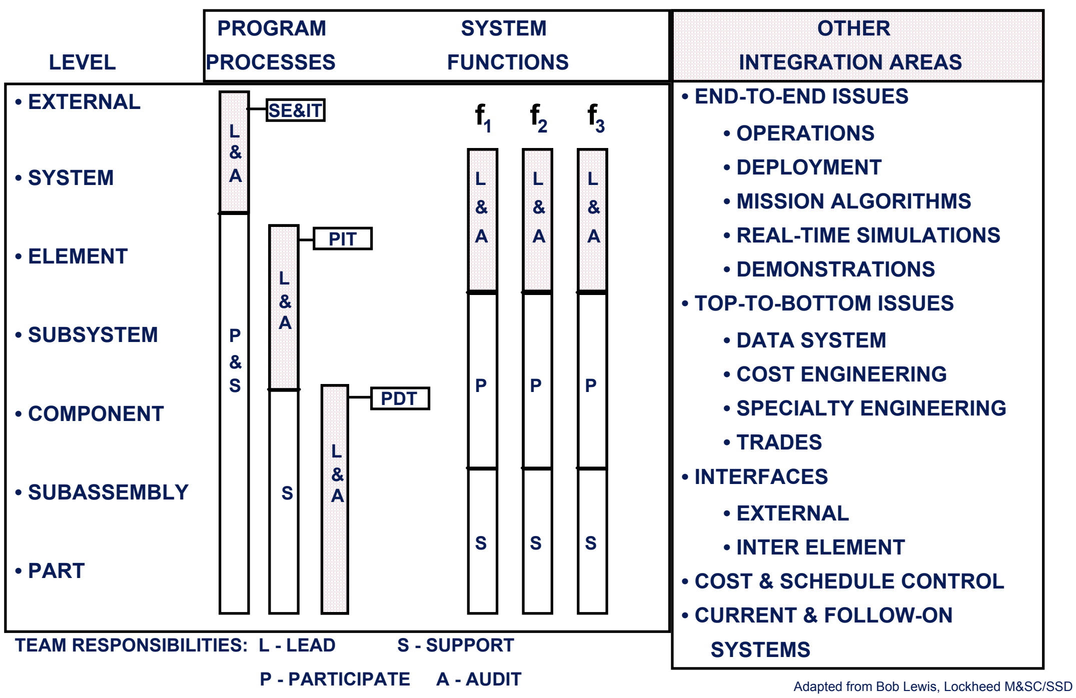


Figure 7-4. Examples of Complementary Integration Activities of PDTs

The column at the right side of Figure 7-4 shows other integration areas where all teams will have some involvement. The roles of the various teams must also be coordinated for these activities, but they should be similar to the example.

7.3 STEPS IN ORGANIZING AND RUNNING AN IPDT

The basic steps necessary to organize and run an IPDT on a project are listed in Table 7.1. Each step will be discussed in turn, with a summary of the key activities that should take place during the step.

Table 7.1 – Steps in Organizing and Running an IPDT

Step	Description
1.	Define the PDT teams for the project
2.	Delegation of responsibility and authority to the PDT teams
3.	Staff the IPDT
4.	Understand the team's operating environment
5.	Plan and conduct the "Kick-off meeting"
6.	Team training
7.	Define the team vision and objectives
8.	Each team expands the definition of its job
9.	Process assessment and continuous improvement
10.	Monitor team progress via metrics and reports
11.	Sustain and evolve the team throughout the project
12.	Documentation of team products
13.	Project closure and follow-on activities

7.3.1 STEP 1, DEFINE THE PDT FOR THE PROJECT

The first major task is organizing the PDT teams for the project to establish a comprehensive team of teams that efficiently covers all project areas. The goal of defining the PDTs for a project is to create a process-oriented division of effort along natural products, including their design, development, manufacturing, test, delivery, and operational support. The products and product teams can include subsystems, assemblies, components, elements, parts and system integration and test. Some guidelines for defining the PDT teams include:

- Select the teams so they are as self-contained as possible, with minimum dependence on other teams to get their job done.
- Select products/teams such that the interfaces between them minimize complexity.
- Use the Venn diagram to help visualize a best division.
- Avoid defining too many teams, such that individuals must divide their time between more than one team.
- Use representatives from PDT on the PIT and the PIT representatives in the SEIT.

There are many ways to organize the PDT teams. Figure 7-5 shows three examples from different size satellite programs within Boeing and Lockheed Martin. The small program example on the left side of the figure is an IPDT arrangement for a small satellite program. It uses a SEIT, PDT and PDT for major hardware assemblies and major functional areas, such as system test. The PDT Council, headed by a chief engineer, controls requirements and budget allocations and adjudicates interface contentions. Council members are leaders of the PDT.

In the center of the figure is a product team organization used on medium to large satellite programs. Supporting organizations are not shown. This general model is used as the standard throughout this section.

On the right is the IPDT organization for very large programs. At the Tier 1 level the program was managed by the program VP and General Manager with a Team Program Office (TPO) containing the senior program manager level representatives from all three major participating companies. The TPO includes major product managers as well as senior representatives from all the functional disciplines involved. The weapon Analysis and Integration Team (AIT) consists of a director and 16 sub-teams that provide analytical support and/or integration that can not be provided by a lower level team. For example, design-to-cost analyses and allocations emanate from the Tier 1 level.

Although these three organization charts are greatly simplified, similarities between them are apparent. The AIT teams on the very large program cover the Systems Engineering functions that the other programs cover with their SEIT. The PIT on the medium to large program is similar to the large program IPT teams. The small program does not need PIT teams.

The organizations on the left and right are discussed in detail in papers listed in the references below.

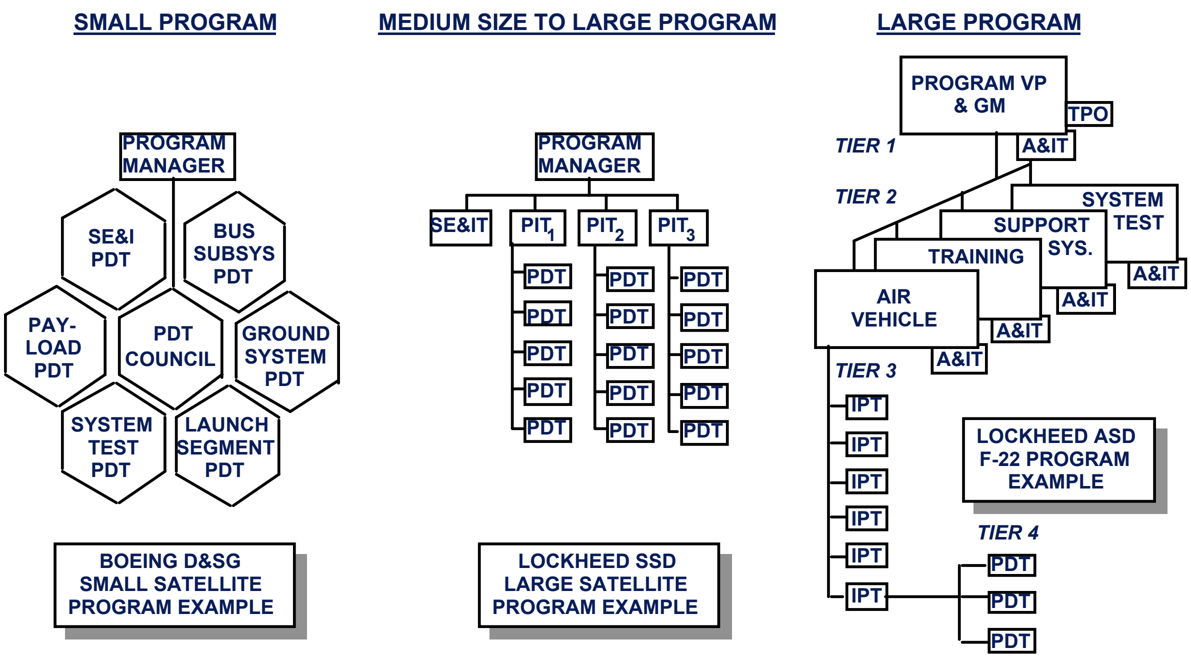


Figure 7-5. IPDT Organization Alternatives - Examples

REFERENCES

1) Churchill, M.J.; McPherson, S.A; and Weston, T.C.; *System Engineering in a Low Cost, Concurrent Engineering, Small Project Environment*, Boeing Corporation, Proceedings of the 3rd annual NCOSE International Symposium, July 1993.

2) Cox, J.D., *Organizational Challenges in Integrated Product Team Implementation*, Lockheed ASD, Proceedings of the 3rd annual NCOSE International Symposium, July 1993.

7.3.2 STEP 2, DELEGATION OF RESPONSIBILITY AND AUTHORITY TO PDTS

This is an essential management activity. The project management, with support from Systems Engineering and others should identify the teams and select the initial team leaders. When possible, management should select an experienced team leader. Then, a project management document identifying the team, its initial leader, functions and responsibilities, resources, project tasks and preliminary schedule, and limits on team authority should be approved by both the project manager and the selected team leader at the earliest possible moment in the program. This empowers the team and its leader and sets the stage for rapid, creative response to the assignment.

Tasks, budget availability, and schedules must sometimes change frequently as company and project management adapt to the dynamic business environment. For continuity of effort and minimum loss of productive time, management should try to avoid unnecessary changes.

For companies and projects that routinely use the same subdivision of product teams, many of the items listed in the chart can be incorporated by reference to appropriate standard company/project management procedures.

7.3.3 STEP 3, STAFF THE IPDT

Most engineering team leaders know how to staff an engineering team. Even so, management assistance is helpful to get the right people for the job. Staffing problems are compounded when trying to staff a cross-functional team. The team leader and his management may not know qualified people from the other disciplines (especially on the first IPDT). They also may not know how much effort will be required from each discipline at each stage of the program.

In this environment of uncertainty, specialty areas may "sell" more effort than the team really needs. Project management can keep the lid on staffing by holding the line on overall budget and forcing the team to make the tough decisions. No organization can afford to have unnecessary people on their teams. Therefore, the team, the team leader, and the various specialty support areas should be "challenged" to put together a lean, efficient team and evolve it as project needs change. Some key thoughts for staffing an IPDT follow.

- Balance competency and <u>full time</u> commitment for core team members.
- Balance competency and availability of part-time team members.
- Identify the program stages in the development and support cycles when competencies are required.
- Identify issues critical to the team that require additional emphasis or specialized expertise.
- Consider the ability of a candidate to work well with other team members.
- Keep the budget in focus.
- Inform all team members of scheduled meeting, their agendas and the results of prior meetings, but only request the attendance of those who are needed to support the current agenda.

7.3.4 STEP 4, UNDERSTAND THE TEAM'S OPERATING ENVIRONMENT

Many issues of the operating environment influence the team. Many of these are indicated in Figure 7-6. One should always try to understand and anticipate these influences and communicate them to team members.

The influence of other teams, other projects within the company, the company or division's current situation, and the external (outside the company) situation may be indirect, but they can all have a strong influence on a team's situation. For example, the relative value of international currency may force you to produce key parts in a country with lower labor rates. Obviously, setting up a new factory would have a major impact.

The message of the figure is to recognize the position of the team in this nest of influences and be sensitive, not just to project influences, but to other indirect influences as well. For example, if your team plans to use some company test facilities that are also used by other projects, you must insure that they are reserved for you when needed or you will have a schedule impact.

The better one can anticipate potential opportunities and impacts to the team, the better the team can adapt to do a good job. The failure of any team to accomplish its objectives imperils the other teams, the project, and the higher-level organization. So, all of these organizations should support each other in the appropriate manner.

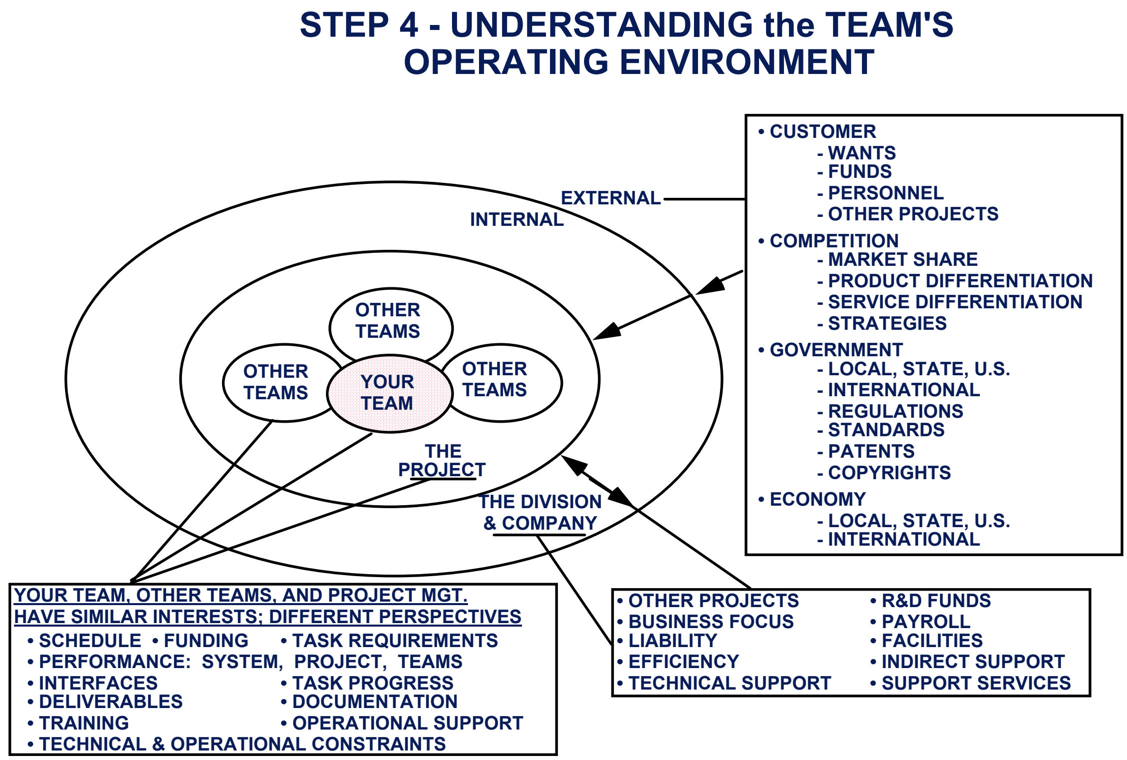


Figure 7-6. Understanding the Team's Operating Environment

7.3.5 STEP 5, PLAN AND CONDUCT THE "KICKOFF MEETING"

There are two "kickoff meetings", one for all project personnel, followed by each team's kickoff meeting. The project meeting obviously covers general project issues, but the team meeting focuses on team-specific issues.

The team kickoff meeting is identified as a specific step because of its importance to the success of the team effort. It is the culmination of preparation by the team leader and perhaps others to launch the team activity. The team leader should have a week or more of preparation for the meeting; working on staffing and getting briefing charts to cover topics in the agenda.

The objective of the meeting is to establish a climate for the successful conduct of the project, by establishing a rapport between the team members that creates an environment for teamwork.

Topics for the agenda of this meeting should include:

- Introduction to the project and personnel

- Introduction to the “Team of teams” concept, the teams and the team leaders
- Introduce the general plan of operation and operating environment (from step 4)
- Review the preliminary schedule and budget
- Provide an overview of the meeting plans
- Discuss each PDT, the Team Vision and Objectives, expanded definition and scheduling of team tasks and the next activities with due dates
- Discuss the training requirements for each team to get up to speed
- Discuss basic project infrastructure including IT-support, process, schedule and budget review and control procedures.

After the meeting each team should further expand their tasks and validate or negotiate revised budgets and schedules that the team are committed to meet.

For some of the team, the Kickoff Meeting will be their first exposure to the leader and other team members. It is important that the team leader appear to be organized and competent. The leader must extract loyalty, dedication, and quality outputs from the team. While the leader need not know all the answers, he/she should have a strong idea of where the meeting is going and see that it gets there (while getting everyone to participate!).

The team kick-off meeting is very important in setting the stage for professional team conduct. It should move fast, in a business-like manner. It, and all future team meetings, should have a posted (or regular) agenda (that attendees know in advance). The next meeting could be a short training session, discussed next as step 6, to cover tools and techniques the team will use in its cooperative activities.

7.3.6 STEP 6, TEAM TRAINING

Some of the items covered in team training, may already have been covered at either the project or team kick-off meetings. If so, they may be simply reviewed or eliminated from any subsequent training. It is recommended that a booklet of these charts or other project/team direction material be maintained so that absent or new team members can brief themselves and rapidly come up to speed on a project.

The objective of team training is to prepare members to act as a team, using common terminology, techniques and tools to meet the expectations of management and the team leader. A typical agenda might include the following topics:

- Review the project organization, product nomenclature and terminology
- Explain the “Team of Teams” concept
    - Provide a handout with a listing of all teams, leaders and contact information
    - Discuss the functions and responsibilities in detail for interfacing teams
- Discuss the operating procedures for the PDT
    - Regularly scheduled meetings
    - Status and reporting methods and schedules
    - Project and team documentation requirements
- Introduce key techniques and tools to be used in the execution of team activities

The kickoff meeting and early team training sessions provide excellent opportunities to establish high standards for team performance by establishing creative procedures to let people work to their best capabilities with simple, well-defined controls that minimize interruption of their work.

Stretch out training on key techniques and tools such that an item is presented just prior to its need (when interest is highest) rather than a few lengthy (probably boring) sessions.

Procedures should be established for document and design drawing control and release, interface definition and control (consistent with project), requirements reviews and design reviews, maintenance of baseline schedule (for the team), maintaining documentation on the baseline design (accessible to all project personnel), etc. Establishing these items of project level "infrastructure" and training your team in their use are critical to project success.

7.3.7 STEP 7, DEFINING TEAM VISION AND OBJECTIVES

At the first private meeting of the team it is beneficial to spend an hour or so (not days) focusing on team vision and objectives. Make it a collaborative, brainstorming process to involve the entire team and get their input. It also provides the first opportunity to work together as a team and learn others' perspectives.

After a short general discussion, you could use an Affinity Diagram tool to quickly converge on important ingredients for the vision and objectives. Then summarize and restate them in an organized fashion. The team can then use its new vision and objectives in constructing a detailed plan of action.

There is heavy interaction with other teams, management, and customers. Many of these meetings require overview briefings on team activities and status updates. If this recognized from the outset, the team leader can enlist team support in preparing and maintaining a master set of briefing charts for external presentations.

7.3.8 STEP 8, EACH TEAM'S EXPANDED DEFINITION OF ITS JOB

In expanding the definition of the team's job, the operating environment constraints should include technical as well as budget/manpower and schedule. In mapping out a plan of action, a tree chart approach can be used to organize identified tasks and subtasks into greater detail.

Once a breakout of tasks and subtasks has been developed, schedule them, establish reporting milestones and identify the responsible person on each activity; provide status at least weekly.

Once a more-detailed plan has been developed, the team leader may need to renegotiate budgets (more or less) as necessary to accomplish the tasks (or adjust the tasks for compatibility with the available budget). Resolve all budget problems quickly; the problems only increase with time.

Emphasize that team members must be accurate, factual, and quantitative in reporting status. Simply stating that "everything's fine" does not do it. Rather, "we have accomplished 85% of the items scheduled for this period; what's missing is ... and these items will be accomplished by <a stated date>, which puts us two days behind plan, etc.”

7.3.9 STEP 9, PROCESS ASSESSMENT AND CONTINUOUS IMPROVEMENT

A process is an integrated set of activities that accomplish a stated objective. Before you can improve processes, you must identify the ones you are using. Next, assess the maturity of your processes. Focus on process improvements that appear to have high payoffs. Remember, although you're looking for improvements, you must also consider the adverse consequences for your team and others. Some "improvements" have turned out to be disasters when a proven, reliable process was changed and process control was lost.

Recommended Activities

The following steps are useful for achieving continuous improvement.

1. From the outset instil an attitude in the team to achieve continuous improvement.
2. Encourage individuals and groups to come forward with ideas for improvements to both process and product.
3. Schedule a bi-monthly team session to discuss team interactions and processes and how they might be improved.
4. Assess the maturity of team key processes by performing quick subjective assessments as a team, ranking each process from 1 to 6, where 1 corresponds to an initial level (undocumented processes), 2, to documented, and the highest level, 6, to optimizing.
5. Identify candidate improvements, including schedule, cost, performance, quality, risk, personnel changes, facility/equipment changes, training required, impact on other project elements, impact on the customer, etc. Score as a team or separately then average, or discuss and re-evaluate.
6. Evaluate the improvements for their contribution to team objectives. Develop a simple scoring system and use Pareto chart plots of the results.
7. Consider implementing improvements after careful cost vs. benefit analysis of the ideas with the highest Pareto ranking.
8. Select the improvements to be implemented and review with project management as appropriate.
9. Implement the improvements using the Shewhart cycle (Plan-Do-Check-Act). Step 4, "Act" in the Shewhart Cycle, can also mean eliminate the proposed change, or adjust and try again. As you implement improvements, remember that, after Plan-Do-Check-Act, you start the process over -- to continuously seek useful improvements.

7.3.10 STEP 10, MONITOR TEAM PROGRESS - METRICS AND REPORTS

Six categories of metrics that each product team can use to status its current and projected progress are illustrated in the four charts of Figure 7-7. Coordination with the SEIT on parameters, techniques, and units of measure to be used for commonality throughout the entire project allows results to be quickly "rolled up".

Chart 1 shows team schedule status on each major task. The task bar is darkened to indicate percent completion, or days ahead (behind) schedule. The scheduled and actual completion dates (day and month) are shown by each task and milestone, including all deliverables. Lots of intermediate milestones should be shown.

Chart 2 gives status of actual team expenditures (including all commitments) vs. plan. At the bottom of the chart, the arrow shows milestone status. If the team is behind on some milestones, the arrow stops short of the current dateline by the number of days required to complete overdue milestones.

Chart 3 is representative of any number of design efficiency metrics, such as weight, required power, envelope dimensions (volume), errors per design drawing, or rework time, etc. Several may be required for adequate status.

Chart 4 gives another metric for design efficiency -- production cost of the first unit. Team status vs. its negotiated design-to-cost goal is shown. Also, performance status vs. key performance and quality measurements should be shown. This is a form of Technical Performance Measurement (TPM) for the team and should cover critical performance and quality parameters.

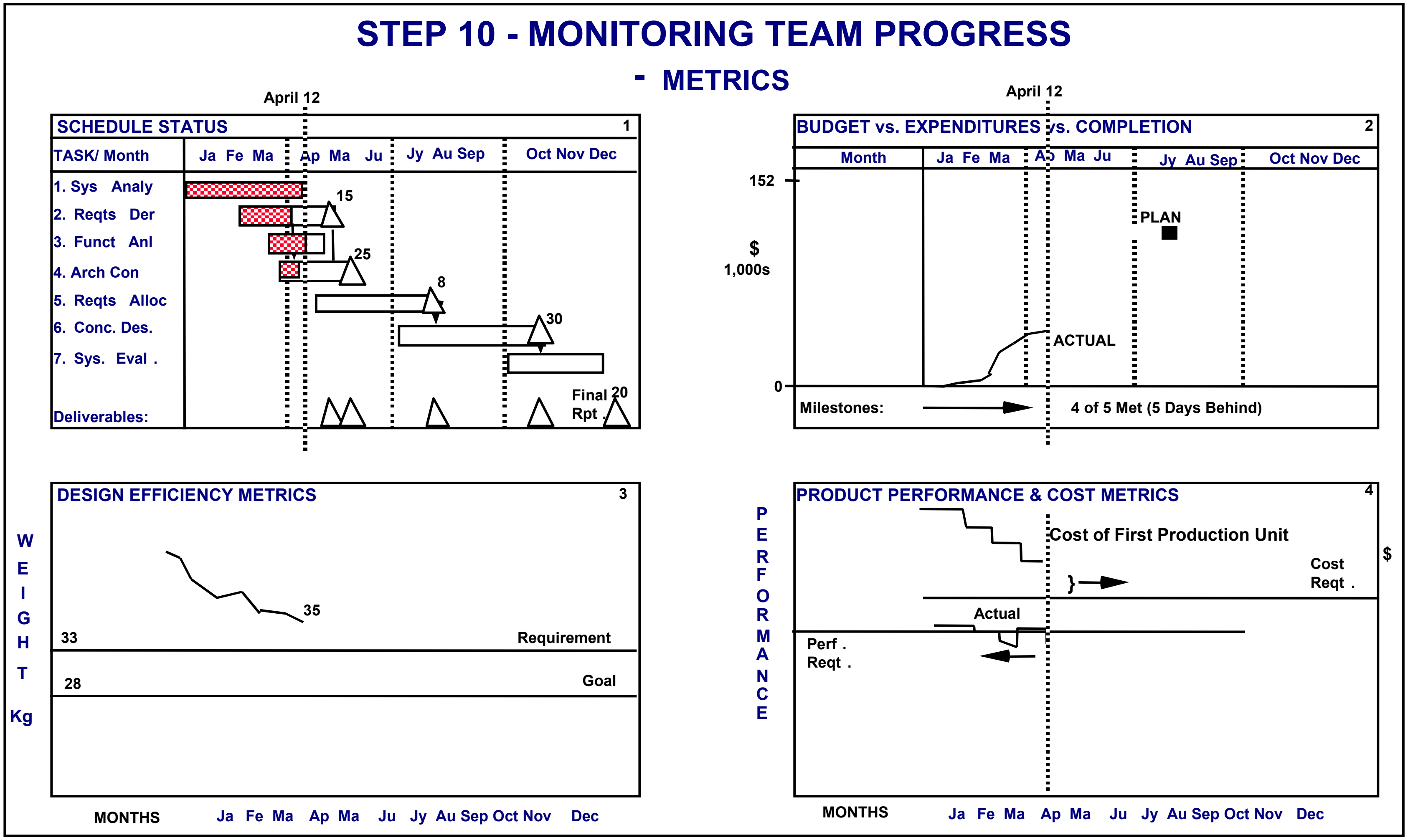


Figure 7-7. Monitoring the Team Progress

7.3.11 STEP 11, SUSTAINING AND EVOLVING THE TEAM THROUGHOUT THE PROJECT

The personnel assignments to a team will probably vary over the project cycle. If personnel adapt to the project's changing needs, perhaps they remain, but certainly the needs for skills varies during the cycle, as shown in Figure 7-8. The chart depicts the relative emphasis for various skills on a project that has a heavy emphasis on both hardware and software.

Obviously, requirements development is a primary focus during early conceptual design. Then the cross-functional disciplines are brought in later in the conceptual design phase but early enough to make major changes with insignificant cost impact. These cross-functional specialists identified with a team during conceptual design should continue periodic reviews of team progress, including detailed sessions with the other team members.

Specialty engineering may include reliability, maintainability, human factors, materials and processes, engineering standards writers, life cycle cost analysts, Electromagnetic Interference/Electromagnetic Compatibility (EMI/EMC), configuration management, etc.

Functions such as marketing, program controls, procurement, finance, legal, and human resources will generally support the team at a steady, low level of effort, or as required.

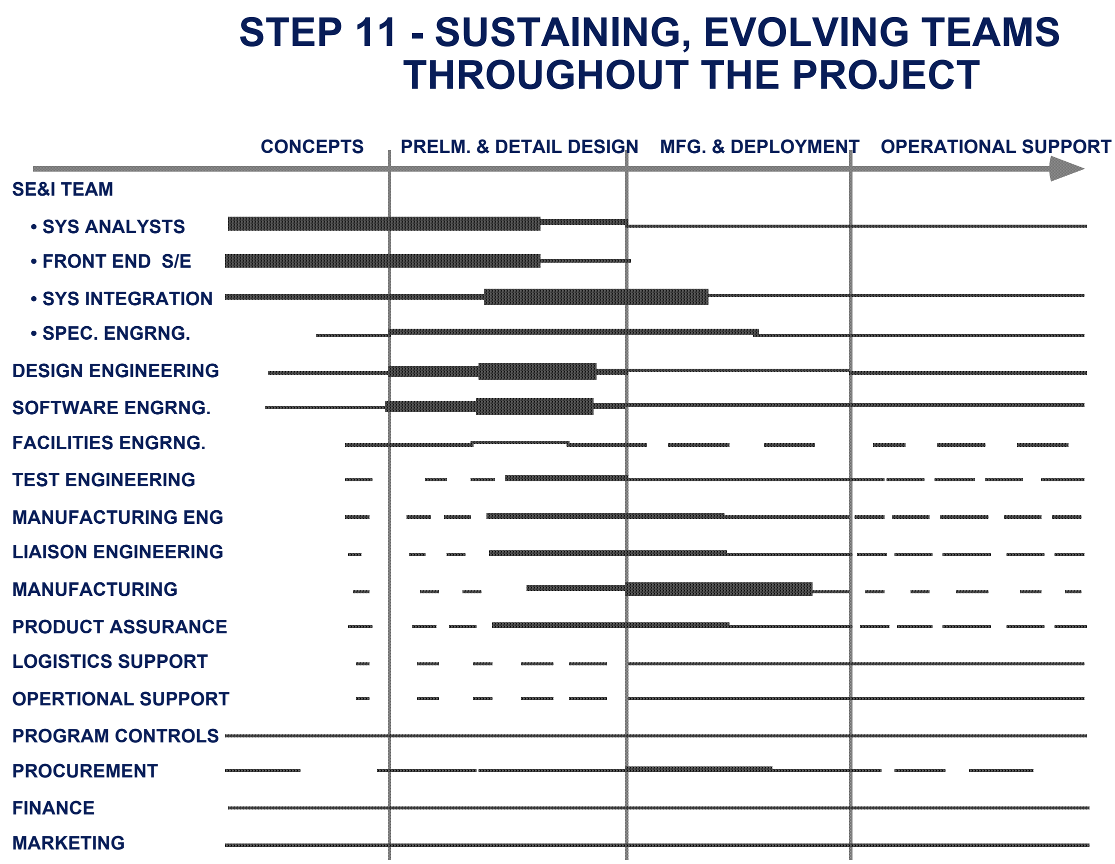


Figure 7-8. Sustaining, Evolving Teams Throughout the Project

7.3.12 STEP 12, DOCUMENTATION OF TEAM PRODUCTS

The primary documentation requirements do not change significantly for an IPDT. What does change is the amount of cross-organizational correspondence required. Ideally, it is greatly reduced or eliminated.

On documents that are likely to be updated, one person should be designated as the overall responsible author. This person is responsible for preserving the previous edition and collecting revisions for the next edition. Different people on the team should be designated as responsible for different documents. With cross-functional teams, there are many areas of expertise represented on the team. The team often has within its membership the capability to prepare the required documentation; delegating sections to various members, then integrating and editing the inputs to form the final document.

It is also recommended that the IPDT team leader maintains a notebook of team activities. This notebook serves as a form of corporate memory, providing continuity for following projects, and a history of the project to-date for use by team members joining after the project has begun. It is sometimes necessary on long projects to transition personnel on and off the team, including the team leader, and this notebook can be an invaluable technique for bringing new members up to speed.

The team leader can/should delegate preparation and maintenance of various parts of the notebook to various members of his/her team. The team leader should periodically review the notebook to insure that it remains up-to-date and a viable reference resource.

The contents of the team leader's notebook may vary, but the key items that should be considered for collection in the team leader notebook(s) are as follows.

- The team mandate
- Team members names, responsibilities, tel., fax, e-mail, building address
- Schedule (latest revision) and team's current schedule status
- Team budget (latest revision) and team's current spending status
- Key decisions, resolved issues, or management direction and rationale
- Outstanding action items
- Summary of Interface Agreements (details maintained in database)
- Supplier data; individual notebook for each supplier contract
- Design characteristics: summary characteristics of team products
- Key concerns and approach
- Notes of general interest

7.3.13 STEP 13, PROJECT CLOSURE AND FOLLOW-ON

In closing down a team, the main thing is to leave a team historical record on file with the project/program office. If the team leader kept his notebook(s) up-to-date during the program, there will be little left to do, except to possibly write a summary assessment of the team's activities and the status of things when the team's activity ceased.

Other Project Closure items should also be provided, including lessons learned and how to contact key team members for several years in the future. The rationale for maintaining these records is to support analysis of in-service problems; to possibly assist other programs with similar situations; and to maintain records in case there is ever another start-up of the team/project.

A follow-on activity could be anything from extended production of the same products, modifications, or entirely new applications. This dictates how far back into the product life cycle the program must go and what type of product development teams it should have.

If extensive re-engineering is required -- as in design modifications or new applications, problems can occur if the operational support teams attempt to address these without substantial engineering help.

7.4 POTENTIAL IPDT PITFALLS VERSUS HIGH PERFORMANCE

There are some things teams should watch out for. Table 7-2 describes eight. Table 7-3 lists Ten Techniques for High Performance in an IPDT. There are ample opportunities to get off track before team members and leaders go through several project cycles in the IPDT framework and gain the experience of working together.

Table 7-2, Pitfalls of using IPDT

	IPDT Pitfalls	What to do
1.	Spending too much time defining the vision and objectives	Converge and move on
2.	Insufficient authority – PDT members must frequently check with management for approval	Give team leader adequate responsibility, or put the manage on the team
3.	PDT members are insensitive to management issues and over commit or overspend	Team leader must remain aware of overall project objectives
4.	Teams are functionally-oriented rather than cross-functional, process-oriented	Review step 1
5.	Insufficient continuity of team members throughout the project	Management should review staffing requirements
6.	Transition to the next phase team specialists occurs too early or too late in the schedule	Review staffing requirements
7.	Overlapping assignments for support personnel compromises their effectiveness	Reduce the number of teams
8.	Inadequate project infrastructure	Management involvement to resolve

Obviously, some things do require checking with higher authority. Encourage team members to anticipate these from the outset. Functional managers/supervisors, if any, must stay aware of major team issues and coach/guide/train participants until they gain the requisite experience.

Project managers should review team staffing plans to ensure proper composition and strive for continuity of assignments. It has been observed that the advantages of a full time contributor outweigh the work of many part-time team members. The loss of a key team member who knows how and why things are done can leave the team floundering.

On product teams it is important to have people who can work well together and communicate. But team results may be condemned to mediocrity by avoiding those outstanding technical/specialist professionals who can really make a difference.

Table 7-3. Ten Techniques for High Performance in IPDTs

	Recommended technique
1.	Careful selection of staff – excellent people do excellent work
2.	Establish and maintain positive team interaction dynamics; all should know what is expected of the team and each individual, all should strive to meet commitments, interactions should be informal but efficient, and a “no blame” environment where problems are fixed and the team moves on
3.	Generate team commitment and buy-in to the vision, objectives, tasks and schedules
4.	Breakdown the job into manageable activities that can be accurately scheduled, assigned and followed-up on weekly
5.	Delegate and spread out routine administrative tasks among the team; frees the leader to participate in technical activities, give every team member some administrative/ managerial experience.
6.	Create a “world class” analysis and simulation capability for requirements and performance to be better than the competition
7.	Schedule frequent team meetings with mandatory attendance for quick information exchanges; everyone is current; assign action items with assignee and due date
8.	Maintain a Team Leader’s Notebook
9.	Anticipate and surface *potential* problems quickly (internally and externally)
10.	Acknowledge and reward good work

THIS PAGE
INTENTIONALLY
LEFT BLANK

8 REQUIREMENTS DEFINITION PROCESS

Requirements are the foundation of the project. They form the basis for design, manufacture, test, and operations. Each requirement carries a cost. It is therefore essential that a complete, but minimum set of requirements be established early. Changes in requirements later in the development cycle can have a significant cost impact on the project, possibly resulting in cancellation. This section discusses methods for developing requirements from user objectives and the customer's preliminary requirements.

The objective of requirements analysis is to identify and express verifiable requirements that state user needs in appropriate terms to guide system concept development. Performing the mission analysis against measurable parameters ensures that an appropriate system sizing (of communication links, data processing throughput and capacity, number of computers and personnel, facility space) can be achieved. Requirements analysis, like the total Systems Engineering process, is an iterative activity in which new requirements are identified and constantly refined as the concept develops and additional details become known. These are analyzed and deficiencies and cost drivers are identified and reviewed with the customer to establish a requirements baseline for the project.

A second objective of the requirements analysis is to provide an understanding of the interactions between the various functions and to obtain a balanced set of requirements based on user objectives. Requirements are not developed in a vacuum. An essential part of the requirements development process is the concept of operations, the implicit design concept that accompanies it, and associated demands of relevant technology. Requirements come from a variety of sources, some come from the customer/user, some come from regulations/codes, and some come from the corporate entity. Figure 8-1 illustrates this environment.

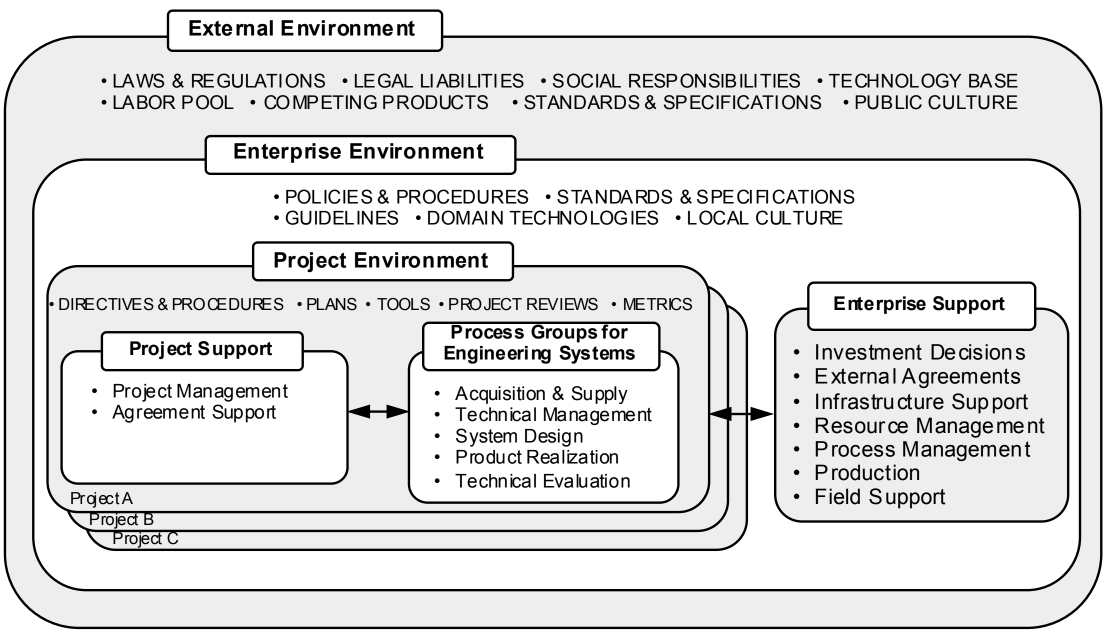


Figure 8-1. Sources of Requirements

Requirements definition is a complex process that employs performance analysis, trade studies, constraint evaluation and cost/benefit analysis. System requirements cannot be established without checking their impact (achievability) on lower level elements. Therefore, requirements definition is an iteration and balancing process that works both "top-down" and "bottom-up". Once the top-level set of system requirements has been established, it is necessary to allocate and flow them down to successively lower levels. As the allocation and flowdown process is repeated, it is essential that traceability be maintained to assure that all system level requirements are satisfied in the resulting design. The resulting requirements database usually contains many attributes for each requirement, and is also used in verification.

To describe the requirements analysis process in more detail, this section is broken down into four subsections: capturing source requirements, Concept of Operations definition, refinement, and requirements allocation and traceability.

8.1 CAPTURING SOURCE REQUIREMENTS

The Systems Engineering team leaders extract, clarify, and prioritize all of the written directives embodied in the source documents relevant to the particular phase of procurement activity. Examples of typical inputs include (but are not limited to):

a. New or updated customer needs, requirements, and objectives in terms of missions, measures of effectiveness, technical performance, utilization environments, and constraints

b. Technology base data including identification of key technologies, performance, maturity, cost, and risks

c. The outputs from the preceding acquisition phase

d. Requirements from contractually cited documents for the system and its configuration items

e. Technical objectives

f. Records of meetings and conversations with the customer

The source requirements gained by carrying out this function are only a portion of the total system requirements. They will be expanded by a number of activities as follows:

- The steps described below to break down the broad requirements statements will reveal the need for additional clarification which will lead to either revision of the written source material or supplement by additional source documents such as clarification meeting minutes, etc.

- The Concept of Operations Definition function, covered below in Section 8.2 will reveal the need for additional clarification.

This function is a continuing activity throughout the life of the project, hence the need for a solid foundation. The methods used for maintenance and revision of the databases are dependent on the change control procedures adopted for the project and will not be explicitly covered here.

The primary objective is to establish a database of baseline system requirements derived from the source, to serve as a foundation for later refinement and/or revision by subsequent functions in the Systems Engineering process and for a non-ambiguous and traceable flow down of source requirements to the system segments. This database foundation needs to be as complete and accurate as possible and must be fully traceable to source documentation. As a minimum, this foundation must include the following:

a. Project requirements
b. Mission requirements
c. Customer specified constraints
d. Interface, environmental, and non-functional requirements
e. Unclear issues discovered in the requirements analysis process

f. An audit trail of the resolution of the issues raised
g. Verification methods required by the customer.

Prerequisites for the successful performance of this function are:

a. Empower a systems analysis team with the authority and mission to carry out the function. (See Section 7 on IPPT)
b. Assign experienced Systems Engineer(s) to lead the team.
c. Assign experienced team members from relevant engineering, test, manufacturing, and operations (including logistics) disciplines to be available on call to the team.
d. Establish the form of the design decision database mechanisms and any supporting tools; select and obtain necessary SE tools for the function.
e. Complete the relevant training of team members in the use of tools selected for the function.
f. Define the formats of the output deliverables from this function (to permit the definition of any database schema tailoring which may be needed).

The following activities are involved in capturing source requirements:

Task	Paragraph
Organizing the Effort	8.1.1
Initializing the Database	8.1.2
Identifying Issues	8.1.3
Generation of the System Requirements Document (SRD)	8.1.4

8.1.1 ORGANIZING THE EFFORT

The final objective is the initial establishment of the decision database. Systems Engineering must be empowered by the project management office to act as activity leader. Good communication needs to be developed and maintained between the system leaders and the project/program management authority to ensure rapid identification and resolution of open issues as they are discovered in the Requirements Analysis process. Of equal importance is the identification of team members from the design disciplines. Using an interdisciplinary team in the analysis phase to supplement the generalist skills of the system leaders is a key contributor to discovery of open issues early in the Requirements Analysis process.

Recommended Activities

1. Identify system analysis team leader(s) and participants/delegates from other disciplines.

2. Identify the communication and management procedures/standards that will apply to the requirements analysis function.

3. Clearly state and document the project objectives(s) for the requirements analysis function.

4. Assemble and prioritize the relevant source documents.

5. Choose the media, tools and procedures for the decision database and enter the source documents into this database in a manner which provides access to the data by all authorized team members and permits traceability from source requirements to eventual Configuration Item (CI) specifications.

6. Determine the work breakdown for analysis of source documents and assign responsibility for analysis of each part of the source document database to a work group (or person). Each work group performs the same steps outlined below.

8.1.2 INITIALIZING THE DATABASE

The decision database must first be populated with the source documents that provide the basis for the total set of system requirements that will govern its design. Source documents used as inputs will include statements of user objectives, customer requirements documents, marketing surveys, systems analysis, concept analyses and others. These source or originating requirements should be entered in the design decision database and disseminated to all team members assigned to the requirements analysis team. The information should also be accessible for rapid reference by other project personnel.

Recommended Activities

1. Take the highest priority source document (partition) assigned and ensure that it is recorded in the database in a manner such that each paragraph in the source document is recorded as a separate requirements object. Record information to trace each such requirements object back to the identity of:
    - The source document identity
    - The paragraph title
    - The paragraph number

   (One reason for selecting paragraphs as the parent requirements object is to evaluate later change impact analyses. Most changes to source documents are flagged by a change bar against paragraphs which have been modified or deleted.)

2. Analyze the content of each parent requirement object produced in the previous step. Based on its engineering content determine the following:
    - Does the parent object contain any information on requirements or systems objectives? If so, is it completely clear, non-conflicting with other requirements and uniquely assignable to a single system function or architectural component or performance measurement index or system constraint? If so, bypass the mini-steps below and move to the next parent requirement object. If not, based on the engineering content, determine a strategy for decomposing the parent requirement object into separate but related pieces with the objective of achieving a family of simple, clear, non conflicting statements each of which can be individually assigned to single system function or component or performance measurement index or system constraint.

    - Record information in the database to provide vertical traceability from the parent requirement object to the child requirement object using the Project Unique Identifier (PUID) discussed in Section 8.4.

    - Repeat the procedure with child objects as necessary, creating and recording traceability to grandchild, great grandchild, etc. Stop fragmentation at the level when the objective has been achieved. This is called a leaf node requirement. As the requirements are flowed down, this will eventually end at: for hardware, the Configuration Item (CI) level; for software, the Computer Software Component (CSC) or Computer Software Unit (CSU) level.

3. Repeat steps 1 and 2 for lower priority source documents.

8.1.3 IDENTIFYING KEY ISSUES

The Systems Engineering process analyzes the mission objectives and requirements to insure that they are feasible and cost effective, and adds functional detail and design requirements with the goal of achieving the proper balance among operational, economic, and logistic factors. This analysis employs a sequential and iterative methodology to reach cost-effective design alternatives. Inconsistent or questionable items should be identified for resolution with the customer or by internal trade studies to help select the most cost effective approach.

Recommended Activities

1. During the decomposition of requirements objects, it is normal for discovery of the need for additional clarification or an apparent conflict with other requirements objects in the database.

   a. Record issue objects in the database, together with information that provides the horizontal traceability from the requirements object where they were discovered.
   b. Record a clear statement of the nature of each issue in the issue object.
   c. Communicate the existence of the issue to the appropriate authority for resolution.

2. Frequently, complete resolution of issues may take time. Also, issue objects may be decomposed into children, grandchildren, etc. which may be resolved at different times. To avoid holding up the process of analysis it is recommended that for each leaf node issue in the database there be a corresponding decision object, wherein are recorded the alternative potential resolutions of the issue. Information should be recorded to provide traceability from the issue object to the decision object. If a time delay is anticipated in resolving an issue it may be desirable to provide traceability to a temporary object in the database named TBR/TBD (to be resolved (value known but not agreed upon)/ to be determined (value unknown)). This permits analysis and generation of draft documentation to proceed. It also permits management tracking of the age, priority, assignment for resolution, etc. of outstanding issues.

When an issue is officially resolved by the appropriate authority (such as a change board, customer contact, etc.), the written record of the event should be recorded in the database (possibly as a type of source object) and information should be recorded to trace the authorizing event to the decision object and also to retire the TBR/TBD object to a lower priority than the authorized decision object. The decision object should be modified to record the authorized solution and the date of resolution. The linkage in the database providing traceability from the TBR/TBD should be modified to trace from the authorized decision object.

8.1.4 GENERATION OF THE SYSTEM REQUIREMENTS DOCUMENT (SRD)

The output of this function will be a baseline set of complete, accurate, non-ambiguous system requirements, recorded in the decision database, accessible to all parties.

To be non-ambiguous, requirements must be broken down into constituent parts in a traceable hierarchy such that each individual requirement statement is:

- Clear, unique, consistent, stand-alone (not grouped), and verifiable
- Traceable to an identified source requirement
- Not redundant, nor in conflict with, any other known requirement.
- Not biased by any particular implementation.

Note that these objectives may not be achievable using source requirements. Often requirements analysis is required to resolve potential conflicts and redundancies, and to further decompose requirements so that each applies only to a single system function.

Recommended Activities

1. Periodically during the analysis process, it is desirable to be capable of generating a "snapshot" report of clarified system requirements. To aid this process it may be desirable to create a set of clarified requirement objects in the database with information providing traceability from its corresponding originating requirement. Clarified requirements may be grouped as functional, performance, constraining, and non-functional for easy access by other team databases.

2. Generate a draft System Requirements Document (SRD) if one does not already exist. Use of an automated database will greatly facilitate this effort, but is not explicitly required. This is the highest level document to be created by the project to represent the customer/user requirements. If a SRD already exists, review it internally and with your customer to insure that it is valid and that you understand it. The SRD should be generalized to fit the range of real-world situations.

8.2 CONCEPT OF OPERATIONS DEFINITION

A Concept of Operations (ConOps) document is produced early in the requirements definition process to describe what the system will do (not how it will do it) and why (rationale). It should also define any critical, top-level performance requirements or objectives (stated either qualitatively or quantitatively) and system rationale. The ConOps should contain a preliminary functional block diagram of the system with only the top-level functional "threads" specified. No attempt is made at this stage to define a complete operational concept or to allocate functions to hardware or software elements (this comes later). This concept of operations is essentially a functional concept definition and rationale *from the user and customer perspective.*

Objective

The primary objective is to communicate with the end user of the system during the early specification stages to ensure that operational needs are clearly understood and incorporated into the design decision database for later inclusion in the system and segment specifications.

Other objectives are:

a. To provide traceability between operational needs and the written source requirements captured.

b. To establish a basis for requirements to support the system over its life, such as personnel requirements, support requirements, etc.

c. To establish a basis for test planning, system-level test requirements, and any requirements for environmental simulators.

d. To generate operational analysis models to test the validity of external interfaces between the system and its environment, including interactions with external systems.

e. To provide the basis for computation of system capacity, behavior under/overload, and mission-effectiveness calculations.

f. To validate requirements at all levels and to discover implicit requirements overlooked in the source documents.

Recommended Activities

The concept of operations consists of describing system behavior, starting with outputs generated by external systems (modified as appropriate by passing through the natural system environment) which act as stimuli to the system, causing it to take specified actions and produce outputs which are absorbed by external systems. These single threads of behavior are traced from source document statements and cover every aspect of operational performance, including logistical modes of operation, operation under designated conditions, and behavior required when experiencing mutual interference with multi-object systems.

Aggregation of these single threads of behavior represents a dynamic statement of what the system is required to do. In some cases, the word “scenario” is used to describe a single thread of behavior and in other cases it describes a superset of many single threads operating concurrently.

1. Starting with the source requirements statements, deduce a set of statements describing the higher-level, mission-oriented system objectives. Record them in the design decision database.

2. Review the system objectives with end users and operational personnel. Disagreements may be recorded in the decision database as issues, resulting in requirements to the source requirements to include operational details outlined.

3. Define the boundaries of the operational models. Identify the different models; e.g., military operational mission model, deployment model, training modes, models, etc.

4. For each model, generate a context diagram to represent the model boundary. Show the system as a top-level, root function within the context of the model boundary. Establish information in the database to provide traceability as follows:
   - System object performs system root function
   - System root function is in context of model
   - System root function traces from functional source requirements

5. Add concurrent functions to the context diagram, which are performed by the sections of external systems that send input stimuli to the system or receive outputs from the system. Add traceability information to the database to record what external systems perform the functions, traced from functional source requirements.

6. Identify all of the possible types of observable input and output events that can occur between the system and its in interacting external systems. Record them as input and output events in the database including information to trace the reason for their existence to originating requirements.

7. If the inputs/outputs are expected to be significantly affected by the environment between the system and the external systems, add concurrent functions to the context diagram to represent these transformations and add input and output events to the database to account for the differences in event timing between when it is emitted to when it is received.

8. Record the existence of a system interface between the system and the environment or external system.

9. For each class of interaction between a part of the system and an external system, create a functional flow diagram to model the sequence of interactions as triggered by the stimuli events generated by the external systems.

10. Add information to trace the function timing from performance requirements and simulate the timing of the functional flow block diagrams (FFBD) to confirm operational correctness or to expose dynamic inconsistencies. In the latter case, record inconsistencies in the design decision database as issued and resolve them following the procedures of 8.1 above.

11. Review the FFBD with end users and operational personnel.
12. Develop timelines, approved by end users, to supplement the source requirements.
13. Draft ConOps are prepared in early project phases, such as concept definition studies or pre-proposal studies. Usually requirements will evolve and prior drafts, should be updated for the next project phase.

Input
The following typical source documents serve as inputs for the ConOps (sometimes called "Operational Concept Document" or OCD):

- Mission Need Statements (MNS)
- Statement of Operational Need
- Technical Operational Requirements
- System Operational Requirements Documents
- Statement of Operational Objectives
- System Requirements Document
- Statement of Work
- Customer Standard Operating Procedures (SOP)
- Internal requirements documents from, for example, manufacturing, product support, or supplier management.

Output
A ConOps comprising:

- A top-level operational concept definition containing approved operational behavior models for each system operational mode (which can be documented as functional flow diagrams), supporting time lines, and event transcripts, which are fully traceable from source requirements
- Trade Analyses
- Operational Procedures with supporting rationale
- System Test Plan test threads and key test features
- Environmental Simulation Plan
- Design Reference Mission

End Result
Understanding of operational needs will typically produce:

- Diminished risk of latent system defects in the delivered operational systems.
- Enhanced probability of meeting schedule and budget targets.

This activity is generally concluded when the Concept of Operations Document (ConOps), sometimes also called the Operational Concept Document (OCD), is released and approved by System Requirements Review.

Metrics
1. Functional Flow Diagrams required and completed;
2. Number of system external interfaces;
3. Number of unresolved source requirement statements;
4. Missing source documents;
5. Number of significant dynamic inconsistencies discovered in the source requirements.

Methods/Techniques
Interviews with operators of current/similar systems and potential users, Interface Working Group meetings, Context Diagrams, Functional Flow Diagrams (FFD), time-line charts, N2 charts.

Example
An example of a ConOps Overview diagram is given in Figure 8.2. The concept description may include multiple diagrams such as this with the top-level data and timing specified for each of the functional threads.

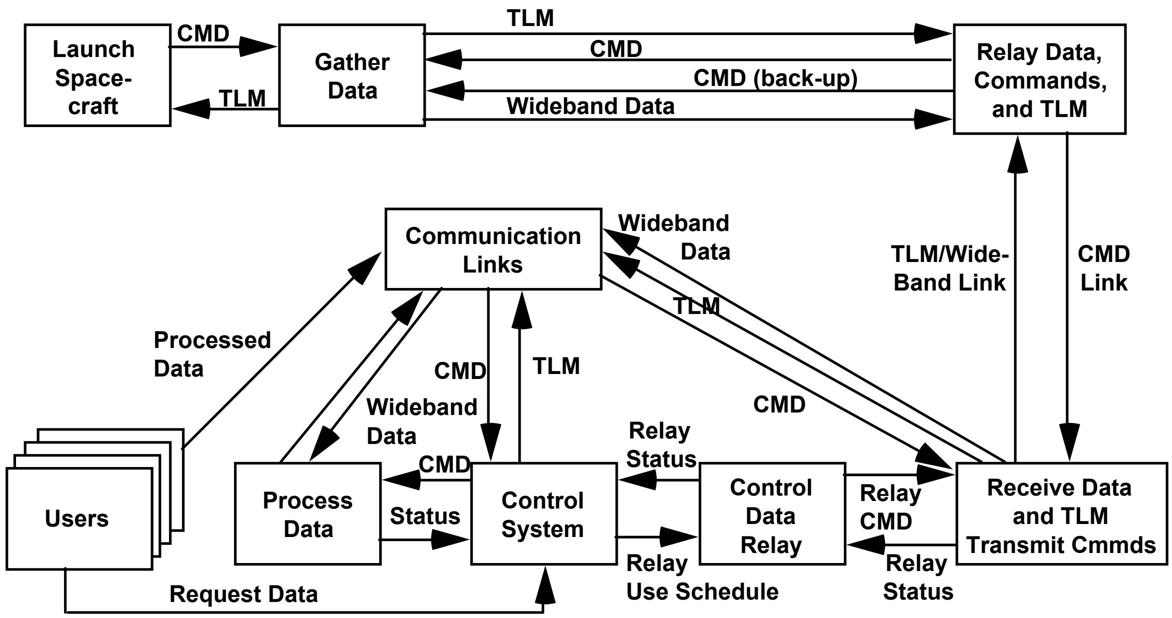


Figure 8-2. Concept of Operations (ConOps) Overview Example

8.3 DEFINE/DERIVE/REFINE FUNCTIONAL/PERFORMANCE REQUIREMENTS

At the beginning of the project, Systems Engineering is concerned primarily with user requirements analysis – leading to the translation of user needs into a quantifiable set of performance requirements that can be translated into design requirements. These objectives are then quantified in broad terms, and basic functions are identified that could fulfil the need.

Functional/Performance requirements definition/derivation/refinement covers the total system over its life cycle, including its support requirements. These need to be formally documented, quantified, performance-based requirements that define the functions and interfaces and characterize the system by performance requirements that can be flowed down to hardware and software designers.

A. Participation/Stakeholders

All Systems Engineering groups will be involved in this activity. In the early phases (up through SRR), this is the primary Systems Engineering activity, with significant support from the design engineering organizations. The customer is also a key stakeholder and validates the work as it progresses.

B. Recommended Activities

Establishing a total set of system requirements is a complex, time consuming task involving nearly all project areas in an interactive effort. It must be done early, since it forms the basis for all design, manufacturing, test, operations, maintenance, and disposal efforts, and therefore determines the cost and schedule of the project. The process is iterative for each phase, with continuous feedback as the level of design detail increases. The overall process is shown in Figure 8-3. The complex interaction of requirements development is best illustrated in an N^2 chart (Figure 8-4).

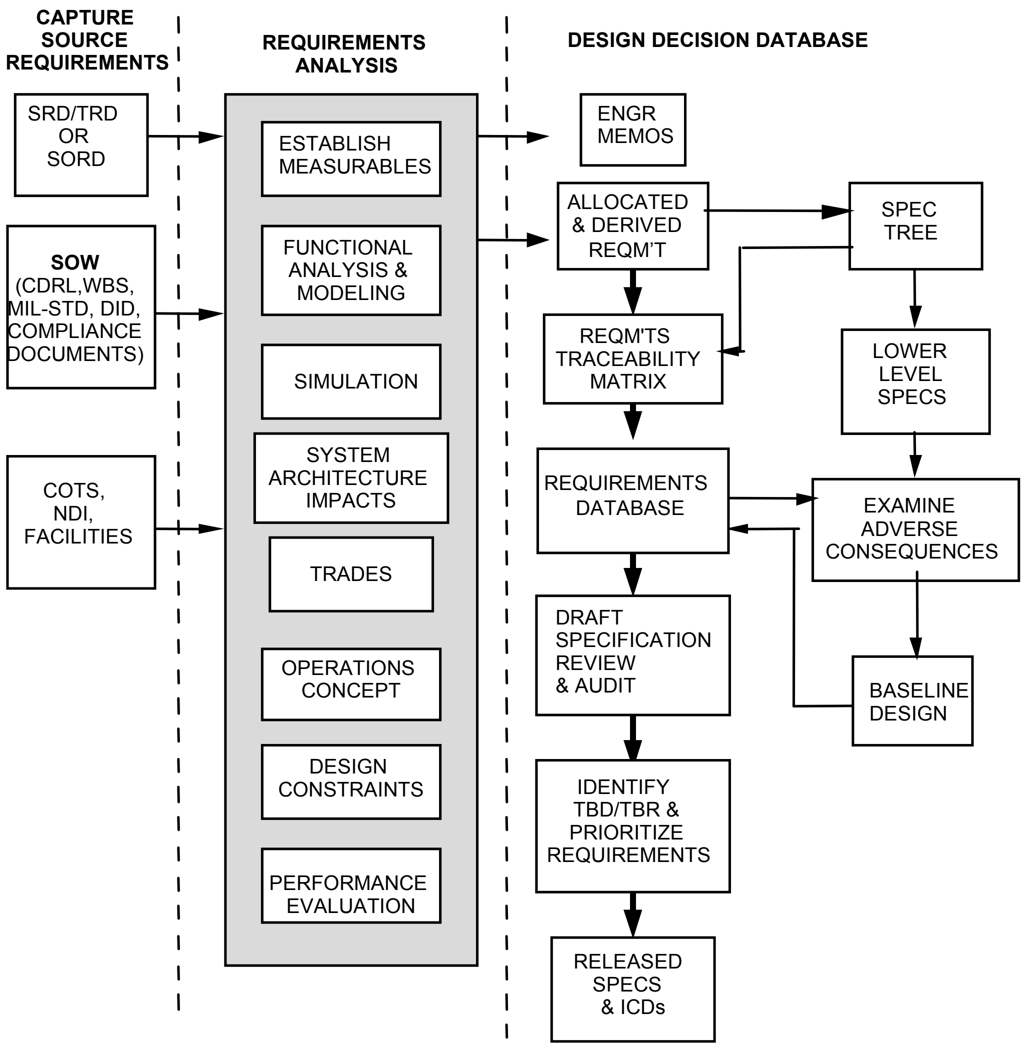


Figure 8-3. Requirements Derivation, Allocation, and Flowdown Process

Capture Source Documents	Requirements	Required Capability	Mission Parameters			External Interfaces, Compliance Documents	Mission Requirements	Mission Requirements, External Interface Requirements
	Develop Ops Concept	Operations	Scenarios	Equipment Experience		Organization & Personnel		Timeline Requirements
		Functional Analysis	Functional Sequences, Timelines	Functional Areas	Required Functionality		Required Capability	Derived Reqts, Functional Reqts, Internal Interface Requirements
			Simulation	Sizing	Sizing		Quantified Performance Capability	Derived Parameter Times
			Configuration	**System Architecture**	Candidate Approaches		Configuration Item Definition	Allocated Path
				Selected Design	**Trade Studies**		Quantified Performance	Quantified Requirements
Tailored MIL-STDs & MIL-SPECs				Ops Environment, GFE, Cost	Cost	**Design Constraints**	Interface Requirements	Engineering Specialty Reqts
Verification of Compliance			Margins & Deficiencies	Margins & Deficiencies	Adverse Consequences	Margins & Deficiencies	**Performance Evaluation**	Requirements
Traceability			Functionality	Functionality	Baseline Capability Required	Traceability of Flowdown	Required Capability	**Requirements Database**

Note: This is an N-squared chart. Outputs are on the horizontal, inputs on the vertical.

Figure 8-4. Functional Interactions in System Requirements Development

The following paragraphs describe the process steps; however, some steps are concurrent and others are not always done in the order shown.

1. The starting point is the set of source requirements developed as described in Section 8.1. Establish constraints on the system including:

- Cost
- Schedule
- Use of Commercial Off-The-Shelf (COTS) equipment
- Use of Non-Development Items (NDI)
- Use of Existing Facilities
- Operational Interfaces with other systems or organizations
- Operational environment

As a result of this activity, a number of functional and performance requirements will be identified.

2. The mission should then be examined and characterized in measurable requirement categories such as: Quantity, Quality, Coverage, Timeliness, and Availability. An example of typical measurables for various systems is shown in Figure 8-5. Actual systems will have many measurables under each attribute, and additional attributes such as communications, command and control, security, etc.

	MEASURABLE			
ATTRIBUTE	**SURVEILLANCE SATELLITE**	**COMMUNICAT'N SATELLITE**	**SUBMARINE**	**AIRCRAFT**
QUANTITY	Frames/Day, Sq Mi/Day	Throughput (BPS)	No. of Missiles Carried	Wt. of Bombs or Armaments (lb)
QUALITY	Resolution (Ft)	S/N or BER	Targeting Accuracy (ft)	Navigation Accuracy (ft)
COVERAGE	Latitude & Long. (deg)	Latitude & Long. (deg)	Range (mi)	Range (mi)
TIMELINESS	Revisit Time (hr), Process/ Delivery Time(sec)	Channel Availability on Demand (min)	Time to get on-station (hr)	Time to acquire target (sec)
AVAILABILITY	Launch Preparation Time (days)	Bandwidth Under Stressed Conditions (Hz)	Cruise Duration (days)	Flight Prep Time (min)

Figure 8-5. Examples of System Attributes And Measurables

3. The above analysis is usually directed at the mission or payload requirements, and does not consider the total system requirements that include communications, command and control, security, supportability, life expectancy. It is necessary to expand the analysis to include supporting areas in order to obtain the total system requirements. Model the system based on the functional analysis to establish all the functions and sub functions to be performed by the system. Graphical models are typically used to represent all functions that the system must perform in executing its mission, and should, in the case of CBSE tools, be capable of execution to provide time line analysis.

4. Use the detailed functional analysis to extract new functional requirements, particularly those required to support the mission. This includes items such as power, propulsion, communications, data processing,

attitude control or pointing, commanding, and human interaction and intervention. This will eventually result in the conversion from mission parameters (targets/sq mi) into parameters that the hardware and software designers can relate to, such as effective radiated power (ERP), Received Signal Strength Intensity (RSSI), etc. Functional decomposition tools such as functional block diagrams, functional flow diagrams, time lines, control/data flow diagrams are useful in developing requirements. Quality Function Deployment (QFD) is also useful, particularly where the "voice of the customer" is not clear (See Appendix A). As requirements are derived, the analysis that leads to their definition must be documented and placed into the decision database.

5. For larger systems, develop a high-level system simulation evolved from the system architecture. The simulation should contain sufficient functional elements that the interactions can be properly assessed. The purpose of the simulation is to establish measurable parameters for the functional requirements developed above, and convert them wherever possible from functional requirements to performance requirements. This provides the necessary guidance to the designers on the size and capability required of their equipment. In addition, these parameters will be used as an integral part of the verification process in establishing the capability of the equipment (and the system) to satisfy user needs. The simulation will be used to quickly examine a range of sizes and parameters, not just a "Point Design". This will insure that the "best" solution is obtained - the system is the proper size throughout, with no choke points. Exercise the simulation using scenarios extracted from the concept of operations with inputs based on system requirements. A number of scenarios should be run to exercise the system over the possible range of mission activities. Monte Carlo runs may be made to get averages and probability distributions. In addition to examining nominal conditions, non-nominal runs should also be made to establish system reactions or breakage when exposed to extraordinary (out-of-spec) conditions. This will establish (or verify) timeliness requirements.

6. Examine any adverse consequences of incorporating requirements:

- Is unnecessary risk being introduced?
- Is the system cost within budget limitations?
- Is the technology ready for production?
- Are sufficient resources available for production and operation?
- Is the schedule realistic and achievable?

7. Where existing user requirements cannot be confirmed, trade studies should be performed to determine more appropriate requirements, and achieve the best-balanced performance at minimum cost. Where critical resources (Weight, Power, Memory, Throughput, etc.) must be allocated, trade studies may be required to determine the proper allocation.

8. Revise the simulation as a result of the trade studies and rerun. Evaluate the performance of the candidate solutions and compare to the original results. When User needs are satisfied, establish the baseline set of system performance requirements.

9. Incorporate revised and derived requirements and parameters into the decision database and maintain traceability.

10. Prepare the complete system/segment specification(s) and submit to all organizations for review.

11. Use an interdisciplinary team to audit the specification to assure good requirements practices, including the following:

- Traceability to source documentation
- Clarity of requirements statements

- Capability of requirement to be verified
- Completeness of requirements set
- That the requirement states "what", not "how"
- Each requirement contains a shall
- Each requirement is unique and not redundant
- All requirements have parents
- All requirements are verifiable by methods such as inspection, test, etc.
- That the flowdown is correct and complete

12. Assess requirements as to degree of certainty of estimate, and place a TBD, TBR, TBS flag on items requiring further attention. Maintain a list of all TBD/TBR/TBS items with responsibilities and closure dates in the decision database. Enter the TBDs/TBRs/TBSs into the Risk Register, as appropriate.

13. Prioritize all requirements as to the criticality of mission success. Since resources on any project are limited, this identifies where the effort should be concentrated in refining, deriving, and flowing down requirements.

14. Incorporate audit findings and appropriate comments into the decision database, and generate a specification for final review and approval.

15. Generate specification documents as needed, enter the document into the formal release system, and maintain it under configuration management control. Any further changes will require Configuration Control Board (CCB) approval.

Input

System Requirements Document, Statement of Work, Company Policies and Procedures, Concept of Operations Document (or Operations Concept Document), Design Concept, System Hierarchy, and Data Item Description.

C. End Result

The result of performing this requirements analysis function should be a baseline set of complete, accurate, non-ambiguous system requirements, recorded in the decision database, accessible to all parties, and documented in an approved, released System Specification.

Metrics

1. Number or percent of requirements defined, allocated, and traced;
2. Time to issue draft;
3. Number of meetings held;
4. Number and trends of TBD, TBR, and TBS requirements;
5. Product - Approval time;
6. Number and frequency of changes (additions, modifications, and deletions).

D. Methods/Techniques

Functional decomposition using a system hierarchy, functional block diagrams, functional flow diagrams, time lines, control/data flow diagrams, trade studies, requirements allocation sheets, and Quality Function Deployment.

8.4 REQUIREMENTS ALLOCATION AND TRACEABILITY

Traceability should be maintained throughout all levels of documentation; traceability is both vertical and horizontal for specifications (CI and interface), and should include traceability to the test program (plans, procedures, test cases, and reports) to provide closed loop verification.

1. Allocate all system requirements to hardware, software, or manual operations;
2. Ensure that all functional performance requirements or design constraints, either derived from or flowed down directly to a system architecture component, have been allocated to a system architecture component;
3. Ensure that traceability of requirements from source documentation is maintained through the project's life until the test program is completed and the system is accepted by the customer; and
4. Ensure that the history of each requirement on the system is maintained and is retrievable.

Recommended Activities

1. While requirements can be traced manually on small projects, such an approach is generally not considered cost-effective, particularly with the proliferation of requirements management tools. A requirements traceability tool that augments the decision database should be accessible to and usable by all technical personnel on the project. This includes subcontractors who are preparing specifications and verification data. Inputs to the database will include draft specifications, comments, approvals, status data, change data, and requests.

The tool should generate the following directly from the database:

a. Requirements Statements with Project Unique Identifiers (PUID)
b. Requirements Traceability Matrices (RTM)
c. Verification Cross Reference Matrices (VCRM)
d. Lists of TBD, TBR, and TBS
e. Specifications
f. Requirements metrics (e.g., requirements stability)

The tool must have configuration management capability to provide traceability of requirements changes, and ensure that only properly authorized changes are made

2. Use the system level requirements defined in Section 8.3 as the starting point for the allocation process. Give each defined and derived requirement and design constraint a PUID.

Each requirement must be traceable using a Project Unique Identifier (PUID). The specification tree provides the framework for parent-child vertical traceability (tree-down or tree-up) used for specifications. For interface documents such as Interface Control Documents (ICDs) the traceability is horizontal - in some cases over several levels. Thus, the specification tree does not adequately portray interface traceability. However, the decision database tool must have capability for both vertical and horizontal traceability. The PUID is an alpha numeric assigned to each requirement. The alphas employed are similar to acronyms in order to provide an easily recognizable identification of the functional area. This is particularly useful when requirements statements are extracted from many specifications as part of the audit process. The numeric portion is assigned within individual documents.

3. Identify the functions and sub functions for which each area is responsible, and the top level system requirements associated with those functions. Assign a PUID to each of the functions (system actions) and sub functions developed in Section 9.1. For each system action, identify functional/performance requirements to be associated with it. Capture this association in the decision database. For each function and sub function, identify which system component in the system architecture (Spec Tree) is responsible for it, and capture this information in the decision database.

4. The most difficult part of requirements flowdown can be the derivation of new requirements, which often involves a change in the parameters as appropriate to the level in the hierarchy (targets per sq. mi - a system parameter - has little meaning to the hardware designer). Repeat the process at each level until the CI level is reached. At the lowest (CI) level, the parameters specified must be relevant to that particular equipment item, and provide adequate direction to the designer.

5. As each requirement is identified at the lower level, assign a PUID to it, and enter it into the decision database. The traceability should include the following attributes:

a. The requirement identification number (PUID)
b. The source of the requirement, such as the customer's document paragraph number or the engineering report documenting the analysis that derived the requirement.
c. The full text of the requirement
d. For allocated or derived requirements, a pointer to the requirement from which it was derived, or "parent" requirement.
e. A pointer to the next lower-level area that this requirement was allocated to during the allocation process
f. Verification level, method, and category
g. The Test Plan name & number controlling the verification
h. The Test Procedure name & number performing the verification
i. The date and results of the final verification
j. The name of the responsible engineer.

With the completion of specifying the requirements for a functional area that include the unique requirement designator, the requirement text, the source requirement designator, the next lower level allocation designator, and the logical function charts, the entire system can be reviewed in a logical manner. This can assure that all system requirements are allocated and traceable to some function, and that all lower-level requirements can be traced upward to a "parent" and ultimately to a source requirement.

6. Throughout the requirements identification, derivation, definition process (including not only functional/performance but also design constraints) provide configuration management and configuration control maintenance of the decision database. For each requirements change, ensure that changes and modifications have been approved by personnel and organizations appropriate to that level. If changes affect only one functional area (system Component), ensure that review and approval is accomplished by responsible design engineers in that area. If the change affects two or more functional areas, ensure that the change is coordinated through all areas and if there is arbitration needed, that the appropriate level of engineering decision is addressed and decisions made.

7. Audit the specifications as they are produced to verify that the allocation process is correct and complete. Use the Requirements Database to generate audit reports that contain the flowdown of requirements statements. Identify proposed corrections and changes, and process them through the proper approval channels.

8. Generate Requirements Traceability Matrices (RTM) from the database.

End Results
Traceability is achieved when all requirements have been placed in the database, and all specifications have been released. A complete set of allocated requirements should be found in specifications, with a Requirements Traceability Matrix (RTM).

Input

Specification Tree and SRD or System Specification

The initial definition of system requirements from the source documents defined in 8.1 is completed using a combination of graphical functional analysis tools and simulations as described in Section 9.1. As the requirements are developed, a design concept and a concept of operations (8.2) are developed concurrently. The output of this effort is a set of requirements statements, which are placed in the System Specification as described in 8.3. A specification tree is developed first that identifies all requirements documents on the program and provides the hierarchy for requirements flowdown and traceability.

Output

- **Specifications** - The primary output of the Requirements Database is specifications. Draft specifications are generated by the database, and distributed to reviewers. The copies are returned with comments as appropriate, to the author. When all comments are resolved, the document is formally released. The Requirements Database tool should generate the specification directly from the database without manual intervention, thereby preserving the integrity of the decision database.

- **Audit Reports** - Auditing is a major Systems Engineering effort during the specification preparation phase. Audit reports can take many forms, from a simple check for missing parents or attributes to a complete tree-down from the system level to the CI level. The latter usually involves an interdisciplinary team to insure that the flow-down and allocated requirements provide complete satisfaction of the upper level requirement and that they are abstract (no implementation) and are clearly stated. Each requirement statement and its children is extracted from the database, and reviewed in sequence throughout the specification.

- **Requirements Traceability Matrices** - The Requirements Traceability Matrices (RTMs) are generated directly from the database, and are also used as part of the audit process.

- **Status Reports** - As the system acquisition cycle proceeds, increasing effort will be directed toward verification that the demonstrated capability of the system meets its requirements as expressed in specifications. The database plays a major role in this by incorporating the verification data in its attribute files, either directly or by pointer to other databases where the data are located. Status reports on verification progress, TBD/TBR/TBS elimination, and requirements changes can be obtained by sorting the appropriate attribute listings.

Metrics

1. Number and trends of requirements in the database;
2. Number of TBD, TBR, and TBS requirements.
3. Number (or percent) of system requirements traceable to each lower level and number (percent) of lower level requirements traceable back to system requirements.

Methods/Techniques

A large variety of tools are available for requirements management and systems architecting. Since the information on these tools becomes outdated approximately every six months, INCOSE has elected to maintain a current database on SE tools available to anyone at its World Wide Web site. This site can be accessed at URL = http://www.incose.org/.

Example

An example of a Requirements Traceability Matrix is shown in Figure 8-6.

SYSTEM TYPE A SPECIFICATION				SEGMENT TYPE A SPEC				ELEMENT TYPE B1 SPEC				B2/B5 SPEC	
PARA	PUID	PARA	PUID	PARA	PUID	PARA	PUID	PARA	PUID	PARA	PUID	PARA	PUID
3.2.1.1.1	SYS0010	3.7.1.1	SAT0001	3.2.1.2.3	SAT0010	3.7.3.1	EPS0020	3.2.1.2	EPS 0020	3.7.2.1	PDS0034	3.2.1.2	PDS0090
													PDS0095
													PDS0098
										3.7.2.1	PDS0035	3.2.2.2	PDS0100
													PSD0110
													PSD0120
										3.7.2.2	CTR0045	3.2.1.5	CTR0056
													CTR0089
						3.7.3.2	EPS0021	3.2.3.2	EPS0021	3.7.2.2	PDS0045	S3.2.2.4	PDS0045

Figure 8-6. Requirements Traceability Matrix

Where:

CTR	= Control	EPS	= Electrical Power Subsystem
SYS	= System	PDS	= Power Distribution
SAT	= Satellite	0020	= Req't no. in specification
TYPE A	= System Specification	TYPE B	= Development Specification

8.5 DEVELOPMENT OF SPEC TREE AND SPECIFICATIONS

Create a Spec Tree for the system and specifications for each configuration item of the system under development. This activity represents the establishment of the documented baseline of a particular level of the system design. For complex systems there may be multiple iterations performed wherein the definition/design process is successively applied in a hierarchical manner down to the level of hardware and software configuration item definition. These baselines are captured in configuration item specifications. The road map and hierarchical representation of the specifications is the Spec Tree. The specifications document the set of configuration items (hardware, software, operations, etc.), which will implement the system.

The objective is to create a specification baseline for each of the configuration items and place these specifications in a flowdown hierarchy. This will allow the further definition of each configuration item to proceed independently, in parallel with all the others, while maintaining requirements traceability and compatibility of all items that make up the system. This section concentrates on the production and quality factors of the Spec Tree and the specifications.

The most common source of development problems is flaws in the system specification process. The importance of the system specification(s) in providing the framework for the entire development effort should be stressed. It is widely recognized that the cost of correcting errors is least early in the development cycle and that specification errors found late in the development cycle have a higher average cost to repair. The influence of the system specification(s) on establishing a development and test direction is not widely appreciated.

This lack of understanding is manifested in one of several ways. First, the document is deemed to be something required by management or the customer and irrelevant to the design, development, and testing of the system. Second, the document is erroneously perceived as intended to capture both the requirements and the design. The first problem tends to lead to a development without adequate design direction, which often results in programs running into severe problems late in their development. These cases often involve ``over-designed'' systems in which the design was directed by engineers' wishes rather than requirements. The second problem leads to voluminous specifications, which often results in specifications not being kept up to date and misdirected test efforts.

The end-result of either scenario is often an increase in cost, schedule, and technical risk for the program. On complex developments, it is imperative to develop and maintain a proper hierarchy of specifications. The concomitant delay in achieving baselines adversely affects project schedule. Even worse, when schedule pressures force the project to proceed with implementation before specifications are acceptable, unintended or abandoned requirements in the delivered system may result. If corrections are attempted at later stages in the program, associated costs grow significantly.

In practice, requirements engineering is not just a front-end to the system development process but a complex communication and negotiation process involving the parties that will use the system, i.e., the customers; the parties that will provide parts or all of the system, i.e., the developers and vendors; and the parties that will test the system, i.e., the test group(s). Systems Engineering acts as the translator in this communications process with the system specifications being the key written embodiment of this communication. Some of the major challenges facing the Systems Engineer in performance of this requirements engineering task are:

- An envisioned system is seldom, if ever, designed to work totally independent of the other systems in the customer's environment. This means that the environment in which the system is to operate must be known and documented as thoroughly as the system itself.

- Off-the-shelf solutions or components play a major role in defining the system. While requirements are supposed to be independent of solution, being able to achieve an implementable solution within the resource constraints available is the *primary* requirement.

- Every aspect of an envisioned system's function and performance cannot practically be specified. Thus, a level of requirement specification must be established which represents a cost-effective balance between the cost of generating, implementing, and testing requirements versus the risk of not getting a system meeting customer's expectations. In each case, the cost of non-performance is a major driver. For example, a life support system merits far more development rigor than a prototype digital camera system.

The Systems Engineering process is a bridging process translating an identified need into a system solution composed of specified implementable hardware and software elements. The process is very much a communication process with all the potential flaws of any communication plus the added uncertainty of the customer's real desires and the risks associated with achieving an implementation. In this environment, it is not surprising that so many system development efforts have problems. It is the purpose of this section to guide that communication process resulting in a proper set of system specifications. Specifications exist to assist the customer in early visualization of the emerging system reducing the risk of not meeting his desires; and configuring an easily implemented system thus reducing the development risk.

A. Participation

This function is lead by Systems Engineering, with support from design engineering and the supporting disciplines.

Systems Engineering creates the Spec Tree, the outlines for each of the specifications, crafts the requirements, and establishes traceability. Systems Engineering also ensures that the supporting disciplines are present and active, scopes their participation, and ensures that their contributions are coordinated and integrated.

Design engineering provides technical definition data for derived and reflected requirements, and documents design decisions.

Supporting disciplines monitor implementation of requirements in each specialty area, identify derived and reflected requirements, and review the results of the requirement definition process.

B. Recommended Activities

1. Derive the Spec Tree from the system architecture configuration

The system hierarchy should be a balanced hierarchy with appropriate fan-out and span of control. Appropriate fan-out and span of control refers to the number of elements subordinate to each element in the hierarchy. Hierarchies are organizational representations of a partitioning relationship. The hierarchy represents a partitioning of the entity into smaller more manageable entities.

System hierarchies are analogous to organizational hierarchies. Both can suffer from improper balance; that is, too great a span of control or excessive layers in the hierarchy. A "rule of thumb" useful in evaluating this balance is that an entity should have 7 $\pm$ 2 entities reporting to it. *What is an entity? In most cases, an entity is a configuration item represented by a specification; however, it may represent a major purchased item not requiring a specification.* A design level with too many entities reporting suffers from too much complexity. The design and corresponding test activities run the risk of running out-of-control or acquiring an informal partitioning that guides the work without proper control or visibility. A level of design with too few entities likely does not have distinct design activity, and both design and testing activities contain redundancy. Figure 8-7 shows a typical specification tree.

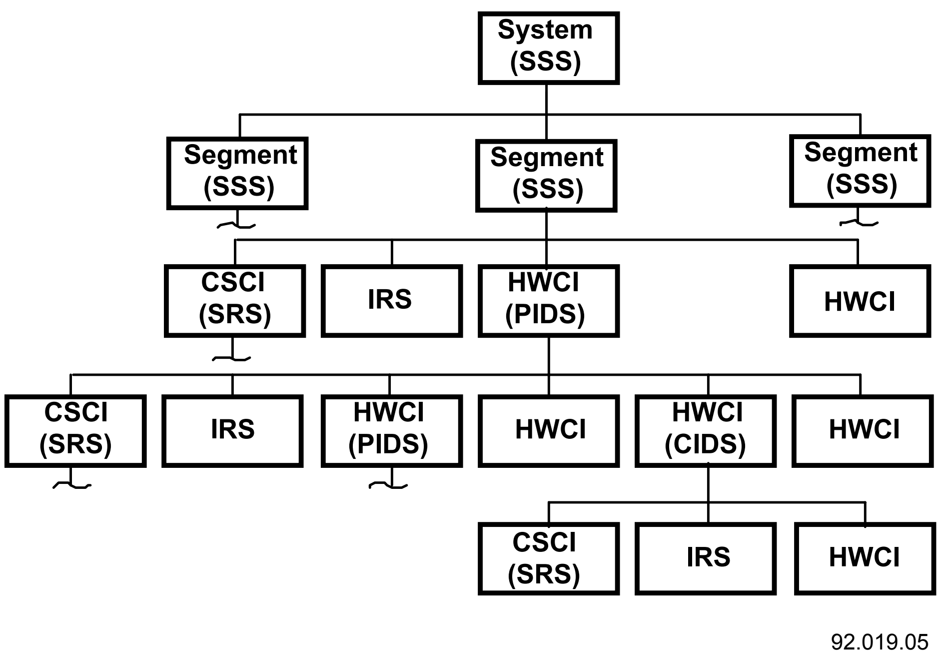


Figure 8-7. Typical Project Specification Tree

Developing the specification tree is one element of system design whereby the system is decomposed into its constituent parts. This process has major ramifications on the development of the system in that it essentially determines the items to be purchased versus those to be developed and establishes the framework for the integration and test program. The objective in the design is to achieve the most cost-effective solution to the customer's requirements with all factors considered. Generally, this is achieved by identifying existing or implementation units as early as possible in the tree development. At each element or node of the tree a specification is written, and later on in the project a corresponding individual test will be performed. When identifying elements, it is useful to consider the element both from a design and a test perspective. The element should be appropriate from both perspectives.

Specifications must be written, in some form, for every item of hardware and software comprising the system. The specification and the supporting design documentation establish the configuration of the system. Off-the-shelf items (non-configuration items) are items with standard part numbers whose supporting documentation and configuration are controlled by the manufacturer. All items of hardware or software that we develop or have developed require a specification, supporting design documentation, and configuration control.

Specification and design documentation represent a minimum to develop the system. The customer may require additional documentation depending on his plans for life cycle support and additional production. The Specification Tree should be carried to a level where an individual unit in hardware or an individual computer software configuration item (CSCI) is specified.

As a second check on configuration item size, a configuration item should not be larger than something that can be developed by 7 ± 2 people. The number of levels in the tree is then determined by the number of levels required to decompose to the unit level while maintaining appropriate span of control and assuring that the element specifications contain appropriate complexity.

2. For each specification in the Spec Tree, create an outline using a standard specification template and the definition of the configuration item.

Specification outlines or templates may be obtained from several sources. The most useful and commonly used are previous similar specifications prepared by your organization. These are often the source of useful material, parts of which can be used with minimal modification. In addition, there are Standard formats and IEEE formats recommended for system, hardware, and software specifications.

3. Craft requirements for each specification, fulfilling all flowdown and accommodating derived and reflected requirements emerging from the definitions of each configuration item.

A specification represents a design entity and a test entity. The specification should represent appropriate complexity from both the design and the test perspective. Many factors contribute to the appropriate selection of elements. However, as a measure of complexity, a requirements specification should not have too many or too few requirements. As a "rule of thumb" 50-500 functional/ performance requirements in a specification is appropriate. Requirements in the physical or environmental areas would be in addition to the functional/ performance variety.

In defining the requirements in a specification, care should be exercised to assure the requirement is appropriately crafted. The following questions should be considered for every requirement:

1. **Is each requirement clear?** Requirements must convey what is to be done to the next level of development. Its key function is to communicate. Is the requirement clear, compatible and complete?

Is it possible to interpret the requirement in multiple ways? Are the terms defined? Does the requirement conflict or contradict another requirement?

2. **Is each requirement a proper requirement?** A requirement's specification is a demand on the designer (or implementer) at the next level. Is this requirement at the proper level? Customer requirements may be imposed at any level they desire; however, when customer requirements specify design, it should be questioned. When generating requirements, the requirements should be targeted at the next lower level and no lower (except when carrying forward a legitimate customer design requirement). A proper requirement should deal with the entity being specified as a "black box" describing what transformation is to be performed by the "box". The requirement should specify "what" is to be done at that level, not "how" it is to be done at that level.

3. **Is the requirement necessary?** Every requirement generates extra effort in the form of processing, maintenance, and testing. Only necessary requirements should be written. Unnecessary requirements are of two varieties: (1) unnecessary specification of design which should be left to the discretion of the designer, and (2) a redundant requirement covered in some other combination of requirements.

4. **Is each requirement consistent with product standards?** In many instances, there are applicable government, industry and product standards, specifications, and interfaces with which compliance is required. An example might be additional requirements placed on new software developments for possible reusability. Another might be standard test interface connectors for certain product classes.

5. **Is each requirement achievable?** It is imperative that the implementing designer participate in requirements definition. The designer should have the expertise to assess the achievability of the requirements. In the case of items to be subcontracted, it's important that the expertise of potential subcontractors be represented in the generation of the requirements. Additionally, participation by manufacturing and customers/users can help assure achievable requirements. IPPTs and requirements reviews provide mechanisms to achieve these perspectives.

6. **Do the requirements pass the traceability test?** Do all requirements trace to the higher level specification? Are there requirements at the higher level not allocated (or allocated, but not picked up)? Those with no allocation may be satisfied at that level of the specification. Requirements with either deficiency should be corrected.

7. **Is each requirement verifiable?** Each requirement must be verified at some level by one of the four standard methods (test, demonstration, analysis, or inspection). A customer may specify, "The range shall be as long as possible." This is a valid but unverifiable requirement. This type of requirement is a signal that a trade study is needed to establish a verifiable maximum range requirement. Each test requirement should be verifiable by a single test. A requirement requiring multiple tests to verify should be broken into multiple requirements. There is no problem with one test verifying multiple requirements; however, it indicates a potential for consolidating requirements. When the system hierarchy is properly designed, each level of specification has a corresponding level of test during the test phase. If subsystem specifications are required to appropriately specify the system, subsystem verification should be performed.

Requirements must be written with extreme care. The language used must be clear, exact, and in sufficient detail to meet all reasonable interpretations. A glossary should be used to precisely define often-used terms or terms that could have multiple interpretations. In most writing, it is desirable to substitute words that are more or less synonymous in order to avoid the constant repetition of a word. However, because few words are exact synonyms, requirements should be written using the same wording with exact meaning established. Care must be taken in utilizing clear, unambiguous phraseology and punctuation. A misplaced comma can have dramatic ramifications.

Often requirements are written in a vague manner when the author is not sure of what is required. However, a vague requirement is left open to interpretation, first by the next level designer who is likely to be less qualified to establish the requirement and later by customer test personnel who are likely to make the most stringent interpretation. The effort should be expended to establish the exact requirement at the time required or, at a minimum, flag the requirement as a critical issue for early resolution.

Verb tense and mood in requirements specifications are very important. The following describes the common use of the forms of the verb "to be" as they apply to specifications:

- "Shall" - Requirement specifications are demands upon the designer or implementer and the resulting product, and the imperative form of the verb, "shall", shall be used in identifying the requirement.

- ``Will" - statement containing "will" identifies a future happening. It is used to convey an item of information, explicitly not to be interpreted as a requirement. "The operator will initialize the system by …" conveys an item of information, not a requirement on the designer of his product. However, some organizations have dropped the distinction between "shall" and "will" in specifications, and treat either word as a means of stating a requirement.

- "Must" - "Shall" is preferable to the word "must". If both are used in a requirements specification, there is an implication of difference in degree of responsibility upon the implementer.

- Other forms - "To be", "is to be", "are to be", "should" and "should be" are indefinite forms of the verb, and they have no place in requirement specifications.

The imperative mood may be used as well in specifying requirements. For example, "The database shall be dumped to magnetic tape every four hours." Requirements done in table format, usually express the processing requirements in the imperative mood. Judicious use of the imperative mood can eliminate many words and enhance the readability of specifications.

Use tables where possible. They usually convey requirements clearly and concisely.

There are words whose use should be avoided in requirements in that they covey uncertainty. These include:

- <u>Pronouns.</u> Pronouns should be avoided or used with care in that they can lead to ambiguity or confusion as to exact meaning. Words such as "he", "she", "this", "they", "their", "who", "it", and "which", should be used sparingly, if at all.

- <u>Adjectives and adverbs.</u> Adjectives and adverbs generally convey an indefinite degree. Words such as "timely", "real-time", "precisely", "appropriately", "approximately", "various", "multiple", "many", "few", "limited", and "accordingly" should be avoided in requirements.

- Other indefinites. Words or phrases such as "etc." and "and so on" usually indicate an unbounded list. "To be determined" is generally an official flag of uncertainty. If used, "to be determined" along with "to be supplied" and "to be reviewed" should be logged and documented in a table at the end of the specification with an assigned person for closure and a due date. The word "process" needs to be used with care. When used, that processing must be clearly defined.

Input

1. System requirements and functional architecture as defined by previous steps.
2. System configuration, with sufficient technical supporting data.

Output

1. Specification Tree.
2. Specifications for each configuration item.

C. End Result

The result is the Spec Tree and the set of specifications for all of the configuration items that implement the system.

Completion Criteria

1. All specifications identified and located on the Spec Tree.
2. Each specification adequate to proceed with the next stage of development or procurement.

D. Methods/Techniques

The design and development methods described in the earlier sections apply to this step. As for the actual generation of the Spec Tree and the Specifications, templates and previously completed specifications are useful starting points for document generation.

Metrics

For the Specification Tree:

1. Its completeness as measured by its inclusion of all items required in the system
2. Its balance as determined by its span of control and fan-out from each element.

For the Specifications,

1. TBDs and TBRs in specifications.
2. Number of requirements in the specification (50-250 functional/performance requirements is the ideal range).
3. Stability of the requirements as the development progresses.

Tools

There are a number of commercially available requirement generation and maintenance support tools available. Check the INCOSE website for current tool availability information.

9 FUNCTIONAL ANALYSIS/ALLOCATION

This section discusses the activities associated with Functional Analysis and Allocation activities. See also Appendix D *Methods for Functional Analysis and Allocation* with Key supporting Methodologies.

Introduction to Functional Analysis/Allocation

A *function* is a characteristic task, action, or activity that must be performed to achieve a desired outcome. For a product, it is the desired system behavior. A function may be accomplished by one or more system elements comprised of equipment (hardware), software, firmware, facilities, personnel, and procedural data.

The scope of the *Functional Analysis/Allocation* activity can be defined by the following:

1) *Functional Analysis/Allocation* is an examination of a defined function to identify all the subfunctions necessary to the accomplishment of that function. The subfunctions are arrayed in a functional architecture to show their relationships and interfaces (internal and external). Upper-level performance requirements are flowed down and allocated to lower-level subfunctions.

2) This activity should be conducted to define and integrate a functional architecture for which system products and processes can be designed. Functional analysis/allocation must be conducted to the level of depth needed to support required synthesis efforts. Identified functional requirements must be analyzed to determine the lower-level functions required to accomplish the parent requirement. All usage modes must be included in the analysis. Functional requirements should be arranged so that lower-level functional requirements are recognized as part of higher-level requirements. Functional requirements should be arranged in their logical sequence; have their input, output, and functional interface (internal and external) requirements defined; and be traceable from beginning to end conditions. Time critical requirements must also be analyzed.

3) The performance requirements should be successively established, from the highest to lowest level, for each functional requirement and interface. Time requirements that are prerequisite for a function or set of functions must be determined and allocated. The resulting set of requirements should be defined in measurable terms and in sufficient detail for use as design criteria. Performance requirements should be traceable from the lowest level of the current functional architecture, through the analysis by which they were allocated, to the higher-level requirement they are intended to support.

4) Functional analysis/allocation should be conducted iteratively:

- To define successively lower-level functions required to satisfy higher-level functional requirements and to define alternative sets of functional requirements.
- With requirements analysis to define mission and environment driven performance and to determine that higher-level requirements are satisfied.
- To flow down performance requirements and design constraints.
- With design synthesis to refine the definition of product and process solutions.

5) Trade-off studies should be conducted within and across functions to:

- Support functional analyses and allocation of performance requirements.
- Determine the preferred set of performance requirements satisfying identified functional interfaces.
- Determine performance requirements for lower-level functions when higher-level performance and functional requirements cannot be readily decomposed to the lower level.
- Evaluate alternative functional architectures.

Functional Architecture is defined as the hierarchical arrangement of functions, their internal and external (external to the aggregation itself) functional interfaces and external physical interfaces, their respective functional and performance requirements, and the design constraints.

Functional Analysis/Allocation, as an early step in the Systems Engineering process, defines a baseline of functions and subfunctions and an allocation of decomposed performance requirements. All aspects of system functionality should be addressed, including production, deployment, logistical support, and operations. Functional analysis and decomposition can be performed independently of system architecture, but functional allocation obviously requires a system architectural structure. The functional requirements provide a common basis for the selection and design criteria for system elements and identify areas where tradeoffs between input requirements and engineering development require further consideration.

In the Systems Engineering process flow, the candidate implementation is developed in the *System Synthesis* phase, which follows *Functional Analysis/Allocation*. An independent functional analysis and decomposition will sometimes lead to creative implementation approaches because functional needs are better understood before synthesis begins. After several levels of functional decomposition, it is appropriate to begin the synthesis process to define one or more candidate architectures for evaluation.

Many functions can be decomposed in any of several alternative ways. Some ways make logical sense, and some do not. Some will lend themselves to further decomposition, while others will not. Some lead to economical implementations, while others will lead to complications. For that reason, the Systems Engineer may need to compare and evaluate candidate functional architectures via trade studies. For example, there is a strong trend towards required use of Commercial Off-the-Shelf (COTS) products. There also may be a requirement, perhaps only implicit, to reuse existing software code modules, or Non-Developmental Equipment (NDE). The Systems Engineer must be aware of such constraints when the functional architecture is developed, or the result might be incompatible with a hardware or software architecture based on the desired products. Trade studies are a *Functional Analysis/Allocation* tool to help the Systems Engineer address these and other design constraints.

9.1 PURPOSE OF THE FUNCTIONAL ANALYSIS/ALLOCATION TASK

The objective of *Functional Analysis/Allocation* is to create a functional architecture that can provide the foundation for defining the system architecture through the allocation of functions and subfunctions to hardware/software and operations (i.e., personnel). It should be clearly understood that the term (functional architecture) only describes the hierarchy of decomposed functions and the allocation of performance requirements to functions within that hierarchy. It does not describe either the hardware architecture or software architecture of the system. Those architectures are developed during the *System Synthesis* phase of the Systems Engineering process.

Functional Analysis/Allocation describes what the system will do, not how it will do it. Every function that must be done by the system in order to meet the operational requirements needs to be identified and defined in terms of allocated functional, performance, and other limiting requirements. Then, each of these functions is decomposed into subfunctions, and the requirements allocated to the function are each decomposed with it. This process is iterated until the system has been completely decomposed into basic subfunctions, and each subfunction at the lowest level is completely, simply, and uniquely defined by its requirements. In the process, the interfaces between each of the functions and subfunctions are fully defined, as are the interfaces to the external world.

The *Functional Analysis/Allocation* task should provide added value to the over-all Systems Engineering process above and beyond the development of the functional architecture. This added value includes the identification of missing functional requirements, development of derived requirements, and identification of unrealistic or poorly written requirements.

Functional Analysis/Allocation supports mission and operations-concept analysis in defining functional areas, sequences, and interfaces. *Functional Analysis/Allocation* is also used by engineering specialists and support organizations to develop derived requirements for equipment, software, personnel, facilities, and operational procedures to complete implementation, test, and deployment of the system.

Input Criteria

The more that is known about the system the better. Ideally, Functional Analysis/Allocation should begin only after all of the system requirements have been fully identified. This means that the Requirements Analysis must be completed before this task starts. Often, of course, this will not be possible, and these tasks will have to be done iteratively, with the functional architecture being further defined as the system requirements evolve. The output of the Requirements Analysis task may be incomplete, and the omissions may be well understood, or may not be recognized at all. The Functional Analysis/Allocation task should help to reveal any missing requirements, and help to refine or clarify others.

Representative inputs from the user/customer or program management are:

- Customer needs, objectives, and requirements
- Technology base
- Program decision requirements (such as objectives to reuse certain HW & SW)
- Specifications and Standards requirements
- Concept of Operations

The entire Systems Engineering process, including Requirements Analysis, Functional Analysis/Allocation, System Architecture Synthesis, and Systems Analysis and Control, is carried out many times throughout the system life cycle. This includes various levels of detail during system design. It will occur during early concept development, recur during system procurement, and be repeated from segment down to Configuration Item levels. For that reason, there is no single requirements document that can be cited as a prerequisite for initiating and completing the Functional Analysis/Allocation process. At the highest levels, a system-level specification is desirable. At lower levels a segment-level specification or a Configuration Item or Computer Software Configuration Item specification may suffice. The flow down of system requirements to lower levels is based upon a mission area analysis and system-level Functional Flow Diagrams (FFDs).

Output Criteria

The successful completion of the *Functional Analysis/Allocation* effort will allow the start of the *System Synthesis* phase of the Systems Engineering process. The final criterion for completion of the *Functional Analysis/Allocation* effort is the complete problem definition. This is the process where the functions to be performed by the mission are identified and the requirements that define how well the functions must be performed are generated.

There are various formats that the output products of the *Functional Analysis/Allocation* task can take depending on the specific stage of the process and on the specific technique used to develop the functional architecture:

a. Behavior Diagrams - Behavior Diagrams describe behavior that specifies system-level stimulus responses using constructs that specify time sequences, concurrencies, conditions, synchronization points, state information and performance. This notation provides constructs for control flow, data flow, state transition and state machine characteristics of a system.

b. Context Diagrams - Top-level diagram of a Data Flow Diagram that is related to a specific level of system decomposition. This diagram portrays all inputs and outputs of a system but shows no decomposition.

c. Control Flow Diagrams - A diagram that depicts the set of all possible sequences in which operations may be performed by a system or a software program. There are several types of Control Flow Diagrams which include Box diagrams, flowcharts, input-process-output (IPO) charts, state transition diagrams.

d. Data Flow Diagrams - they provide an interconnection of each of the behaviors that the system must perform. All inputs to the behavior designator and all outputs that must be generated are identified along with each of the data stores that each must access. Each of the Data Flow diagrams must be checked to verify consistency with the context Diagram or higher level Data Flow Diagram.

e. Data Dictionaries - Documentation that provides a standard set of definitions of data flows, data elements, files, databases and processes referred to in a Data Flow Diagram set for a specific level of system decomposition. This is an aid to communications across the development organizations.

f. State Transition Diagrams - A diagram that shows the possible states that are possible and the way a system may change from one state to another. The symbols used are circles that indicate system states and line segments that indicate how the change from one state to another can occur.

g. Entity Relationship Diagrams - These diagrams depict a set of entities (functions or architecture elements) and the logical relationship between them. The attributes of the entities can also be shown.

h. Functional Block Diagrams - These block diagrams relate the behaviors that have to be performed by a system to each other, show the inputs and outputs and provide some insight into flow between the system functions.

i. Models: Models are abstractions of relevant characteristics of a system, used as a means to understand, communicate, design, and evaluate (including simulation) a system. They are used before the system is built and while it is being tested or in service. A good model has essential properties in common with the system/situations it represents. The nature of the properties it represents determines the uses for the model. A model may be functional, physical, and/or mathematical

j. Timing Analysis Results - The results of the analysis of the time required to perform concurrent functions that provide inputs to a subsequent function. Provides an assessment of the time that the system will require to accomplish an desired output behavior when concurrency of functions has to be considered.

k. Simulation Results - The output from a model of the system that behaves or operates like the system under interest when provided a set of controlled inputs.

l. Functional and Concurrent Thread Analysis Results - The output of a system or model that demonstrates a specific sequence of functions of interacting system objects. A system's overall behavior is the time response of all its threads to initial conditions, its environment, and control (mode) parameters.

m. IDEF Diagrams - Process control diagrams that show the relationship between functions by sequential input and output flows. Process control enters the top of each represented function and lines entering the bottom show the supporting mechanism needed by the function. IDEF is an acronym for Integrated DEFinition. It is also used with ICAM DEFinition, where ICAM means Integrated Computer-Aided Manufacturing.

These various output products characterize the functional architecture. There is no one preferred output product that will support this analysis. In many cases, several of these products are necessary to understand the functional architecture and the risks that may be inherent in the subsequent synthesis of system architecture. Using more than one of these formats allows for a "check and balance" of the analysis process and will also aid in the communication across the system design team.

9.2 MAJOR STEPS IN THE FUNCTIONAL ANALYSIS/ALLOCATION PROCESS

Even within a single stage in the system life-cycle, the Functional Analysis/Allocation process is iterative. The functional architecture begins at the top level as a set of functions that are defined in the applicable requirements document or specification, each with functional, performance, and limiting requirements allocated to it (in the extreme, top-level case, the only function is the system, and all requirements are allocated to it). Then, as shown in Figure 9-1, the next lower level of the functional architecture is developed and evaluated to determine whether further decomposition is required. If it is, then the process is repeated. If not, then the process is completed and System Synthesis can begin.

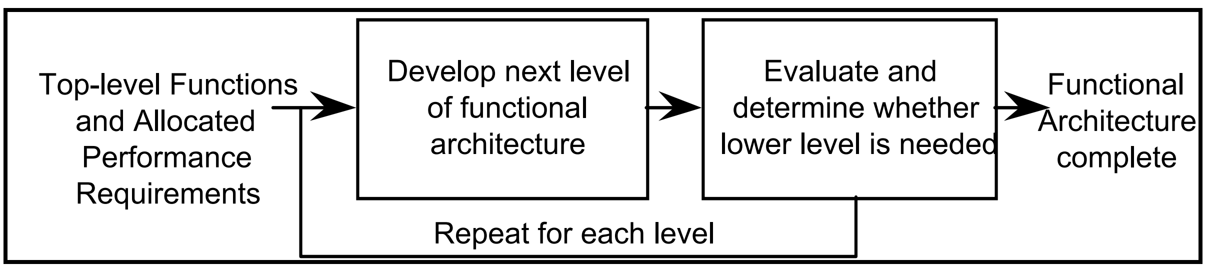


Figure 9-1. Functional Analysis/Allocation Process

The Functional Analysis/Allocation process is iterated through a series of levels until a functional architecture is complete. At each level of the *Functional Analysis/Allocation* process, alternative decompositions and allocations may be considered and evaluated for each function and a single version selected. After all of the functions have been treated, then all the internal and external interfaces to the decomposed subfunctions are established.

These steps are each described briefly in the following paragraphs. While the shaded portion of the diagram in figure 9-2 shows performance requirements being decomposed and allocated at each level of the functional decomposition, it is sometimes necessary to proceed through multiple levels before allocating the performance requirements. Also, sometimes it is necessary to develop alternative candidate functional architectures, and conduct a trade study to determine a preferred one.

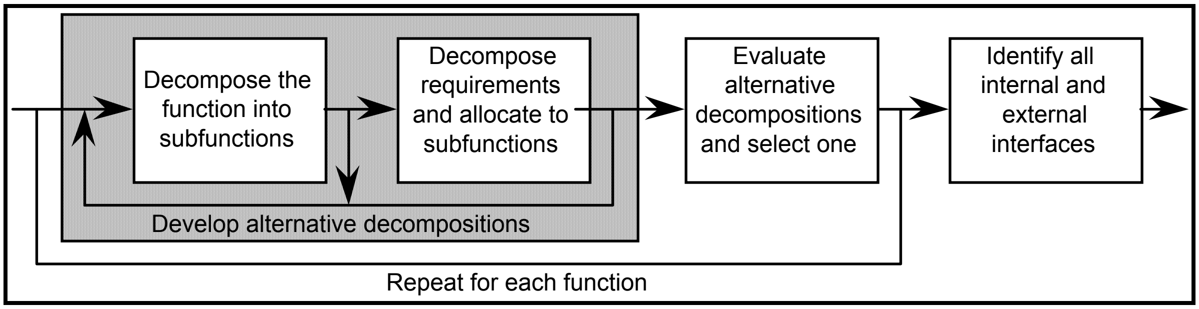


Figure 9-2. Alternative Functional Decomposition Evaluation and Definition

During each iteration of Functional Analysis/Allocation, alternative decompositions are evaluated, and all interfaces are defined.

9.2.1 DECOMPOSE EACH FUNCTION TO LOWER-LEVEL FUNCTIONS: FUNCTIONAL FLOW DIAGRAMS

The work of a function is accomplished by one of the system or segment elements of equipment, software, facilities, or personnel. Functional identification and decomposition can be performed with respect to logical groupings, time ordering, data flow, control flow, state transitions, or some other criterion. The stepwise decomposition of a system can be viewed as a top-down approach to problem solving.

While this discussion talks about functions and subfunctions, it should be clearly understood that all of these meet the basic definition of "function", and the distinction only addresses relationships between levels of the hierarchy. What were subfunctions during the first iteration of the process become functions during the second iteration. It is important to determine the inputs required by each function and the outputs it generates and to ensure consistency in the definitions during decomposition.

Objective

The objective of this process is to develop a hierarchy of Functional Flow Diagrams (FFDs) that meet all the functional requirements of the system. Note, however, that this hierarchy is only a portion of the functional architecture. The architecture is not complete until all of the performance and limiting requirements have been appropriately decomposed and allocated to the elements of the hierarchy.

Recommended Activities

For the initial iteration of *Functional Analysis/Allocation*, the baseline requirements and operational concept have been identified during *Requirements Analysis*. First, determine the top-level system functions. This is accomplished by evaluating the total set of baseline requirements as they map to the system-level design, keeping in mind the desire to have highly cohesive, loosely coupled functions. The result is a set of top-level functions which, when grouped together appropriately, provide the required capabilities of each component in the system-level design. Each of the top-level functions is then further refined to lower-level functions based upon its associated requirements.

Decomposition of the function involves the creation of a network of lower-level "child" functions, each of which receives its allocated portion of the "parent's" functional requirements. In this process, each functional requirement is decomposed into lower-level requirements, and each of these is allocated to a lower-level function (i.e., a subfunction) in the next-level FFD. Functional interfaces fall out of this process.

Develop a description for each function in the hierarchy. This description must include the following:

1. its place in a network (Functional Flow Diagram or IDEF0/1) characterizing its interrelationship with the other functions at its level
2. the set of functional requirements that have been allocated to it and which define what it does
3. its inputs and outputs, both internal and external

This process may use various graphical methods to capture the results of the analysis, including structured analysis, such as Data Flow Diagrams, IDEF0/1 diagrams, and Control Flow Diagrams, or other modern techniques. These are all forms of the Functional Descriptions.

“Stop” Criteria

In undertaking the *Functional Analysis/Allocation* process, it is important to establish criteria for completion of the functional decomposition. The usual criteria are to continue until the functional requirement is clear and realizable in hardware, software, and/or manual operations. In some cases, the engineer will continue the effort beyond what is necessary until funding for the activity has been exhausted. In establishing the stop criteria, recognize that the objective of pushing the decomposition to greater detail is to reduce the program risk. At some point, the incremental risk reduction becomes smaller than the cost in time or money of the effort to further decompose. Each program will be different, so it is impossible to set forth all-purpose stop criteria. The program manager and Systems Engineer who understand their specific program’s risks need to establish their own stop criteria early in the process and ensure that the decomposition efforts are reviewed frequently.

Constraints

Ideally, *Functional Analysis/Allocation* is a pure exercise based on a logical analysis of the requirements. In fact, there are always constraints on the analysis that serve to limit the choice of decompositions. For instance, the project may be required to use COTS hardware or software, or Non-Developmental Items from other programs. In those cases, the functions and subfunctions and their allocated performance requirements had better be consistent with the capabilities of the target products.

Sometimes as the hierarchy develops it becomes apparent that there are similar subfunctions at different points in the hierarchy. This could be true, for instance, of a database-management subfunction, a human-computer interface subfunction, or a communications subfunction. In such cases, it is incumbent upon the Systems Engineer to tailor those subfunctions so that they are very similar, so that they can all be implemented by the same component during the *System Architecture Synthesis*.

9.2.2 ALLOCATE PERFORMANCE AND OTHER LIMITING REQUIREMENTS TO ALL FUNCTIONAL LEVELS

Requirements allocation is the further decomposition of system-level requirements until a level is reached at which a specific hardware item or software routine can fulfill the needed functional/performance requirements. It is the logical extension of the initial functional identification and an integral part of any functional analysis effort.

Objective

Functional requirements have been fully allocated to functions and subfunctions in the previous step. The objective of this step is to have every performance or limiting requirement allocated to a function or subfunction at the next level in the hierarchy of FFDs. Some performance requirements will have been decomposed in order to do this. Additional requirements may have to be derived.

Recommended Activities

In this step, performance, and other limiting requirements are allocated to the functions in the next level FFD. Some straightforward allocation of functional requirements can be made, but the procedure may involve the use of supporting analyses and simulations to allocate system-level requirements. An example of the need for additional analysis is the allocation of availability goals to configuration items. These goals can only be expressed as maintainability and reliability requirements. Allocations and trade studies will be made by these parameters (maintainability and reliability), but only in conjunction with analytical and/or computer simulation to ascertain the impact of a given set of allocations on system availability.

If a requirement cannot be allocated as a single entity, then it must be decomposed and the derived requirements allocated. Often this step requires some anticipation of the results of the *System Architecture Synthesis* because decomposition of response-time or noise-level requirements is equivalent to developing timing or noise budgets. In some cases, it will be necessary to defer decomposition of performance and limiting requirements until multiple stages of functional hierarchy have been developed, or, in a worst case, until the *System Architecture Synthesis*.

DESIGN CONSTRAINT REQUIREMENTS

Insure that all constraints are identified to the designer prior to start of detailed design. This should prevent the need for redesign due to unidentified constraints. Include all Systems Engineering groups, but primarily the Engineering Specialties: Reliability, Maintainability, Producibility, Human Engineering, EMI/EMC, System Safety, Survivability, Support, Security, Life Cycle Cost/Design-to-Cost

Recommended Activities

1. Identify from the SOW all design constraints placed on the program. This particularly includes compliance documents.

2. Identify the groups defining constraints and incorporate them into the Systems Engineering effort.

3. Analyze the appropriate standards and lessons learned to derive requirements to be placed on the hardware and software CI design.

4. Tailor the compliance documents to fit overall program needs.

5. Identify the cost goals allocated to the design.

6. Define system interfaces and identify or resolve any constraints that they impose.

7. Identify any COTS or NDI CIs that must be used, and the constraints that they may impose.

8. Document all derived requirements in specifications and insure that they are flowed down to the CI level.

9. Insure that all related documents (operating procedures, etc.) observe the appropriate constraints.

10. Review the design as it evolves to insure compliance with documented constraints.

DESIGN CONSTRAINTS

Design constraints recognize inherent limitations on the sizing and capabilities of the system, its interfacing systems, and its operational and physical environment. These typically include power, weight, propellant, data throughput rates, memory, and other resources within the vehicle or which it processes. These resources must be properly managed to insure mission success.

Design constraints are of paramount importance in the development of derivative systems. A derivative system is a system that by mandate must retain major components of a prior system. For example, an aircraft may be modified to increase its range while retaining its fuselage or some other major components. The constraints must be firmly established: Which components *must* remain unmodified? What can be added? What can be modified? The key principle to be invoked in the development of derivative systems is that the requirements for the system as a whole must be achieved while conforming to the imposed constraints.

The usual process is for Systems Engineering to establish a Control Board, with the Chief Systems Engineer as the chairman. Meetings are scheduled monthly or more frequently if the situation warrants, and the agenda is prepared by Systems Engineering. Preliminary allocations are made to the subsystems, based on mission requirements, configuration concept, and historical data. A percentage contingency is usually applied to critical parameters, based on historical growth data. This contingency is expected to be "used up" as the design evolves.

A margin may also be withheld by Systems Engineering personnel to accommodate unforeseen problems. The latter is held at the system level. In communication links, typically a 3 dB system margin is maintained throughout the development phase. These allocations are analyzed by engineering personnel to verify their achievability. As the design progresses, the current status of the allocations is reviewed at the control board meetings. Care must be exercised that "margins-on-margins" are not overdone, resulting in too conservative (possibly too expensive) a design. This is the Chief Systems Engineer's responsibility.

When allocations cannot be met by the current design, it is necessary to reallocate, redesign, use some of the margin, or revisit the requirements to determine if they can be reduced. If cost is the primary driver, then the design becomes capability driven, rather than performance driven. These are the decisions that the Control Board must make and implement.

Constraints placed on the system by interfacing systems are surfaced in Interface Working Group meetings organized by Systems Engineering. Operational constraints are established through analyses and simulations developed by Systems Engineering.

ENGINEERING SPECIALTY CONSTRAINTS

Care must be exercised that the myriad of engineering specialty requirements and constraints are incorporated into appropriate specifications. Incorporation of engineering specialties personnel into the Systems Engineering and Integration Team (SEIT) of an Integrated Product and Process Team (IPPD) organization or into all appropriate Product Development Teams (PDTs), are ways of insuring that their requirements are incorporated into specifications.

9.2.3 EVALUATE ALTERNATIVE DECOMPOSITIONS AND SELECT ONE

Not all functional decompositions are of equal merit. It is necessary to consider alternative decompositions at each level, and select the most promising. Because of the reality of system design constraints or target COTS or NDI components, it is often desirable to produce multiple alternative functional architectures that can then be compared in a trade study to pick the one most effective in meeting the objectives.

Objective

Eventually, each subfunction in the lowest levels of the functional architecture is going to be allocated to hardware, software, interfaces, operations, or a database, and then to a specific configuration item. In addition, each of these functions will have to be tested. The objective here is to select those decompositions that lend themselves to straightforward implementation and testing. Also, we may be able to come up with decompositions that allow a single function to be used at several places within the hierarchy, thereby simplifying development.

Recommended Activities

This is a task that requires best engineering judgment. There are various ad hoc figures of merit that can be applied to evaluate alternative decompositions. The degree of interconnectivity among functions is one possible measure. There are several measures for software-intensive systems that can be applied, such as high cohesion and low coupling. The Systems Engineer needs to be aware of opportunities for use of NDI hardware and software. That means that a subfunction that has already been implemented in a compatible form on another system may be preferred to one that has not.

9.2.4 DEFINE/REFINE FUNCTIONAL INTERFACES (INTERNAL AND EXTERNAL)

All of the internal and external interfaces must be completely defined.

Objective

Each function requires inputs in order to operate. The product of a function is an output. The objective of this step is to identify and document where within the FFD each function (or subfunction) will obtain its required inputs and where it will send its outputs. The nature of the flows through each interface must be identified.

Recommended Activities

N^2 diagrams can be used to develop interfaces. These apply to systems interfaces, equipment (hardware) interfaces, or software interfaces. Alternatively, or in addition, Data/Control Flow Diagrams can be used to characterize the flow of information among functions and between functions and the outside world. As the system architecture is decomposed to lower and lower levels, it is important to make sure that the interface definitions keep pace, and that interfaces are not defined that ignore lower-level decompositions.

9.2.5 DEFINE/REFINE/INTEGRATE FUNCTIONAL ARCHITECTURE

It may be necessary to make some final modifications to the functional definitions, FFDs, and interfaces in order to arrive at a viable allocation. The product of this activity is a final FFD hierarchy with each function (or subfunction) at the lowest possible level uniquely described. The functional flow diagrams, interface definitions, and allocation of requirements to functions and subfunctions constitute the functional architecture.

9.3 TOOLS USED TO SUPPORT FUNCTIONAL ANALYSIS/ALLOCATION

Tools that can be used to perform the four steps in *Functional Analysis/Allocation,* are described in more detail in Appendix D, and include:

- Analysis tools
- Modeling tools
- Prototyping tools
- Simulation tools
- Requirements traceability tools

See the INCOSE web page for a current listing of applicable tools.

9.4 METRICS USED IN FUNCTIONAL ANALYSIS/ALLOCATION

This paragraph lists some metrics that can be used to measure the overall process and products of *Functional Analysis/Allocation*. Candidate metrics include the following:

1. Number of trade studies completed as a percent of the number identified
2. Percent of analyses completed
3. Maximum time between raising a system issue and getting it resolved
4. Percent of issues currently unresolved
5. Average time between identifying a risk item and getting it mitigated
6. Remaining number of risk items that are unmitigated
7. Maximum days a risk item has remained unmitigated
8. Depth of the functional hierarchy as a percentage versus the target depth
9. Percent of performance requirements that have been allocated at the lowest level of the functional hierarchy
10. Percent of analysis studies completed (schedule/progress)

9.5 EXAMPLE OF FUNCTIONAL ANALYSIS/ALLOCATION

The stepwise decomposition of a system can be viewed as a top-down approach to problem solving. This top-down approach is illustrated in Figures 9-3, -4, and -5, which show a system being separated into a string of subfunction states and associated events/actions.

Each functional subdivision satisfies an allocated portion of the basic system functions. Collectively, these functions constitute a complete system at each level. When these functions are separated, as they actually may be in a physical sense, then the required interface connections are exposed and the boundaries of where one function begins and another one ends becomes apparent or at least is exposed and must be defined.

As the functions are decomposed to the next lower level (subfunctions), the number of subfunctions greatly increases, each with its own interfaces. This process continues until the lowest level is reached at which discrete tasks (such as Command Payload Transmitter ON) can be defined and satisfied. Note that traceability is maintained throughout by a decimal numbering system.

One of the most important advantages of top-down development is that the most difficult design area can be attacked first throughout its total hierarchy at the start of the development to reduce risk.

The entire flight mission of the STS and its Payload can be defined in a top-level Functional Flow Diagram (FFD), as shown in Figure 9-3. Note that the numbers in this figure correspond to the element numbers in Figure 9-4 and 9-5. Each block in the first-level diagram can then be expanded to a series of functions, as shown in the second-level diagram for (Perform Mission Operations). Note that the diagram shows both input (Transfer Shuttle to Ops Orbit) and output (Transfer To STS Orbit), thus initiating the interface identification and control process. As the block diagrams are separated, an Event is identified (above the line) and a corresponding next Action identified below the line, which helps to define the beginning and ending of a Function. Each block in the second-level diagram can be progressively developed into a series of functions, as shown in the third-level diagram on Figure 9-5. These diagrams are used to develop requirements and to identify profitable trade studies. The FFDs also incorporate alternate and contingency operations, which improve the probability of mission success. The FFDs provide an understanding of total operation of the system, serve as a basis for development of operational and contingency procedures, and pinpoint areas where changes in operational procedures could simplify the overall system operation. In certain cases, alternate FFDs may be used to represent various means of satisfying a particular function until data are acquired, which permits selection among the alternatives.

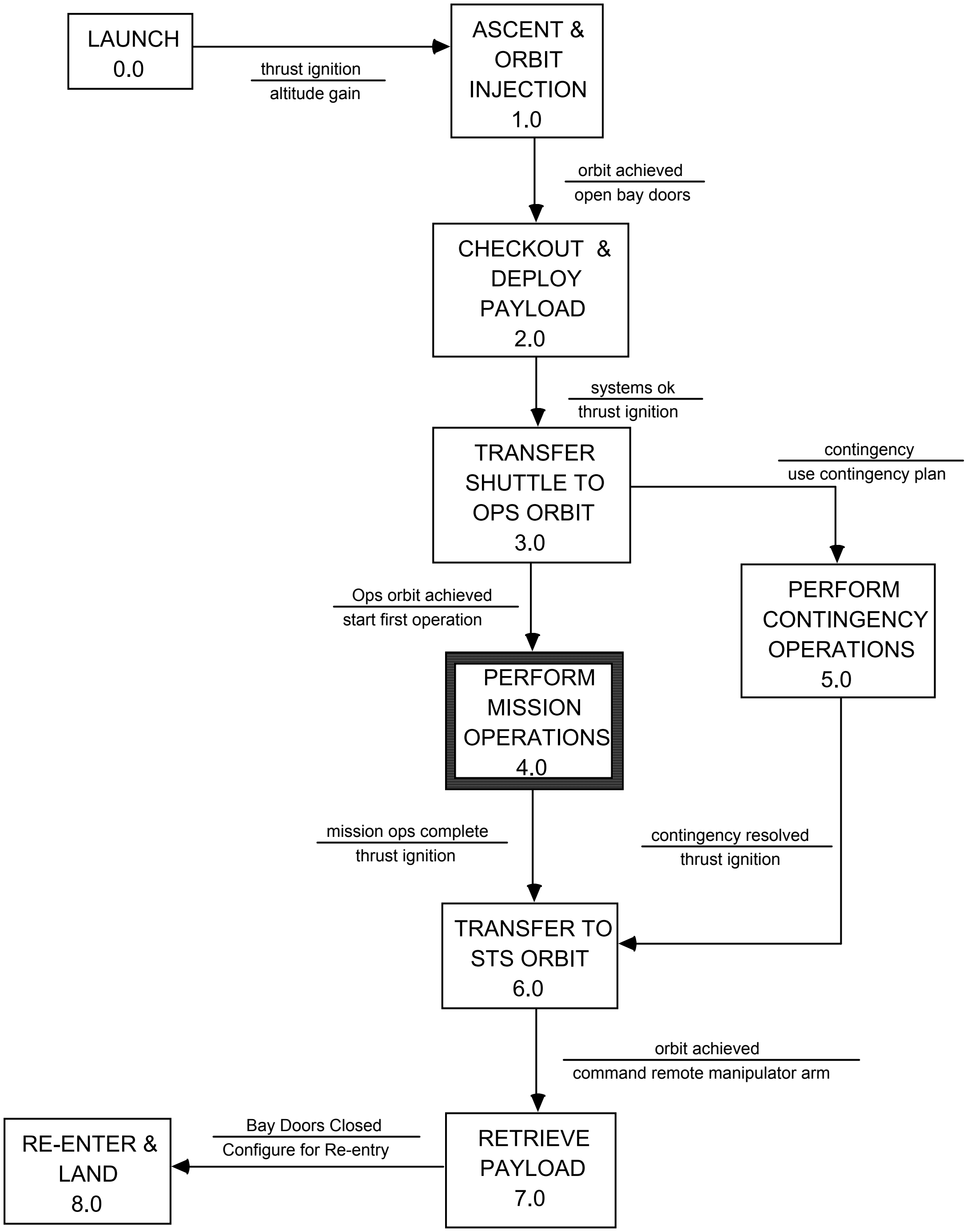


Figure 9-3. Functional Decomposition - Top Level - STS Flight Mission

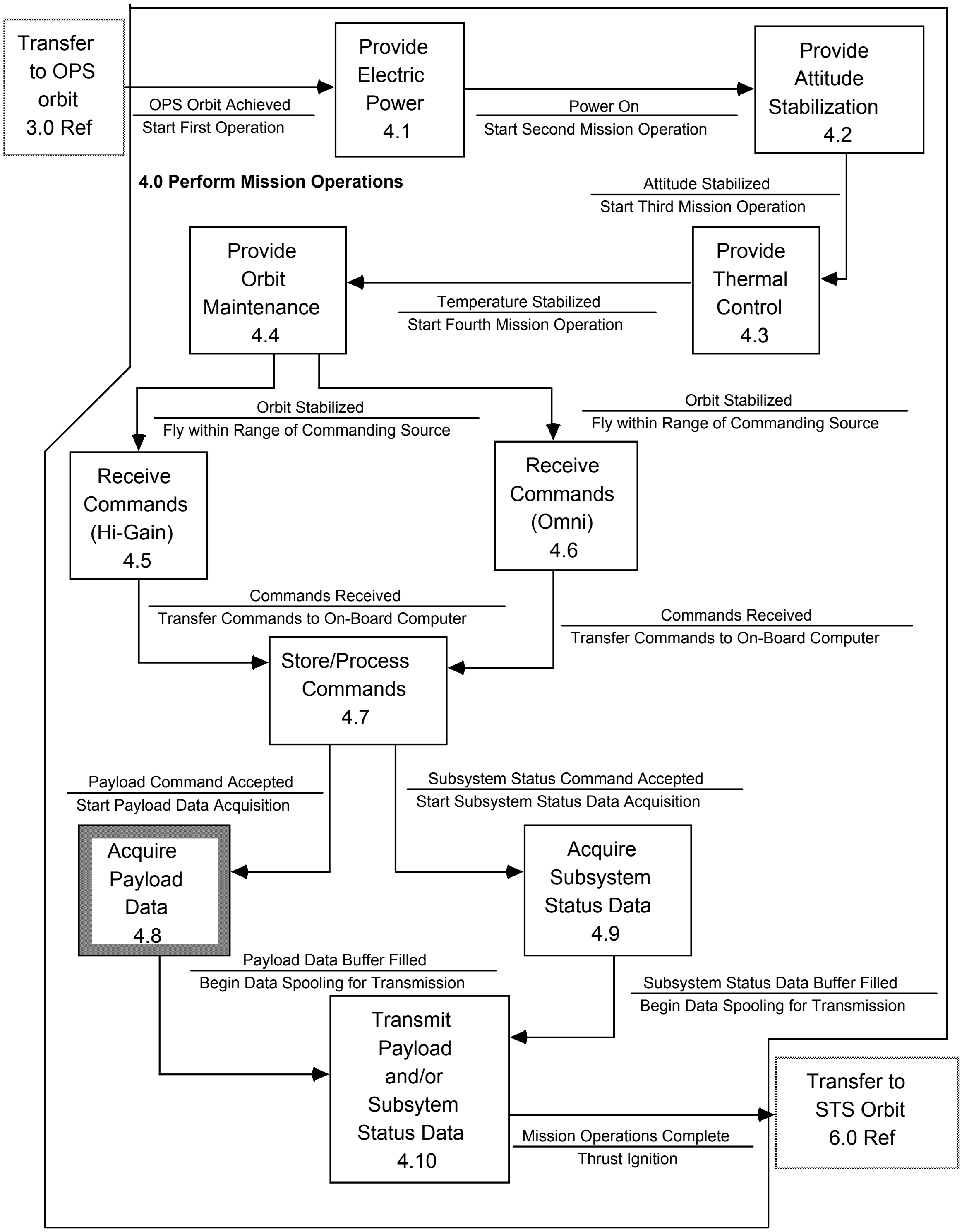


Figure 9-4. Functional Decomposition - Second Level - 4.0 Perform Mission Operations

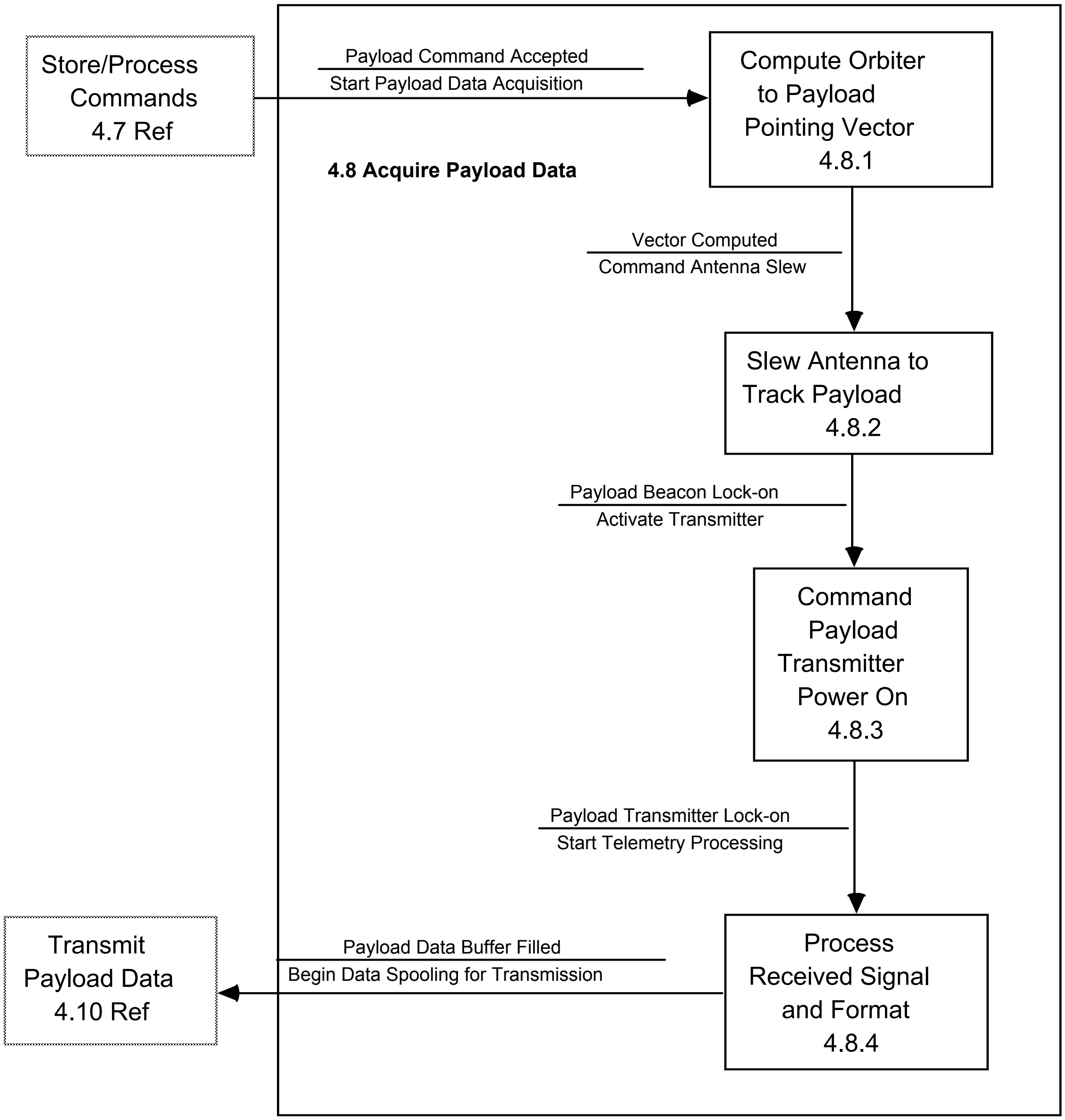


Figure 9-5. Functional Decomposition - Third Level - 4.8 Acquire Payload Data

An N^2 chart example is shown in Figure 9-6. An example of a high-level time line for a satellite ground station contact is shown in Figure 9-7.

Related system timeline requirements are that requests for data are permitted up to 35 minutes prior to the start of data take and that the data must be processed and delivered to users within 30 minutes of acquisition.

In addition to defining detailed subsystem/component or software requirements, the time line analysis can also be used to develop trade studies. For example, should the spacecraft location be determined by the ground network or by on-board computation using navigation satellite inputs?

Generic Space-craft	-Payload Data -Spacecraft Status and Ranging	-Payload Data -Spacecraft Status and Ranging		
- S/C Commands	Remote Ground Stations		-Payload Data -Spacecraft Status and Ranging	
- S/C Commands		Tracking and Data Relay Satellites	-Payload Data -Spacecraft Status and Ranging	
	- Schedules - Directives - S/C Commands - Pred. S/C Positions	- TDRS Vectors - TDRS Commands - S/C Commands	Data Relay and Control	- Payload Data - Data Products -Schedules
			- Data Requests - Status - Surface Truth	Primary Users

Figure 9-6. N^2 Chart Example

The following examples illustrate the allocation of requirements:

Pointing Error - Allowable pointing error is a critical issue on all missile and spacecraft programs. Typical errors range from several tenths of a degree to a few arc seconds for astronomical observatory spacecraft. In defining the error budget, it is necessary to first establish those hardware and software characteristics that contribute to the error, otherwise known as error sources. Individual values for errors would be obtained from specifications for candidate components, experience from similar projects, or extrapolation of experimental data. Where data are totally lacking, values for errors could be obtained through analysis. Typically, a minus-two-sigma (0.9 probability) value is stated in the specification. This assumes normal distribution with a 95 percent confidence in the error being less than stated. For the above example, the error sources are root-sum squared to arrive at a total, since they are random and uncorrelated. The allocated pointing requirements would be placed in subsystem and component specifications, as appropriate.

Electrical Power - Electrical power is a support requirement determined by summing the individual component loads. It is usually defined by average load, peak load, and a profile of power demands over the total mission sequence. In developing this profile, all electrical items in the design must be identified and a mission operational scenario developed to define equipment operation and duration. Total power requirements in each mode are established and a power profile is developed. The peak and average power requirements are then defined to size the power subsystem. Because some items may be based on only a conceptual design, and because power needs tend to increase, a power control plan is often used that incorporate margins early in the design process to allow for contingencies that may arise. The plan also provides for periodic review of requirements.

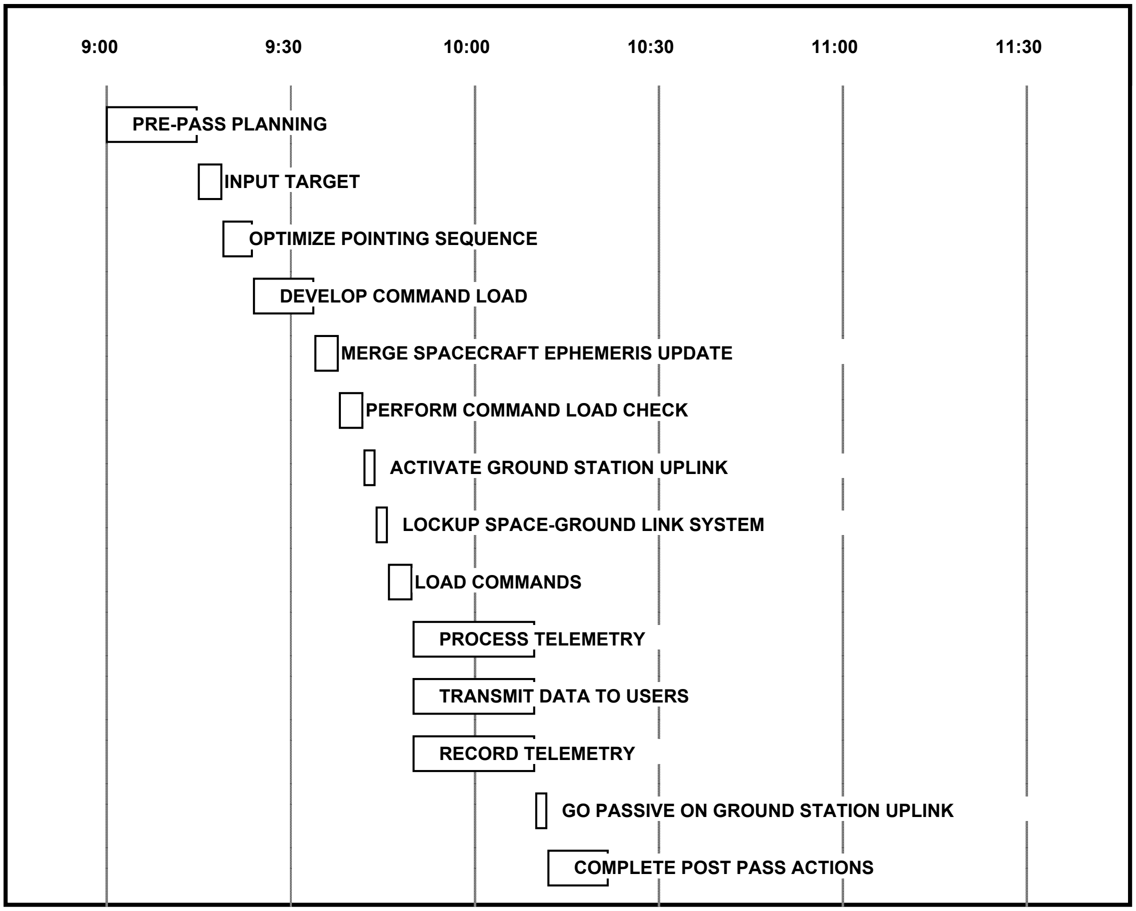


Figure 9-7. Time Line Chart

10 SYSTEM ARCHITECTURE SYNTHESIS

The overall objective of System Definition is to create a System Architecture (defined as the selection of the types of system elements, their characteristics, and their arrangement) that meets the following criteria:

1. Satisfies the requirements and external interfaces.
2. Implements the functional architecture.
3. Is acceptably close to the true optimum within the constraints of time, budget, available knowledge and skills, and other resources.
4. Is consistent with the technical maturity and acceptable risks of available elements.
5. Is extensible, i.e., accommodates system growth and introduction of new technologies.
6. Provides the base of information which will allow subsequent steps of system definition and implementation to proceed. The system architecture and operational concept, element descriptions, and internal interfaces, are all adequately defined.
7. Is robust, i.e., allows subsequent, more detailed system definition to proceed with minimum backtracking as additional information is uncovered.

System Architecture Synthesis (Definition) is part of the overall process of system design, which includes Requirements Analysis and Functional Analysis. This process can be viewed as a search through a highly non-linear design space of very large dimension. This search process is highly iterative. An initial set of functions is defined to carry out the system's mission. Requirements quantify how well the functions must be performed, and impose constraints. An architecture is chosen to implement the functions and satisfy the requirements and constraints. The realities of a practical architecture may reveal need for additional functional and performance requirements, corresponding to architecture features necessary for wholeness of the design, but not invoked by the original set of functions. The initial functional and performance requirements may prove infeasible or too costly with any realizable architecture. Consequently, the search process involves a mutual adjustment of functions, requirements, and architecture until a compatible set has been discovered.

For this process to converge to an adequate approximation of the true optimum in reasonable time requires considerable skill and judgment in balancing depth and breadth within the search strategy. Breadth is required since the design space may be very lumpy, i.e., contain several localized regions of good system design imbedded in a matrix of poor or infeasible designs. Thus, the region containing the optimum may be isolated. A search that begins in some region of moderately good system designs, using a conservative search strategy, may remain stuck there and miss the true optimum. To avoid this problem, a broad search strategy that explores the whole design space with some reasonable sampling density is needed. At the same time, depth of analysis of each option must be sufficient to ensure robustness of the final choice (no unknown show-stoppers lurking anywhere). Thoroughness in searching the breadth and probing the depth must be limited so as not to consume excessive time and budget in analysis of all the candidates. Literature on the design of such search strategy is extensive and should be consulted.

The process of System Architecture Synthesis flows as shown in Figure 10.1. The limitations of text force a sequential description of these functions, although in practice, the process usually proceeds in a highly-interactive, parallel manner with considerable iteration. In addition, for clarity and completeness, a rather formalized description is provided. A large project, with numerous participating organizations at separate locations, may require a high level of formality and discipline for coordination and concurrence, whereas a small unified team can function in a much more informal manner.

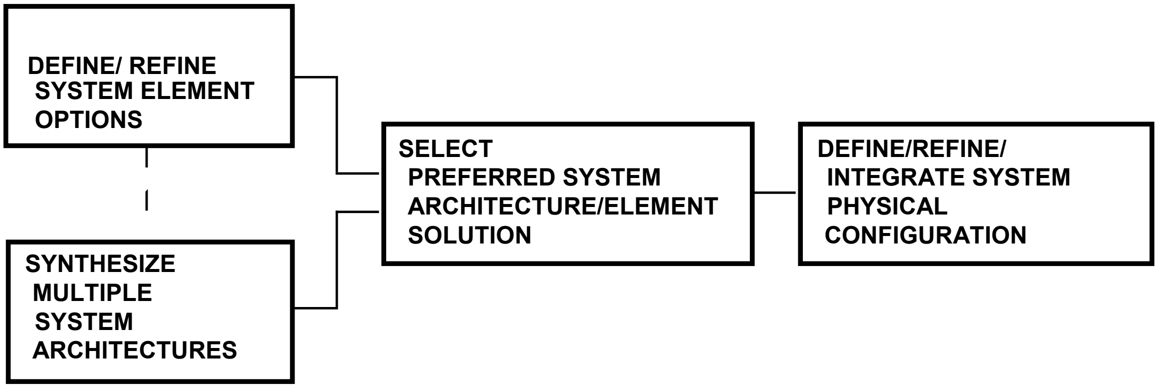


Figure 10-1. System Architecture Synthesis Process Flow

Regardless of the style appropriate for a particular project, all members of the system definition team must understand and accept the entire process as described here to ensure success. This process description is a generalized baseline, to be tailored to the needs of a specific project without risk that important steps will be overlooked or improperly executed.

The process of System Architecture Synthesis is essentially a tradeoff, performed at a grand scale, leading to a selected system architecture baseline as the final output. The objective is to select the best from among a set of System Architecture candidates, which have been constructed in a manner that assures (with reasonable certainty) that one of the candidates is acceptably close to the true (usually unknowable and unattainable) optimum.

The process works as follows. A bottom-up approach may be selected, starting with a menu of options for each element of the system (see Section 10.1), from which a set of system architecture options is created (see Section 10.2). Alternatively, in a top-down approach, a set of system architecture options is created, each providing a framework into which element options may be inserted. These two approaches usually work together. It is difficult to conceive of system element options in a vacuum without some system architecture concept in mind. Likewise, abstract system architecture, without some concrete ideas about the elements it will be made of, is difficult to envision. In general, the initial notions about system architecture constrain the range of element options but also suggest paths for expanding the list. Similarly, a set of system elements both constrains the possible system architectures and suggests new architectures resulting from novel combinations of elements. The creation of system architecture options can follow existing practice or be highly innovative, and can involve both deductive and creative thinking.

The starting point for system architecture synthesis can vary considerably. One extreme is a completely fresh start with no existing system and a minimal definition of the system concept. The other extreme is a minor modification to an existing, large operational system. It also depends on the amount of work already done by the customer or carried over from earlier studies. The process description here assumes a fresh start, since that viewpoint provides the clearest and most complete overall explanation. The process can be tailored in obvious ways to fit other cases.

The process continues with selection of the preferred system architecture from the set of candidates (Section 10.3). This is carried out using the trade-off methodologies described in Section 11, Trade Studies. It is important to keep the selection criteria simple and few in number, relating only to top-level considerations, and with resolution no finer than necessary to distinguish between the options being compared. More complex approaches waste time and resources. Pursue opportunities to create hybrids among options, combining their best features. Where possible, obtain quantitative selection criteria and associated data for each alternative. As the selection narrows, consequences not anticipated in the criteria, both beneficial and adverse, should be considered.

Finally, in preparation for subsequent more detailed steps of system definition, the definitions of the elements and internal interfaces of the selected system architecture are completed, and they are integrated into a consistent and fully-defined system description, as described in Section 10.4.

10.1 DEFINE/REFINE SYSTEM ELEMENT ALTERNATIVES

The system elements include the hardware, software, information, procedures, and people that make up the system.

The purpose of this activity is to identify element options, one level down from the top of the system hierarchy, which constitute a set of building blocks from which System Architecture options will be assembled, as described in Section 10.2.

The range of element options in the set may be defined by either or both of the following:

a. Expand a range of various types of elements, representing diverse approaches to implementing the system functions (e.g., if the function is communications, the types of elements might include microwave relay, satellite link, or fiber optics)

b. Variations of design parameters within any given element type (e.g., number and thrust level of thrust chambers to produce a required total thrust for a launch vehicle).

The objective is to create a set of element options that satisfy the following criteria:

- With reasonable certainty, spans the region of design space which contains the optimum
- Supports analysis which efficiently closes on the optimum
- Contains all relevant design features necessary to provide a firm baseline for the subsequent round of system definition at the next level of detail

A. Stakeholder Participation

This function is lead by Systems Engineering with major support from Hardware and Software Design and Operations. Other engineering support disciplines participate as appropriate to support the definition of system element options, in particular to anticipate issues which will become important later in the system definition process. Within the framework of concurrent engineering, all disciplines are kept informed as the process unfolds. Specific comments may be solicited, and any discipline is free to contribute at any time. The objective is to include all disciplines in identifying design drivers, defining selection criteria, and detecting show-stoppers. In addition, participation in these earlier stages of system definition lays the groundwork for knowledgeable contributions later in the process.

B. Recommended Activities

The following steps, although listed in sequence, are highly interactive and usually evolve in a parallel and iterative manner.

1. Create a list of the elements that will make up the system. These may be derived from the functional requirements or from a decomposition of the existing system architecture. The initial version of this list may be considered preliminary, and will mature as the process is iterated.

2. Identify a set of option descriptors for each element in the list. These are the definitive attributes (design features and parameters) that distinguish one element option from another. The descriptors are

the minimal set of significant element characteristics which allows a unique identification for every element choice in the design space. They should include the design drivers, derived from the Requirements Analysis and Life Cycle Operations Concept, which are the object of subsequent optimization and design analysis. They should be orthogonal in the sense that the range of values that can be adopted by any one descriptor is not constrained by the values of the others. These descriptors become the dimensions of the design space in which the sets of element and System Architecture options exist. Also, if properly chosen, the descriptors will relate directly to the parameters in the models used to evaluate system architecture options in the next activity. Example: for a structural element, the descriptors might include material (aluminum or composite) and structural arrangement (monocoque or truss). In this case, (two descriptors, each having two possible values) four different element choices are defined.

3. Define the envelope of design space (range of design features and parameter values) which is to be scanned. In the example just above, the design space for the structural element is limited to two choices of material and two choices of structural arrangement.

4. Develop a process to generate a range of element options, providing both diversity of element types and range of design parameters within a given type. Demonstrate that the range of element options created is both exhaustive and lean:

   - Exhaustive – no good options have been left out and the optimum is somewhere within the envelope of options under consideration (the definition of optimum includes satisfaction of all requirements and factors such as acceptable design maturity, compatibility with the development schedule, minimum cost, acceptable risk, etc.)
   - Lean – the number of options to be analyzed is small enough to support efficient selection and closure on the optimum

   Borrow from similar existing systems or create new element options through application of the appropriate structured creativity methods (e.g., brainstorming, morphological analysis, synectics, etc. See Adams, and other references on structured creativity.). Any element previously defined or inferred by the Requirements Analysis and Life Cycle Operations Concept, or Functional Analysis must be included.

5. Generate a set of element options which populates the design space envelope. In general, the options selected should satisfy all requirements, but it is useful to include some which may challenge the requirements in ways leading to a better system concept. This includes relaxing requirements of marginal utility, which are costly to implement, or extending requirements where added capability can be purchased cheaply. Include a range of technical maturity (well proven old standard items to unproven and innovative) to allow tradeoffs among cost, performance, development time, and risk. The set of element options may be expressed as a list of discrete choices or as a recipe for generating any possible option by selecting parameter values. If a list is used, it may be complete (explicitly listing every possible option) or it may represent a sampling scattered throughout the design space.

   Find the proper balance between existing practice and fresh innovation. On the one hand, slavishly following past designs can lock in a suboptimum solution. On the other hand, pursuing radical innovations can waste time and resources. The opportunity to challenge existing practice should be provided, but limited so the process can move on quickly. In planning this activity, analyze the potential payoff of exploring innovations versus the cost of doing so.

6. Develop the attendant data describing each element option and its interfaces with other elements, as needed to support the selection process and subsequent system definition activity. This data should include estimates for cost, performance, development time, and risk descriptions for each option.

Input

1. Requirements and Life Cycle Operational Concept, Functional Architecture
2. Examples of existing systems or elements which perform similar functions.

Output

1. Set of descriptors that define the dimensions of the design space.
2. Definition of the envelope of the design space to be scanned.
3. Set of element options, each characterized by a description of its salient features, parameter values, and interactions with other elements.
4. Documentation of the rationale which justifies the selection of the descriptors, the design space envelope, and the menu of element options, demonstrating that the basis has been established for efficient selection of the optimum architecture, i.e., assuring that the options selected will meet the requirements, that the optimum is somewhere within the range of options to be analyzed, and that it can be found quickly and with reasonable certainty.

C. End Result

The result of performing this function is a set of element options with descriptive and supporting documentation that provides:

a. For each option, a description of its salient features, parameter values, and interactions with other elements as necessary to characterize it for analysis and as a potential baseline component. This can take the form of diagrams, schematics, concept drawings, tabular data, and narrative.

b. Identification of design drivers (a limited set of top-level parameters which dominate definition of the design), definition of selection criteria related to elements which will be used in the evaluation process and detection of issues which contain possible show-stoppers

c. For the set of options as a whole, demonstration, with reasonable certainty, that the set spans the region of design space which contains the optimum, that the selected set will support efficient selection and closure on the optimum, and that the descriptive data (features and parameters) are adequate to support subsequent work.

Completion Criteria

1. Descriptors of element and system options, selected to define the design space envelope, which will be searched, are necessary, sufficient, and orthogonal

2. Demonstration that the design space envelope to be searched includes the optimum, with reasonable certainty, without being excessively large

3. Set of elements is a reasonable sampling of the design space envelope, and exhausts the full range of each descriptor

4. All of the options are capable of meeting the requirements, or represent desirable changes to the requirements

5. Reasonable opportunities for innovation have been exercised to the satisfaction of all concerned parties

6. The options span a reasonable range of technical maturity, allowing tradeoffs among cost, risk, and performance

7. Each element option is adequately defined to support the development of System Architecture options, the selection of the baseline, and subsequent work based on the selected baseline

D. Methods/Techniques

Some useful methods include brainstorming, morphological analysis, synectics, (see,Adams and other references on structured creativity), literature search, surveys, inventory of existing concepts, and vendor inquiries.

Metrics

1. Technical performance, schedule spans, costs, and risk estimates for each alternative
2. Above Completion Criteria adequately satisfied
3. Cost and schedule variance for the completion of this function

Tools

Quality Functional Deployment (QFD), see Appendix A, provides a framework to organize the data and test the completeness of the analysis.

References

Adams, James L, (1990). *Conceptual Blockbusting*, 3.*ed.* San Francisco Book Company, Inc.

10.2 SYNTHESIZE MULTIPLE SYSTEM ARCHITECTURES

A System Architecture consists of a selection of the types of system elements, their characteristics, and their arrangement. This process uses the set of element options created by the process described above, and the design space of possible System Architecture arrangements of those elements.

The objective is to provide a set of candidate System Architecture options from which the final optimized and robust System Architecture will be selected or will evolve in an efficient manner.

A. Participation

This activity is lead by Systems Engineering with major support from Hardware and Software Design and Operations. Other engineering support disciplines participate as appropriate to support the definition of system element characteristics and creation of arrangement options, and in particular, to anticipate issues which will become important later in the system definition process.

B. Recommended Activities

1. Assemble candidate System Architectures.

   a. Examine the System Architecture of existing systems that perform similar functions and adopt, in existing or modified form, any which appear suitable.

   b. Create and apply a search methodology to generate System Architecture options by combining elements from the set of element options. Utilize the products of Functional Analysis. For each top-level system function, identify a range of means by which it is implemented (choice of implementing element type or design). Build a set of integrated system concepts which incorporate all the element choices. The methodology should rule out absurd or obviously non-optimal combinations of elements, and seek particularly appealing new combinations of the elements. Apply structured creativity methods as appropriate.

2. Verify that the resulting System Architecture options meet the following criteria:

   a. Perform all the functions of the system as
   b. Capable of meeting requirements
   c. Resource usage is within acceptable limits
   d. Elements are compatible
   e. Interfaces are satisfied

   If not, identify where in the process the shortcoming is introduced and make the necessary correction.

3. Screen the set of System Architecture options generated so far, retaining only a reasonable number of the best. Modify the options as necessary to distribute them with reasonable separation throughout the most promising region of the design space. If some promising regions of design space are poorly represented, create more options to fill the void.

4. Use engineering judgment and formal analysis to ensure that each option is a viable contender, i.e., is feasible, is capable of acceptable performance against the requirements and functions, exemplifies a degree of design elegance, and is a good starting point for subsequent optimization.

5. At this point in the overall system definition process, a review of the Requirements Analysis results may be prudent. The work done in creating element options and synthesizing architectures may have exposed the need for the definition of additional requirements and functions, or may have raised questions regarding the suitability of the existing requirements and functions. It is at this point that the requirements and design definition loops of the Systems Engineering process engine are closed.

Input

1. Menu of elements and combinations.
2. Examples of existing systems which perform similar functions.
3. System Requirements and Functional Architecture.

Output

1. Set of System Architecture options.
2. Documentation that demonstrates, with reasonable certainty, that the set of options is adequate to support successful and expeditious completion of subsequent activities, as defined by the following criteria:
   - contains the optimum
   - supports analysis which efficiently closes on the optimum
   - is an adequate description of the system to enable subsequent system definition activity

C. End Result

The result is a set of System Architecture options that spans the region of design space containing the optimum, with sufficient descriptive and supporting documentation that provides:

- For each System Architecture option, identification of the elements making up that option, their arrangement, the interactions among the elements, and a description of the salient features and parameter values, as necessary, to characterize the option for analysis and as a potential System Architecture baseline. This can take the form of diagrams, schematics, concept drawings, tabular data, and narrative.

- For the set of candidate System Architecture options as a whole, demonstration with reasonable certainty that the set spans the region of design space which contains the optimum, that the selected set will support efficient selection and closure on the optimum, and that the descriptive data (features and parameters) are adequate to support subsequent work.

Completion Criteria

1. Appropriate number of options: large enough to represent a reasonable sampling of the design space envelope, small enough to analyze efficiently (three to five).

2. All of the options are capable of meeting the requirements, resource allocations, and interfaces, or represent desirable changes to the requirements.

3. Reasonable opportunities for innovation have been exercised to the satisfaction of all concerned parties.

4. The options span a reasonable range of technical maturity, allowing tradeoffs among cost, risk, and performance.

5. Each option is adequately defined to support the selection of the baseline and subsequent work based on the selected baseline.

D. Methods/Techniques

Some useful methods include brainstorming, morphological analysis, synectics, literature search, surveys, inventory of existing concepts, and vendor inquiries.

Metrics

1. Completion Criteria above adequately satisfied.
2. Technical performance, schedule spans, cost, and risk estimates for each alternative.
3. Cost and schedule variance to complete this function.

Tools

Check the INCOSE website for current references of applicable tools. Quality Functional Deployment (QFD), described in Appendix A, provides a framework to organize the data and test the completeness of the analysis. Other techniques (many of them described in appendix D) include:

System Hierarchy (functional decomposition)
Functional Flow Diagram
System Schematic
N2 Chart
Layout Sketches
Operational Scenario
Decision Trees (Analytic Hierarchy Process models, e.g., Expert Choice)

10.3 SELECT PREFERRED SYSTEM ARCHITECTURE/ELEMENT SOLUTION

The objective of this process step is to select or evolve the preferred System Architecture from the set of System Architecture options developed in the previous processes steps. The selected baseline System Architecture should be acceptably close to the theoretical optimum in meeting requirements, with acceptable risk, within available resources, and robust, i.e., allows subsequent, more detailed system definition to proceed with minimum backtracking as additional information is uncovered.

A. Participation

This function is conducted by Systems Engineering with support from specialists as necessary to support the definition of selection criteria, and the modeling and analysis used to make the selection.

B. Recommended Activities

The selection of the preferred System Architecture is essentially a tradeoff among the options, using the tradeoff process with modeling. It includes the possibility of combining the best features of several options, and modifying top contenders to further improve their desirability.

1. Define selection criteria and their method of application. The selection criteria are the quantifiable consequences of system implementation and operation. They are derived from the requirements, operational concept, and functions, and from programmatic considerations such as available resources (financial and otherwise), acceptable risk, and political considerations. These selection criteria include:

   a. Measures of the system's ability to fulfill its mission as defined by the requirements
   b. Ability to operate within resource constraints
   c. Accommodation of interfaces
   d. Costs, economic and otherwise, of implementing and operating the system over its entire life cycle
   e. Side effects, both positive and adverse, associated with particular architecture options
   f. Measures of risk
   g. Measures of quality factors
   h. Measures of subjective factors which make the system more or less acceptable to customer, users, or clients, e. g., aesthetic characteristics

   In the interest of efficient analysis, strive to identify the minimal set of criteria that will do the job. Include only the most significant ones, those that are sufficient to distinguish the optimum from the other contenders, and no more.

   The set of criteria usually contains several different types, which are not directly comparable. The application of the criteria involves converting each to a common scale which establishes equivalence according to its relative importance to the final outcome, as described in step 3 below.

2. Create models which map each option's characteristics onto measures of success against the criteria. The models should be as objective and analytical as possible. However, the detail and precision of the models need be sufficient only to clearly distinguish between the options, i.e., the models are used only to produce a clear ranking of the options and not as a design tool. Excessively detailed or accurate models are a waste of resources at this stage. Include capability to assess resource usage and interfaces.

3. Use the Trade Studies methods to compare and rank the options.
   Frequently a simple weighted scoring approach, with subjective evaluation of options against the criteria, will be adequate. With this method, the criteria are of two types: go/no-go criteria, which

must be met, and criteria used to evaluate the relative desirability of each option on a proportional scale.

The go/no-go criteria are applied first as an initial screening. Any option that fails any of these criteria is ruled out of further consideration. Then each option receives a proportional score against the remaining criteria, representing its position between minimally acceptable and perfection. These criteria may be weighted according to their relative importance. The total weighted score for each option represents its success in satisfying the composite set of criteria.

If one option is a clear standout, it becomes the selected baseline. If several options are close to each other, further analysis involving adjustment of weights and scores, and introduction of new criteria, is done to increase the spread in the ranking.

Use formal techniques such as Analytical Hierarchy Process or Multi-Attribute Utility Analysis if the criteria are numerous, difficult or controversial to evaluate.

4. Modify options or combine the best features of several options to correct shortcomings and advance the capability of the leading contenders. Also, look for adverse consequences and potentially unacceptable risks associated with the top contenders. Either correct such conditions or eliminate options that cannot be corrected.

5. Perform sensitivity analysis to test the robustness of the final selection. Examine the effects of variation in the definitions and application of the criteria, the methods of analyzing and evaluating the options, and any assumptions inherent in the analysis. Look for plausible scenarios that could result in a different selection. If two or more of the options are closely ranked or the ranking can be changed by plausible means, then look for ways to arrive at a clear decision by strengthening the options or improving the selection method, perhaps by expanding the set of criteria.

6. Document the process, providing a clear description of how each step is implemented, justifying all choices made, and stating all assumptions.

A situation that occurs frequently and requires resolution is to find strong interdependencies among the major elements of the system architecture, which prevent independent selection of element type or optimization of element characteristics. This can occur because certain global design drivers are shared by several of the elements, or the design optimization of particular elements depends on the choice of design driver values for other elements. In such a case, Systems Engineering must work out a strategy for closure on a design definition, and supervise the design teams responsible for the individual elements (see Figure 10.2).

The first step in such a situation is to sort out the interdependencies, and schedule the work accordingly. The interdependence is a two-way relationship. A particular element may be either (a) a major design driver for another element, or (b) dependent on another element for its design definition. The elements may be categorized according the following scheme, which guides the strategy for solving the interdependencies.

	NOT DEPENDENT ON OTHER ELEMENTS	DEPENDENT ON OTHER ELEMENTS
DRIVES OTHER ELEMENTS	MAJOR DESIGN DRIVER DO THESE FIRST	INTERDEPENDENT, REQUIRES ITERATIVE PROCESS
DOES NOT DRIVE OTHER ELEMENTS	INDEPENDENT OF OTHER ELEMENTS, DO ANY TIME	DEPENDENT ON OTHER ELEMENTS, DO LAST

Figure 10-2. Element Design Interdependencies

The next step is to plot out the design interdependency relationships and identify the parameters which carry the interdependencies. Then schedule the sequence of work. Clearly the elements which drive other elements and are not dependent on others should be done first.

This allows work to proceed on the elements which are dependent on others but do not drive any others. The independent elements, if any, can be done at any time.

The interdependent elements, which are both drivers and dependent on other elements, require a process which seeks a group solution. If models are available which allow an analytic or numerical solution, of course this path should be followed. Usually this is not the case, and an iterative process is necessary. This process begins with selection of a set of trial values for all the parameters involved in the interdependency. Fortunately, design judgment and intuition come to the rescue here.

A good approach is to assemble a team of about half a dozen members who are knowledgeable about the design of the elements involved, as well as have a sense of proportion about the whole system. Through discussion, supported by informal analysis as appropriate, the team selects an initial set of values for the interdependency parameters. Each element design team then optimizes their particular element, using the trial values for the other elements, and reports back new parameter values. The process is then repeated with the new set of values. In most cases, closure will occur quickly, because the intuition of experienced designers is remarkably good. If closure is not reached quickly, there is no universal cure. Analyze the situation to discover the cause of the difficulty and adjust the strategy accordingly. Ultimately the resolution may require higher-level engineering judgment from the system-level technical management.

Input

1. Requirements and Operational Concept, Functional Architecture, and System Architecture options from Section 10.2.
2. Additional information: programmatic, political, economic, etc., needed to define the criteria.
3. Technical information needed to create models and enable evaluation of options.

Output

1. The selected System Architecture baseline
2. Documentation of the selection process to:

   - Justify the selection
   - Enable its review
   - Support subsequent system development, modification and growth throughout its life cycle

C. End Result

The result is a System Architecture baseline, with sufficient descriptive and supporting documentation that provides:

- Identification of the elements (type and principle design characteristics), their arrangement, the interactions among the elements, and a description of the system's salient features and parameter values, as necessary to characterize the System Architecture baseline. This can take the form of diagrams, schematics, concept drawings, operational and life cycle scenarios, tabular data, and narrative.

- Demonstration, within reasonable certainty, that the selected System Architecture baseline is adequately close to the theoretical optimum, that it is robust, and that the descriptive data (features and parameters) are adequate to support subsequent work.

Completion Criteria

1. Concurrence of all responsible parties in the selection process and final result

2. Completion of all supporting documentation

Metrics

1. Many of the selection criteria mentioned in Step 1 can be used as metrics.
2. Completeness of the documentation
3. Schedule and cost variance to perform the function.

D. Methods/Techniques

The general tradeoff methods as described in Section 11 are used. In this case, to economize on the effort expended, the depth of detail and fidelity of modeling is limited to that necessary to clearly separate the options.

Tools

1. Weighted scoring spreadsheet
2. Software for Multi-Attribute Utility Analysis or Analytical Hierarchy Process
3. Models for converting option parameters to scores against criteria

Example

A tradeoff was performed to determine the best system architecture for a Maintenance Workstation to be used aboard Space Station Freedom. The functions of this workstation are to provide a fixed work surface with the capability to enclose the work volume for contamination control while allowing a crew member to perform the maintenance task, and to provide support services (power, cooling, etc.) to the item which is under maintenance. Stringent size and weight constraints apply. The top-level elements of the system are (1) the structural frame, (2) the contamination control system consisting of heat exchanger, filters, charcoal absorber, blower, ducting, and associated controls, and (3) the work volume including work surface, contamination containment enclosure, and service connections.

The design of the structural frame and contamination control system were fixed by external constraints and previous design studies. The geometry of the work volume was in question, with contending options advanced by different factions within the customer's organization. A design space was created which included one fixed-geometry, and several extendible-geometries with both rigid and flexible enclosures. The viable combinations within this set of parameters were shown to include all possible designs that were consistent with the constraints. Criteria were developed that included satisfaction of key functional requirements, reliability of the contamination containment envelope, cleanability of the work volume, usability by the crew, and cost. The result was the clear superiority of one of the options, to the satisfaction of rival factions within the customer organization.

10.4 DEFINE/REFINE/INTEGRATE SYSTEM PHYSICAL CONFIGURATION

After the System Architecture has been selected, sufficient detail must be developed on the elements to (1) ensure that they will function as an integrated system within their intended environment, and (2) enable subsequent development or design activity as necessary to fully define each element. During this process step establish the physical, software, and operational implementations, at the next level of detail, for the elements in the selected architecture. Identify the defining interface parameters and, to the degree possible for the current stage of development, define the values of those parameters. The objective of this function is to allow the further definition of each configuration item to proceed on its own, in parallel with all the others.

A. Participation

This function can be lead either by Systems Engineering or by Design Engineering, depending on the technical maturity of the design at the time. In either case, the discipline not in the lead has a strong supporting role.

Systems Engineering provides the process for generating and selecting options for each configuration item, performs analyses and trades, identifies and coordinates interfaces, integrates the results, and ensures that all requirements are implemented. Systems Engineering establishes the documentation framework for compiling interface information and making it available to other disciplines, polls the other disciplines for the identity of interfaces and the definition and values of interface parameters, reviews the overall interface definition to identify missing interfaced definitions and data, performs analysis to define interfaces when necessary, resolves disputes over interfaces, and reviews overall integration. Systems Engineering also ensures that the supporting disciplines are present and active, scopes their participation, and ensures that their contributions are coordinated and integrated.

Design Engineering creates design options for configuration items and their arrangement as a system, develops technical definition data, performs analyses and trades, and documents design decisions. Design engineering performs analysis to define interface parameters and their values, and provides documentation of design decisions relevant to interfaces.

Supporting disciplines can propose options for configuration items or their features, monitor implementation of requirements in each specialty area, and review the results of the system definition process. At this stage in system development, certain key disciplines may emerge, such as design for manufacturing or human factors and operability, and these should be identified. Supporting disciplines also can provide information for interface definition in each specialty area, and review the results of the interface definition process. At this stage in system development, certain key disciplines may emerge, such as imposed system environment, EMI/EMC, and these should be identified.

B. Recommended Activities

Note that this function often proceeds in parallel with Define/Derive/Refine Functional/Performance Requirements, as the system development proceeds into a more detailed level of definition.

1. Create a system-level description of system operation, using appropriate tools and notation, to enable a thorough analysis of the system's behavior at the interfaces among all of its elements. Prepare system interface diagrams.

2. Enter the available data about the elements into the system-level description. Obtain interface identification and definition from design engineering and supporting disciplines. Determine what additional data are needed in order to support analysis of system operation.
3. Perform design activity on the elements as needed to provide the additional data identified in Step 2 above.
4. Perform liaison with customer representatives regarding definition of interfaces with the system's operating environment throughout its life cycle.
5. Analyze system operation to verify its compliance with requirements. Modify elements and system architecture, and resolve interface issues as needed to bring the result into compliance.
6. Identify additional data required for each element (functions, requirements, configuration, and interfaces) as input to its design process. Conduct activities as necessary to create that information.
7. Compile data.

Input

1. System architecture, configuration item, and interface identifications.
2. Customer's definition of external interfaces.
3. Technical data on interfacing items.

Output

1. Selected design concepts for configuration items to implement all of the system elements, and identification of their interfaces.
2. Documented definition of all interfaces.
3. Documented justification for the selected concepts.

C. End Result

The result is the definition of the set of configuration items (selected technology, configuration, design parameter values, and arrangement) and the definition of their interfaces, integrated as a system.

Completion Criteria

1. Option selections satisfy all concerned parties.
2. Definition of each configuration item and its interfaces adequate to allow further development of all configuration items, in a parallel process.

D. Methods/Techniques

The method is to establish a systematic framework for identifying interfaces and tracking descriptive data, acquiring updates as they occur, and displaying a consistent set of data in a uniform format to concerned parties.

Metrics

1. System requirements not met (if any) by selected concept.
2. Number or percent of system requirements verified by system operation analyses.
3. Number of TBDs/TBRs in system architecture or design.

4. Number of interface issues not resolved.
5. Percent of identified system elements that have been defined.

Tools

- N^2 Chart
- System Schematic
- Interface diagrams
- Tables and drawings of detailed interface data

THIS PAGE
INTENTIONALLY
LEFT BLANK

11 SYSTEMS ENGINEERING ANALYSES

Systems Engineers support the development process with specialized analyses. These are frequently called "ilities" because of the word endings of many of the studies. This section describes the most frequently performed analyses, how to conduct trade studies and a brief overview of system modeling.

11.1 DEPLOYMENT ANALYSIS

A deployment analysis supports the development of products and processes necessary to deploy system end-items. Deployment analysis includes:

a. Factors for site/host selection and activation/installation requirements,
b. Operational and maintenance facilities, equipment, and personnel requirements,
c. Compatibility with existing infrastructure (e.g., computer-communication systems),
d. Determination of environmental impacts (environment impacts on the system and system impacts on the environment) at deployment sites,
e. Early deployment of training items,
f. Initial provisioning and spares,
g. Packaging, handling, storage, and transportation.

11.2 DESIGN ANALYSIS

Design analysis supports all design-related activities, including concept definition, alternative concept trade studies, synthesis, modeling and simulation at/below the subsystem level, design sizing and evaluation. Evaluation includes analytical determination of design response to normal and abnormal inputs as well as to loads and perturbations such as acceleration, acoustic, electrical, pressure, thermal, vibration, shock, and weight.

Some examples are the preliminary and detailed modeling of electrical circuit boards, and entire packages; the finite element modeling of a structural or thermal control system; the modeling and simulation of a missile dynamics and control system; or a structural dynamic model to evaluate structural response to vibration and shocks.

The availability of large, high speed computers, sophisticated programming tools and existing software has led to high fidelity modeling and simulation support for almost any design activity. The simulation model represents the engineer's knowledge of his system as well as its predicted response to various normal and abnormal stimuli. During the development process, as engineering models and prototypes are built and tested, the results can be compared to results predicted by modeling and simulation.

Use of design analysis is extremely important in time and cost savings during product development and in operational element failure analysis. The analyst, with his/her computer program, can run many trial cases to quickly and efficiently select the preferred design configuration. To accomplish the same result by fabrication and testing many models of a complex product could be prohibitively expensive.

11.3 ELECTROMAGNETIC COMPATIBILITY AND RADIO FREQUENCY MANAGEMENT ANALYSIS

Electronic items must be able to perform their mission in their intended electromagnetic environments. Electromagnetic compatibility analysis is performed on electric or electronic items so that they can perform their mission in their intended electromagnetic environments. Analysis also ensures that items that are intentional radiators of radio frequency energy comply with military, governmental, and relevant international policies for radio frequency spectrum management.

11.4 ENVIRONMENTAL IMPACT ANALYSIS

Environmental impact analysis (EIA) is that part of the system design which deals with the system's impact on the environment. In 1970 the Congress of the United States enacted the National Environmental Protection Act (NEPA) which created the Environmental Protection Agency and legally mandated environmental impact statements (EIS). The act forces any Federal Agency project to include a full disclosure of the environmental consequences of the project and to build environmental planning into the decision-making process of Federal Agencies and programs. These environmental regulations impact many commercial systems as well.

In this same time frame, the European Union adopted strong environmental regulations, with strict limitations on the use of components containing, for instance, mercury, lead, cadmium, and chromium 6. These restrictions have global impact for suppliers in world markets. Another restriction the EU adopted for implementation in 2006 is that manufacturers and suppliers retain a degree of life-time responsibility for decommissioning systems they build and sell, which is a substantial change from past practice in the EU. These new regulations can have a profound impact on Systems Engineering responsibilities.

Environmental Protection Agencies worldwide identify the issues and concerns regarding the environment. These may be classified into two broad categories: 1) those impacts which concern the natural environment and 2) those which impact on the human element; there is considerable overlap in these categories. The first category represents the traditional concerns of the environment, such as the habitat, the flora and fauna; there is particular attention to endangered and rare species of flora and fauna, and general deterioration of the habitat from project activity and pollution. The second category determines the effects on the human element. Although "safety" was discussed in the previous section, system features such as "noise pollution", electromagnetic intensities, traffic, growth inducement, impacts on the community infrastructure, and other socio-economic impacts have become major considerations in the environmental assessment of a project.

Because the EIA can have a strong influence on the design of the project, indeed the project cannot move forward until the EIS is approved, the draft EIS should be available no later than the Design Concept Review (DCR) for the project and approval of the final EIS should be no later than the Preliminary Design Review. At this stage, the project will include the recommendations from the EIS which will minimize the environmental impacts and possibly enhance the quality of the environment associated with the project.

Environmental Analysis should proceed in parallel with the conceptual design phase of the project. With inputs and analysis from the members of the interdisciplinary team, design concept changes will likely be recommended to the project. There are three key points that could influence the project:

1. Attaining the environmental goals may demand substantial design changes,
2. The project may include an imbedded environmental monitoring element,
3. There may be an environmental feedback loop over the project's life cycle affecting the project's operation.

The development of the EIA/EIS will proceed like any SE process and minimally it includes the following steps (see Figure 11-1):

1. Establish the environmental goals,
2. Perform a functional analysis of the project and the environment reflecting the goals,
3. Establish the requirements and a set of metrics to measure the affect the project will have on the environment,
4. Measure, with assessments and models, the project's impact on the environment,
5. Synthesize a set of designs for the project that will satisfy the environmental goals.

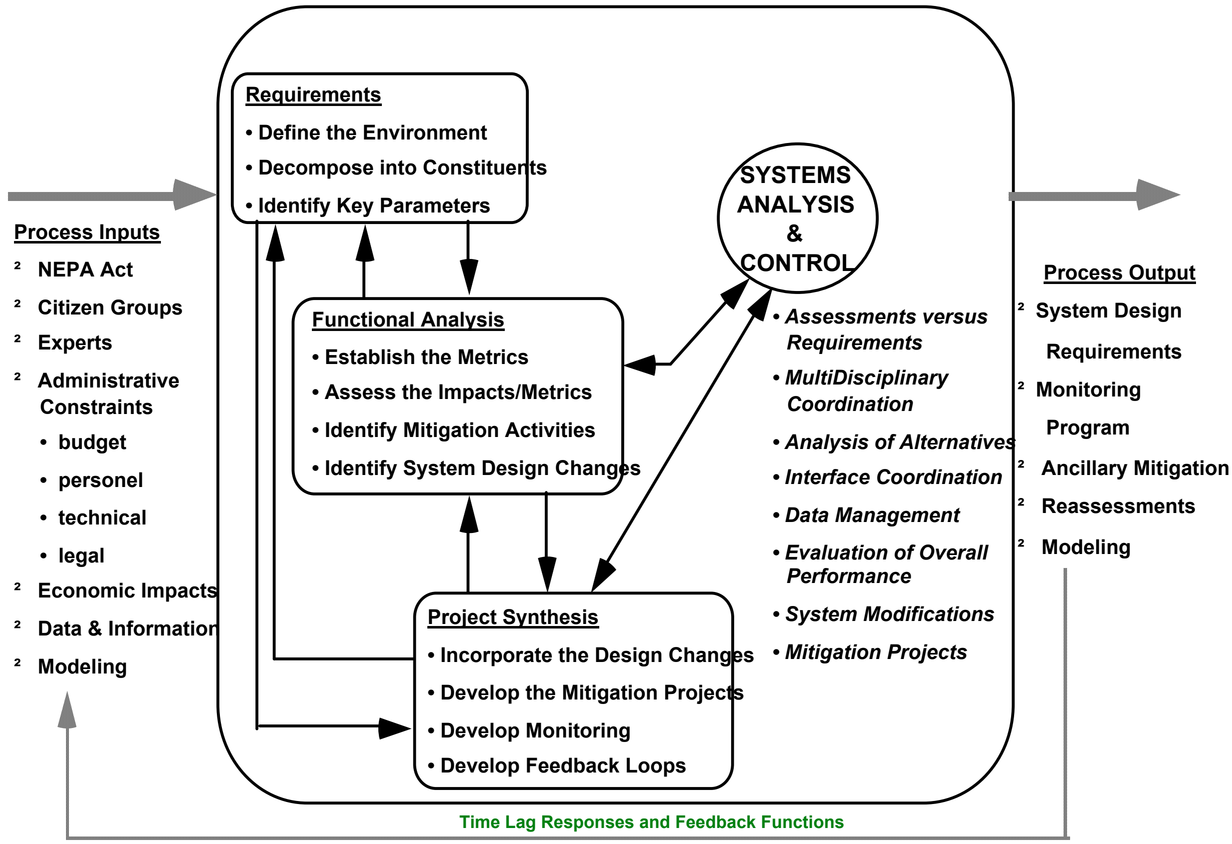


Figure 11-1. A Systems Engineering Process for EIA

Incorporating the EIA/EIS into the Systems Engineering process for a project is a new and major step. It has evolved from the gradually expanding awareness and concern for environmental issues. It is expected that the process of incorporating the environment into projects will grow in familiarity and maturity. The SE process will greatly facilitate designing for the environment and the above summary of procedures is a start in that direction. The generic nature of this process makes it equally appropriate for integrating the environmental legislation imposed in other parts of the world.

References

1. EPA 600/5-74-006 *Environmental Impact Requirement in The States: NEPA's Offspring*, April 1974
2. Hear, J. E. and Hagarty, D. J.; *Environmental Assessments and Statements;* Van Nostrand Reinhold Environmental Engineering Series, 1977
3. Mallory, R. S.; *Legally Required Contents of a NEPA Environmental Impact Statement*; Stanford Environmental Law Society, 1976

11.5 HUMAN SYSTEMS ENGINEERING AND ANALYSIS

The Human Engineering (HE) or Human Systems Engineering (HSE) effort affects every portion of the system that has a person-machine interface. A detailed coverage of this topic is contained in Appendix B.

It is essential to integrate human system factors into the design of items. The objective is to achieve a balance between system performance and cost by ensuring that the system design is compatible with the capabilities and limitation of the personnel who will operate, maintain, transport, supply, and control the system. Requirements and designs should minimize characteristics that require extensive cognitive, physical, or sensory skills; require the performance of unnecessarily complex tasks; require tasks that unacceptably impact manpower or training resources; or result in frequent or critical errors.

11.6 LIFE CYCLE COST ANALYSIS

Life cycle cost (LCC) analyses are performed to help understand the total cost impact of a program; to compare between program alternatives; and to support tradeoff studies for system decisions made *throughout* the system life cycle.

The LCC of a system includes the development of hardware and software, production or operations, support, and personnel costs through development, acquisition, operational support, and, where applicable, disposal. An LCC estimate task is initiated in order to identify cost "drivers" or areas where resources can best be applied to achieve the maximum cost benefit. These LCC studies should examine those performance parameters where small changes in the parameters produce significant changes in development and operational costs. For example, sometimes a relatively small change in mean-time-to-repair (MTTR) or mean-time-between-failures (MTBF) results in large savings in operational costs.

Life cycle cost analyses are used in system cost/effectiveness assessments. The LCC is not necessarily the definitive cost proposal for a program. LCC estimates are often prepared early in a program's life cycle -- during Concept Definition. At this stage, there is insufficient detail design information available to support preparation of a realistic, definitive cost analysis. These are much more detailed, and prepared perhaps several years later than the earliest LCC estimates. Later in the program, life cycle LCC estimates can be updated with actual costs from early program phases and should be more definitive and accurate due to hands-on experience with the system.

In addition to providing information for the LCC estimate, these studies also help to identify areas in which emphasis can be placed during the subsequent sub phases to obtain the maximum cost reduction.

Adequate documentation requires three basic elements:
1. data and sources of data on which the estimate is based;
2. estimating methods applied to that data; and
3. the results of the analysis.

Background

LCC normally includes the following, which are depicted in Figure 11-2.
1. Research and Development (R&D) phase costs
2. Investment (Production and Deployment/Installation) phase costs
3. Operation and Support (O&S) phase costs
4. Disposal and Termination costs

The above costs should include hardware, software, material, personnel, support agencies and suppliers, operations, and logistics.

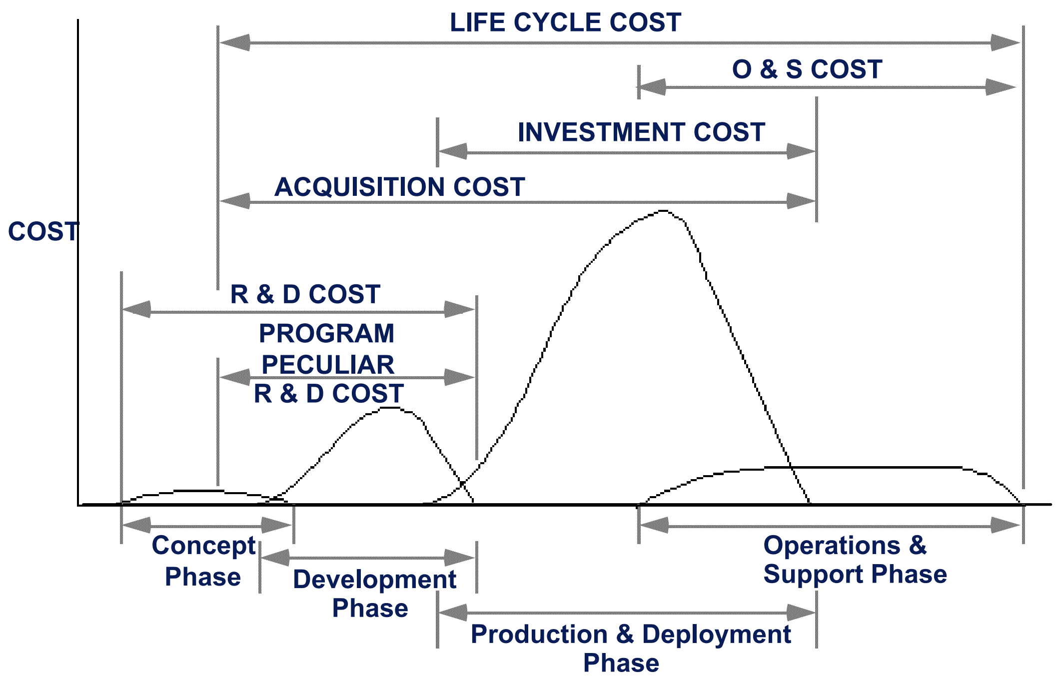


Figure 11-2. Life Cycle Cost Elements (Prod & Deploy)

A description, as complete as possible, or parametric equations, learning curves, cost-performance analysis, and factor derivations or build-up techniques for each part of the estimate provides continuity and consistency and facilitates tracking for future estimates. Comparison to prior estimates and analysis of reasons for differences make up an estimate track. The explanation of differences should be quantitatively expressed, if possible.

Recommended Activities

1. Obtain a complete definition of the system, elements, and their subsystems.

2. Determine the total number of units of each element, including operational units, prototypes, spares, and test units to be procured. If it is desired to develop parametric cost data as a function of the number of operational units, define the minimum and maximum number of operational units and how, if any, the number of spares and test units will vary with operational unit size.

3. Obtain the life cycle program schedule, including spans for R&D, Production & Deployment, and O&S phases. The Production and Deployment phase length will vary with the number of operational units.

4. Obtain manpower estimates for each phase of the entire program and, if possible, for each element and subsystem. These are especially important for cost estimating during R&D and O&S. Paygrade distribution is also important.

5. Obtain approximate/actual overhead, G & A burden rates and fees that should be applied to hardware and manpower estimates. Usually this is only necessary for effort within your on company; suppliers will have already factored it into their cost estimates. This data is not required to the accuracy that your finance department would use in preparing a formal cost proposal.

6. Develop cost estimates for each subsystem of each system element for each phase of the program. This is, of course, the critical step. Generally, it should be done as accurately as time and resources allow. Sometimes the argument is heard that the LCC estimates are only to support internal program tradeoff decisions and therefore the estimates must only be accurate enough to support the tradeoffs (relative accuracy), and not necessarily realistic (absolute accuracy). This is usually a bad practice. The analyst should always **attempt** to prepare accurate cost estimates. These estimates are often reviewed by upper management and customers. It enhances the credibility of results if reviewers sense the costs are "about right", based on their past experience.

   6a. Both R&D and O&S costs can usually be estimated based on average manpower and schedule spans. Include overhead, G & A, and fees, as necessary.

   6b. Investment costs are usually prepared by estimating the cost of the **first** production unit, then applying **learning curve** formulae to determine the reduced costs of subsequent production units. For an item produced with a 90 percent learning curve, each time the production lot size doubles (2, 4, 8, 16, 32, ... etc.) the average cost of units in the lot is 90 percent of the average costs of units in the previous lot. A production cost specialist is usually required to estimate the appropriate learning curve factor(s).

   6c. R&D and Investment costs can sometimes be scaled by "complexity factors" or Cost Estimating Relationships (CERs) from accurate costs of existing items. This entails fact finding with experts familiar with the item. For example, if the expert estimated the item was 120 percent more difficult to develop than an existing item whose costs are known:

   R&D cost (new item) = 1.2 x R&D cost (existing item)

   Similarly, CERs are simple, heuristic equations which can be used to scale costs up (or down) over a limited range of primary parameters which drive cost. For example, if you are developing an optical sensor with a certain primary mirror diameter, number of optical elements (mirrors or lenses), and number of detectors, its cost could be approximately scaled from a known sensor by use of the cost drivers in a CER, such as:

   $$\text{1st Unit Prototype cost (new sensor)} = k_1(\text{Dia.}) + k_2(\text{Elements}) + k_3(\text{Detectors})$$

   where the k( )s sum to the 1st unit prototype cost of the known sensor when its characteristics are plugged into the equation.

7. Attempt to obtain customer guidelines for system costing. These guidelines should include the categories of costs they expect to see. An example of these cost categories for one government agency is shown in Table 11-1.

8. Document results, including assumptions, approaches, rationale, and overall cost accuracy estimates for each program phase.

Table 11-4. Example of LCC Element Categories

- **RESEARCH & DEVELOPMENT**
  - Development Engineering
  - Producibility Engineering & Planning
  - Tooling
  - Prototype Manufacturing
  - Data
  - System Test & Evaluation
  - System/Project Management
  - Training
  - Facilities
  - Other
- **INVESTMENT**
  - Non-Recurring Investment
  - Production
  - Engineering Changes
  - System Test & Evaluation
  - Data
  - System/Project Management
  - Operational/Site Activation
  - Training
  - Initial Spares & Repair Parts
  - Transportation
  - Other
- **OPERATIONS & SUPPORT COSTS**
  - Operational Personnel
    - Operator Pay & Allowances
    - Maintenance Pay & Allowances
    - Indirect Pay & Allowances
    - Relocation Costs
  - Consumption
    - Replenishment Spares
    - Consumables Costs
    - Unit Training & Supplies
  - Depot Maintenance
    - Labor
    - Material
    - Transportation
  - Modifications, Material
  - Other Direct Support Operations
    - Maintenance Labor
    - Other Direct
  - Indirect Support Operations
    - Personnel Replacement
    - Transportation
    - Quarters, Maintenance & Utilities
    - Medical Support

C. Methods / Techniques

1. Expert Judgment - which is consultation with one or more experts (good for sanity check, but may not be consistent).

2. Analogy - which is reasoning by comparing the proposed project with one or more completed projects that are judged to be similar, with corrections added for known differences (may be acceptable for early estimations).

3. Parkinson Technique - which defines work to fit the available resources.

4. Price-To-Win - which focuses on providing an approach at or below the price judged necessary to win the contract.

5. Top-Down - which is based on developing costs from the overall characteristics of the project (from the top level of the architecture).

6. Bottom-Up - which identifies and estimates costs for each component separately and computes the sum of the parts.

7. Algorithmic (parametric) - which uses mathematical algorithms to produce cost estimates as a function of cost driver variables, based on historical data. Often supported by commercial tools/models.

8. Design-To-Cost (DTC) or Cost-As-An-Independent-Variable (CAIV) - which works on a design solution that stays within a predetermined set of resources.

Tools

• Parametric Models	• Activity-Based Costing Tools
• Spreadsheets	• Decision Support Tools

Metrics

1. Project Cost Estimates	7. Estimate of Cost-To-Complete
2. Schedule Estimates	8. Impact Estimate for Modification or Change
3. Estimate of Maintenance Cost	9. Estimate of Operations Cost
4. Estimate of Support Costs	10. Cost Variance
5. Schedule Variance	11. Size Measures
6. Cost Risk	12. Schedule Risk (Quantified in Days or Dollars)

11.7 MANUFACTURING AND PRODUCIBILITY ANALYSIS

The capability to produce a hardware item satisfying mission objectives is as essential as the ability to properly define and design it. For this reason, production engineering analysis and trade studies for each design alternative form an integral part of the Systems Engineering process beginning in this sub phase. It includes producibility analyses, production engineering inputs to system effectiveness analysis, tradeoff studies, and the identification of special tools, test equipment, facilities, personnel and procedures. A key element is to determine if existing proven processes can do the job since this could be the lowest risk and most cost-effective approach. Critical producibility requirements are identified during system analysis and design and included in the program risk analysis, if necessary. Where production engineering requirements create a constraint on the design, they are included in the applicable specifications. Manufacturing test considerations are fed back to the engineering efforts and are taken into account in Built-In-Test (BIT), Automated Test Equipment (ATE), and manual test trade studies and design. Long-lead-time items, material limitations, special processes, and manufacturing constraints are evaluated and documented.

Manufacturing analyses support the development of manufacturing product and process requirements and solutions necessary to produce system end items. Manufacturing analyses include producibility analyses and manufacturing and production inputs to system effectiveness, trade-off studies, and life cycle cost analyses. Analyses evaluate alternative design and capabilities of manufacturing. These analyses and manufacturing product and process alternatives are considered interactively with other system products and processes. Design features and associated manufacturing processes critical to satisfying performance and cost needs are clearly identified. High-risk manufacturing elements, processes, or procedures are identified and manufacturing risks are included in technical risk management efforts. Long lead-time items, material source limitations, availability of materials and manufacturing resources, and production cost are also considered, assessed, and documented.

Producibility analysis is a key task in developing low cost, quality products. Multidisciplinary teams should simplify the design and stabilize the manufacturing process to reduce risk, manufacturing cost, lead time, and cycle time; and to minimize strategic or critical materials use. Design simplification should address ready assembly and disassembly for ease of maintenance and preservation of material for recycling. The selection of manufacturing methods and processes should be included in early design efforts.

a. Prior to full rate production, the contractor must ensure that the product design has stabilized, the manufacturing processes have been proven, and rate production facilities, equipment, capability, and capacity are in place (or are about to be put in place) to support the approved schedule.

b. Value engineering concepts assist in the identification of requirements that add cost to the system, but add little or no value to the users.

During system analysis and design, the specific tasks that are addressed are the following:

a. Evaluate production feasibility
b. Assess production risk
c. Identify manufacturing technology needs
d. Develop manufacturing strategy
e. Determine availability of critical materials
f. Develop initial manufacturing plan
g. Evaluate long-lead procurement requirements
h. Develop initial manufacturing cost estimate
i. Define manufacturing test requirements

11.8 MISSION OPERATIONS ANALYSIS

Operational analyses support the development of products and processes for operations. Operational analyses at the systems level address the operational mission. Other operational analyses address the operating mode (or mission) of non-mission items. Operational analysis addresses the operational use of the item, reflecting the way the item will be used to accomplish required tasks in its intended environment. These analyses include the host system (if any), multiple systems, and other external systems required to execute identified operational functions and applicable joint and combined operations. Analyses should address all modes of operational employment and operational deployment of the system and its interactions with other systems.

11.9 RELIABILITY, MAINTAINABILITY AND AVAILABILITY ANALYSIS

Reliability and availability analyses are performed to assure that the system under development meets the reliability and availability objectives of the user.

Diagnostics must be incorporated to unambiguously detect and isolate mission, safety, and maintenance faults known or expected to occur in the system while the system is operational. Integrated diagnostics factors include embedded testability, built-in-test (BIT), and automatic and manual testing. A major measure in support is Availability, which in turn is a function of MTBF and MTTR. These factors are iterated to achieve an acceptable balance among availability, LCC, MTTR, and MTBF.

Maintainability analyses look into the proper approach to maintaining each element, including locations and levels of repair, and types of scheduled maintenance, repair, or replacement to meet mission objectives in a cost/effective manner. The design process must be monitored to ensure inclusion of adequate maintenance considerations in mission equipment, including handling and support equipment, test and checkout equipment, facilities, and logistical plans.

Emphasis is placed on:

1. Determining requirements based on the user's system readiness and mission performance requirements, physical environments and resources available to support the mission.

2. Managing the contributions to system reliability made by system elements. Some measures include: Failure Rate, Mean-Time-Beween-Failures (MTBF), Mean-Time-To-Repair (MTTR), and Mean Error Isolation Time.

3. Ability to find and isolate errors after failures and repair them.

4. Preventing design deficiencies (including single point failures), precluding the selection of unsuitable parts and materials, and minimizing the effects of variability in the manufacturing process.

5. Developing robust systems, acceptable under specified adverse environments experienced throughout the system's life cycle, repairable or restorable under adverse conditions and supportable under conditions consistent with the ILS plan.

6. Requirements for parts, software, materials, and processes should be developed that insure the reliability standards for the program can be obtained. Standards and Specifications should be incorporated into program specifications, where appropriate.

7. Monitoring and managing the contributions to system availability from system reliability, maintainability, supportability, and the overall ILS plan.

11.10 SAFETY AND HEALTH HAZARD ANALYSIS

Safety analysis identifies and reduces hazards associated with elements of the system. The safest possible item is designed consistent with requirements and cost effectiveness. Risks associated with identified hazards are documented to establish criteria for defining and categorizing high and serious risks. Materials categorized as having high and serious risks should also be characterized in terms of the risks related to producing, fielding, operating, supporting, and training with system end items using such materials.

Systems engineers identify and reduce hazards associated with system items. The safest possible item is designed consistent with requirements and cost effectiveness. Risks associated with identified hazards are documented to establish criteria for defining and categorizing high and serious risks. Materials categorized as having high and serious risk are characterized in terms of the risks related to producing, fielding, operating, supporting, and training with system end items using such materials. If the use of hazardous materials is essential, a program for containment and/or possible substitution should be developed and implemented. Handling and disposal of hazardous materials should also be included in the life cycle cost estimates.

It is imperative that all elements of safety associated with the system be thoroughly analyzed. The use of hazardous materials should be avoided to the extent practical. If the use of hazardous materials is essential, a program for containment and eventual substitution should be developed and implemented. Handling and disposal of hazardous material should be included in life-cycle cost estimates.

The Safety Analysis process should be tailored as appropriate to meet individual program requirements and cost-effectiveness objectives. Additionally each system safety engineering team may have in-house and customer-based standards, processes and methodologies that should be integrated within the basic process.

It is useful to prepare a System Safety Program Plan, SSPP, which delineates hazard and risk analysis methodologies; tasks; significant milestones; verification, test, and certification; applicable documents;

and prime areas of responsibilities and authority. If used, a preliminary SSPP should be completed by the System Specification Review (SSR) and finalized by the System Functional Review.

Preliminary analysis of requirements enables identification of potential critical areas, system safety and health hazards. These are incorporated into the preliminary hazard list (PHL), which is updated as other potential hazards and criticalities are identified throughout the design process. The PHL is used as the basis for the preliminary hazard analysis (PHA) and/or Health Hazard Analysis (HHA). The primary purpose of system safety and health hazard analyses is to identify and evaluate potential safety problems to enable timely, cost-effective, and appropriate corrective action. The hazard analysis can be performed on both the system and subsystem level depending on program requirements. Potential areas of concern are evaluated for failure modes, critical design features, human error inputs, and functional relationships. To qualitatively evaluate a potential hazard two criteria are used: hazard severity and hazard probability. A hazard risk index (HRI) for each potential hazard is developed using these criteria to prioritize hazards. These criteria are also used to determine whether or not the controls applied to a potential hazard, using the system safety order of precedence are sufficient. Results of the PHA and HHA are documented in separate respective documents and are presented at the PDR.

System and subsystem test and evaluation activities are generally the closure mechanism for the safety analyses process. Test criteria are derived from the tailored checklists and system safety design. System safety test plans may be incorporated into the coordinated system test plan. Dedicated system safety testing during both development and operational testing may or may not be implemented. Through coordination with the primary test group, support by the system safety engineers is determined. Test and evaluation activities supported by the system safety engineering component of the program as necessary include quality test, system test, subsystem test, and software test. Testing may be performed iteratively or collectively.

11.11 SUPPORTABILITY, AND INTEGRATED LOGISTICS SUPPORT ANALYSIS

Planning for support begins in system analysis and design with the development of supportability criteria. These criteria, which include reliability and maintainability requirements as well as personnel, training, facilities, etc., are placed in the system specification to ensure that they are considered in the system design.

Supportability analyses are used to assist in the identification of data and procedures needed in specifications and other development documentation to provide system life cycle support (e.g., additional interface information and verification requirements for utilization of "used" parts). Supportability analyses addresses:

a. All levels of operation, maintenance and training for system end-items.
b. The planned life cycle to ensure that system end-items satisfy their intended use.
c. Identification of supportability related design factors.
d. The development of an integrated support structure.
e. Support resource needs including parts, software, data, people and materials.
f. Determine requirements for system restoration, fix, or operational work-arounds.

Integrated Logistics Support (ILS) analysis focuses on assuring that developed items are supportable. The following factors are incorporated during the application of the Systems Engineering process:

a. Developing support requirements that are related consistently to readiness objectives, to design, and to each other.

b. Interactively integrating support factors into item and system element design with the design of support products and processes.

c. Identifying the most cost-effective approach to supporting all items (hardware, software, and data) when they are deployed/installed. This includes repair or replacement determination for all parts and assemblies of each element at the appropriate level (field, forward support, depot, or factory).

d. Ensuring that the required support structure elements are identified and developed so that the item is both supportable and supported when deployed/installed.

e. Planning for post-production support to ensure economic logistics support after cessation of production.

11.12 SURVIVABILITY ANALYSIS

Certain weapon systems and other items that must perform critical functions in hostile environments must survive the threats defined for the specified levels of conflict or other situations. Threats to be considered for weapon systems may include conventional, electronic, nuclear, biological, chemical, high power microwave, kinetic energy weapons and directed energy weapons, and terrorism or sabotage. If degraded performance is acceptable under certain threat levels, this should be specified. A commercial system such as an automobile should consider threats such as weather extremes, road hazards, collisions, shipping and handling environments, etc. A commercial communications satellite design must consider high energy electromagnetic emissions, space launch environments, and various long-term, on-orbit environments, including those from atmospheric/exo-atmospheric nuclear tests by rouge nations in non-compliance with test ban treaties.

Critical survivability characteristics should be identified, assessed, and their impact on system effectiveness evaluated by the Systems Engineer.

Survivability analysis is performed when items must perform critical functions in a man-made hostile environment. Survivability from all threats found in specified levels of conflict are analyzed. Threats to be considered include conventional, electronic, nuclear, biological, chemical, high-power microwave, kinetic energy weapons, directed energy weapons, and terrorism or sabotage. Critical survivability characteristics are identified, assessed, and their impact on system effectiveness evaluated.

For items hardened in order to meet a survivability requirement, hardness assurance, hardness maintenance, and hardness surveillance programs are developed to identify and correct changes in manufacture, repair, or spare parts procurement; and maintenance or repair activities that may degrade item hardness during the system's life cycle.

11.13 SYSTEM COST/EFFECTIVENESS ANALYSES

System cost/effectiveness analyses support the development of life-cycle balanced products and processes. Systems/cost effectiveness analyses tasks are integrated into the Systems Engineering process. During the conduct of the analyses, critical requirements and verifications identified serve as constraints on impacted areas. These requirements and verifications should be included in pertinent requirements documentation and specifications. These analyses are conducted throughout the life cycle and serve as the analyses supporting design decisions at appropriate times.

System cost/effectiveness analyses are the primary means of deriving critical system performance and design requirements. Some examples of critical cost/effectiveness analyses are:

1. An airline's studies of the desired aircraft performance features (types, size, speed) to increase its market share at lowest overall cost over its route structure.
2. A communications company's studies of the desired characteristics of a communications satellite to serve specified markets most economically.
3. A world-wide corporation's studies of the most cost/effective networking scheme.
4. A city's studies of the most cost/effective method(s) to improve its transportation infrastructure, including bus, train, mass transit, new freeways, routes, and departure schedules.

Analysis should be conducted and integrated, as required, to:

1. Support the identification of mission and performance objectives and requirements;
2. Support the allocation of performance to functions;
3. Assist in the selection of preferred product and process design requirements;
4. Provide criteria for the selection of solution alternatives;
5. Provide analytic confirmation that designs satisfy customer requirements;
6. Impact development decisions on the determination of requirements, designs, and selection of preferred alternatives for other system products and processes;
7. Support product and process verification; and
8. Support modification and enhancement decisions in later stages of the life cycle;

11.14 SYSTEM MODELING

The function of modeling is quickly and economically to create data in the domain of the analyst or reviewer, not available from existing sources, to support decisions in the course of system development, production, testing, or operation.

A model is a mapping of the system of interest onto a simpler system which approximates the behavior of the system of interest in selected areas. Modeling may be used to represent:

- The system under development
- The environment in which the system operates

Objective

The objective of modeling is to obtain information about the system before significant resources are committed to its design, development, construction, testing, or operation. Consequently, development and operation of the model must consume time and resources not exceeding the value of the information obtained through its use.

Important areas for the use of models include the following:

- Requirements Analysis: determine and assess impacts of candidate requirements

- System Synthesis Tradeoffs: evaluate candidate options against selection criteria

- Design & Development: obtain needed design data and adjust parameters for optimization

- Test and Verification: simulate the system's environment and evaluate test data (model uses observable data as inputs for computation of critical parameters that are not directly observable)

- Operations: simulate operations in advance of execution for planning and validation

Results

The result of modeling is to predict characteristics (performance, reliability, operations and logistics activity, cost, etc.) across the spectrum of system attributes throughout its life cycle. The predictions are used to guide decisions about the system's design, construction, and operation, or to verify its acceptability.

Criteria for Completion

a. Validation of the model through an appropriate method to the satisfaction of responsible parties

b. Documentation of the model including background, development process, a complete description of the model itself and its validation, and a record of activities and data generated by its use, sufficient to support evaluation of model results and further use of the model

c. Output data delivered to users

Tools

Standard tools for all types of modeling are now available commercially for a wide range of system characteristics.

Recommended Activities

The general steps in application of modeling are:

1. Select the appropriate type(s) of model
2. Design the model
3. Validate the model
4. Obtain needed input data and operate the model to obtain desired output data
5. Evaluate the data to create a recommendation for the decision in question
6. Review the entire process, iterating as necessary to make corrections and improvements
7. Evolve the model as necessary

(Also review the discussion of Modeling and Simulation in Appendix D)

11.14.1 SELECTING THE MODEL TYPE(S)

A particular application may call for a single type of model or several types in combination. The selection depends on the nature of the object system, its stage in its life cycle, and the type of information to be obtained from the model.

Criteria

Quick: get the needed information to support the decision in a timely manner.

Economical: resources used to create and operate the model must be in proportion to the value of the information.

Accurate: the model must be proven to be a sufficiently faithful representation, i.e., validated, so the information it provides adequately represents the actual system to follow.

Precautions

Complex systems with non-linearities often exhibit surprising and counter-intuitive behavior. In these cases modeling may be the only way to get the needed information, but the lack of previous experience with similar systems provides no assurance that the model is valid. Assurance depends on careful analysis of the problem and selected modeling methodology, subject to independent review, to determine that all relevant factors have been adequately represented. Tests of the model against known or independently analyzed test cases are advised.

In many cases the fundamental phenomena of interest are not directly observable. A superficial model based only on observable data may apply to the specific case represented but may not extend to more general situations. Fluid dynamics and heat transfer provide illustrations, in which dimensionless ratios such as Reynolds number, Mach number, or thermal diffusivity are the parameters of interest, rather than the observable physical quantities such as fluid velocity, temperature, or density.

Modeling can create an activity trap, absorbing excessive time and resources in developing the model and running numerous cases. The type of model selected and the detail it represents should be carefully assessed to determine its cost effectiveness, and the modeling process should be carefully managed to prevent overruns. On the other hand, the appropriate use of modeling can avoid costly mistakes or extended development activity later in the program.

Types of Models

Models fall into one of two general categories – representations and simulations. Representations employ some logical or mathematical rule to convert a set of inputs to corresponding outputs with the same form of dependence as in the represented system, but do not mimic the structure of the system. Validity depends on showing, through analysis or empirical data, that the representation tracks the actual system in the region of concern. An example might be a polynomial curve fit, which relates centrifugal pump head to flow over a specific flow range. Simulations, on the other hand, mimic the detailed structure of the simulated system. They are composed of representations of subsystems or components of the system, connected in the same manner as in the actual system. The validity of a simulation depends on validity of the representations in it and the faithfulness of its architecture to the actual system. Usually the simulation is run through scenarios in the time domain to simulate the behavior of the real system. An example might be the simulation of a fluid control system made up of representations of the piping, pump, control valve, sensors, and control circuit.

The type of model selected depends on the particular characteristics of the system which are of interest. Generally, it focuses on some subset of the total system characteristics such as timing, process behavior, or various performance measures.

Representations and simulations may be made up of one or several of the following types: Physical, Graphical, Mathematical (deterministic), and Statistical.

Physical models exist as tangible, real-world objects which are identical or similar in the relevant attributes to the actual system. The physical properties of the model are used to represent the same properties of the actual system.

Examples of physical models include:

- Wind tunnel model
- Mock up (various degrees of fidelity)
- Engineering model (partial or complete)
- Hanger queen

- Testbed
- Breadboard/brassboard
- Prototype
- Mass/inertial model
- Scale model of section
- Laser lithographic model
- Structural test model
- Thermal model
- Acoustic model
- Trainer

Graphical models are a mapping of the relevant attributes of the actual system onto a graphical entity with analogous attributes. The geometric or topological properties of the graphical entity are used to represent geometric properties, logical relationships, or process features of the actual system.

Examples of graphical models include:

- Functional flow charts
- Behavior diagrams
- Plus function flow charts
- N^2 charts
- PERT charts
- Logic trees
- Document trees
- Time lines
- Waterfall charts
- Floor plans
- Blue prints
- Schematics
- Representative drawings
- Topographical representations
- Computer-aided drafting of systems or components

Mathematical (deterministic) models use closed mathematical expressions or numerical methods to convert input data to outputs with the same functional dependence as the actual system. Mathematical equations in closed or open form are constructed to represent the system. The equations are solved using appropriate analytical or numerical methods to obtain a set of formulae or tabular data defining the predicted behavior of the system.

Examples of mathematical models include:

- Dynamic motion models
- Structural analysis using finite elements
- Thermal analysis
- Vibrational analysis
- Electrical analysis as in wave form or connectivity
- Eigen value calculations
- Linear programming
- Cost modeling
- Network or nodal analysis
- Decision analysis
- Operational or Production Throughput Analysis
- Flow field studies
- Work Flow Analysis
- Hydro-dynamics studies
- Reliability & Availability Models
- Control systems modeling
- Maintainability Analysis
- Computer aided manufacturing
- Process Models
- Object-oriented representations
- Entity Relationship Models

Statistical models are used to generate a probability distribution function for expected outcomes, given the input parameters and data. Statistical models are appropriate whenever truly random phenomena are involved as with reliability estimates, whenever there is uncertainty regarding the inputs such that the input is represented by a probability distribution, or whenever the collective effect of a large number of events may be approximated by a statistical distribution.

Examples of statistical models include:

- Monte Carlo
- Logistical support

- Process modeling
- Manufacturing layout modeling
- Sequence estimation modeling
- Discrete
- Continuous

Rapid Prototyping

A rapid prototype is particular type of simulation quickly assembled from a menu of existing physical, graphical, or mathematical elements. Examples include tools such as laser lithography or computer simulation shells. They are frequently used to investigate form and fit, human-system interface, operations, or dynamic envelope or producibility considerations.

Rapid prototyping is probably the best way to include human engineering and account for the users. Rapid prototyping of the user interface, tested with representative users, is one of the best ways to get user performance data and evaluate alternate concepts. Objective and quantitative data on performance times and error rates can be obtained from higher fidelity interactive prototypes.

11.14.2 DESIGN OF THE MODEL

Care is needed in the design of the model to ensure that the general criteria are met. Usually this requires some degree of fundamental analysis of the system:

1. Identify the relevant system characteristics which are to be evaluated through use of the model.

2. Determine the relevant measurable parameters which define those characteristics, and separate them from irrelevant parameters.

3. Define the scope and content of data needed to support the decision economically and accurately.

It is particularly important that the model be economical in use of time and resources, and that the output data be compact and readily understandable to support efficient decisions. The Taguchi Design of Experiments process (identifying the sensitivity of the results to variation of key parameters and adjusting the spacing of sampling so that the total range of results is spanned with the minimum number of test points) can be very effective in determining the bounds and the limits of the model. This data can be used to estimate the value of the information gained by producing the model.

The model itself can be considered as a system to which the Requirements Analysis, Functional Analysis, and System Synthesis steps of the Systems Engineering Process Engine are applied to determine the requirements for the model and define the approach.

This analysis provides an overall description of the modeling approach. Following its review and approval, the detailed definition of the model can be created according to usual practice for the type of model selected.

11.14.3 MODEL VALIDATION

It is crucial to prove that the model is trustworthy, particularly in cases where a feel for system behavior is absent, or when serious consequences can result from inaccuracy. Models can be validated by:

1. Experience with application of similar models in similar circumstances
2. Analysis showing that the elements of the model are of necessity correct and are correctly integrated

3. Comparison with test cases in the form of independent models of proven validity or actual test data

11.14.4 MODEL APPLICATION

Obtain needed input data to set the model's parameters to represent the actual system and its operating environment. In some situation, defining and acquiring the basis model data can be a very large effort, so care in design of the model is needed to minimize this problem. Perform as many runs as are needed to span the range of the system parameters and operating conditions to be studied, and in the case of statistical models, to develop the needed level of statistical validity.

11.14.5 DATA EVALUATION

Reduce the output data to a form which concisely supports the decision to be made, and draw the appropriate conclusions.

11.14.6 REVIEW

Review the entire process to ensure that it supports the conclusion reached. Explore the sensitivity of the result to changes in initial assumptions, data, and processes. If the result is an adequate level of confidence in an unambiguous decision, then the task is complete. Otherwise, look for corrections or improvements to the process and iterate.

11.14.7 EVOLUTION OF THE MODEL INTO A COMPONENT OF THE SYSTEM

In some cases, a model, created initially to support analysis of the system, evolves to become a deliverable portion of the system. This can occur in cases such as a model of system dynamics which becomes the core of the system control system, or an operations simulation model which evolves into a tool for system operations planning used in the operational phase. The potential for the model to evolve in this manner should be a factor in initial selection and design of the model; anticipation of future uses of the model should be included in its initial conception.

11.15 SYSTEM SECURITY ANALYSIS

System security analysis identifies, evaluates, and eliminates or contains item vulnerabilities to known or postulated security threats (documented for contractual use). Item susceptibility to damage, compromise, or destruction is identified and reduced. Computer security is explicitly addressed early in the acquisition of items that have a potential to emanate sensitive information. All items and their processes are evaluated for known or potential vulnerabilities for the entire life cycle. The Government establishes the level to which the vulnerability is be reduced.

The customer or system prime contractor should identify, evaluate, and eliminate or contain item vulnerabilities to known or postulated security threats (documented). Item susceptibility to damage, compromise, or destruction should be identified and reduced to acceptable levels. Computer security should be explicitly addressed early in the acquisition of items that have a potential to emanate sensitive information. All items and their processes should be evaluated for known or potential vulnerabilities for the entire life cycle. The customer and/or prime contractor should establish the requirements and goals for the levels to which vulnerabilities should be reduced.

Security analysis of a system is required when accreditation or certification of the system is a goal. Accreditation is the official authorization to operate an AIS or network: a) in a particular security mode; b) with a prescribed set of administrative, environmental, and technical security safeguards; c) against a defined threat and with stated vulnerabilities and countermeasures; d) in a given operational environment; e) under a stated operational concept; f) with stated interconnections to other AISs or networks; and g) at an acceptable level of risk for which the accrediting authority has formally assumed responsibility. The designated accrediting authority (DAA) formally accepts security responsibility for the operation of an AIS or network and officially declares that it will adequately protect intelligence against compromise, destruction, or unauthorized alteration. The DAA is assigned by the concerned organization with which your system will interconnect.

Certification is the comprehensive evaluation (testing) of the technical and non-technical security features of an AIS or network and other safeguards, made as part of and in support of the accreditation process, that establishes the extent to which a particular design and implementation meets a specified set of security requirements.

Security requirements for a particular system, as defined by the DAA, determine what the threats and vulnerabilities are to a system. In the past these requirements were generally based on Government security guidance (for further information, see reference list). Various security analysis documents are usually a requirement for accreditation. Examples of these are the Threat Analysis document, which describes the threats and vulnerabilities of the system, a formal Security Policy document, which documents the security requirements of the system to be accredited, and a Security Analysis Report, which describes whether or not the security requirements are met by the system, and how the system meets these requirements. An Accreditation Test Plan, Procedures, and Accreditation Test Report documents are also required. Other operational documentation may also be required.

It is the job of a computer security (COMPUSEC) engineer to provide guidance to the systems and software developers in designing the system in a secure manner. COMPUSEC engineers also perform the required security analysis to identify and document the threats and vulnerabilities of a system, and generate the documentation required by the DAA for accreditation and certification.

11.16 TRADE STUDIES

Trade studies provide an objective foundation for the selection of one of two or more alternative approaches to solution of an engineering problem. The trade study may address any of a range of problems from the selection of high-level system architecture to the selection of a specific COTS processor.

In developing a design, it is tempting to select a design solution without performing a formal trade study. The selection may seem obvious to us--the other possible alternatives appear unattractive, particularly to other IPPD Team members (e.g., design, manufacturing, quality, and other "ility" engineering disciplines). However, it will be far easier to justify the selected solution in a proposal or at a formal design review if we have followed certain procedures in making the selection. Use of a formal trade study procedure will provide discipline in our decision process, and may prevent some ill-advised decisions. It is important, also, to recognize when a formal trade study is not needed in order to reduce project costs.

Whenever a decision is made, a trade-off process is carried out, implicitly, if not explicitly. It is useful to consider trade studies in three levels of formality:

Formal. These trades use a standardized methodology, are formally documented, and reviewed with the customer or internally at a design review.

Informal. These trade studies follow the same kind of methodology, but are only recorded in the engineer's notebook and are not formally reviewed.

Mental. When a selection of any alternative is made, a mental trade study is implicitly performed. The trade study is performed with less rigor and formality than documented trades. These types of trade studies are made continuously in our everyday lives. These are appropriate when the consequences of the selection are not too important; when one alternative clearly outweighs all others; or when time does not permit a more extensive trade. However, when the rationale is not documented, its soon forgotten and unavailable to those who may follow.

One chooses the level of trade study depending on the consequences to the project, the complexity of the issue, and on the resources available. The resources to perform trades are allocated based on the overall life-cycle cost differences (with provision for risk coverage) in alternative selection for the potential trades. Those with the largest overall life-cycle cost deltas are performed first. Since more informal trades can be performed with fewer resources than formal trades, the number and selection of trades and their formality need to be decided with the customer and with the necessary IPPD Team members who might find some design solutions favorably or unfavorably impacting manufacturability, producibility, reliability, testability, maintainability, etc. Remember, it takes minimal effort to document the rationale for informal and "mental" tradeoff conclusions.

Recommended Activities

There are multiple techniques for performing trade studies. These include Multi-Attribute Utility Analysis (MAUA), Decision Trees, and Maximum Expected Utility (MEU). There is no need to standardize on any one. One might be better for one trade study, another better in another situation.

The key components of a formal trade study are the following:

1. A list of viable alternative solutions to be evaluated.
2. A list of selection criteria, i.e., a set of factors that characterize what makes a specific alternative desirable. This should include cost, risk, and performance factors.
3. For each of the selection criteria, a metric characterizing how well the various solutions satisfy that criteria.
4. Weighting values assigned to each of the selection criteria, reflecting their relative importance in the selection process.

With these components, an objective measure of the suitability of each alternative as a solution to the problem is obtained. If this process is performed correctly and objectively, then the alternative with the best score is the best overall solution.

11.16.1 IDENTIFYING ALTERNATIVES

The first step in performing a trade study is the selection of a number of candidate alternative design solutions. In practice, there may be times when as few as only two alternatives need to be considered. However, in general, the trade study should consider between four and seven reasonable alternatives. This will tend to assure that the study will not overlook a viable alternative, while at the same time keeping the cost of the study within reasonable bounds.

It is important that the design solutions being considered be comparable in completeness, i.e., that one can be substituted in our system directly for the other. Where that is not possible, the selection criteria and weighting values need to take into account the disparity.

Do not include alternatives that cannot meet minimum specifications just to expand your trade study. If it ca not meet spec, do not include it. However, if you find that no solution is going to meet your specification, you had better inform higher levels of the problem. Then you might include all viable alternatives, and assign an appropriate metric and weighting value to how close each one comes to meeting the spec. Design alternatives should include those that meet the performance specification, but may be more easily produced, or more reliable, maintainable or supportable.

11.16.2 DETERMINING SELECTION CRITERIA

In most cases, there should be no difficulty in determining the selection criteria. There are usually key characteristics that you are looking for in your solution. In almost every trade study, cost and risk are certain to be significant factors. Risk may be decomposed into cost risk, schedule risk, and performance risk if it appears that these vary separately among the alternatives. If they are not independent, then keep them as a single criterion. Where possible select quantifiable selection criteria; these can be used in decision models.

Make sure that the performance criteria you select are essentially independent. For instance, CPU clock rate and Whetstone performance are closely coupled computer parameters--do not use both. Select only those performance criteria that accurately reflect the needs of your system.

Do not overlook life-cycle cost factors that may be significant to your customer. Manufacturability may be a key factor. Is the solution maintainable? Is it reliable? Will replacement parts be available in the future? Is the software portable to the platforms that will be available in future years? Also, physical parameters such as size, weight, and power consumption could be relevant criteria. Is the solution expandable or scalable? Are design elements or software reusable or already available off-the-shelf?

11.16.3 ASSIGNING METRICS

Assigning metrics to each of the criteria can be very subjective. In order to standardize the interpretation, we will use a scale of one to ten.

One represents total dissatisfaction, while ten represents all we could ever want. The subjective component in assigning metric values arises in determining how to score (i.e., assign values to) various levels of performance. If one processor has a Whetstone rating half of what an ideal one has, do we give it a score of five? Probably not -- more likely a one; unless of course, our modeling studies have indicated that half of the ideal is more than adequate for the task at hand, in which case it might even get a ten. The Systems Engineer will have to use his best engineering judgment in assigning scores. It is essential, however, that he be consistent in how he applies the metrics to the various solutions. Two processors with the same Whetstone rating better have the same score for that criterion.

11.16.4 WEIGHTING VALUES

The weighting values for each criterion distinguish the degree of importance to our design decision. Values should be assigned in the range of one to ten, with ten applying to the most critical criteria for selection. It is important that all parties interested in the decision reach consensus in the assignment of weights. In order to achieve objectivity, this consensus should be reached before the alternative solutions have been scored.

Establishing weighting values can be a difficult task and can become very subjective. For important trades where weightings are particularly difficult to establish, consider using the Analytical Hierarchy Process described in Appendix D.

11.16.5 DETERMINE ADVERSE CONSEQUENCES

It is important to consider the adverse consequences that may be associated with the leading alternatives. These adverse consequences may have been reflected in the attributes selected; however, to assure that they are all considered, a separate step is appropriate. In many cases, where the risk is considerable, this step corresponds to a risk assessment and may be continually tracked as a risk. In any case, the methodology utilized in performing an adverse consequences analysis is the risk assessment methodology.

11.16.6 SENSITIVITY ANALYSIS

For the final evaluation and selection, a sensitivity analysis should be performed. A sensitivity analysis is performed to determine if a relative small variation in scoring is affecting the outcome. If the decision is based primarily on scoring of an individual factor, that score needs to be given extra care since it essentially determines the selection. The sensitivity analysis should concentrate on the criteria most affecting winner selection.

11.16.7 PRESENTING THE RESULTS

The results of the formal trade study need to be both presented and explained in a report. A summary presentation would include the following:

- A summary description of each of the alternative solutions
- A summary of the evaluation factors used
- A graphical display of the overall scores as illustrated in Figure 11-3.
- A summary of the evaluation factors used, and an explanation of why or how the specific weighting values were selected
- A detailed description of each of the alternative solutions
- A summary description of why or how the specific scores were assigned to each of the alternatives for each of the criteria
- A copy of the spreadsheet
- A graphical display of the overall scores as illustrated above
- A graphical display of the weighted scores for each criterion for each of the alternatives, as shown in the example of Figure 11-4.

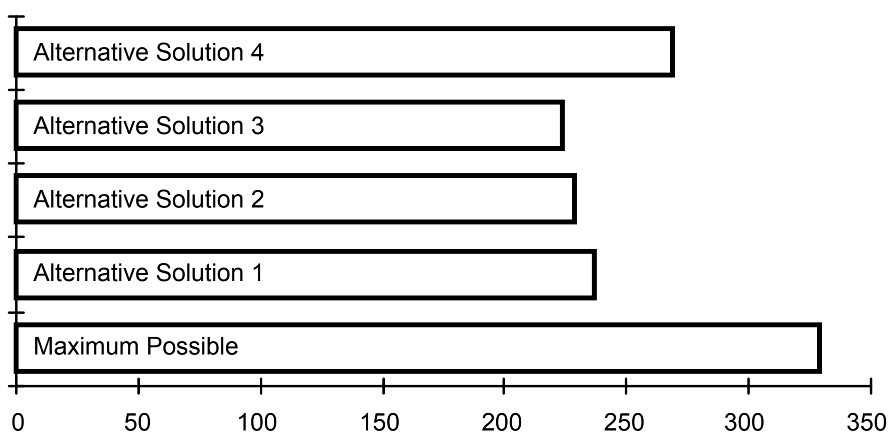


Figure 11-3. Alternative Solution 4 has the highest weighted score

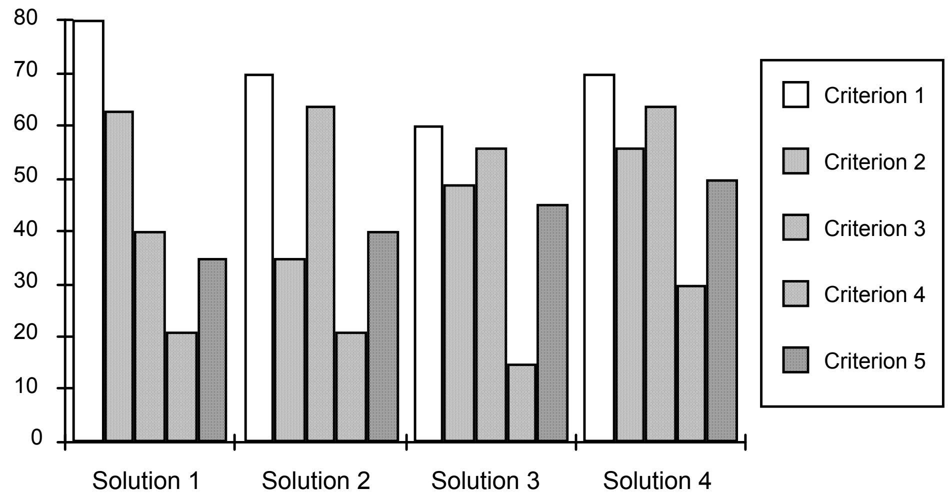


Figure 11-4. Weighted Scores For Each Criterion For Each Alternative

1. Scope
2. Tradeoff Study Team Members
   A.
   B. (List Names and Specialties Represented)
   C.
3. Functional and Performance Design Requirements
   A.
   B.
   C.
4. Design Approaches Considered and Significant Characteristics
   A.
   B.
5. Comparison Matrix of the Design Approaches

Feature or Design Requirement	Alternative 1	Alternative 2	Alternative 3	Alternative 4
Requirements 1 (weight)				
Requirements 2 (weight)				
Requirements n (weight)				

3. Functional and Performance Design Requirements
   A.
   B.
   C.

CE-010-33

Figure 11-5. Tradeoff Study Report Format

11.16.8 PREPARATION OF FORMAL TRADE STUDY REPORTS

Trade studies provide visibility into the Systems Engineering effort and the reasons for selection of one alternative over another. For the most important trades, a report is prepared and the trade result is presented at a customer design review. An example format for a tradeoff study report is shown in Figure 11-5. What follows is a discussion of what information is to be included in each of the paragraphs listed in the figure.

a. Paragraph 1—State the scope of the report.

b. Paragraph 3—Identify and list the functional and technical design requirements which are germane to the tradeoff. In each subparagraph, state the functional requirement first and then identify the related technical design requirements. Immediately following each requirement (and in the same paragraph), a reference should be made which identifies the source of the requirement. This reference consists of the title, file number, date, page number, and paragraph number from which the requirement statement was extracted.

c. Paragraph 4—List the possible design approaches and identify the significant characteristics and associated risks of each design approach. Only reasonably attainable design approaches should be discussed in detail, considering technical capabilities, time schedules, resource limitations, and requirement constraints.

   Characteristics considered must relate to the attributes of the design approaches bearing most directly on stated requirements. These characteristics should reflect predicted impact on such factors as cost, effectiveness, supportability, personnel selection, training requirements, technical data, schedules, performance, survivability, vulnerability, growth potential, facilities, transportability, and producibility. List the less achievable alternatives with brief statements of why they were not pursued.

d. Paragraph 5—Present a comparison matrix of design approaches. The purpose of the matrix is to compare the characteristics for each design approach to determine the degree to which the design approaches satisfy the functional and technical design requirements. The objective is to facilitate rapid comparison and evaluation of potential design approaches, and to allow preliminary screening out of those design approaches that are inconsistent with the functional and technical design requirements. Where applicable, include cost-effectiveness models and cost analysis data as enclosures.

e. Paragraph 6—Recommend the most promising design approach and provide narrative to substantiate the recommendation. Include schematic drawings, outline drawings, interface details, functional diagrams, reliability data, maintainability data, safety data, statistical inference data, and any other documentation or data deemed necessary to support the recommendation. The narrative must cover the requirements that the recommended approach imposes on other areas of the system.

Because there may be a large number of tradeoff study reports prepared during a system development cycle, an index should be prepared which assigns an identification number to each tradeoff study report that has been completed.

11.17 TRAINING ANALYSIS

Training analyses support the development of products and processes for training users of system end-items. Training analysis includes the development of personnel capabilities and proficiencies to accomplish tasks at any point in the system life cycle to the level they are tasked. These analyses address initial and follow-on training necessary to execute required tasks associated with system end-item use.

11.18 DISPOSAL ANALYSIS

Disposal analysis is often a major element for the Environmental Impact Analysis, discussed above. Traditional landfills for non-hazardous solid wastes have become less available within the large city areas and the disposal often involves transporting the refuse to distant landfills at considerable expense. The use of incineration for disposal is often vigorously opposed by local communities and citizen committees, and poses the problem of ash disposal; the ash from incinerators is sometimes classified as hazardous waste. Local communities and governments around the world have been formulating significant new policies to deal with the disposal of non-hazardous and hazardous wastes.

Vast arrays of regulations govern the management of hazardous wastes. The basic tenet for hazardous waste is the "womb-to-tomb" control and responsibility for preventing unauthorized release of the material to the environment. The disposal of radio-active materials has additional constraints and the disposal options for these wastes are limited and costly.

The design of the project should, where appropriate, include a detailed analysis for the disposal of its products, residues, and structure. A goal of the project design should be to maximize the economic value of the project residue and minimize the generation of waste materials destined for disposal. Because of the potential liability that accompanies the disposal of hazardous and radioactive materials the use of these materials must be carefully reviewed and alternatives used where and whenever possible.

The following prioritized guidelines are helpful for managing waste disposal:

1) Design for minimum waste generation, e.g., source reduction,
2) Design for reuse,
3) Design for recycling,
4) Design for transformation, (composting, incineration, bio-degradation, et al),
5) Design for disposal of non-hazardous, non-polluting residue.

Recommended Activities

Managing waste disposal is a concurrent SE process as described in this handbook; it should be an element of the SEMP. Waste management requires a coordinated, top-down systems and subsystems process and it is an integral part of the life-cycle analysis for the project. Disposal could become a significant cost factor to a project and your organization could become a "deep-pocket" for decontaminating the environment from unauthorized releases of pollutants. A senior Systems Engineer, with good management access, should be responsible for analyzing the disposal issues and preparing the project's Waste Management Plan (WMP).

The WMP covers the analysis and recommended design changes to the project that reflect the project goals for disposal and waste management. He/she should also be a member of the EIA interdisciplinary team. The disposal analysis begins with and is concurrent with the concept design of the project. WMP should be included in the Design Concept Review and the Preliminary Design Review. The disposal analysis and waste management is planned for and continues through the life of the project. Waste management processes should include flexibility for dealing with the evolving pattern and concepts for managing waste disposal.

The following steps in disposal engineering and preparing the WMP are:

1. Review the regulations governing the management of wastes and their disposal,
2. Establish an interdisciplinary team of experts on waste management as a resource to the SE responsible for preparing the WMP,
3. Establish the goals for managing the generation and disposal of wastes for this project,
4. Perform a functional analysis to achieve these goals,
5. Establish the requirements and metrics to measure the projects disposal functions,
6. Evaluate the projects disposal functional and performance requirements,
7. Synthesize project design changes to meet the established disposal goals for the project.

In establishing the goals for managing waste generation and disposal for the project there are at least three major elements that need consideration:

1. Handling and disposal of materials used in the project's operations,
2. Waste generated during the fabrication and assembly of the project,
3. End-of-life disposal of the project residuals.

Each of these major elements should be examined in detail as follows:

a) Identify those materials requiring special handling and disposal procedures,
b) Recommend alternative designs and materials to minimize the materials requiring special handling and disposal,
c) Prepare the disposal procedures for materials requiring special handling,
d) Recommend alternatives in the design and use of materials that will promote the reuse and recycling of materials and minimize the disposal and transformation of materials used in project.

The disposal planning has three major schedule milestones:
1) Presentation of the draft WMP,
2) Presentation the draft design recommendations for the project,
3) Presentation of the final WMP with the design recommendations.

Because the WMP can have a strong influence on the design of the project, the draft WMP should be available no later than the Design Concept Review (DCR) for the project and approval of the final WMP should be no later than the Preliminary Design Review. At this stage, the project will include the recommendations from the WMP which will promote the disposal and waste management goals of the project.

Metrics

Each of the system cost/effectiveness analyses may have cost and schedule metrics associated with planning and performing the analyses as well as progress metrics with respect to completion of the analyses. Each type of analysis will also have specific technical metrics related to the topic under analysis.

12 INTEGRATION, VERIFICATION, AND VALIDATION

Figure 12.1 puts the Systems Engineering activities into perspective for the overall project cycle. The "V" depiction shows the relationship of the definition/decomposition/validation process on the left side of the "V" with the corresponding integration/verification process on the right side of the "V". A proper development process will have direct correspondence between the definition/decomposition/validation activities and the integration/verification activities. For every specification, there should be an independent integration and verification activity.

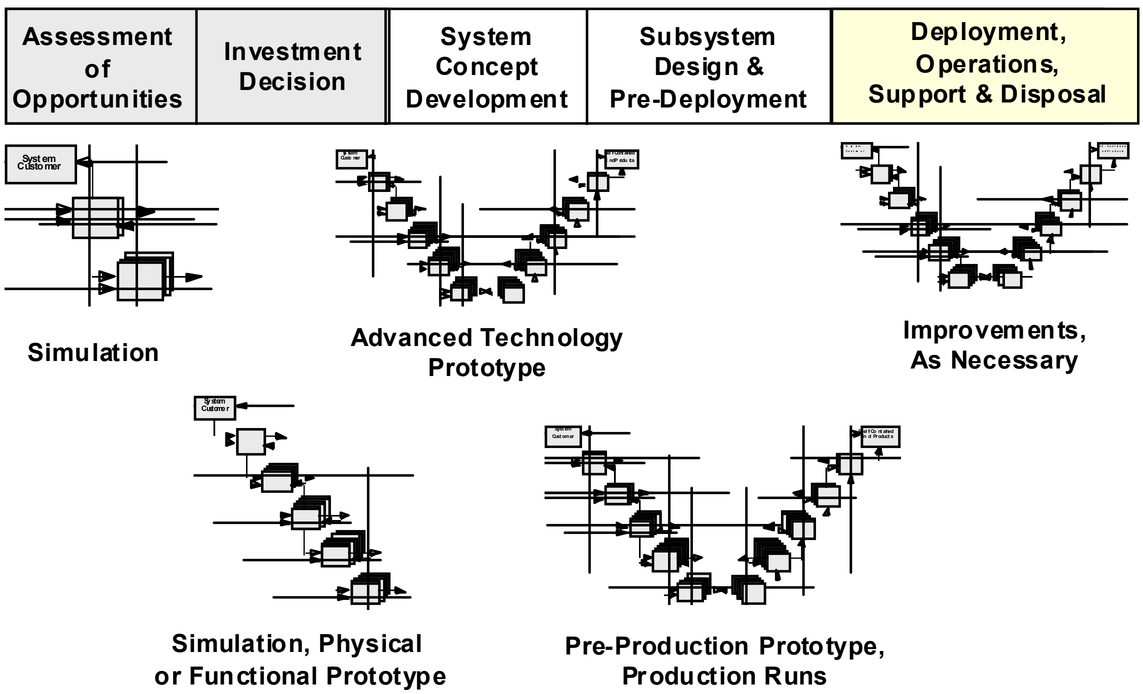


Figure 12-1. System Development "V"s through the Life Cycle

This section will discuss considerations related to System Integration, Verification/Validation Functions and Verification Analysis.

12.1 SYSTEM INTEGRATION

The System Integration (SI) function is to establish system internal interfaces and interfaces between the system and larger program(s). The SI function includes the integration and assembly of the system with emphasis on risk management and continuing verification of all external and internal interfaces (physical, functional and logical).

System Integration is performed on the system and its subsystems, and the system and interfacing external systems. The objective is to ensure that subsystems are integrated into the system and that the system is fully integrated into the larger program.

A discussion of these activities are divided into the internal interfaces among the components and subsystems comprising the system, entitled System Build; and the external interfaces between the system and other systems, entitled System Integration with External Systems.

12.1.1 SYSTEM BUILD

This process addresses the System Integration internal to the system - i.e., the integration of all the elements comprising the system. System build is bottom-up. That is, elements at the bottom of the system hierarchy are integrated and tested first.

Recommended activities

1. Obtain the system hierarchy. The system hierarchy shows the relationship between the system segments and elements, which are structured functionally to form the system. The process begins with a good knowledge of this system structure. In addition to the system hierarchy, obtain the systems and CI design specifications, functional block diagrams, N^2 charts, and any other data which defines the system structure and its interfaces.

2. Determine the interfacing subsystems and components.

3. Ascertain the functional and physical interfaces of the system, subsystems and components within the system. This will require a detailed assessment of the functions flowing in both directions across the interfaces, such as data, commands and power. It will also require a detailed assessment of the physical interfaces such as fluids, heat, mechanical attachments and footprints, connectors and loads.

4. Organize Interface Control Document(s) or drawing(s) to document the interfaces and to provide a basis for negotiation of the interfaces between/among the parties to the interfaces.

5. Work with producibility/manufacturing groups to ensure functional and physical internal interfaces.

6. Conduct internal interface working groups (IFWGs) as required. These would involve all the relevant engineering disciplines. There may be a series of subgroups by discipline, or one group, depending on the size and complexity of the system.

7. Review test procedures and plans which verify the interfaces.

8. Audit design interfaces.

9. Ensure that interface changes are incorporated into specifications.

12.1.2 SYSTEM INTEGRATION WITH EXTERNAL SYSTEMS

This process addresses the System Integration external to the system - i.e., the integration of all the system under development with interfacing external systems.

Recommended activities

1. Obtain system hierarchy, and the systems and CI design specifications, functional block diagrams, N-squared charts and any other data which defines the system structure and its interfaces.

2. Determine the interfacing systems by reviewing the items in 1 above.

3. Obtain interfacing programs' ICDs, SEMPs and relevant interface documents.

4. Ascertain the functional and physical interfaces of the external systems with the subject system. This will require a detailed assessment of the functions flowing in both directions across the interface, such as data, commands and power. It will also require a detailed assessment of the physical interfaces such as fluids, heat, mechanical attachments and footprints, connectors and loads.

5. Organize an Interface Control Document to document the interfaces and to provide a basis for negotiation of the interfaces between/among the parties to the interfaces.

6. Conduct interface working groups (IFWGs) among the parties to the interfaces. These can be one group covering all interfaces for a smaller program, or it can be broken into engineering disciplines addressing the interfaces for larger programs.

   The ICD is developed over a series of meetings/telecons in which the representatives of each side of the interface directly present the performance or needs for their side of the interface. One party takes the lead to be the author of the ICD, and to ensure that copies are available to other parties before a meeting. All parties sign the ICD when agreement has been reached. After the document is signed it is released and comes under formal change control.

7. Review test procedures and plans which verify the interfaces.

8. Audit design interfaces.

9. Incorporate interface changes into specifications.

Metrics
Percentage of released interface drawings.
Number and type of interface issues resolved and unresolved.

Methods/Techniques
Performance of standard configuration management processes will document a concurrent baseline that is consistent with the output of the program. Alternatively, create a baseline document, which contains drawings, specifications, published analyses, and deliverable documents which show the current baseline.

Additionally, ensure that all internal and external interfaces and interactions are included. Interface working groups (IFWG) are established to review interface statements/drawings, and are a good means of ensuring direct interaction of all parties to the interface, as discussed above.

Tools
N^2 charts; Traceability database; and SEMP

12.2 VERIFICATION/VALIDATION FUNCTIONS
System verification and validation activities are very similar, but they address different issues. Verification addresses whether the system, its elements, its interfaces, and incremental work products satisfy their requirements. Validation confirms that the system, as built (or as it will be built), will satisfy the user's needs. Verification ensures the conformance to those requirements, and validation ensures the requirements and the system implementation provide the right solution to the customer's problem. (ANSI/EIA-731). In other words, verification ensures that "you built it right;" while validation ensures that "you built the right thing."

Overview of Verification

Verification is the tasks, actions and activities performed to evaluate progress and effectiveness of the evolving system solutions (people, products and process) and to measure compliance with requirements. Analysis (including simulation, demonstration, test and inspection) are verification approaches used to evaluate: risk; people, product and process capabilities; compliance with requirements, and proof of concept.

The primary function of verification is to determine that system specifications, designs, processes and products are compliant with top-level requirements that spell out customer (internal or external) expectations of the capabilities, performance and characteristics of the developed system and that the processes by which these are developed have adhered to the order and content called out in the SEMP). This serves to assure that the system ultimately developed satisfies the expressed expectation of the customer and that development procedures have been carried out in accordance with plans. As segments and sub segments of the system under development are iteratively allocated, specified, designed, simulated and tested, a hierarchical sequence of specifications, designs and test plans, appropriate to each phase of the development process, is produced. It is intended that verification be an integral part of incremental, subsystem, software and process specifications, and major reviews.

A secondary function of verification is to determine by means appropriate to the level, and to document that the system and subsystem representations at each level are fully compliant with the specifications and requirements in effect at the preceding level. This acts as a guarantee that, as each phase of the development process is completed, the next phase can be executed without omitting desired system properties or embarking on erroneous development step(s) which would cause this and/or subsequent levels of development activity to be substantially redone at some later stage.

Overview of Validation

The hardware and software are validated at the system integration level. This is a step beyond the software and hardware verification processes. Validation is interpreted as the validation of the design and implementation to the requirements, utilizing mission-type hardware to the extent possible. Validation is a determination that a system does all the things it should and does not do what it should not do. Validation is often performed by an independent third party, beyond the developer and customer. Validation may be performed in the operational environment or a simulated operational environment.

A form of validation sometimes used referred to is “requirements validation.” This is conducted to provide early assurance that the requirements are the “right” requirements for guiding the development process to a conclusion which satisfies the customer or system users in its intended environment. Requirements validation is often based on requirements analysis; exploration of requirements adequacy and completeness; assessment of prototypes, simulations, models, scenarios, and mockups; and by obtaining feedback from customers, users or other stakeholders.

Validation activities may include only the product or it may include appropriate levels of the product components that are used to build the product. The functions performed for validation are similar to verification tasks, such as test, analysis, inspection, demonstration, or simulation. End users and other stakeholders are usually involved in validation activities. Both validation and verification activities often run concurrently and may use portions of the same environment. Most of the discussion of verification below can be applied to validation.

The objects of validation are the designs, prototypes, and final products, as well as the documentation which describe the system. It is intended that validation be an integral part of incremental, subsystem, software and process specifications, and major reviews.

Objective

The objective of the verification process is to ensure conformance of the implemented product and processes to all source and derived requirements and that the planned development process has been followed.

The objective of the validation process is to ensure the implemented product functions as needed in its intended environment, including its operational behavior, maintenance, training, and user interface.

A. Participation

The key groups participating in the verification process are Systems Engineering, design engineering, test engineering and where appropriate manufacturing, reliability and maintainability. It is the responsibility of Systems Engineering to assign an individual/group to conduct/oversee the steps of the verification process.

B. Recommended Activities

Three different verification activities are presented to illustrate the work to be done.

C. End Result

Successful verification and validation confirms that the development process has provided a system consistent with customer expectations. Additionally, verification provides safeguards so the development project does not backtrack. It also ensures a development product and processes that meet the applicable functional, behavioral, timing, weight, power, performance, configuration, reliability, and maintainability requirements.

12.2.1 Product Verification – Recommended Activities

1. In conjunction with project management, the Systems Engineering verification designee will establish for each phase the customer-supplied documents and the development process work products that will allow the verification process to be performed in a timely and effective manner. A schedule should be developed for both the baseline document availability and the verification process (both verification and documentation of verification) such that all appropriate scheduled reviews undergo a verification process. As part of this process, prior to performing the verification activities specified, the following items (as they apply) are to be determined and documented:

   a. All equipment - system or subsystem(s) to be exercised, stimulus, measurement and recording devices, computers, software, test scenarios and/or written operator instructions needed for verification

   b. Input stimuli required to perform verification

   c. Means to record or document system or subsystem(s) response

   d. Criteria for successful verification

2. The verification process should include specification which of the acceptable means (analysis, demonstration, test, and inspection) will be applied to each development activity requiring verification. The specification of the verification method, defined in Section 12.3, should include the rationale for the selection of the specific means chosen.

   It is the intent of the verification process that, should the sequence of verification activities be repeated, similar results are obtained.

3. The verification process is subject to the same audit requirements as any other part of the development process. As such, it should be possible to determine, via a documented audit trail, the steps that were taken to assure that the verification process was followed and that the verification decisions were sound.

4. As a result of the verification process, there should be an appropriate document for each verification process describing the requirement to be met, the documents (baseline) submitted to determine that the requirement has been met, the means used, appropriate analysis, test, simulation or observation support for the conclusion reached, names of individuals involved in the verification process and the decision reached on verifiability of the subject item(s).

   The verification process for each review should be completed prior to conducting the review. The results of the pertinent verification process(es) should be introduced as a formal part of the review. In the event that a verification process fails to yield desired, acceptable or anticipated results, the steps, resources and timetable to arrive at successful verification should be a documented output of the review process.

12.2.2 PROCESS VERIFICATION – RECOMMENDED ACTIVITIES

1. In conjunction with project management, the Systems Engineering verification designee will establish for each phase the customer-supplied documents and the development process work products that will allow the verification process to be performed in a timely and effective manner. A schedule should be developed for both the baseline document availability and the verification process (both verification and documentation of verification) such that all appropriate scheduled reviews undergo a verification process. Prior to performing the verification activities specified, the criteria for successful process verification should be determined and documented.

2. For process verification the verification process should include specifying which of the acceptable means (analysis, demonstration, test, and inspection) will be applied to each process activity requiring verification.

3. The verification process is subject to the same audit requirements as any other part of the development process. As such, it should be possible to determine, via a documented audit trail, the steps that were taken to assure that the verification process was followed and that the verification decisions were sound.

4. As a result of the verification of the process, there should be an appropriate document describing the requirement(s) to be met, the documents (baseline) submitted to determine that the requirement(s) have been met, the means used, appropriate method to support the conclusion reached, names of individuals involved in the verification process and the decision reached on verifiability of the subject process.

The process verification may consist of review of a process description by a integrated product team. It may also include a demonstration to an IPT of a process.

12.2.3 MANUFACTURING PROCESS VERIFICATION

Manufacturing Process Verification is a special case of Process Verification. The steps needed to perform manufacturing process verification are similar to those of Process Verification described above. The Systems Engineering role in Manufacturing Process Verification is substantially the same as for Process Verification generally.

Input

System hierarchy including segments and elements and their position in the hierarchy.
System architecture and internal interfaces

Output

Interface control documents for the interfaces of elements comprising the system

Criteria for Successful Completion

Signatures of interfacing parties on the Interface Control Documents (ICDs).
Update of specifications documenting interfaces reflecting this task.

Metrics

Percentage of released interface drawings
Number and type of interface issues resolved and unresolved.

Methods/Techniques

Performance of standard configuration management processes will document a concurrent baseline that is consistent with the output of the program. Alternatively, create a baseline document, which contains drawings, specifications, published analyses, and deliverable documents that show the current baseline. Also ensure that all internal and external interfaces and interactions are included.

Tools

Functional analysis tools (e.g., N^2 charts, functional flow diagrams, IDEF0/1 diagrams); Concurrent Engineering tools; and Traceability database; and SEMP

12.3 VERIFICATION ANALYSIS

Verification analyses should be conducted to support the development of products, services, and processes necessary to verify that system end-items satisfy their requirements. Verification analyses should address verification requirements and criteria for solution alternatives; definition of verifications to demonstrate proof of concept; and development, qualification, acceptance and pertinent operational, and other testing. These analyses should also consider the requirements and procedures needed to verify critical verification methods and processes (e.g., verification of key methods and assumptions and data used in verifications by analysis).

Verification analysis can/should be initiated when a design concept has been established. The verification analysis may be drawn from the Test and Evaluation Master Plan (TEMP), and supports its development. The objective is to define all verification activities that will demonstrate the system's capability to meet the requirements of its specification. These activities must be fully integrated to insure that adequate data will be provided at minimum cost, within the allotted time frame. A continuing feedback of verification data throughout product development, test, and evaluation is necessary to reduce risk and to surface problems early. The goal is to completely verify system capability to meet all requirements prior to production and operational use. Basic verification activities are:

Inspection (I): 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, etc.).

Analysis (A): 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.

Analysis (including simulation) may be used when such means establish that the appropriate requirement, specification, or derived requirement is met by the proposed solution.

Demonstration (D): a qualitative exhibition of functional performance, usually accomplished with no or minimal instrumentation. Demonstration (a set of test activities with system stimuli selected by the system developer) may be used to show that system or subsystem response to stimuli is suitable. Demonstration may be suitable when requirements or specifications are given in statistical terms (e.g., - locate the position of a transmitter up to 5000 meters from the base station with an accuracy of plus or minus 3 meters in the x or y direction 95% of the time, mean time to repair, average power consumption, etc.).

Test (T): 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.

In commercial programs, a fifth verification method is often used: certification. This refers to verification against legal and/or industrial standards by an outside authority without direction to that authority as to how the requirements are to be verified. For example, this method is used for CE certification in Europe, and UL certification in the US and Canada. Note that any requirement with a verification method of “certification” is eventually assigned one or more of the four verification methods listed above.

Verification should be performed throughout the life cycle to assure the system is “on track” and likely to meet its end requirements. It is important to perform verification early when development decisions can have great impact on the system’s life cycle. During testing phases, verification tests are performed incrementally, as required. In general, a distinct verification test is appropriate for each distinctive level of specification.

It is highly desirable that system performance be established by test under actual (or simulated) operating conditions. This may not, however, be possible until the system is deployed. Problems uncovered at that stage are very costly to correct, and a combination of inspection, analysis, and test is therefore often employed during project development to surface problems early. This reduces risk and helps insure a successful, low cost program.

The design of the verification project is usually accomplished in the Program Demonstration & Risk Reduction project phase. The effort involved is therefore a matter of choosing the most cost-effective mix of simulations and physical testing, and integrating test results to avoid unnecessary redundancy. Complete simulation of the system (both performance and design) has become common-place in major system development, and has resulted in reduced development time and cost. However, the assumptions upon which the simulations are developed must be fully verified to insure that resulting outputs will accurately represent actions of the system.

The basis for the verification program is the requirement statements contained in the system, segment, element, or subsystem specification. Each requirement should be given unique identifier and listed in a Verification Cross Reference Matrix with the method of verification identified, together with the category of test employed and its level. The unique identifier can be used for traceability to the test plans, test procedures, and test reports to provide a closed loop verification process from demonstrated capability back to the requirement.

The basic test categories are:

Development: Conducted on new items to demonstrate proof of concept. Testing may be done on breadboard, brassboard, engineering prototype, or partial model. Often used to reduce risk and prove feasibility. Some considerations for performing development testing are shown in Figure 12-2.

Qualification: Tests are conducted to prove the design on the first article produced, has a predetermined margin above expected operating conditions, for instance by using elevated environmental conditions for hardware. The hardware qualification test items cannot generally be used in an operational test due to overstress.

Acceptance: Conducted to prove workmanship and materials on the second and succeeding articles. Tests conducted are a subset of the qualification tests, performed at lower stress levels. Some considerations for acceptance testing are shown in Figure 12-3.

Operational Tests: Conducted to verify that the item meets its specification requirements when subjected to the actual operational environment. Some requirements, such as radiation hardening, may be fully verified at the parts level by testing. Many requirements at the system level may be verified only by simulation (supported by test data from lower levels).

	APPROACH	CONSIDERATIONS	TESTS
1.	Identify Problems Early	Specification Requirements Advanced Technology	System Compatibility Interface
2.	Risk Reduction	Design Maturity	Performance
3.	Develop Packaging & Fab. Techniques	Schedule, Costs Loads Data	Power Under/Over Voltage
4.	Establish Confidence Margins & Failure Modes	Mission Criticality Reliability EMI	Exceed Design Limits Life, Environments
5.	Develop H/W and S/W Simulations	Establish Analytical Model	Operating Parameters

Figure 12-2. Development Test Considerations

	HIGHLIGHTS	CONSIDERATIONS	TESTS
1.	Formally Demonstrate Hardware Acceptability For Delivery	To Specification	Functional Critical Parameters Redundancy
2.	Detect Workmanship, Material, and Quality Deficiencies	All Hardware	Environment • Thermal-Vacuum • Dynamic • Leak
3.	Determine Infant Mortality Failures	Pass/Fail	Burn-In

Figure 12-3. Acceptance Test Considerations

In developing the most cost-effective verification program, a number of trades must be considered, as shown in Figure 12-4. The objective in conducting these trades is to achieve a minimum cost verification program while maintaining low risk.

TRADEOFF	RISK FACTORS		POTENTIAL IMPACT
1. Verification Method: Analysis & Simulation vs. Test	Weight Increase -No Demonstrated Ultimate Capability	vs.	Cost, & Schedule Growth
2. Test Levels of Assembly: Part, Board, Component vs. Vehicle	Failures Detected Late at Vehicle Level	vs.	Cost, Spares, Availability, Schedule
3. Software Validation: Early Using Simulators	Early Maturity of Software Program	vs.	Cost, Schedule

Figure 12-4. Verification Approach Tradeoff Considerations

The hardware and software test plans are identified on a Test Plan Tree similar to the Specification Tree described in Section 8. The verification program should be defined in the SEMP, and detailed in a separate System Verification Plan. Since much of the verification on large systems will be accomplished by means of simulations, this is a critical document. It must identify how the input data to the simulations will be obtained and validated. The simulations themselves must be fully documented, including a description of the model employed, the assumptions made in its development, and the means of verification of those assumptions. Test cases having a known outcome must also be defined to demonstrate that the simulation accurately represents the system design (or portion thereof).

The development of personnel capabilities is usually covered in a Training Plan. Training needs are established by Human Engineering personnel who develop task descriptions, operational sequence diagrams, and evaluate the man-machine interface to establish the human interactions with the hardware and software. Verification analysis insures that tests have been established using realistic scenarios to demonstrate human reaction times satisfy mission requirements. Maintainability demonstrations must also be planned to insure a sufficient number of tests and problem areas to provide a high confidence level of meeting maintainability parameters (Mean-Time-To-Repair).

It is also important that processes that are new or have not been previously applied to this application be verified before any production or testing is attempted. Tests must be devised to demonstrate capability and repeatability of results.

APPENDICES

THIS PAGE
INTENTIONALLY
LEFT BLANK

APPENDIX A - QUALITY FUNCTION DEPLOYMENT (QFD)

A.1 INTRODUCTION

Quality Function Deployment (QFD), sometimes called The House of Quality, is a requirements flowdown technique developed by the Japanese. Reportedly, since all Japanese engineers and technicians were not fluent in the languages of the countries with which they wished to do business, QFD was developed as a process whereby customer requirements and specifications could be quickly translated into an actionable format. It is said to have started in a Japanese shipyard that wanted to bid on modifications to old Liberty ships after W.W.II, but the engineers could not understand U.S. drawings.

QFD is a technique for deploying the "Voice of the Customer." It provides a fast way to translate customer requirements into specifications and systematically flowdown the requirements to lower levels of design, parts, manufacturing, and production.

A.2 PROCESS DESCRIPTION

The process starts on the left hand side of Figure A-1, with entries into the "What?" column. This is where the key system requirements (Voice of the Customer) are entered. Some examples of "Whats" at the system level for a new automobile might be:

- Top speed 120 mph
- Endurance 300 mi. (55 mph @ sea level)
- World Class passenger comfort & convenience
- Beautiful exterior & interior appearance

The features to be specifically implemented in the design (the "Hows?") are listed in the vertical columns (write sidewise). Entries into these columns should be the primary features planned to achieve the "Whats" in the left column. For example, to address the system requirement of a 120 mph top speed, the key planned features might be:

- Large displacement engine
- Low aerodynamic drag
- Lightweight vehicle

World Class passenger comfort & convenience could be a system requirement (What), but an entire QFD diagram might be required to flow down the key features, because it is such a broad requirement. Another approach would be to view World Class passenger comfort & convenience as a higher-level objective and list key features on the system-level chart. This sometimes permits better utilization of the benefits of the QFD diagramming process. Some key features for passenger comfort & convenience include:

- 30 db External noise attenuation (Specified frequency spectrum)
- 20 db Shock and Vibration attenuation (Specified frequency spectrum)
- Easy entry & egress (Flowdown to Headroom, Legroom, Door pull, etc.)

- **A PLANNING TOOL FOR TRANSLATING CUSTOMER REQUIREMENTS INTO SPECIFICATIONS**
- **DEPLOYS THE "VOICE OF THE CUSTOMER"**
- **A FAST, SYSTEMATIC WAY TO DERIVE & FLOW DOWN REQUIREMENTS TO DESIGN, PARTS, MANUFACTURING, AND PRODUCTION**

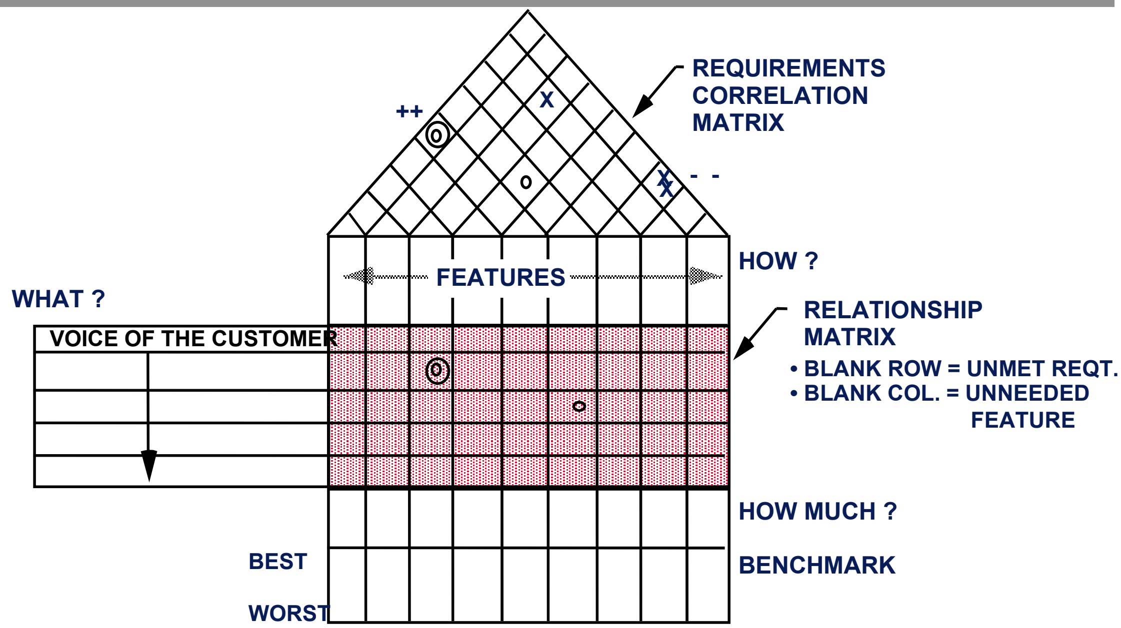


Figure A-1. Quality Function Deployment (QFD); The House of Quality

The shaded Relationship Matrix shows the correlation between features and requirements. Two concentric circles (double circle) are used to indicate a strong correlation between the feature and the requirement. A modest contribution is indicated by a single circle. A blank column indicates an unnecessary feature relative to the listed requirements. Similarly, a blank row indicates an unaddressed requirement.

At the bottom of the features column (the Hows) is a row for "How Much?". This is where the design features are quantified. For example, under the above feature "Large Displacement Engine", 3 Liters might be added to quantify the engine displacement. Similarly, 0.31 drag coefficient could be entered under "Low aerodynamic drag," and 3,000 lb. empty weight under "Lightweight vehicle". Each of these numerical requirements might be the product of extensive system analysis and tradeoff study to determine how best to meet system requirements (Whats).

The "How Much" data is then "Benchmarked" against the competition in the next row. In Benchmarking, the same features of competitor's models are surveyed and ranked on a scale from best to worst. The "How Much?" is then plotted on that scale to show how the design compares to the competition across all its key features. Obviously, to be "World Class", the design should be nearly the best in all features.

The terminology "House of Quality" comes from the requirements (features) correlation matrix shown at the top of the diagram. In this matrix, the features are compared against all other features to indicate if they are supportive (correlate) or in opposition. This correlation matrix gives the system engineer important information to use in requirements balancing. For example, if three features are positively correlated in addressing one or more customer requirements, the system engineer could perform a cost/effectiveness study of the best combinations of those features to meet the specific requirements. Perhaps the highest cost/lowest contribution feature could be reduced or eliminated.

Features that are highly correlated are also shown with a double circle. Features that are strongly negatively correlated are shown with a "XX", or, for moderate negative correlation, an single "X". These negative correlations indicate key tradeoffs that should be performed by the system engineer. For example, one feature for passenger convenience and comfort might be "Easy opening doors, with less than ten pounds pull required. However, another feature for passenger comfort is tight sealing doors - to minimize rain leakage and road noise. If these features required twenty pounds or more pull to open the doors, a tradeoff study could be conducted to find the best way to meet all requirements - including alternate designs.

The QFD charts usually become quite complicated when completed. Sometimes they are a real eye test. Nevertheless, they contain a tremendous amount of information, all on one page. They are a wonderful resource for the systems engineer. The matrix should not exceed about 30 x 30. At the system level, it is recommended that the top ten to fifteen requirements and the top ten to twenty features be displayed.

In contracting with the U.S. government, through agencies such as the DoD and NASA, written specifications are still required, so QFD represents ***additional*** work for the engineer. In America, we still prefer the precise language of specifications as a basis for contracts. QFD may not replace specifications in the near term, however it is a very convenient and useful methodology for displaying top-level requirements and design features, and flowing them down to lower levels in the design and manufacturing process.

A.3 QFD FLOWDOWN

QFD flowdown is shown in Figure A-2, where the "How" and "How Much" from the higher level become the "What?" input for the next lower level. The process is then repeated. In a complex system, three or four tiers of QFD flowdown may be required at the design level to flowdown requirements to actionable levels. This process can be continued for parts, manufacturing, and associated processes.

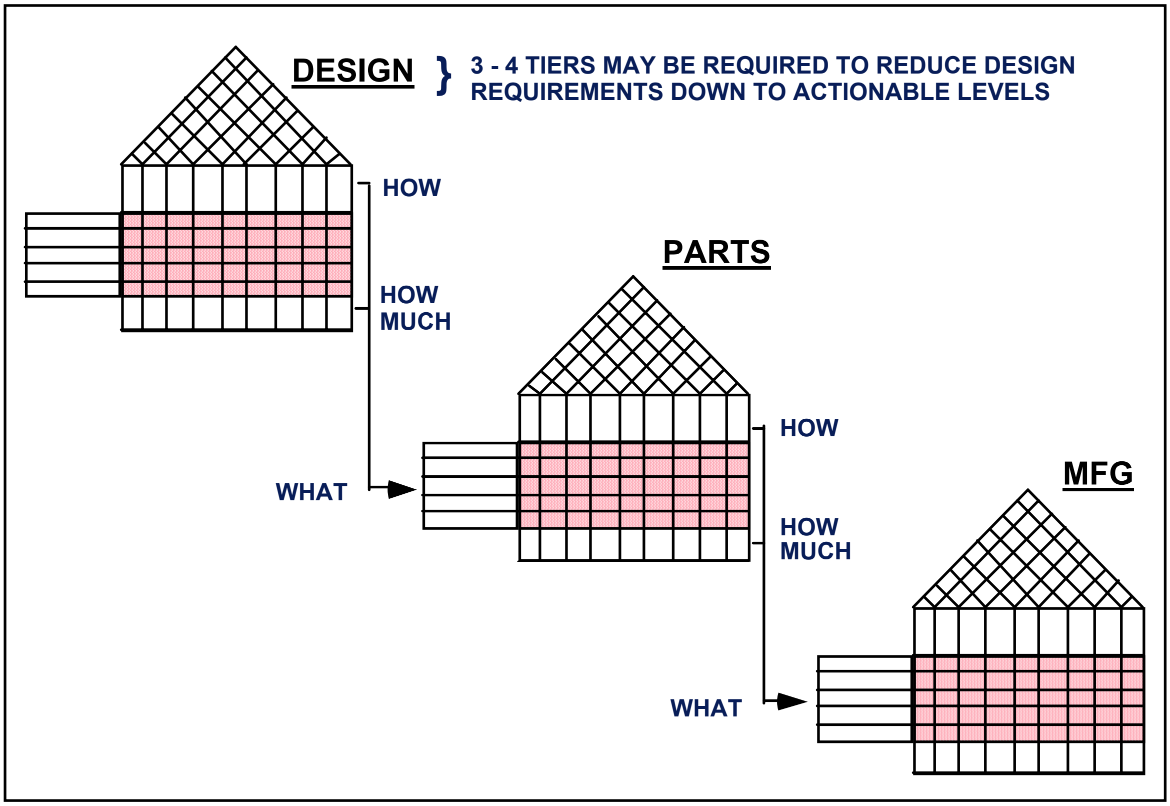


Figure A-2. QFD Flowdown

Since the Japanese use the QFD process extensively, a set of QFD charts for a new automobile design might be several inches thick. Until we in America become more comfortable with the process, it is suggested that system engineers use QFD at least to display the *key* requirements and *features* at any level of the system hierarchy. Management must be sensitive to the problems of requiring full engineering documentation in <u>both</u> specifications and QFD. This documentation load could be counter-productive.

APPENDIX B - HUMAN SYSTEMS ENGINEERING

B.1 INTRODUCTION

"Human" Engineering specializes in the interaction of the human with other humans and with the system. The "Human" engineer specializes in job and task design between the human, others and the system. The primary goal of the "human" engineer is to determine the performance of the human in order to optimize the performance of the overall system. This appendix describes significant interactions that occur between human engineers and systems engineers during system development. These interactions include information that must be shared, decisions that must be made, and actions or decisions that require approval. This information is not intended to serve as an overall description of Human Engineering (HE) or Human Systems Engineering (HSE) or as a "how-to" guide to perform human engineering tasks. It is intended to provide guidance for the systems engineer in the integration and understanding of human engineering efforts. The information in this appendix was developed from task analyses of both systems engineering and human engineering. The task analyses were completed to support the development of the Manning Affordability Initiative's (www.manningaffordability.com) Human Centered Design Environment (HCDE).

An effort has been made to describe the interactions in a stand-alone manner that does not require familiarity with any specific systems engineering or human engineering process. However, it should be noted that the perspective taken is generally from the systems engineer's point of view. Throughout the descriptions, the terms "systems engineer" and "human engineer" are used. Although these are the singular forms, the terms could equally be pluralized or described as engineering teams. Many definitions exist for what qualifies a person to be labeled a "systems engineer," but within the context of this appendix some specific criteria apply (see Figure B-1). Due to either past training and experiences of the systems engineer, or to the type of system under development, the systems engineer may be focused on a particular area of the process, such as software engineering or requirements analysis. But the systems engineer is the individual who has responsibility for the design of the system as a whole. The systems engineer may have a very active role in the definition of requirements or system functions, but his or her responsibilities change during the physical design of the system. At this point, the purpose of the systems engineer is that of an integrator, and he or she is responsible for combining and deconflicting proposed designs submitted by engineers who specialize in particular disciplines or are responsible for particular subsystems. The human engineer plays one of these roles. The human engineer specializes in job and task design and the interaction of humans with one another and with automation, and his or her responsibility covers the human subsystems within the system to be designed.

The next section of this appendix briefly discusses what appear to be the most significant ways in which systems engineers and human engineers can interact. The main section describes the interactions in greater detail. The description of each interaction includes references to how these interactions relate to the systems engineering processes described in IEEE 1220-1998, the Standard for Application and Management of the Systems Engineering Process, and in EIA-632, Processes for Engineering a System.

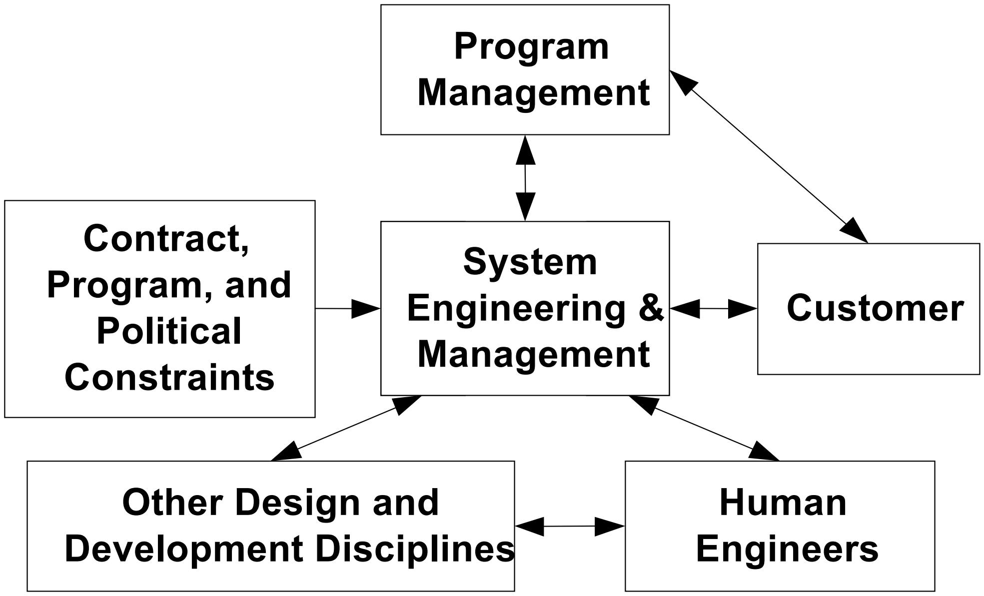


Figure B-1. Context of Interactions between the Systems Engineer and the Human Engineer.

B.2 SIGNIFICANT INTERACTIONS

Based on the interactions described in this report, four overarching interactions or themes have been selected as significant.

- Scenario Definition and User Review
- Participation in Function Analysis
- Function Allocation Decisions
- Compatibility of Models

These interactions are not meant to represent the bulk of the human engineer's work; they are intended to represent the most important ways in which the human engineer must interact with the systems engineer or other designers. The interactions do not necessarily represent what is currently planned or carried out in system development, but they instead represent key interactions through which human engineering can be better integrated within systems engineering. Although the level of human engineering participation will vary with different design stages, the human engineering team should have end-to-end involvement in the system development process.

B.2.1 SCENARIO DEFINITION AND USER REVIEW

The human engineer is often required to extend previous scenarios or build new scenarios in order to identify and provide details about how the operators and users interact with the rest of the system. Different phases or modes of operation can be described, and scenarios may cover both typical conditions and worst-case situations. While many scenarios used in system development or testing may only cover conditions and events external to the system, the human engineer is more interested in scenarios that describe how the system will respond and operate. Scenarios that describe only events and conditions external to the system can be expanded to include system operation and functionality from the perspective of the user.

These scenarios are used to build task and job analyses for the operators and users and to test designs and procedures. Since these scenarios are written from the perspective of the users and operators, they can be excellent vehicles for soliciting feedback during user reviews. Scenarios can be simply represented as written descriptions or storyboard sequences and therefore, they can be used in early stages of system development. The detailed inner workings of the hardware and software do not need to be defined because such details are irrelevant from the user's perspective. Reviewers such as potential users typically are able to provide better and more detailed feedback from a descriptive scenario than from a list of requirements or functional description.

The review of user-centered scenarios with representative users or other appropriate individuals can provide feedback on the system's physical design, functional capabilities, or even performance requirements. Without this sort of review, the system engineer can only assume that the system's requirements are compatible with the needs and limitations of the users or operators.

The human engineer is typically the designer who is best suited to perform user reviews of scenarios and system designs. To allow scenarios to be used in this way, the human engineer must have scenarios that accurately represent the operation of the system. The human engineer must also be prepared to collect feedback on issues such as requirements and system functions in addition to control and display configurations. Understanding of these issues, such as how the system is intended to work and the proposed tasks for users and operators, is essential to comprehend the human roles in the overall operation of the system. With adequate interaction between the human engineer and the systems engineer, scenarios and user reviews can allow for early and rapid feedback on system requirements, functions, and designs.

B.2.2 Participation in Function Analysis

Since the decomposition of functions and definition of the functional architecture is largely performed without regard to the allocation of the system's functions, it may be seen as an area that requires little if any human engineering participation. There are, however, two distinct reasons for human engineering participation that can reduce the potential for having to change the function analysis at a later date. First, the human engineer can assist in identifying functions that must be included because of the presence of humans within the system. Some functions, such as life-support or communications, may be required regardless of the humans' assigned responsibilities. Other functions will become apparent once some preliminary allocations are made, including those allocations that may be assumed from the system's initial concept of operations. Second, much of the human engineer's later work in task design and analysis will be driven by the results of the function analysis. Any information on the timing, sequence, or interaction of functions can be highly useful in the design of human tasks and jobs. Timing and overlap of tasks will influence workload, and unpredictable task sequencing can greatly increase cognitive performance. Without human engineering participation, the function analysis is likely to contain insufficient details for functions and subfunctions to be optimally allocated to humans. The human engineer is then left to make potentially incorrect assumptions about the information or to continue the function analysis.

B.2.3 Function Allocation Decisions

Since accurate allocation of functions to system elements requires consideration of the capabilities and limitations of humans, the participation of the human engineer is essential. The human engineer can provide reasonable estimations of what functions or portions of functions should and should not be allocated to humans. Until functions and subfunctions have been defined to significant detail, most functions will be allocated to "combinations" and not "fully manual" or "fully automated," but the human engineer can help to describe how the human and technology can interact to accomplish the function optimally.

The systems engineer and other participants in the function allocation process are likely to have a good idea of the capabilities and limitations of humans in general, but the human engineer is likely to know more about the specific capabilities and limitations of the intended users. The earlier this participation occurs, the better the result is likely to be, as it can prevent improper decision decisions that are costly or impossible to change at a later date. The human engineer can assist in identifying functions or portions of functions that are required to have a human or non-human allocation. Reasons for such decisions include functions that are beyond the capabilities of the anticipated users, assumptions made as part of the system's initial concept, and grouping of functions that will benefit job design. Making these mandatory allocations as early as possible helps define the system in greater detail and also prevents these allocations from being made to the wrong system element or component.

B.2.4 Compatibility of Models

Proposed designs of systems, subsystems, or components can be evaluated before the system is constructed through accurate modeling. Although often limited in scope or detail when compared to models of other disciplines, human engineering models can provide useful information about how humans interact with one another or with the rest of the system. Such models can help the human engineer optimize the performance of humans within the system. The main goal of the human engineer, however, should be to determine the performance of the human in order to optimize the performance of the overall system. To accomplish this, the human engineering models need to be compatible with other models used in the design of the system. Compatibility can permit both the interoperability of different models and the extension of existing models by the human engineers. Without such compatibility, the human engineering models will not include an accurate representation of the system's hardware and software. Model compatibility is required for the human engineer to produce accurate models of human performance and to be able to model how human performance impacts the performance of the overall system.

B.3 INTERACTION DETAILS

This section outlines all of the systems and human engineering interactions uncovered from task analyses of the two processes. Each interaction begins with contextual information to characterize the design process at the time of the interaction. Additional detailed information about the interaction follows, as well as the implications for the process. Finally, references to IEEE 1220-1998 and EIA-632 are provided.

B.3.1 Mission Analysis

The mission analysis phase of system development includes the determination of the overall system capabilities and the system's mission or purpose. Scenarios or mission profiles are created. The boundaries of the system need to be identified, as do the interactions of the system with its environment and with other external systems.

B.3.1.1 Selection of Comparison Systems

A frequently used approach in system development is comparison of the system under design to predecessor systems. All or part of the current system may be compared to all or part of some previous system that served a similar function, had a similar goal, or included similar components. Although it may be informal or even unintentional, some comparison is performed any time the developers have prior experience with the development or use of similar systems. The human engineering practitioner may observe or otherwise analyze the performance of the comparison systems to establish design goals or human performance requirements. Among the different types of data that may be collected are historical data, observational data, user data or feedback, and data from experimental prototypes. Information on past performance of multiple comparison systems may be used to select or narrow options for designs.

While the comparison systems must be similar to the current system in either mission or implementation, a system that is useful to the human engineer may not be useful at the overall system integration level. The human engineer, however, should address systems selected by the systems engineers or others as a baseline for comparison. Systems or subsystems that the systems engineer considers relevant for the human engineer must be assessed by the human engineer to confirm their similarity and applicability to the system under design. An early identification of comparison systems will allow the subsequent recommendations to have a more effective influence on design decisions.

IEEE 1220-1998:	6.1.2 – Define project and enterprise constraints
	6.1.3 – Define external constraints
EIA-632:	Requirement 4 – Process Implementation Strategy
	Requirement 13 – Information Dissemination

B.3.1.2 SYSTEM USE SCENARIOS

Tools such as system scenarios, design reference missions, and mission profiles or timelines are used by a variety of disciplines during system design. Information from these sources can be used to identify required interactions with external systems, determine functional requirements for a system, and establish performance requirements for interaction with external systems. Once designs are complete, such scenarios and timelines may be used to evaluate or validate system design options.

In order to adequately account for the users and operators of the system under development, some scenarios must be defined from their perspective. System use scenarios describe, from the user's point of view, detailed events of the system mission, including identification of mission phases, mission time scale, and events external to (and their interactions with) the system. Scenarios from the user's perspective are powerful tools for eliciting user or subject matter expert feedback early in the design process.

System use scenarios defined by the human engineer will often be extensions or subsets of scenarios developed or approved by the systems engineer. The definition of system use scenarios will typically require assumptions on the part of the human engineer that further define the system. These scenarios, therefore, should be either approved or at least reviewed by the systems engineer. The human engineer must ensure that the scenarios accurately reflect a potential or achievable design.

Without realistic and valid system use scenarios, it will be difficult, if not impossible, to account for users and operators in the design process. As scenarios extend assumptions about system design, those assumptions must be verified or accepted by other disciplines.

IEEE 1220-1998:	6.1.4 – Define operational scenarios
EIA-632:	Requirement 4 – Process Implementation Strategy
	Requirement 16 – System Technical Requirements
	Requirement 24 – Risk Analysis

B.3.1.3 USER ENVIRONMENT CHARACTERISTICS AND EFFECTS

The design of the system must account for the environmental conditions under which the system will be employed. Once the conditions are identified, the effects of those conditions and any resultant design constraints should be ascertained.

The human engineer will need to assess the environmental conditions and determine whether or not all conditions that significantly affect humans have been identified. The human engineer will need to quantify the effects of environmental characteristics on human performance and provide the data to the systems engineers and other design disciplines for use in design decisions. In some cases, the human engineer will need to determine how to mitigate, eliminate, or compensate for environmental effects. As more of the system's physical design is completed, additional induced environmental factors will become apparent or better defined. The human engineer must therefore iteratively review system designs to continue to identify induced factors and determine how external environmental factors may affect humans.

Once the effects of environmental factors have been assessed, it must be determined whether or not desired levels of system and human performance can be achieved. In some cases, the performance effects of the environment will need to be included in system or component models and simulations.

IEEE 1220-1998:	6.1.8 – Define utilization environments
EIA-632:	Requirement 16 – System Technical Requirements
	Requirement 24 – Risk Analysis

B.3.2 REQUIREMENTS ANALYSIS

During requirements analysis, source requirements are identified, clarified, and prioritized. The requirements are broken down or decomposed into greater detail. Each lower-level requirement must be traceable to higher-level requirements. As the requirements are defined in greater detail, they will become more specific to the planned implementation of the system, and the involvement of designers within different disciplines becomes necessary.

B.3.2.1 HUMAN ENGINEERING CONSTRAINTS

Constraints are implied requirements that restrict the design of a system. They are not created directly from a specification, but they are instead the result of external limitations.

Some constraints will arise due to design decisions or analyses by the human engineer. Many constraints will come from the inherent limitations of humans in general, such as sensory capabilities, endurance limits, and strength. Once the characteristics of the user population become more certain, other constraints may become apparent. As they arise, these constraints must be identified and passed on to other design disciplines. In some cases, constraints from different disciplines must be developed and documented in parallel, requiring collaboration between design disciplines.

IEEE 1220-1998:	6.1.2 – Define project and enterprise constraints
	6.1.3 – Define external constraints
EIA-632:	Requirement 5 – Technical Effort Definition
	Requirement 14 – Acquirer Requirements
	Requirement 15 – Other Stakeholder Requirements
	Requirement 16 – System Technical Requirements
	Requirement 19 – Specified Requirements

B.3.2.2 HUMAN PERFORMANCE REQUIREMENTS AND HUMAN ENGINEERING DESIGN REQUIREMENTS

During requirements analysis, requirements from a variety of sources and disciplines must be analyzed to resolve conflicts. The human engineer is primarily responsible for two types of requirements, human performance requirements and human engineering design requirements. Human performance requirements include times and accuracies for tasks assigned to humans. The human engineer must ensure that the proposed requirements are in fact achievable by the intended operators and users. The human engineer may in some cases, define the human performance requirements based on external requirements, specifications of other system components, or the capabilities and limitations of the prospective operators and users. The human engineering design requirements concern specific aspects of the hardware and software that are necessary to fit the operators and assist them in their assigned tasks. These requirements define what must be designed and constructed to permit the operators and users to interact with one another and the rest of the system.

Human performance requirements are frequently derived from or at least bounded by other performance requirements within the system. The accuracy, response time, and other attributes of the operator tasks will affect similar attributes at the system level. The requirements produced by the human engineer should therefore be in a format similar to that of the system-level requirements. Common format, both visually and electronically, will make the derivation of human performance requirements easier, and it will also make the verification or approval of those requirements a simpler task. In the same way, the human engineering design requirements should share a common format. In the case of these requirements, a common format is even more important as they must be reviewed or followed by system designers in other disciplines. As designs become more detailed, a continuous interaction between the human engineer and other disciplines becomes more advantageous. The implementation of the requirements needs to be verified, and additional design decisions need to be made as the design progresses.

IEEE 1220-1998:	6.1.11 – Define performance requirements
	6.1.14 – Define design characteristics
EIA-632:	Requirement 4 – Process Implementation Strategy
	Requirement 5 – Technical Effort Definition
	Requirement 10 – Progress Against Requirements
	Requirement 13 – Information Dissemination
	Requirement 14 – Acquirer Requirements
	Requirement 15 – Other Stakeholder Requirements
	Requirement 16 – System Technical Requirements
	Requirement 19 – Specified Requirements
	Requirement 25 – Requirement Statements Validation
	Requirement 26 – Acquirer Requirements Validation
	Requirement 27 – Other Stakeholder Requirements Validation
	Requirement 28 – System Technical Requirements Validation
	Requirement 29 – Logical Solution Representations Validation

B.3.3 FUNCTION ANALYSIS

Function analysis involves the conversion of the system's requirements into a functional architecture that defines how the system will meet those requirements. The functional architecture does not include references to allocation or implementation, but some functions will be included because of implementation decisions.

B.3.3.1 FUNCTIONAL DECOMPOSITION

Individual functions can often be decomposed in a variety of ways, and determining the best decomposition is often dependent on an adequate definition of the function's parameters. As functions are decomposed into greater detail, it becomes possible to allocate those functions to specific types of system components (including hardware, software, and humans). Allocating the functions may allow their parameters to be specified in greater detail and serves to verify the decomposition. Although the definition and decomposition of functions is independent of allocation and may be seen as not relevant to the human engineer, the results of the decomposition and analysis will be used in later design work. Much of the information that is critical to the human engineer may not be of interest to those performing the decomposition. Timing requirements, available information, required information, and other inputs may be necessary for subsequent human engineering design decisions. The optimal way to ensure that the necessary information is defined during decomposition is to have the human engineer work in conjunction with other designers.

IEEE 1220-1998	6.3.2 – Functional decomposition
EIA-632:	Requirement 17 – Logical Solution Representations

B.3.3.2 REVIEW OF FUNCTIONAL ARCHITECTURE

As with functional decomposition, the functional architecture is highly relevant to the human engineer despite the fact that it does not explicitly include any allocation decisions. The functional architecture does, however, imply some allocation decisions. It is the human engineer's responsibility to review the functional architecture and ensure that it includes all aspects relevant to the inclusion of humans in the system and their projected roles. In the case of top-level system functionality, the human engineer can provide feedback as to whether or not additional high-level functions need to be added to account for the role of humans proposed in the system concept. While it is likely that few if any functions will be added at this level, additional functions may be catalogued for inclusion during functional decomposition. The functional flow of the system needs to be assessed to ensure that it is compatible with the inclusion of humans in the system. Enhanced analysis is possible as more allocation decisions are made and as greater levels of decomposition are reached.

IEEE 1220-1998:	6.3.3 – Establish functional architecture
EIA-632:	Requirement 17 – Logical Solution Representations

B.3.4 FUNCTION ALLOCATION

One of the goals of function allocation is to effectively distribute the functions of the system between humans and technology. Much of this responsibility falls into the realm of human engineering. One way for the human engineer to go about this task is to identify the capabilities and limitations of both the potential operators and human engineering technologies and then weigh the various options to determine possible allocations. The human engineer first determines which functions must be allocated specifically to a human or machine, and then conducts the tradeoffs to develop additional potential allocations.

B.3.4.1 CONSIDERATION OF HUMAN ENGINEERING TECHNOLOGIES

In order to make the best decisions about which functions should be allocated to technology, it is important to be aware of the types of technology available and their inherent capabilities and limitations. The systems engineer conducts studies to assess the general capabilities and limitations of the technology available that may be useful for the particular system under design. Similarly, the human engineer conducts research and analysis to identify the technologies specifically applicable to human engineering and then further defines their capabilities and limitations. Relevant technologies include decision support systems, human performance models, and human-computer interaction techniques. An accurate assessment of the potential human engineering technology allows the human engineer to tradeoff these factors with the capabilities and limitations of the operator.

IEEE 1220-1998:	6.5.5 – Assess technology requirements
EIA-632:	Requirement 5 – Technical Effort Definition
	Requirement 16 – System Technical Requirements

B.3.4.2 EARLY IDENTIFICATION OF MANDATORY ALLOCATIONS

One of the first steps in allocation is the identification of functions that must be allocated specifically to a human or a particular technology. For example, if there is a complicated numerical calculation that must be completed very quickly, this should probably be allocated to software. On the other hand, if there is an important decision that must be made, such as whether or not to fire on a potential enemy, it may be determined that this function should not be left to a machine but should be the sole responsibility of an operator. The systems engineer will make these mandatory allocation decisions, based in part on recommendations from the human engineer.

There are a number of information sources that might be important for the human engineer to consider while developing mandatory allocation decisions. Information external to the design may include documents such as the Concept of Operations or human engineering literature applicable to the design domain. Sources of information from within the human engineering process that might be useful are the system use scenarios or the variety of documents outlining requirements, constraints, and capabilities/limitations. When dealing with higher-level functions, specific implementation-level allocations may be inappropriate, but the type of interaction between humans and technology (e.g., supervision, decision support, or automated assistance) can be defined. If the mandatory allocation decisions are finalized early, this can prevent wasted effort on designs that do not match the mandatory requirements.

IEEE 1220-1998:	6.5.1 – Group and allocate functions
EIA-632:	Requirement 17 – Logical Solution Representations
	Requirement 18 – Physical Solution Representations

B.3.4.3 DEVELOPMENT AND APPROVAL OF FUNCTION ALLOCATION RECOMMENDATIONS

Both the mandatory function allocations and the additional allocations that follow must be developed by taking into account a number of factors and considering a variety of information from the systems engineering process, the human engineering process, and sources external to the design process. This can be a complicated step in the design where conflicting costs and benefits require careful tradeoffs. If the allocation decision is ambiguous, systems engineering trade studies or human engineering studies, such as user review or performance and workload estimation, may need to be performed.

Once the recommendations are developed, they must be approved by the systems engineer. If the systems engineer was also involved in development, then the approval should be a simple step. However, if the human engineer developed the recommendations independently, the systems engineer may have feedback or suggestions for changes. In addition, the systems engineer should be aware of other influential decisions that might have been made or are being considered. Thus, the systems engineer should be able to take into account the objectives of the human engineer's suggested allocation and the objectives or constraints of the activities of other disciplines. This may be an iterative process of refinement until the systems engineer and human engineer can agree on a set of allocations.

IEEE 1220-1998:	6.5.1 – Group and allocate functions
EIA-632:	Requirement 17 – Logical Solution Representations
	Requirement 18 – Physical Solution Representations

B.3.5 TASK DESIGN AND ANALYSIS

Once the functions of a system have been assigned to particular system components, the functions can typically be defined to greater resolution of detail. Functions that have been allocated to humans are commonly referred to as tasks. Given the constraints of the system's requirements and functional architecture, the human engineer needs to define precisely how the humans within the system will carry out their assigned tasks.

B.3.5.1 Development of the Task List

Prior to analyzing the tasks to be performed by humans, it is necessary to compile a complete list of the tasks to be considered. This process may also include the decomposition of tasks, if such decomposition would be useful. Most likely, the human engineer will be responsible for creating the task list; however, he or she may want to work with the systems engineer or other designers to achieve a better understanding of the tasks. The human engineer will assess the information from the systems engineer and other design engineers and devise a complete list of human tasks. Additional inputs to the development of the task list include the approved function allocations and interface-specific tasks, if applicable. Interface-specific tasks are those that are created as a function of the interface that is chosen. Interface-specific tasks are normally defined following task design; however, due to the iterative nature of the design process, the human engineer may redevelop the task list in light of later decisions.

IEEE 1220-1998:	6.5.2 – Identify design solution alternatives
EIA-632:	Requirement 17 – Logical Solution Representations
	Requirement 18 – Physical Solution Representations

B.3.5.2 Identification of Task Characteristics, Interactions, and Sequences

Once the task list has been generated, the particular characteristics of each task must be outlined. This further definition facilitates a better understanding of the individual tasks, and can be used in other steps of the task design and analysis process. The task design and analysis portion of the human engineering process might be highly iterative, and the results of both these identifications can act as inputs for each other. The human engineer's task definition is dependent on the system design, since this design will impact the possible ways to accomplish the tasks. The human engineer can create the most useful set of task characteristics only with a correct understanding of the system design. The most accurate representation of the system design is probably embodied in the systems engineer's current candidate physical architectures. The systems engineer's functional decomposition will also be useful to consider. If the decomposition is not to the level of detail required by the human engineer, a further functional analysis may be necessary.

IEEE 1220-1998:	6.5.2 – Identify design solution alternatives
EIA-632:	Requirement 17 – Logical Solution Representations
	Requirement 18 – Physical Solution Representations

B.3.5.3 Selection of Modeling Tools and Techniques

Modeling techniques are typically used to evaluate or compare candidate designs. The utility of modeling techniques and executable models in particular can be significantly increased if models used by different designers are interoperable. Systems engineers can then create higher-level models of the system by combining models developed for different subsystems or within different disciplines.

An important step for the human engineer in task design and analysis is to select appropriate task-level tools and techniques that will result in a useful and appropriate model. The tools and techniques should be chosen early enough to ensure that they could support the inclusion of relevant information from the task analysis. These modeling tools and techniques will determine how the task list, task characteristics, and task interactions and sequences will be used to create task models. Given the importance of resource allocation to support system and subsystem modeling, overall project plans should include human engineering modeling as a programmed milestone.

IEEE 1220-1998:	6.5.2 – Identify design solution alternatives
	6.5.11 – Develop models and prototypes

EIA-632:	Requirement 5 – Technical Effort Definition
	Requirement 13 – Information Dissemination
	Requirement 23 – Tradeoff Analysis

B.3.5.4 TASK AND FUNCTION AUDIT

In synthesizing the physical architecture, allocations between humans and machines will be reflected in the design of interfaces. The designers will have to verify that all functions in the functional architecture can be traced to tasks performed by either humans or automation. A review of the task list – including interface- and team-specific tasks – should therefore find all of the tasks drawn from the function allocation in the interface and team concepts and designs. This review may be thought of as an audit of the interfaces with a mandatory consideration of all of the tasks from the analyses and simulations.

IEEE 1220-1998:	6.5.15 – Finalize design
EIA-632:	Requirement 17 – Logical Solution Representations
	Requirement 18 – Physical Solution Representations

B.3.6 HUMAN INTERFACE AND TEAM DEVELOPMENT

Designs and concepts for the interfaces between humans and software, hardware, and other humans need to be identified and developed. Three different stages of this process may be considered – the design of a single interface with which a human interacts, the design of the sum of the user interfaces for a single operator, and the ways in which multiple operators interact as a team. Each of these stages of interface development will occur iteratively or concurrently at the conceptual and detailed design levels. Three levels of interfaces are described starting with individual interfaces that represent a particular interaction based on the task analysis as well as performance and design requirements, then combinations of interfaces for a design at the individual operator level. These individuals are then assembled into crews or teams employing multiple operator interface designs and concepts. The creation of the separate levels of interfaces may be performed in any order depending on the availability of resources and the priority of individual user versus crew/team development.

B.3.6.1 POINTS OF HUMAN INTERFACE

Points of human interface may be thought of as the content and the location (origin and destination) of information that may be conveyed between humans or between a human and a machine. Also included are the data to be transmitted, the nodes or elements between which the data is to be transmitted, when the data is transmitted, and other interface-specific constraints, such as special conditions based on times and events. These points will be used in the development of the interface concepts and designs, and will lead to interfaces at the individual level followed by the crew/team level.

The human engineer must identify all of the data to be transmitted and the location, or nodes, to and from which it will be transmitted. This is based on the functional decomposition and allocation, as well as the task analysis (which includes characteristics of tasks and the interactions and sequences), and any available internal and external interface information developed to that point by the systems engineer. These system-level interfaces must be decomposed for application to the level of automation.

IEEE 1220-1998:	6.1.7 – Define interfaces
	6.5.7 – Define physical interfaces
EIA-632:	Requirement 18 – Physical Solution Representations

B.3.6.2 SELECTION OF HUMAN INTERFACE AND TEAM GUIDELINES

For the development of interfaces and teams, human engineers need to be aware of any existing guidelines applicable to the information or material passed between humans or between humans and equipment. The guidelines will also assist in keeping the design in accordance with constraints, heuristics, and prior research of the particular engineering or design community. Guideline topics may include, but are not limited to, short term and working memory limitations, display and control modalities, physical or strength limitations, and group dynamics. Collaboration between the systems engineer and human engineer on the selection and implementation of standards and guidelines will help identify how system-level guidelines may be applicable to human engineering designs. Full application of system-level guidelines will often require the implementation of specific, lower level, detailed guidelines. For example, if a particular computer system architecture is selected, then any associated user interface design guidelines should be implemented.

IEEE 1220-1998:	6.1.3 – Define external constraints
EIA-632:	Requirement 18 – Physical Solution Representations

B.3.6.3 DEVELOPMENT OF INTERFACE AND TEAM CONCEPTS OR DESIGNS

Once an initial physical architecture has been synthesized and approved by the systems engineer, the interfaces between system components – such as humans, hardware, and software – can be developed. The interaction of humans with other system components will be based on the functional architecture, allocation decisions and human engineering inputs.

The human engineer will be responsible for designing and optimizing how individual humans interact with non-human system components and how humans act together as teams. Interface concepts and designs are developed based on requirements for interaction between humans and other system components specified earlier. The concepts are less detailed and concrete than the designs but are highly iterative with their development, as they feed off of each other.

Team and individual interface design will be highly constrained due to other design decisions, such as specific pieces or types of hardware and software that are to be used. The human engineer attempts to develop team and interface designs that provide for optimal system performance within those constraints. The human engineer requires input from the systems engineer on system-level constraints (particularly those imposed by other design decisions), project and enterprise constraints, off-the-shelf availability, make-or-buy alternatives, state-of-the-art capabilities, and design solution alternatives. In some cases, constraints and design decisions that have been made previously may need to be reevaluated based on analysis of human performance within those constraints as well as interaction with other design disciplines to ensure the feasibility of the proposed designs.

IEEE 1220-1998:	6.1.2 – Define project and enterprise constraints
	6.1.3 – Define external constraints
	6.1.7 – Define interfaces
	6.5.7 – Define physical interfaces
EIA-632:	Requirement 18 – Physical Solution Representations

B.3.7 PERFORMANCE, WORKLOAD, AND TRAINING LEVEL ESTIMATION

The systems engineer must evaluate the design or design options proposed by system designers within the different disciplines. Evaluation of a single option is necessary to determine whether or not the system requirements are satisfied, and multiple options may be evaluated in order to make a selection. Overall system performance is an important parameter, but it typically consists of multiple variables that may be measured within different design disciplines. The design evaluations provided by different disciplines would all need to be available to the systems engineer to enable the tradeoff of different design options.

To help in the evaluation of concepts and designs, the human engineer will estimate the physical and cognitive workload levels of individuals and teams within the system. Workload stressors and their effects on human performance and operator coping strategies, as well as the effects of task neglect or delay, need to be defined. Workload and the resultant manning and training requirements are to be optimized to meet required performance levels.

B.3.7.1 INDIVIDUAL AND TEAM WORKLOAD AND PERFORMANCE ESTIMATION

Workload levels can significantly influence the performance of many system components or subsystems, including humans. Once workload levels are predicted, performance measures can be adjusted to determine the impact of workload. Given the tasks allocated to humans, the human engineer needs to estimate the cognitive and physical workload demands of the tasks on the operators and users. Executable models or simulations are typically used, but subjective feedback from test users or subject matter experts may also be employed. In order to be accurate, workload models need to include any operator or user tasks that are required to manipulate or utilize the human-machine interface.

To effectively estimate workload and performance, the human engineer needs up-to-date design data from the systems engineer and other designers. In order to create accurate models of how the humans interact with the rest of the system, the human engineer will need access to models of other system components. Without an accurate simulation of hardware and software functions and performance, the model of the human interactions will not be accurate. Information on other system components may be included as part of an executable model, or it may be used to create a physical prototype of portions of the system with which test users can interact. The true relevance of workload lies in its impact on human and system performance, not as a stand-alone measure, so workload measures should be easily integrated with performance models. Similarly, models of human performance need to be compatible with models that can predict overall system performance. The goal of the human engineer should not be to optimize human performance alone, but to put human performance within acceptable levels to optimize overall system performance. This goal cannot be accomplished without human workload and performance models that are compatible with higher-level system models. Model compatibility will also be important when design changes are made that necessitate alterations to the models.

IEEE 1220-1998:	6.5.11 – Develop models and prototypes
	6.5.15 – Finalize design
EIA-632:	Requirement 10 – Progress Against Requirements
	Requirement 23 – Tradeoff Analysis

B.3.7.2 TRAINING CONCEPT EVALUATION

The resources required to field and maintain a system are typically key concerns of the systems engineer. The overall cost of the system includes the cost to prepare it for use and to maintain it over its life cycle. If the human is considered part of the system, then the resources required to prepare and provide operators and users are just as relevant as the resources required to provide equipment upgrades or to replenish supplies. The users and operators are frequently the most often changed and varied parts of the system. The training required to prepare them for use of the system and to maintain their qualifications as users and operators are important parts of the system life cycle support requirements.

In the development of a particular system, training may or may not be considered part of the human engineer's responsibilities. Even if the human engineer is not directly responsible for developing training requirements or training plans and methodologies, the work of the human engineer has direct and significant impact on these issues. The difference between the knowledge, skills, and abilities required to be a system user and operator and the knowledge, skills, and abilities possessed by prospective users and operators will determine the training and selection requirements. The knowledge, skills, and abilities expected to be available in prospective users and operators must be agreed upon by the human engineer and systems engineer. Requirements and constraints for the life cycle support of the system must be available to the human engineer to ensure that the training and selection requirements are compatible. Requirements such as those for on-the-job training or embedded training must be stated early to reduce the likelihood of design changes to meet these requirements at a later date.

IEEE 1220-1998:	6.1.2 – Define project and enterprise constraints
	6.1.3 – Define external constraints
	6.5.4 – Assess life cycle quality factors
EIA-632:	Requirement 21 – Transition to Use

B.3.7.3 TRADEOFF OF CONCEPTS AND DESIGNS

Once estimates of subsystem or component performance are available, different design alternatives can be traded off to determine the best available option. If multiple alternatives meet the system's functional and performance requirements, then those alternatives should be compared to select the optimal design.

In some cases, a tradeoff may involve the decision of whether or not to redesign portions of the system or the degree of redesign required. In such situations, the availability of resources such as time, money, and personnel become as important as technical feasibility. The systems engineers and designers within different disciplines, such as human engineering, must operate from the same set of resource assumptions in making these decisions. When proposing a design change, the human engineer needs to go beyond simply stating that there is a problem with the current design and provide a potential alternative to the current design. This alternative should be in line with the available resources and the selected design criteria for the project as a whole. Simply because the human engineer has the time and resources to make a design change, does not mean that the other designers required to implement the change have the available resources.

IEEE 1220-1998:	6.7.5 – Define trade-off analysis scope
EIA-632:	Requirement 18 – Physical Solution Representations
	Requirement 23 – Tradeoff Analysis

B.3.8 USER AND REQUIREMENTS REVIEW

Throughout the system development process, the system design must be reviewed with respect to both its requirements and the operational need. The system design must be compared to all requirements, not simply the top-level system requirements. Designers or verifiers within individual design disciplines must carry out some of this verification process.

B.3.8.1 COMPARISON TO HUMAN ENGINEERING REQUIREMENTS

As system designs are generated from requirements, those designs must then be verified to ensure that the requirements are satisfied. This verification is likely to be at least partially included in the responsibilities of designers from different disciplines. It is highly probable that the human engineer will need to assess and verify designs generated by others. The specific human engineering requirements, such as design requirements and human performance requirements, must be used to evaluate the designs. A large amount of the verification process will typically be spent on task or job designs or equipment design specific to human engineering. Other designs, however, will have to be reviewed for compatibility with human engineering requirements. Verification may be performed through a variety of different means, ranging from inspection to modeling and simulation to user-in-the-loop testing.

IEEE 1220-1998:	6.6.2 – Conduct verification evaluation
EIA-632:	Requirement 19 – Specified Requirements
	Requirement 20 – Implementation
	Requirement 29 – Logical Solution Representations Validation
	Requirement 30 – Design Solution Verification
	Requirement 31 – End Product Verification

B.3.8.2 USER REVIEW

Verification that the design of a system conforms to requirements is important, but the system design must also be validated. The system needs to conform to the needs of the users, operators, or purchasers, and precise conformance to written requirements does not always provide such assurance. Reviewing potential designs with intended users and operators through means such as storyboards, simulations, and mock-ups can provide early and rapid validation feedback. Full validation that the system meets the operational need may not occur until the system is operational and fielded.

One of the major roles of the human engineer is to determine the requirements and needs of the intended operators and users. Although reviewers such as representative users and operators or subject matter experts may be able to provide some feedback or requirements and functional descriptions, more effective feedback can be generated from the review of proposed physical designs. Through system use scenarios and static or dynamic models of system operation, the human engineer can elicit feedback that may be used for changes to designs or requirements. Not all feedback will be relevant or valid. Changes to system design or requirements should be based on an objective analysis of information, not on the subjective preferences or opinions of reviewers. The human engineer will need to evaluate the feedback to determine what changes may be considered, and an initial estimate of the impact of those changes on other portions of the system should be made. This information will need to be passed to the systems engineers or other designers.

IEEE 1220-1998:	6.5.11 – Develop models and prototypes
	6.6.2 – Conduct verification evaluation
EIA-632:	Requirement 10 – Progress Against Requirements
	Requirement 11 – Technical Reviews
	Requirement 19 – Specified Requirements
	Requirement 20 – Implementation
	Requirement 30 – Design Solution Verification
	Requirement 31 – End Product Verification
	Requirement 33 – End Products Validation
	Requirement 33 – End Products Validation

B.3.8.3 RECOMMENDATION OF CHANGES TO REQUIREMENTS OR DESIGNS

Deficiencies in system design that are revealed through verification or validation must be addressed by some combination of changes to the design and changes to requirements. These changes can frequently have far-reaching effects, leading to time delays and cost overruns. It is the role of the systems engineer to work to balance the required changes with the available resources to meet the design goals. This requires rapid feedback from designers from various disciplines on the impact of changes.

The human engineer should go beyond singling out design deficiencies and should work to present alternative designs or requirements. In some cases, it may be found that the operators simply cannot meet the specified human performance requirements or that unsatisfactory workload levels exist. This will necessitate either a change to the requirements or an addition to the design to provide additional support. Proposed designs may conflict with requirements that have been specified by the human engineer. In some instances, other designers or the systems engineer may want to delete or ignore some requirements related to human engineering. The human engineer must know which human engineering requirements can be traded away to efficiently meet overall system requirements and which requirements cannot be sacrificed. The human engineer should not blindly hold to requirements to optimize human performance when the overall performance of the system will suffer.

IEEE 1220-1998:	6.7.1 – Assess requirement conflicts
	6.7.3 – Assess design alternatives
EIA-632:	Requirement 10 – Progress Against Requirements
	Requirement 11 – Technical Reviews
	Requirement 19 – Specified Requirements
	Requirement 23 – Tradeoff Analysis

B.4 INTERACTIONS SORTED BY IEEE 1220-1998

IEEE 1220-1998 Paragraph	Human Engineering Appendix Paragraph (B.xxx)
6.1.2 Define project and enterprise constraints	3.1.1 Selection of Comparison Systems
	3.2.1 Human Engineering Constraints
	3.6.3 Development of Interface and Team Concepts or Designs
	3.7.2 Training Concept Evaluation
6.1.3 Define external constraints	3.1.1 Selection of Comparison Systems
	3.2.1 Human Engineering Constraints
	3.6.2 Selection of Human Interface and Team Guidelines
	3.6.3 Development of Interface and Team Concepts or Designs
	3.7.2 Training Concept Evaluation
6.1.4 Define operational scenarios	3.1.2 System Use Scenarios
6.1.7 Define interfaces	3.6.1 Points of Human Interface
	3.6.3 Development of Interface and Team Concepts or Designs
6.1.8 Define utilization environments	3.1.3 User Environment Characteristics and Effects
6.1.11 Define performance requirements	3.2.2 Human Performance Requirements and Human Engineering Design Requirements
6.1.14 Define design characteristics	3.2.2 Human Performance Requirements and Human Engineering Design Requirements
6.3.2 Functional decomposition	3.3.1 Functional Decomposition
6.3.3 Establish functional architecture	3.3.2 Review of Functional Architecture
6.5.1 Group and allocate functions	3.4.2 Early Identification of Mandatory Allocations
	3.4.3 Development and Approval of Function Allocation Recommendations
6.5.2 Identify design solution alternatives	3.5.1 Development of the Task List
	3.5.2 Identification of Task Characteristics, Interactions, and Sequences
	3.5.3 Selection of Modeling Tools and Techniques
6.5.4 Assess life cycle quality factors	3.7.2 Training Concept Evaluation
6.5.5 Assess technology requirements	3.4.1 Consideration of Human Engineering Technologies
6.5.7 Define physical interfaces	3.6.1 Points of Human Interface
	3.6.3 Development of Interface and Team Concepts or Designs
6.5.11 Develop models and prototypes	3.5.3 Selection of Modeling Tools and Techniques
	3.7.1 Individual and Team Workload and Performance Estimation
	3.8.2 User Review
6.5.15 Finalize design	3.5.4 Task and Function Audit
	3.7.1 Individual and Team Workload and Performance Estimation
6.6.2 Conduct verification evaluation	3.8.1 Comparison to Human Engineering Requirements
	3.8.2 User Review
6.7.1 Assess requirement conflicts	3.8.3 Recommendation of Changes to Requirements or Designs
6.7.3 Assess design alternatives	3.8.3 Recommendation of Changes to Requirements or Designs
6.7.5 Define trade-off analysis scope	3.7.3 Tradeoff of Concepts and Designs

B.5 INTERACTIONS SORTED BY SYSTEMS ENGINEERING OSDs (Operational Sequence Diagrams)

EIA-632 Requirement	Human Engineering Appendix Paragraph (B.xxx)
Planning Process	
Requirement 4 – Process Implementation Strategy	3.1.1 Selection of Comparison Systems
	3.1.2 System Use Scenarios
	3.2.2 Human Performance Requirements and Human Engineering Design Requirements
Requirement 5 – Technical Effort Definition	3.2.1 Human Engineering Constraints
	3.2.2 Human Performance Requirements and Human Engineering Design Requirements
	3.4.1 Consideration of Human Engineering Technologies
	3.5.3 Selection of Modeling Tools and Techniques
Assessment Process	
Requirement 10 – Progress Against Requirements	3.2.2 Human Performance Requirements and Human Engineering Design Requirements
	3.7.1 Individual and Team Workload and Performance Estimation
	3.8.2 User Review
	3.8.3 Recommendation of Changes to Requirements or Designs
Requirement 11 – Technical Reviews	3.8.2 User Review
	3.8.3 Recommendation of Changes to Requirements or Designs
Control Process	
Requirement 13 – Information Dissemination	3.1.1 Selection of Comparison Systems
	3.2.2 Human Performance Requirements and Human Engineering Design Requirements
	3.5.3 Selection of Modeling Tools and Techniques
Requirements Definition Process	
Requirement 14 – Acquirer Requirements	3.2.1 Human Engineering Constraints
	3.2.2 Human Performance Requirements and Human Engineering Design Requirements
Requirement 15 – Other Stakeholder Requirements	3.2.1 Human Engineering Constraints
	3.2.2 Human Performance Requirements and Human Engineering Design Requirements
Requirement 16 – System Technical Requirements	3.1.2 System Use Scenarios
	3.1.3 User Environment Characteristics and Effects
	3.2.1 Human Engineering Constraints
	3.2.2 Human Performance Requirements and Human Engineering Design Requirements
	3.4.1 Consideration of Human Engineering Technologies
Solution Definition Process	
Requirement 17 – Logical Solution Representations	3.3.1 Functional Decomposition
	3.3.2 Review of Functional Architecture
	3.4.2 Early Identification of Mandatory Allocations
	3.4.3 Development and Approval of Function Allocation Recommendations
	3.5.1 Development of the Task List
	3.5.2 Identification of Task Characteristics, Interactions, and Sequences
	3.5.4 Task and Function Audit
Requirement 18 – Physical Solution	3.4.2 Early Identification of Mandatory Allocations

EIA-632 Requirement	Human Engineering Appendix Paragraph (B.xxx)
Representations	3.4.3 Development and Approval of Function Allocation Recommendations
	3.5.1 Development of the Task List
	3.5.2 Identification of Task Characteristics, Interactions, and Sequences
	3.5.4 Task and Function Audit
	3.6.1 Points of Human Interface
	3.6.2 Selection of Human Interface and Team Guidelines
	3.6.3 Development of Interface and Team Concepts/Designs
	3.7.3 Tradeoff of Concepts and Designs
Requirement 19 – Specified Requirements	3.2.1 Human Engineering Constraints
	3.2.2 Human Performance Requirements and Human Engineering Design Requirements
	3.8.1 Comparison to Human Engineering Requirements
	3.8.2 User Review
	3.8.3 Recommendation of Changes to Requirements or Designs
Implementation Process	
Requirement 20 – Implementation	3.8.1 Comparison to Human Engineering Requirements
	3.8.2 User Review
Transition to Use Process	
Requirement 21 – Transition to Use	3.7.2 Training Concept Evaluation
Systems Analysis Process	
Requirement 23 – Tradeoff Analysis	3.5.3 Selection of Modeling Tools and Techniques
	3.7.1 Individual and Team Workload and Performance Estimation
	3.7.3 Tradeoff of Concepts and Designs
	3.8.3 Recommendation of Changes to Requirements or Designs
Requirement 24 – Risk Analysis	3.1.2 System Use Scenarios
	3.1.3 User Environment Characteristics and Effects
Requirements Validation Process	
Requirement 25 – Requirement Statements Validation	3.2.2 Human Performance Requirements and Human Engineering Design Requirements
Requirement 26 – Acquirer Requirements Validation	3.2.2 Human Performance Requirements and Human Engineering Design Requirements
Requirement 27 – Other Stakeholder Requirements Validation	3.2.2 Human Performance Requirements and Human Engineering Design Requirements
Requirement 28 – System Technical Requirements Validation	3.2.2 Human Performance Requirements and Human Engineering Design Requirements
Requirement 29 – Logical Solution Representations Validation	3.2.2 Human Performance Requirements and Human Engineering Design Requirements
	3.8.1 Comparison to Human Engineering Requirements
System Verification Process	
Requirement 30 – Design Solution Verification	3.8.1 Comparison to Human Engineering Requirements
	3.8.2 User Review
Requirement 31 – End Product Verification	3.8.1 Comparison to Human Engineering Requirements
	3.8.2 User Review
End Products Validation Process	
Requirement 33 – End Products Validation	3.8.1 Comparison to Human Engineering Requirements
	3.8.2 User Review

B.6 SUGGESTED REFERENCES

The following references each provide further information on human engineering or human factors, primarily in the context of systems engineering.

Human Factors in Systems Engineering; Alphonse Chapanis; 1996. (340 pages)

Part of a series of titles on systems engineering, this book covers the integration of human factors into the development of tools, machines, and systems. It includes sections on systems engineering and systems engineering work products along with human factors methods. General introductions to human physical and mental characteristics and personnel selection and training issues are also included. The conclusion of the book covers the specification of human-system requirements and how to make tradeoffs between competing requirements or designs.

MANPRINT: An Approach to Systems Integration; Harold Booher, Ed.; 1990. (600 pages)

This book is a collection of chapters by various authors on topics relating to the Manpower and Personnel Integration (MANPRINT) program developed for the US Army. Management, design, and integration topics are included. Although sections such as those on design tools lack up-to-date information, the discussion of the principles of human engineering and integration remains relevant.

Introduction to Human Factors Engineering; Christopher Wickens, Sallie Gordon, and Yili Liu; 1997. (750 pages)

Although there is an emphasis on cognition and human information processing, this book provides a broad coverage of human factors issues. Topics include automation, human-computer interaction, safety, and workplace layout.

Human Factors in Engineering and Design (7th ed.); Mark Sanders and Ernest McCormick; 1993. (790 pages)

First published in 1957, this book is commonly used as an upper-undergraduate level or introductory graduate level textbook. It provides a broad overview of human factors and ergonomics topics and sections on how human factors should be applied. Other sections include information input, human output and control, workplace design, and environmental conditions. Information included on human-computer interaction is relatively dated, but the principles illustrated by the examples included remain applicable.

Human Performance Engineering (3rd ed.); Robert Bailey; 1996. (576 pages)

Although sometimes billed as a general human factors reference, this book places significant emphasis on computer-based systems. There is more of a discussion on human factors techniques and methodologies than in other general texts. Design and analysis examples are included, as are several real-world examples of violations of human factors principles.

System Design and Evaluation; Sara Czaja; 1997. In G. Salvendy (Ed.), **Handbook of Human Factors and Ergonomics** (2nd ed.) (pp.17-40)

This book's chapter provides a brief overview of system design and presents a discussion of different approaches to system design that address the presence and role of humans within the system. The basic human factors activities in system design and test and evaluation are also described.

Allocation of Functions; Joseph Sharit; 1997. In G. Salvendy (Ed.), **Handbook of Human Factors and Ergonomics** (2nd ed.) (pp. 301-339)

Part of a section on job design, this book chapter discusses the importance of human-machine allocation of functions during system design. Different procedures for function allocation are covered, as are implications for dynamic allocation – the transfer of functions between humans and machines during system operation. The issues of trust and confidence in automated systems are also covered.

APPENDIX C - THE SYSTEMS ENGINEERING MANAGEMENT PLAN INCLUDING SYSTEMS ENGINEERING SCHEDULES

C- Introduction:

The Systems Engineering Management Plan (SEMP) is a document that identifies the plans and schedules that will be needed to perform the technical effort for the systems development. This document is used to tailor the various activities to the needs of the project/program and is used to control the systems development when completed and approved. This appendix discusses the SEMP in detail and provides guidance for writing such a plan. The information is organized around a proposed table of contents as follows.

Topic	Paragraph
Cover and Title Page	1
Table of Contents	2
Scope	3
Applicable Documents	4
Systems Engineering Process	5
Transitioning Critical Technologies	6
Integration of the Systems Engineering Effort	7
Additional Systems Engineering Activities	8
Notes and Appendices	9

C-1 COVER AND TITLE PAGE

The cover and title page should follow company/organization procedures or style guide as applicable. The minimum information typically required is the words "Systems Engineering Management Plan," and the document title. Other information such as project number and customer may be added.

C-2 TABLE OF CONTENTS

The Table of Contents can be generated automatically by most word processing software to list the title and page number of each titled paragraph and subparagraph. Then, list the title and page number of each figure, table, and appendix, in that order.

C-3 SCOPE

A draft Scope should be written early to express how the project being proposed is intended to be performed. This will most likely need to be modified when the actual SEMP content has been reviewed. The required executive summary could be provided as part of the scope or as the first paragraph of the Systems Engineering Process section. The scope must include a brief description of the purpose of the system to which this SEMP applies, and a summary of the purpose and content of the SEMP.

C-4 APPLICABLE DOCUMENTS

Start with the list of documents from the Request for Proposal. These will be in the format:

Document No.	Title
ISBN	Title of the Document (date and other important notes are optional)

Government documents will be listed first followed by non-Government documents. You should add company/organization or other documents you know will be needed. When the rest of the SEMP has been completed, go back to this section and cross off the documents that are not referenced in your SEMP and add any documents referenced in your SEMP that were not already on the list. This might be a good time to check if any of the crossed-off documents really should have been referenced. If so, put them back in and reference them where appropriate.

C-5 SYSTEMS ENGINEERING PROCESS

This section contains the core Systems Engineering planning information description. It should convey how the company/organization performs Systems Engineering, and should include applicable aspects of the company's/organization's Systems Engineering Policies and Procedures. Include the organizational responsibilities and authority for Systems Engineering activities and control of subcontracted engineering. Define the tasks in the Systems Engineering Master Schedule (SEMS) and the milestones of the Systems Engineering Detail Schedule (SEDS). Further descriptions of the SEMS and SEDS follow in this appendix.

Process Topic	Paragraph
Systems Engineering Process Planning	5.1
Requirements Analysis	5.2
Functional Analysis/Allocation	5.3
Synthesis	5.4
Systems Analysis and Control	5.5

C-5.1 SYSTEMS ENGINEERING PROCESS PLANNING

There are many planning elements for the Systems Engineering activities.

Provide descriptions of:	Paragraph
Process inputs	5.1.2
Decision database (deliverables and results)	5.1.1
Technical objectives	5.1.3
Project work breakdown structure (WBS)	5.1.4
Training	5.1.5
Standards and procedures	5.1.6
Resource allocation	5.1.7
Constraints	5.1.8
Work authorization	5.1.9
Verification planning	5.1.10

C-5.1.1 DECISION DATABASE (DELIVERABLES AND RESULTS)

Major deliverables include a decision database, specifications and baselines, all discussed in the body of the handbook. Software packages for Systems Engineering are available which can organize these three items into a single database. Source documentation is entered into the database and separated into individual requirements expressed as shall statements. These are sometimes called atomic requirements. The source requirements should be identified by a convenient numbering system for traceability.

When the Decision Database material has been collected, a technical requirements document (TRD) can be prepared. The TRD maintains traceability of requirements to original source material and provides the foundation for the remaining specifications in the specification tree.

C-5.1.2 PROCESS INPUTS

This paragraph of your SEMP identifies the source material to be used for the deliverables discussed above. This will include the Statement of Work and the specification from the request for proposal. It also may include previously developed specifications for similar systems and company procedures affecting performance specifications. A table listing the documents to be used and their availability is recommended.

C-5.1.3 TECHNICAL OBJECTIVES

A technical objectives document should be developed. This may be one of the source documents for the decision database described above. The document may be part of a Concept of Operations for the system. It should include success criteria that define when the design is complete. Items to be considered for the document include development, manufacturing, verification, deployment, operations, support, training, and disposal objectives. Since this document will be an additional source for the decision database, it is not necessary to duplicate requirements from the other source material. However, if it is convenient to duplicate for the technical objectives document, then only one entry should appear in the TRD discussed above for each duplicated requirement.

C-5.1.4 WORK BREAKDOWN STRUCTURE.

The Work Breakdown Structure (WBS) is a programmatic requirement and is not included in the technical decision database. The WBS should:

a. Account for the processes for each product within the hierarchy
b. Follow the specification tree
c. Provide a useful structure for reporting progress, performance, and engineering evaluation
d. Provide the framework for relating statements of work, TPM, SEMS, SEDS, contract line items, configuration items, technical and management reports, and the system elements
e. Extend to the lowest level necessary to reach assignable and measurable units of tasks
f. Serve as the framework for your management control system to provide auditable and traceable summaries of internal data generated by your performance measurement procedures (i.e., earned value), if prepared according to the above guidelines
g. Provide a structure for identifying risks, making risk assessments, and identifying critical technical parameters and their dependency trees
h. Assist in developing and evaluating engineering change proposals and specification change notices through the logical depiction of WBS elements

The WBS separates the activities of development, manufacturing, verification, deployment, operations, support, training, and disposal into manageable subtasks. A sample WBS is included in the sample SEMP. One way to prepare a WBS is to construct a hierarchy of tasks to be done. Similar tasks should be grouped by similar identifiers and allowance for addition of tasks to the original structure should be considered. A programmatic decision database may be constructed which parallels the technical decision database. If IPPD is employed, then the WBS should be organized by end items. In this context, an end item is any item being worked by an IPDT.

C-5.1.5 TRAINING

The system being proposed may be complex enough that the customer will require training in order to use it. Your company may also need to train those who will develop, manufacture, verify, deploy, operate, support, do training, or dispose of the system. A plan for this training is required in the SEMP. Include in the training:

a. Analysis of performance
b. Behavior deficiencies or shortfalls

c. Required training to remedy the above
d. Schedules to achieve required proficiencies

Analysis of Performance

This should include a theory of operation, comparison to related or similar systems, what the system can and cannot do. It may also include concept of operations, level of skill required to operate the system, system goals from the technical objectives document above, and any other appropriate detailed analyses.

Behavior Deficiencies or Shortfalls

Behavior deficiencies or shortfalls are important aspects of the training so that unrealistic expectations do not degrade perceptions of the system. They may also be an excellent opportunity for preplanned product improvement. Such deficiencies may represent system vulnerabilities, which should be identified as a major concern.

Required Training to Remedy Deficiencies

Identify the required training to remedy the above behavior deficiencies or shortfalls. These areas are important to explore for optimum customer satisfaction with the system.

Schedules to Achieve Required Proficiencies

Schedules to achieve required proficiencies should list all items to be trained as a schedule item. Scheduling these activities must include how long the task will take, when the trained people will be needed, and what prerequisite knowledge is required. The availability of materials and equipment (including that being developed on the project and that obtained from other sources) must be considered. Many software packages exist which can assist in laying out this schedule and identifying schedule conflicts which need to be solved or worked around. These schedules will become part of the SEDS.

C-5.1.6 STANDARDS AND PROCEDURES

Define the set of your company's standards and procedures that are applicable to this project. These may include Workmanship, Quality Assurance, Engineering Policies and Procedures, and Time Charging Practices. Since the standards and procedures are well documented in the company's policies and procedures these published policies should be referenced.

C-5.1.7 RESOURCE ALLOCATION

Resource allocation includes:

a. resource requirements identification
b. procedures for resource control
c. reallocation procedures

Resource Requirements Identification

Resource requirements identification includes

a. capital equipment needs
b. software needs
c. personnel needs
d. customer-furnished equipment and information
e. time-phased needs for parts of the system being developed.

Make a list of needs by a method such as team brainstorming, then separate by the above categories and define the soonest, optimum, and latest need dates which are required to support the project SEMS.

Procedures for Resource Control

Procedures for resource control vary for the five types of resources identified above. The SEMP should discuss each of these separately. For example, capital equipment requires approval for expenditure of company funds. At what point in the capital planning cycle will this award occur? Will this capital be purchased, rented, or obtained from other company resources? Will it only be purchased if the contract is won?

Reallocation Procedures

Reallocation procedures describe alternate approaches if the identified resources are not available in a timely manner. These contingency plans should be addressed in adequate detail for the highest risk occurrences. In particular, make or buy decisions may need to be readdressed. The results of the reallocation of resources can contribute to defining constraints for the project.

C-5.1.8 CONSTRAINTS

Constraints describe what is not included in the project. These items define work that might be expected but will not be done. Often these are defined in work scope statements given by project contributors during the cost definition process. Constraints may be further adjusted during the project performance. Like behavior deficiencies or shortfalls, these are excellent opportunities for preplanned product improvement. Funding, personnel, facilities, manufacturing capability, critical resources, or other reasons cause the existence of constraints. The reason for each constraint should be understood and documented.

C-5.1.9 WORK AUTHORIZATION

Work authorization is the process by which the project is baselined and financially controlled. In this section a description of the company's procedures for starting work on the detailed parts of the WBS should be defined.

C-5.1.10 VERIFICATION PLANNING

Verification planning is usually done according to a verification matrix which lists all the requirements. The possible methods of verification include inspection, analysis, demonstration, and test. The SEMP should state that a verification plan will be written to define the items to be verified and which methods will be used to verify performance. Detailed procedures are usually not written for inspection and analysis verification methods.
Procedures will normally be written for the demonstrations and tests. A procedure may describe all the demonstrations and tests for the system, or more commonly a related subset of these demonstrations and tests, or in some cases just one test. A given demonstration or test may verify one or several requirements. The demonstrations and tests may be done on the full system or on subsystems or components, and may be done on first article equipment, qualification models, or on every system to be produced.

The plan should define, at least in general terms, which performance items will be verified by which of the above methods. The plan should also define who is to perform and witness the verification of each item. This should also relate to the SEMS or SEDS for time phasing of the verification process. For example, component tests may be done early in the program, and full system tests at the end. Some components or subsystems may be already verified by your company or by suppliers or customers. In this case, only the integration into the system needs to be further verified.

C-5.2 REQUIREMENTS ANALYSIS

The approach and methods should be defined for analyzing missions and environments; identification of requirements for development, manufacturing, verification, deployment, operations, support, training, and disposal; and determination of design constraint requirements. The approach and methods used to define the performance and functional requirements for the following areas of Specialty Engineering should also be documented:

a. Reliability and Availability
b. Maintainability, Supportability, and Integrated Logistics Support (ILS)
c. Survivability including Nuclear, Biological, and Chemical
d. Electromagnetic Compatibility, Radio Frequency Management, and Electrostatic Discharge
e. Human Engineering and Human Systems Integration
f. Safety, Health Hazards, and Environmental Impact
g. System Security
h. Producibility
i. Test and Evaluation
j. Testability and Integrated Diagnostics
k. Computer Resources
l. Transportability
m. Infrastructure Support
n. Other Engineering Specialties bearing on the determination of performance and functional requirements

The above fifteen-point list should be used as a check-list on the source documentation, and any applicable items that were omitted should be included. Some areas may impact requirements analysis only after synthesis efforts identify solution alternatives. If some of the areas are not possible to address until after a portion of the project has been completed, this should be noted in the SEMP.

C-5.3 FUNCTIONAL ANALYSIS

The SEMP should include a clear description of the scope of functional analysis and any affordability constraints that will be encountered during the project. Ideally, a discussion of the functional analysis to be performed would be presented for each and every system requirement. This is not practical, so a minimum subset of this effort must be described. The remainder will then necessarily be covered by a more general plan. Funds available usually limit a rigorous full system analysis to some reasonable subset. A good way to manage this is through the use of risk analysis. High-risk requirements will be defined in some detail, medium-risk requirements will be listed, and low-risk requirements will be included in an overall general summary. In addition to high-risk requirements, those requirements that are most important to the customer, including affordability should be defined in detail. For the items to be defined in detail, the SEMP should include consideration of what type of analysis will be done, what tools will be used, what are the schedule and budget constraints, and what are the completion criterion.

C-5.4 SYNTHESIS

Include the approach and methods to transform the functional architecture into a physical architecture; to define alternative system concepts; to define physical interfaces; and to select preferred product and process solutions. Describe how requirements including "ilities", non-developmental items, and parts control are converted into detailed design specifications. The following topics could be considered:

- Previous experience
- Maintain a System Design Notebook
- Perform trade studies

- Taguchi analysis
- Inclusion of COTS equipment
- Simulation
- Performance models
- Developmental testing
- Concurrent engineering

C-5.5 SYSTEMS ANALYSIS AND CONTROL

This portion of the SEMP describes the specific systems analysis efforts needed including tools necessary for their conduct. Include the approach and methods to arrive at a balanced set of requirements and a balanced functional and physical architecture to satisfy those requirements. Include the approach and methods to control the phase dependent outputs of the Systems Engineering process.

Provide descriptions of:	Paragraph
Trade studies	5.5.1
System/cost effectiveness analyses	5.5.2
Risk management	5.5.3
Configuration management	5.5.4
Interface management	5.5.5
Data management	5.5.6
Systems Engineering Master Schedule (SEMS)	5.5.7
Technical Performance Measurement	5.5.8
Technical reviews	5.5.9
Supplier Control	5.5.10
Requirements traceability	5.5.11

C-5.5.1 TRADE STUDIES

The SEMP should indicate what trade studies will be included in the project. For the SEMP, describe the studies planned to make tradeoffs among stated user requirements, design, program schedule, functional, and performance requirements, and life-cycle costs. Describe the use of criteria for decision-making and trade off of alternative solutions. Include a description of the use of technical objectives, criteria and weighting factors, and utility curves as applicable. These may be presented in matrix form. A sample or partially completed trade matrix for a typical requirement could be included.

C-5.5.2 SYSTEM/COST EFFECTIVENESS ANALYSES

System/cost effectiveness analyses should be included as a part of each trade study. Even if no other trade study is done, an overall evaluation of system/cost effectiveness should be completed. This will assure the customer that no obvious alternatives, which could save significant cost, have been overlooked due to poor planning. Describe the implementation of system/cost effectiveness analyses to support the development of life-cycle balanced products and processes and to support risk management activities. Describe the Measures of Effectiveness (MOE) hierarchy, how the MOEs interrelate, and criteria for the selection of additional MOEs to support the evolving definition and verification of the system. Describe the overall approach for system/cost effectiveness analysis as well as manufacturing analysis, verification analysis, deployment analysis, operational analysis, supportability analysis, training analysis, disposal analysis, environmental analysis, and life cycle cost analysis. Include a description of how analyses will be partitioned into the various areas, if they cannot be conducted integrally, and how analytic results will be integrated.

C-5.5.3 RISK MANAGEMENT

The SEMP should indicate what requirements are considered high, medium or low risk. Alternatively a more detailed assessment than high, medium or low may be desired. A discussion of risk management is contained in Section 6 of this handbook. Describe the technical risk management program including the approach, methods, procedures, and criteria for risk identification, quantification sensitivity assessment, handling, and risk impact integration into decision processes. Describe the risks associated with the development, test, and evaluation requirements. Identify critical risk areas. Describe plans to minimize technical risk (e.g., additional prototyping, technology and integration verification, back up development). Identify risk control and monitoring measures including special verifications, technical performance measurement parameters, and critical milestones. Describe the method of relating TPM, the SEMS, and the SEDS to cost and schedule performance measurement and the relationship to the Work Breakdown Structure.

C-5.5.4 CONFIGURATION MANAGEMENT

The SEMP should include the configuration management plan for the system and related documentation. Configuration management may also include programmatic documents such as the statement of work, and cost records. Describe the approach and methods used to manage the configuration of identified system products and processes. Describe program change control procedures and baseline management.

C-5.5.5 INTERFACE MANAGEMENT

The SEMP should include the interface management plan. Describe the nature of interface control documents, and subsystem, component and vendor specifications. Interface control working groups may be established to manage the documents and the work of diverse participants in the project. Indicate if documents may require customer approval. Describe the approach and methods used to manage the internal interfaces and support activities to ensure that external interfaces are managed and controlled.

C-5.5.6 DATA MANAGEMENT

Describe the approach and methods used to establish and maintain a data management system. Do not duplicate the description of the Decision Database. This section should describe what control systems are in place including applicable company procedures, recent practice on similar programs, and what controls will be used for this project. The company may be evaluated to confirm that the stated controls are being followed.

C-5.5.7 SYSTEMS ENGINEERING MASTER SCHEDULE

The Systems Engineering Master Schedule (SEMS) is a tool for project control. The SEMS is an essential part of the SEMP. A further discussion of the SEMS is given below. Describe the critical path analysis used to derive the SEMS and the supporting Systems Engineering Detailed Schedule (SEDS) and their structure.

C-5.5.8 TECHNICAL PERFORMANCE MEASUREMENT

Technical performance measurement (TPM) is a project control tool. The extent to which TPM will be employed should be defined in the SEMP. Describe the approach and methods to identify and control critical technical parameters. Include update frequencies, level of tracking depth, and response time to generate recovery plans and planned profile revisions (if the customer has not provided such information in the RFP). Include identification of related risks. The interrelationships between the selected critical parameter and lower-level parameters that must be measured to determine the critical parameter achievement value should be depicted in the form of tiered dependency trees and reflect the tie in to the related system performance requirement (critical parameter). Define the correlation of each parameter in the dependency tree a specific WBS element.

C-5.5.9 TECHNICAL REVIEWS

Technical reviews are essential to insure that the system being developed will meet requirements, and that the requirements are understood by the development team. The SEMP should list what technical reviews will be conducted and the methodology to be used in solving problems uncovered in reviews. Typical reviews include:

TIM	Technical Interchange Meeting
SRR	System Requirements Review
SDR	System Design Review
SFR	System Functional Review
IPDR	Internal Preliminary Design Review
PDR	Preliminary Design Review
CDR	Critical Design Review
SWDR	Software Design Reviews
QPR	Quarterly Progress Review
FCA	Functional Configuration Audit
PCA	Physical Configuration Audit

The SEMP should tailor this list to the project being proposed, and may provide information on the items to be completed as a prerequisite for holding the review. Describe the technical reviews and/or audits (major, subsystem, functional, and interim system) applicable to the project phase and the approach and procedures planned to complete identified reviews and/or audits. Describe the tasks associated with the conduct of each review, including responsibilities of personnel involved and necessary procedures (e.g., action item close-out procedures). Describe how compliance with RFP/contract performance and design requirements will be determined; how discrepancies identified as not meeting contractual requirements will be handled; and how system products and processes assessed to have a moderate to high risk of compliance should be addressed in order to comply with the contract, SEMS, and/or success criteria prior to conducting a review. The SEMS will show when these reviews are scheduled.

C-5.5.10 SUPPLIER CONTROL

Supplier control is a subset of interface management (see Paragraph 5.5.5). It may be defined in the SEMP as a part of the interface management plan or may be separated for special emphasis. The potential suppliers should be listed and evaluated. Describe the technical control of suppliers and subcontractors. For example, will a subcontracts manager and/or subcontracts Systems Engineer be assigned? How many subcontracts and purchased items, and approximately what dollar value will be assigned to each subcontracts manager and subcontracts Systems Engineer?

C-5.5.11 REQUIREMENTS TRACEABILITY

Requirements Traceability is done as a part of the decision database. Otherwise, provide a separate description in this section of the SEMP. Each requirement of the system should be traced to an originating requirement. If an originating requirement should change during the course of the project, then this traceability will facilitate identification of all related detail system requirements.

Describe how requirements traceability will be implemented. This includes the traceability between Systems Engineering process activities, work breakdown structures and correlation, as pertinent, and the SEMS and the SEDS. The traceability of requirements through the data management system should be described. Any automated requirements traceability tool should be described along with how this tool supports the process.

C-6 TRANSITIONING CRITICAL TECHNOLOGIES

Describe key technologies for the program and their associated risks. Include the activities and criteria for assessing and transitioning critical technologies from technology development and demonstration programs. When moderate to high-risk technologies are assessed, as required to meet performance and functional requirements, include a description of how alternatives will be identified and selection criteria established to determine when and which alternative will be incorporated in the product. Describe the planned method for engineering and technical process improvement including procedures for establishing preplanned product improvement or evolutionary acquisition strategies. Assess the impact on manufacturing of design and specification changes.

Transitioning critical technologies should be done as a part of risk management. It is called out separately here for special emphasis. Identify what technologies are critical and follow the steps outlined for risk management. Reference the work done (to be done) in this paragraph of your SEMP.

C-7 INTEGRATION OF THE SYSTEMS ENGINEERING EFFORT

This section describes how the various inputs into the Systems Engineering effort will be integrated and how interdisciplinary teaming will be implemented to involve appropriate disciplines in a coordinated Systems Engineering effort. Required Systems Engineering implementation tasks include:

Task	Paragraph
Team organization	7.1
Technology verifications	7.2
Process proofing	7.3
Manufacturing of engineering test articles	7.4
Development test and evaluation	7.5
Implementation of software designs for system end items	7.6
Sustaining engineering and problem solution support	7.7
Other Systems Engineering implementation tasks	7.8

C-7.1 TEAM ORGANIZATION

The SEMP should report the results of the creation of a project team. The results could be organized along the lines of the following typical tasks that the teams need to accomplish. To follow concurrent engineering practice the teams should be multidisciplinary, including customers, suppliers, and key domain experts, and should be end-item oriented. An example of such a team structure is shown in Figure C-1. The overall project is controlled by the System Team of Teams. This team includes the Lead Systems Engineer, Technical Director, and a representative from the segment team of teams. It may include the program manager, financial manager, and other important disciplines for performing the overall program. It may also include a representative from very significant or high-risk subsystems or even end items and if applicable a representative from a tiger team of one or more important performance thread tiger teams that cross many end items. Normal coordination of end-item information will be through a representative from related end-item teams functioning to make up a subsystem team of teams. The Subsystem team in turn sends a representative to the Segment team, and so on until the overall project team is formed.

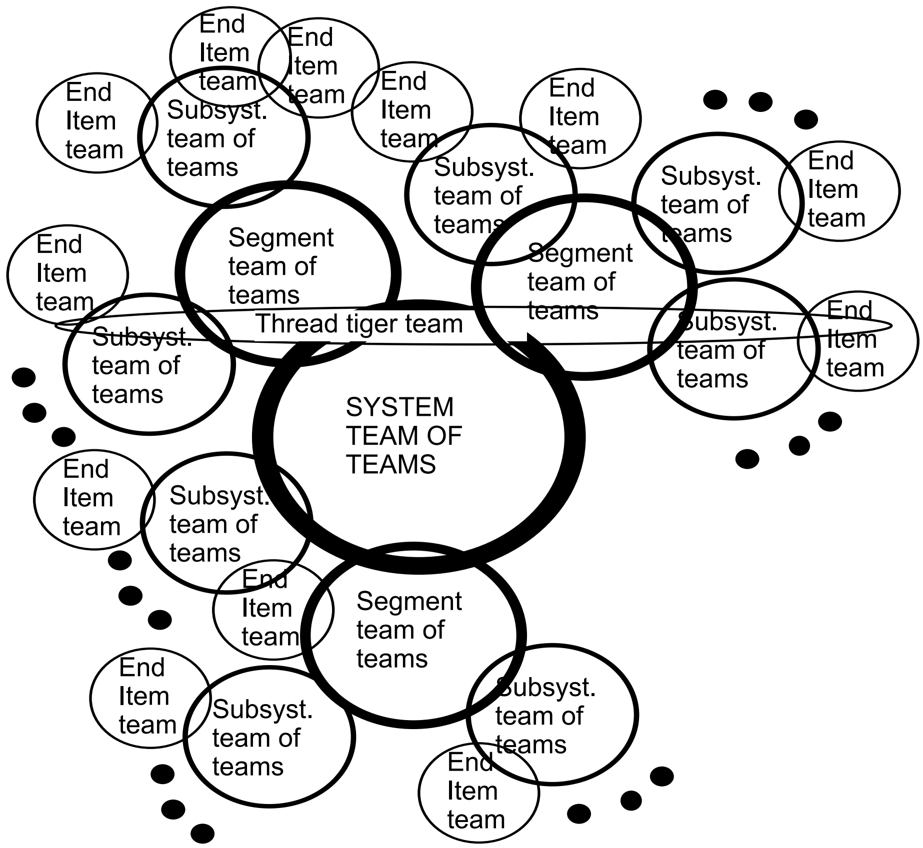


Figure C-1. Team Organization Example

Show how the appropriate disciplines are integrated into a coordinated Systems Engineering effort that meets cost, schedule, and performance objectives. Include how your organizational structure supports team formation; the composition of functional and subsystem teams; and the products each subsystem and higher-level team will support (e.g., teams organized to support a specific product in the WBS and "team of teams" utilized for upper level WBS elements). Describe major responsibilities and authority of the Systems Engineering team members and technical parties by name, and include present and planned program technical staffing. This part may include planned personnel needs by discipline and performance level, human resource loading, and identification of key personnel.

C-7.2 TECHNOLOGY VERIFICATIONS

In this section answer the following questions. What composition of team is needed for technology verifications? Does it include IPPD representation? Is there cost-benefit analysis expertise available? What are candidate technologies being considered for the proposed system?

C-7.3 PROCESS PROOFING

In this section answer the following questions. What composition of team is needed for process proofing? Does it include IPPD representation? What studies has your company done in the past on process definition and process improvement? What efforts are currently ongoing? How do they relate to the proposed system? What process synergy exists between the proposed project and other previous or concurrent projects?

C-7.4 MANUFACTURING OF ENGINEERING TEST ARTICLES

In this section answer the following questions. What composition of team is needed for manufacturing of engineering test articles? Is manufacturability and producibility proposed as a concurrent activity with development engineering? How will the Technology, Process, and Development teams relate to Manufacturing? Or will end-item teams all consider technology, process, and development of their end item?

C-7.5 DEVELOPMENT TEST AND EVALUATION

In this section answer the following questions. What composition of team is needed for development test and evaluation? Can these test articles be built on the actual production line - or at least with production equipment in the IPPD lab? What is expected to be learned from engineering test articles? Is a coordinated study considered for overall life-cycle cost optimization; where the overall cycle includes development, manufacturing, verification, deployment, operations, support, training, and disposal?

C-7.6 IMPLEMENTATION OF SOFTWARE DESIGNS FOR SYSTEM END ITEMS

In this section answer the following questions. What composition of team is needed for implementation of software designs for system end items? Will Software Systems Engineering be done to define software requirements and integrate them with overall system life cycle requirements? Can software be tailored to meet these requirements? What tradeoffs are needed and which teams should perform them? How do software risks fit with other system risks? Will software staff be assigned to each end-item team? If so, will there be an overall software team of teams?

C-7.7 SUSTAINING ENGINEERING AND PROBLEM SOLUTION SUPPORT

In this section answer the following questions. What composition of team is needed for sustaining engineering and problem solution support? What sustaining engineering is anticipated? Does it include hands-on factory labor inputs? Is sustaining engineering consistent with risks identified? Is up front Systems Engineering expected to reduce integration, test, and sustaining engineering and potential future problems?

C-7.8 OTHER SYSTEMS ENGINEERING IMPLEMENTATION TASKS

In this section answer the following questions. What composition of team is needed for other Systems Engineering implementation tasks? Identify what the other tasks applicable to this program are, what team efforts will be necessary to address them, and how the teams interact.

C-8 ADDITIONAL SYSTEMS ENGINEERING ACTIVITIES

Other areas not specifically included in previous sections but essential for customer understanding of the proposed Systems Engineering effort and/or scoping of the effort planned include:

Task	Paragraph
Long-lead items	8.1
Engineering tools	8.2
Design to cost	8.3
Value engineering	8.4
System integration plan	8.5
Compatibility with supporting activities	8.6
Other plans and controls	8.7

C-8.1 LONG-LEAD ITEMS

Describe long-lead items that affect the critical path of the program. Are long-lead items consistent with the risk analysis done? Can delivery time of any of these items be shortened? In particular, are there any workaround, substitution, or contingency possibilities or plans?

C-8.2 ENGINEERING TOOLS

Describe the Systems Engineering tools that will be used on the program. The subject of engineering tools is a dynamic topic. Discussions on this subject must be tempered by recent tool availability, and be realistic about what can be accomplished. Significant accomplishment requires a familiarity of the staff with the tool proposed. Expectations usually become reality only after a reasonable amount of experience in applying the tool.

C-8.3 DESIGN TO COST/COST AS AN INDEPENDENT VARIABLE

In some sense every project is a design to cost effort. There is an upper limit to affordability for every system. A combination of the value of the system and the resources of the customer will establish this limit. In many commercial programs, particularly those for which the market has yet to be established, cost is treated as an independent variable in trade-off studies used to steer the product design and development. In this section demonstrate the extent that these principles been applied in the proposed system solution. Show that you have considered the affordability aspect of the system requirements.

C-8.4 VALUE ENGINEERING

In this section answer the following questions. Can the project be engineered to have significantly more value with minimal additional cost? If so, is the particular significant increase also significant to the customer? Does the customer have the resources for even the modest cost increase for the improvement? Conversely, can the project be made to cost significantly less for only a minor decrease in value? Does the customer consider the specific identified decrease in value to be minor? Would the customer consider the specific identified decrease in value to be acceptable?

C-8.5 SYSTEM INTEGRATION PLAN

Describe the process by which the system is integrated and assembled together with emphasis on risk management. This should flow from or summarize the work discussed above under sections 6 Transitioning Critical Technologies, and 7 Integration of the Systems Engineering Effort.

C-8.6 COMPATIBILITY WITH SUPPORTING ACTIVITIES

Describe compatibility with supporting activities. Team Organization should address this concern. Refer to those discussions here, emphasizing integration of supporting activities into the individual team and overall team of teams organization.

C-8.7 OTHER PLANS AND CONTROLS

Describe any other plans and controls you will use. Since the gamut of known plans and controls are included in the descriptions above, these other plans would relate to specific, unique requirements for the system being proposed. Another possibility is the use of new techniques that were not foreseen at this writing. Either of these may supersede some of the plans and controls above. If so, this should be noted in the SEMP.

C-9 NOTES AND APPENDICES

Notes contain general information that aids in understanding the SEMP (e.g., background information, glossary). Appendices may be used to provide information published separately for convenience in document maintenance (e.g., charts, large drawings, and classified data). As applicable, reference each appendix in the main body of the SEMP where the data would normally have been provided.

Background information may include reasons for taking certain approaches, why certain requirements or risks were emphasized, etc. Include an alphabetical listing of all acronyms, abbreviations, and their meanings as used in the SEMP.

Appendix A may be bound separately. This appendix could provide a definition and plan for the classified requirements of the system. This may be a complete, stand-alone document or a simple listing of parameters that are omitted from the SEMP to make it unclassified.

Appendix B may include drawings or other large items, which would be difficult to include in the main body of the SEMP.

Appendix C etc. as required.

C-10 SYSTEMS ENGINEERING SCHEDULING

This section of appendix C further elaborates the work involved in generating the SEMS and SEDS sections of the SEMP.

A Systems Engineering Master Schedule (SEMS) contains identification of the major project phases and milestones and their entrance and exit criteria. Specific dates are intentionally separated into the Systems Engineering Detailed Schedule (SEDS). The separation is important in that the information in the SEDS can be expected to change; however, information in the SEMS requires customer or executive management approval prior to change. Both the SEMS and SEDS must be updated periodically to accurately reflect programmatic and technical changes. Program progress should be accurately shown on both the SEMS and SEDS to provide visibility into current status.

C-10.1 SYSTEMS ENGINEERING MASTER SCHEDULE

The Systems Engineering Master Schedule is the top-level process control and progress measurement tool to ensure completion of identified accomplishments. The SEMS accomplishments with their supporting criteria include those necessary to provide critical technical inputs and decision data into engineering (technical) and program decision points, demonstrations, reviews, and other identified events. The SEMS accomplishment structure provides the logical flow from necessary accomplishment to the successive accomplishments relying on their predecessor.

Identify the significant accomplishments that must be achieved by established contract events. Include, as a minimum, the events, accomplishment and associated success criteria identified by the customer, if applicable. Include in-process verifications of required accomplishments before proceeding with further technical efforts that rely on successful completion of those accomplishments. Reflect the integration of the efforts necessary to satisfy required accomplishments. Relate each event and accomplishment to an element in the WBS.

a. Identify events in the format of entry and exit events (i.e., Initiate PDR and Complete PDR) or use entry and exit criteria for each event.
b. Use event-related and not time-coincidental or driven accomplishments.

   SEMS accomplishments should have one or more of the following characteristics:

(1) Define a desired result at a specified event that indicates design maturity or progress directly related to each product and process.
(2) Define completion of a discrete step in the progress of the planned development.
(3) Describe the functional activity directly related to the product.

c. Use measurable criteria (for example, “Test Plan Complete” is a measurable criterion, whereas “Test Plan 85% Complete” is not a measurable criterion).

Provide a definitive measure or indicator that the required level of maturity or progress has been achieved. This can include work effort completion that ensures closure of the accomplishment. Examples of SEMS accomplishment criteria include:

(1) Test plan complete.
(2) Safety significant item list finalized.
(3) Supportability requirements implemented in design.
(4) Achievement to date of a technical parameter within TPM tolerance band and current estimate satisfies threshold.

d. Use customer input to identify critical TPM parameters to be used as accomplishment criteria for identified milestones. Select TPM parameters on the basis of risk and to define performance in meeting all critical performance parameters.

e. The SEMS should include:

(1) Schedule phases, critical inputs & outputs for all major activities
(2) CDRL deliverable milestone schedule
(3) Program reviews and audits
(4) Major milestones
(5) Payment milestones

A SEMS task or milestone is complete when all accomplishment criteria identified for it are successfully demonstrated.

f. Develop the SEMS from the "bottom-up" aggregation of WBS task schedules, ensuring that all contract-specified events and milestones are incorporated at the achievable point and other key items, identified by the performing organizations are also included. An example of the SEMS development process is illustrated in Figure C-2.

C-10.2 SYSTEMS ENGINEERING DETAILED SCHEDULE (SEDS)

This section offers a practical routine for the creation of the SEDS. The goal is to develop a calendar-based schedule to support the events and tasks identified in the SEMS.

1. The Systems Engineering Detailed Schedule (SEDS) is constructed in a "bottom-up" fashion, beginning from the expansion of each WBS Task into its constituent subtasks and planning the execution and integration of each of those subtasks into the WBS Task as portrayed in Figure C-2.

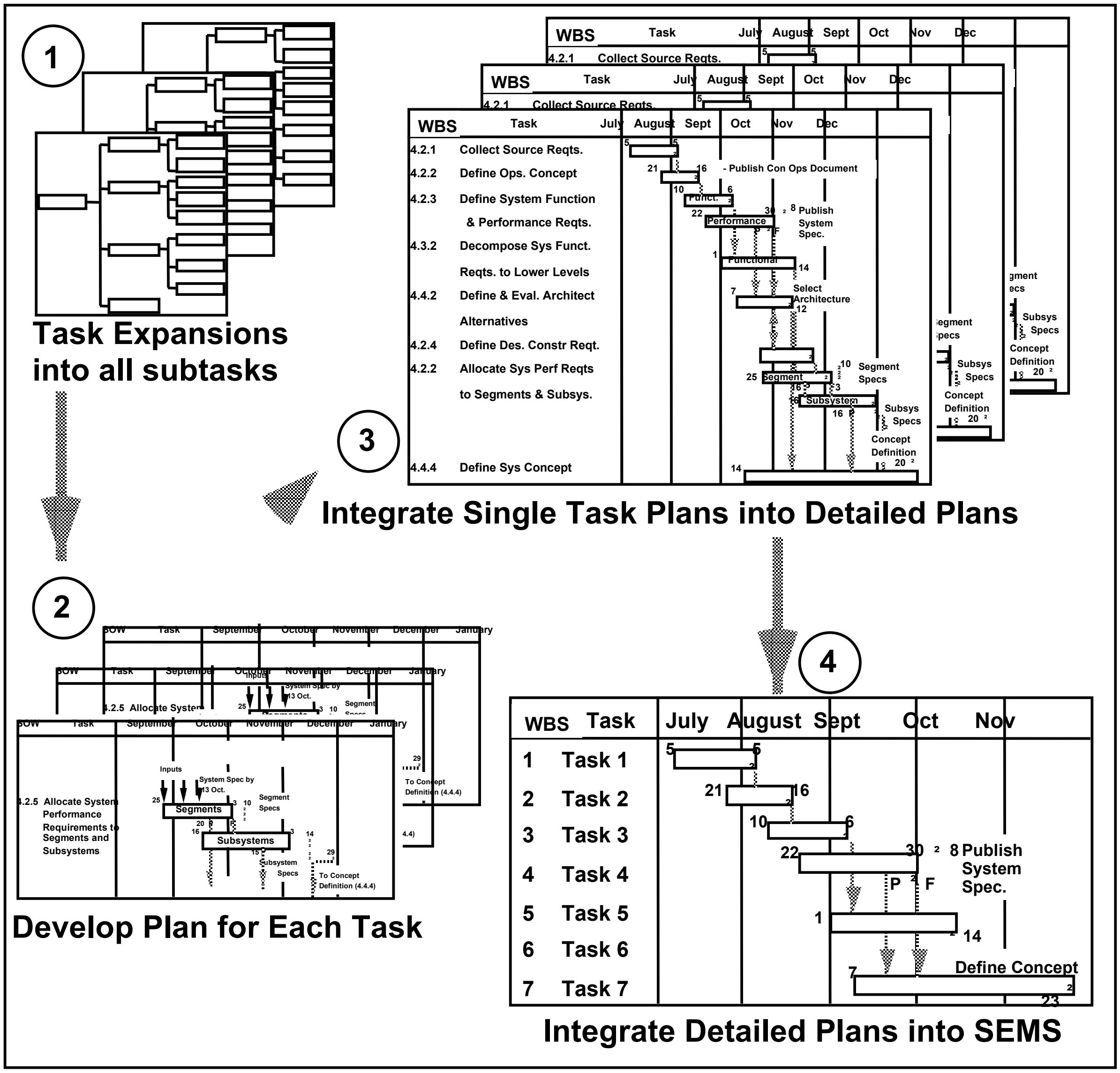


Figure C-2. The SEMS Development Process

2. Use Tree Diagrams or other appropriate means to subdivide each WBS into executable, assignable, budgetable steps. Lay out a calendar-based schedule for integrating those steps. Show what and when inputs and outputs are needed for each subtask. Identify and schedule meaningful milestones - at least one per month per task (everyone on the program should always know which milestone they are working on and when their inputs are needed).

3. Preparing these schedules is a consensus-building effort. The task leader wants more time and resources; the project managers need to ensure meeting contract commitments. In the end the task leader and team must commit that they can deliver the desired product within the agreed time and resources or other sources must be sought.

4. Numerous milestones are essential if a milestone-based status tracking system is used, such as Cost/Schedule Control System Criteria (C/SCSC), where "Earned Value" is based on milestones completed.

5. Show the date of the specific day when each schedule activity begins and ends, including both preliminary and final outputs of the activity. Show specific dates for all milestones. Use the common assumption that the item is due as of the close of business (COB) local time (say 5 pm) unless specifically defined otherwise for the item or by general policy of the project. Milestones are darkened-in when completed.

6. (Optionally/preferably) on the chart, but certainly defined with the WBS item is the person responsible for completion of the task and the resources allocated, i.e., Joe Reed, 2,500 Hr.

7. An example of a schedule summary for one task (or significant subtask) is shown in Figure C-3. Note how delays are shown on the chart. A revised schedule is not prepared to hide the delays of the past. This has the positive effect of providing incentives for people to meet their commitments (especially when their name is displayed as responsible for the task).

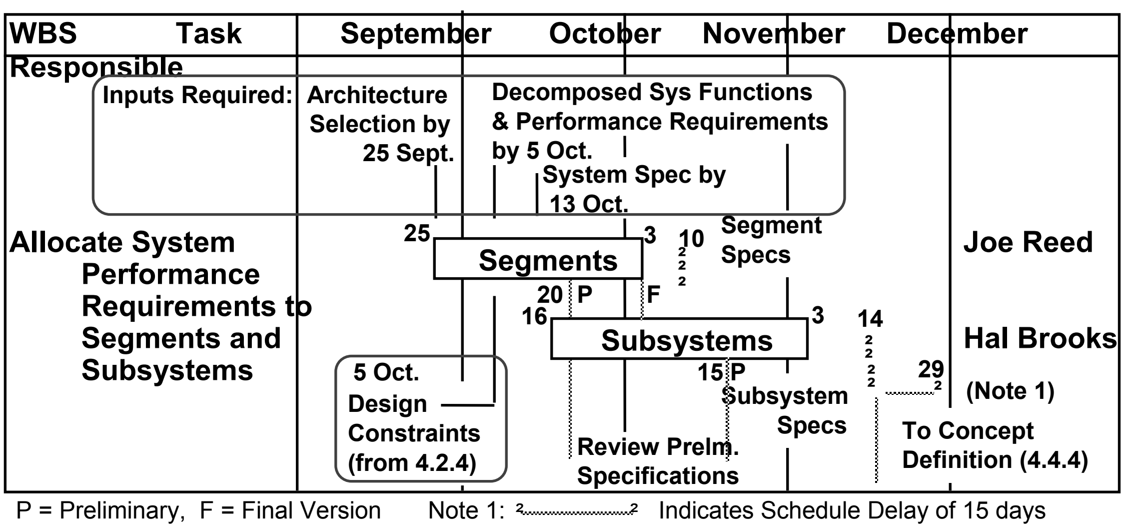


Figure C-3. Example of Task Overview Schedule

8. Individual tasks are integrated to develop the SEDS. At this point there is some manipulation of the individual task schedules (unless coordinated earlier) to achieve a smooth overall task flow.

9. Issues of schedule critical path and slack time should be addressed during the task planning process. If the critical (longest) schedule path can be alleviated by subtask schedule adjustments to avoid slack time, this should be done.

   An example of an Activity Network Diagram is shown in Figure C-4. This is a tool to determine critical path length and slack time. In the example shown, only the nominal time for each task is used for deterministic results. A variation on this technique, known as PERT, provides a statistical estimate of path completion times. Consult Reference 12 in Section 1 for details on these calculations.

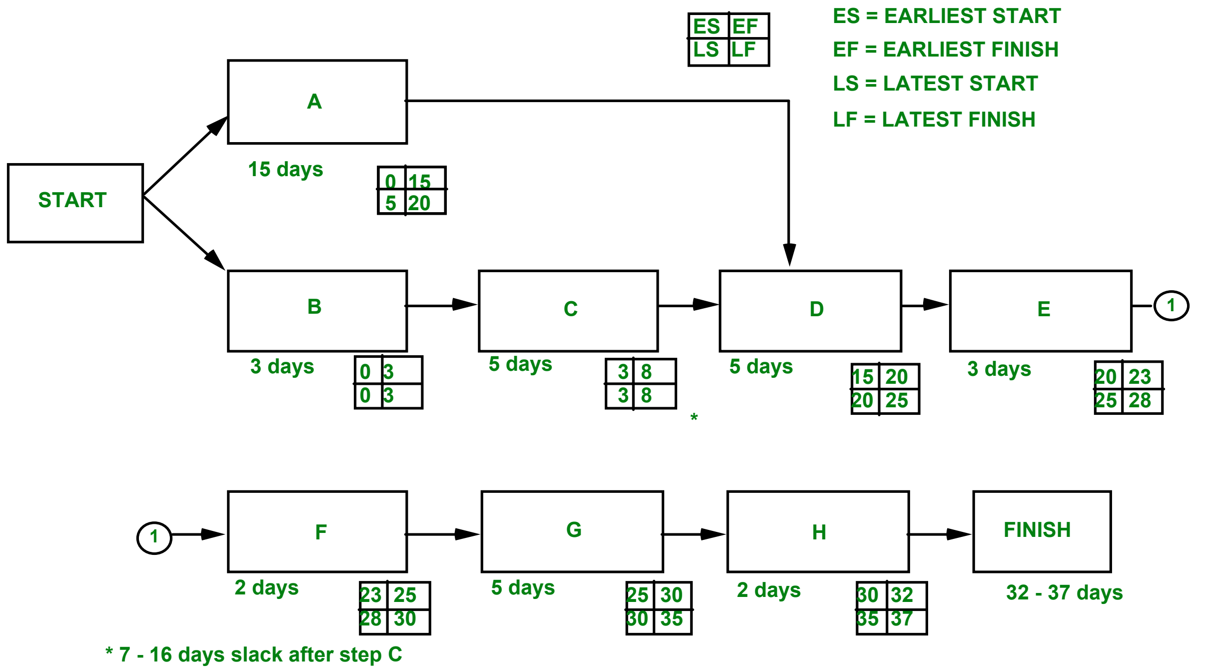
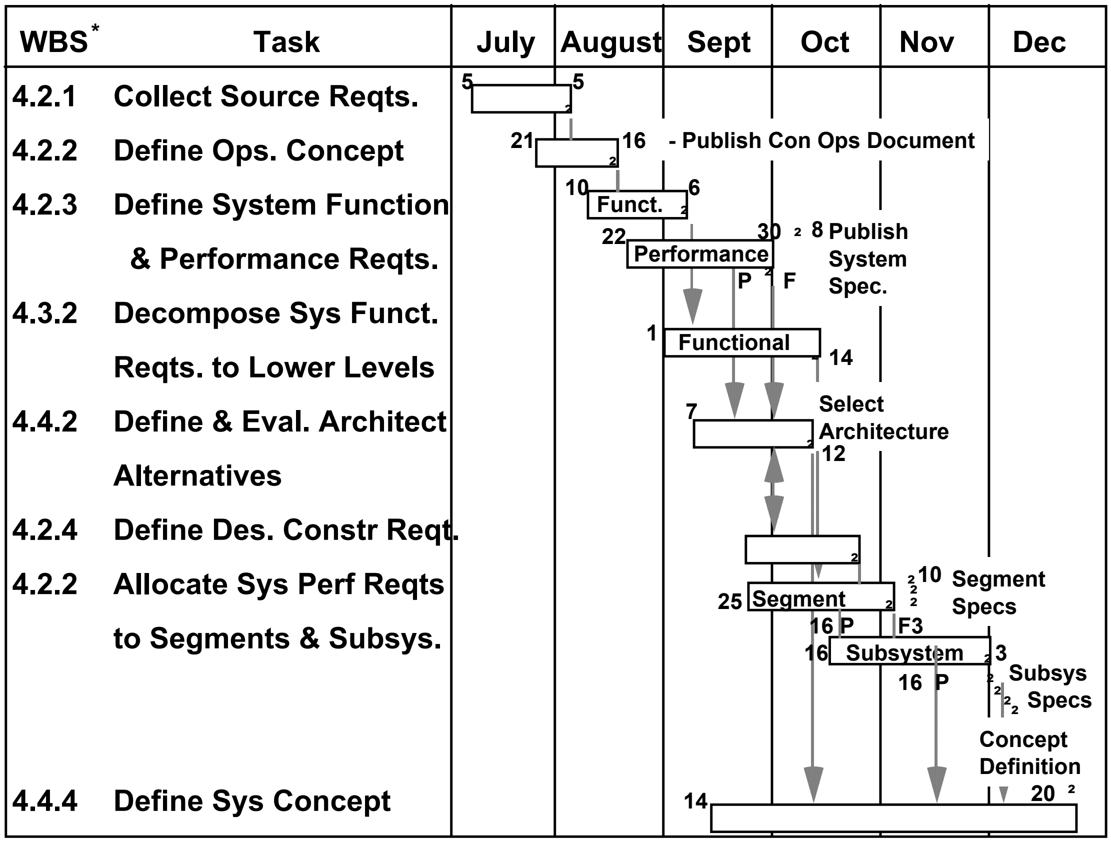


Figure C-4. Activity Network Diagram

10. Other schedule risk reduction measures may be necessary, including resource reallocation, work-around plans, second source suppliers, and carefully-monitored use of concurrent (overlapping) development. Section 7 contains risk management techniques.

An example of an integrated, multi-task, six-month Systems Engineering effort for a Concept Definition phase program is shown in Figure C-5. This is incomplete (even for the Systems Engineering effort) and only part of a simplified SEDS. Similar schedules for supporting efforts (such as the development engineering efforts to define the preliminary concepts for the system's segments and associated subsystems) should also be prepared. It is good practice to also show all major milestones along the bottom of the chart (omitted here) so they can be rapidly identified.



* The "WBS" numbers in this example correspond to this Handbook's SE process sections
P = Preliminary Draft F = Final Output Date

Figure C-5. A Simplified SEDS Example

C-11 SYSTEMS ENGINEERING PROCESS METRICS

This section provides a brief discussion of Process Metrics. For a more comprehensive coverage of Metrics, refer to the INCOSE Metrics Guide. This Section is focused on process metrics, including ways to status cost and schedule performance and some other basic control tools, which can be applied to Systems Engineering activities.

C-11.1 COST AND SCHEDULE PERFORMANCE MEASUREMENT

One of the best ways to accurately measure cost and schedule performance is using an earned value technique. Earned value provides a measurement of work accomplishment compared with resource expenditure (cost or effort). This technique is described in the Earned Value Measurement System (EVMS) ANSI Standard (ANSI/EIA-748) for defense acquisition.

The ANSI/EIA-748 earned value management system guidelines incorporate best business practices to provide strong benefits for program or enterprise planning and control. The processes include integration of program scope, schedule, and cost objectives, and use of earned value techniques for performance measurement during the execution of the program. The system provides a sound basis for problem identification, corrective actions, and management re-planning as may be required. Earned value can be an even more effective indicator of program health when correlated with other WBS based metrics such as Technical Performance Measurements.

The guidelines are purposely high-level and goal oriented as they are intended to state the qualities and operational considerations of an integrated management system using earned value analysis methods without mandating detailed system characteristics. Different companies must have flexibility to establish and apply a management system that suits their management style and business environment. The system must, first and foremost, meet company needs and good business practices.

While EVMS is an excellent project management tool, it is often criticized as being very expensive to implement. The formality required can be costly; however, an earned value system does not have to be expensive to implement. As cited in the new ANSI/EIA Standard, formality needs to be adjusted to project requirements. There is considerable planning effort required to use the system, so it should be the primary cost and schedule data collection and reporting system for the program, not a redundant duplication of other systems.

The basic concept of EVMS is to establish an "Earned Value" for the work accomplished on the project to date. This Earned Value is called the "Budgeted Cost of Work Performed (BCWP)". It is compared with the planned effort (to date) to determine the Schedule Variance (SV), in dollars or days. The planned effort is called "Budgeted Cost of Work Scheduled (BCWS)". The Schedule Variance is computed as follows:

$$SV = BCWP - BCWS \qquad \text{(Eqn. 1)}$$

A negative value of SV indicates a "behind schedule" situation. The SV is computed in Dollars, but can be converted to Days by referring to the plot of BCWP vs. time (days). The Schedule Performance Index (SPI), (BCWP/BCWS), is a good indicator of actual performance efficiency. An SPI<1 indicates a behind schedule situation on the element(s) measured.

BCWP is compared to the Actual Cost of Work Performed (ACWP) to determine the Cost Variance, as follows:

$$CV = BCWP - ACWP \qquad \text{(Eqn. 2)}$$

A negative value of CV indicates the cost overrun to date, in dollars. The Cost Performance Index (CPI), (BCWP/ACWP), is a good indicator of the actual cost efficiency. A CPI<1 indicates an overrun situation on the element(s) measured.

The industry standard metric for completion cost predictability is referred to as the “To Complete Performance Index (TCPI)”. The TCPI is the ratio of the work remaining (Budget – EV) to the funding required to complete that work (Estimate to Complete (ETC) or Estimate at Complete (EAC) – ACWP). With an accurate ETC, a TCPI > Cost Performance Index (CPI) indicates that the remaining work will be performed at a productivity level HIGHER than the program has been experienced. Conversely, a TCPI < CPI indicates that to meet the EAC, a lower productivity is being assumed for the remaining work.

If the productivity level projected by the TCPI is not realistic then the validity of the EAC is in question.

TCPI = Work remaining/funds required

TCPI = (BAC – BCWP) / ((EAC – ACWP)

The relationship of these parameters is shown in their plot vs. schedule time in Figure C-6.

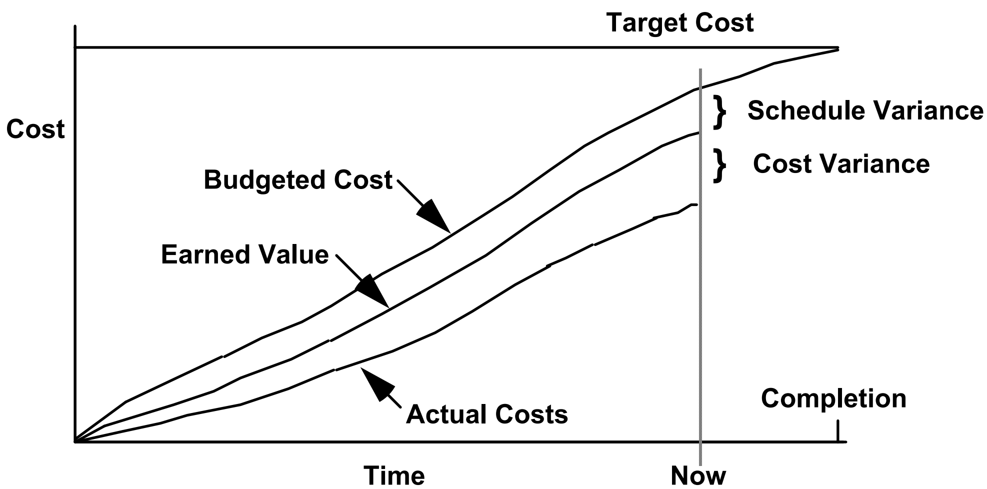


Figure C-6. Cost and Schedule Variance under C/SCSC

C-11.2 OTHER PROCESS CONTROL TECHNIQUES

Four useful techniques are: the Run Chart, the Control Chart, the Process Flow Chart, and the Scatter Diagram. Examples of these are shown in Figure C-7.

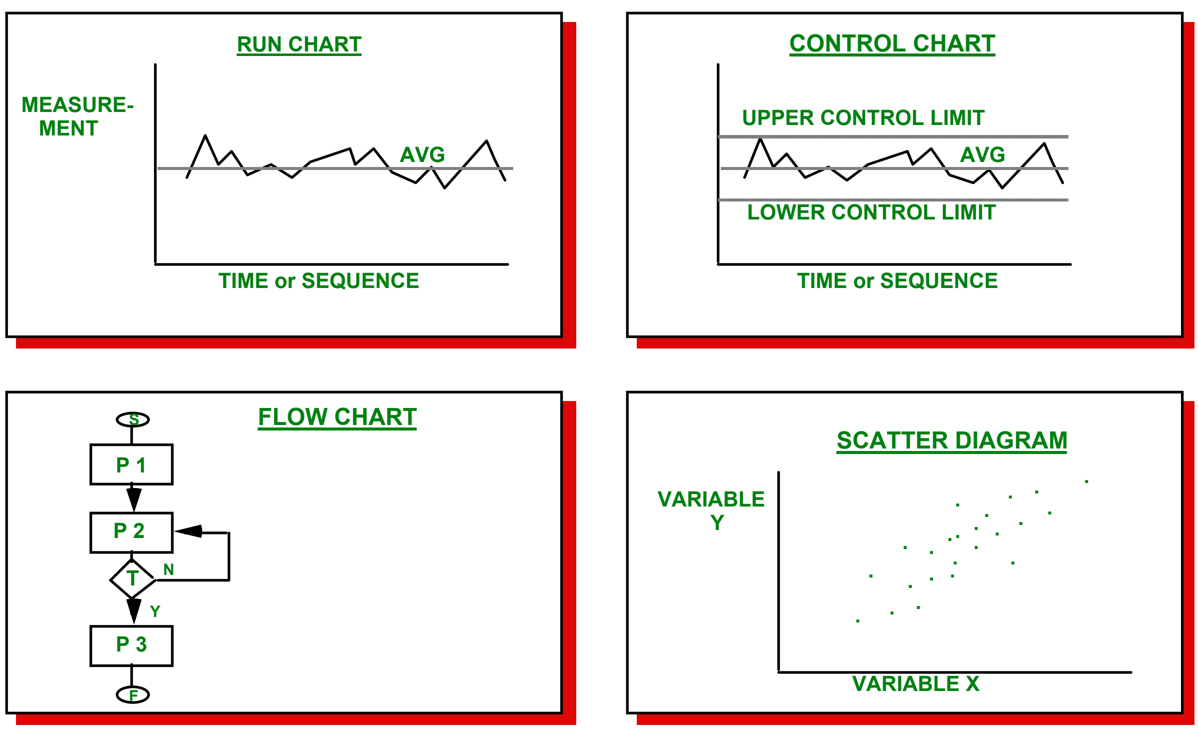


Figure C-7. Other Process Control Techniques

The Run Chart is simply a plot of a key measurement during the run time or sequence. It permits visual display of trends. This can be used for a variety of purposes. Some examples are:

1. Tracking labor charges or lost productive time per week for several months.
2. Plotting a technical parameter vs. time.
3. Tracking design drawing check errors vs. time.
4. Tracking key manufacturing tolerances vs. time

The Control Chart is a Run Chart with statistically-determined limits drawn on both sides of the process average. There are two types of Control Charts: The X Chart and the R Chart. The X Chart is used to monitor the means of a process. The means are sampled and plotted vs. time. The upper and lower control limits are normally set at ± 3. Points that fall outside the limits should be investigated to determine if the process has changed. The R Chart is used to plot the range of the samples. It is used to check on the dispersion of the process. Consult Reference 12 in Section 1 for details on this technique.

The Process Flow Chart uses normal flowcharting techniques to describe a process.

The Scatter Diagram is a plot of all observations of two variables vs. each other to help determine if there is any correlation between them. If the points are randomly scattered around the chart they are not correlated.

C-12 ROLE AND FUNCTION OF REVIEWS AND AUDITS

Select meaningful reviews and audits for your program and schedule them for the appropriate time. The reviews and audits are for both customer and peer review. The major reviews should serve as gates for proceeding to the next stage in the product cycle.

1. After the primary task spans have been determined in the SEDS process, the major milestones can be scheduled, followed by the reviews and audits.
2. Start by considering appropriate times for review and audit of the individual CI or CSCI, then, allowing some reserve time for any subsequent major fixes, schedule the summary system-level reviews.
3. Consult the schedule in Figure C-8 for the appropriate time for major reviews, and the guidelines in Table C-1 for the rationale and critical issues associated with each review. A project may need more or less reviews. Remember that formal, documented reviews, with the customer in attendance can have a significant cost, so also use more-frequent informal, in-house reviews to resolve most issues; strive to exit the major, formal reviews with no major customer-imposed Action Items.
4. After the number, type, and schedule for the reviews and audits has been established, write down the criteria for entering and successfully exiting each review or audit. These criteria must be distributed to all who will have a role in preparing for or participating in the reviews and audits.
5. As each review/audit approaches, a chair and recording secretary are named, along with all other representatives of the review team. Select from experts knowledgeable in the areas being reviewed, both from within and outside the project.
6. Establish a method for recording, reviewing, tracking, and closing Action Items assigned during the review or audit, including a formal signoff by the review chairman, as appropriate.

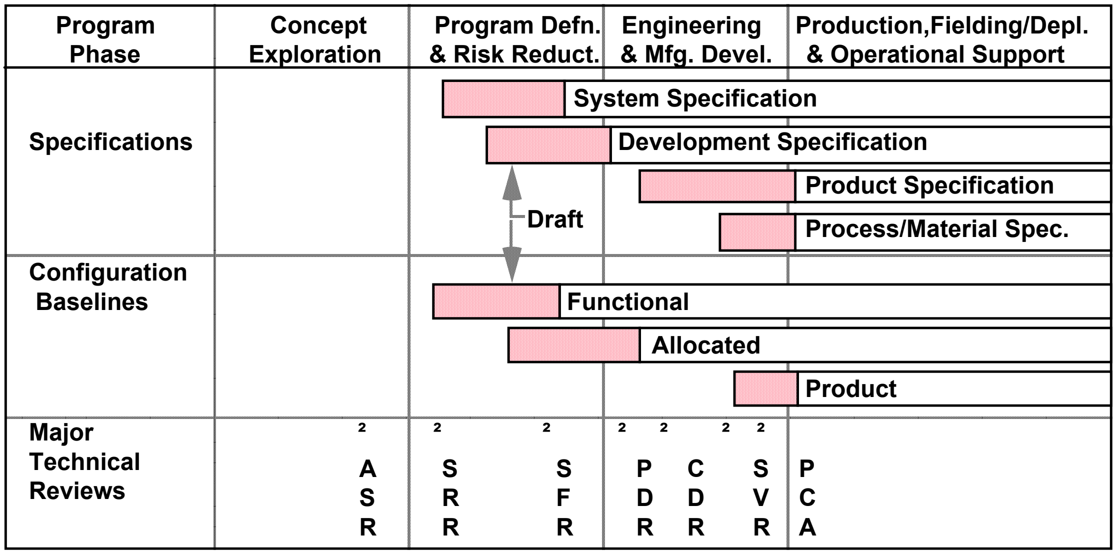


Figure C-8. Typical Schedules for Specifications, Baselines, and Reviews

Table C-5. Examples of Major Technical Reviews

REVIEW	WHEN *	PURPOSE
ASR Alternative System Review	Concept Exploration	Concept Study Complete Assess Future Risk Reduction Plans
SRR System Requirements Review	Early in PD&RR ** PFD&OS ***	Review Complete Draft System Specification Review Draft System Architecture
PDR Preliminary Design Review	PD&RR ** EMD** PFD&OS***	Prelm. Design Satisfactory to Proceed to Detailed Review Each CI and CSCI Hold System PDR after all CI/CSCis PDRs Complete Functional, Physical I/F Reqts. Defined
CDR Critical Design Review	EMD** PFD&OS***	Detail Design Satisfactory to Continue Development Review Each CI and CSCI Hold System CDR after all CI CDRs Establish ICDs Complete CI and CSCI Draft Product Specs. Complete
SVR System Verification Review	EMD** PFD&OS*** w FCA (if held)	Verify The System Is Ready for Production FCA for Each CI/CSCI Complete Verify Each CI/CSCI Conforms to Description Verify Mfg. Proofing and Capacity Verify Product & Process Design Stabilized
FCA Functional Configuration Audit	EMD** PFD&OS***	Establish that All CI Development Spec. Verification Tests are Complete vs. Requirements (New CIs)
PCA Physical Configuration Audit	EMD** PFD&OS***	Specifications and Design Documentation Complete Mfg. Process Reqts. & Documentation Finalized Product Fabrication Specs. Finalized

* See Sect. 9 for Acronyms **For New Developments Only *** For Mods & Product / Process Improvements

THIS PAGE
INTENTIONALLY
LEFT BLANK

APPENDIX D - METHODS FOR FUNCTIONAL ANALYSIS AND ALLOCATION WITH KEY SUPPORTING METHODOLOGIES

D-Introduction:

There are numerous methods that support functional analysis, allocation and decisions. The following is a brief description of some of the most common methods used in systems engineering. The purpose of Functional Analysis and Allocation is to clearly describe the system functionality, analyze the individual function inputs, process and outputs, and parse (divide, break apart) these functions into sub-functions and allocate them appropriately to sub-systems (hardware, software, human, databases etc.). Supporting the Functional Analysis and allocation are methods for making decisions on alternatives that are part of the analysis and allocation methods. The goal of allocation and analysis is to provide a system architecture that is balanced and optimized that meets the intended needs/requirements of the stakeholders. This process is an iterative and continues until the system is fully defined. This appendix provides an overview description of the following methodologies:

Functional Analysis and Allocation:
- D.1 - Functional Flow Diagrams (FFDs)
- D.2 - N^2 Charts
- D.3- Timeline Analysis
- D.4 - Requirements Allocation
- D.5 - Functional Thread Analysis
- D.6- Modeling and Simulation
- D.7 - Real-Time Structured Analysis
- D.8 - Object-Oriented System Modeling Decision Support
- D.9 - Analytic Hierarchy Process
- D.10 - Decision Analysis Technique for Risk Management

D.1 FUNCTIONAL FLOW DIAGRAMS

One of the best tools for functional analysis is the functional flow diagram (FFD). The Functional Flow Diagram is a multi-tier, time-sequenced, step-by-step diagram of the system functional flow. FFDs are usually prepared to define the detailed, step-by-step, operational, and support sequences for systems. But they may also be used effectively to define processes in developing and producing systems. They are also used extensively in software development. In the system context, the functional flow steps might include combinations of hardware, software, personnel, facilities, and/or procedural data actions. An example of a functional flow block diagram is given in Figure D-1.

Here are a few rules for common understanding of FFDs:

1. The top-level functions should be numbered with even integers and zero decimals, i.e., 1.0, 2.0, etc., and cover the complete span of life cycle functions anticipated from initial set-up and check-out through disposal.

2. Inputs to functions come from the left side, outputs from the right side, and lower level functions emanate from the bottom.

3. The name of the function is defined inside the box, replacing F1, F2, etc.

4. A reference function (ref) is indicated at the beginning and end of all functional sequences, EXCEPT AT THE TOP LEVEL.

5. An "OR" gate is used to indicate alternative functions; an "AND" gate is used to indicate summing functions, where all functions are required.

6. A "GO" "NO GO "sequence is indicated by an arrow out the right side with the letter G for GO and an arrow out the bottom with G-bar for NO GO.

7. It is customary, when the second level, or lower, is shown on a separate page, to list the title of the function at the top center of the page for reference.

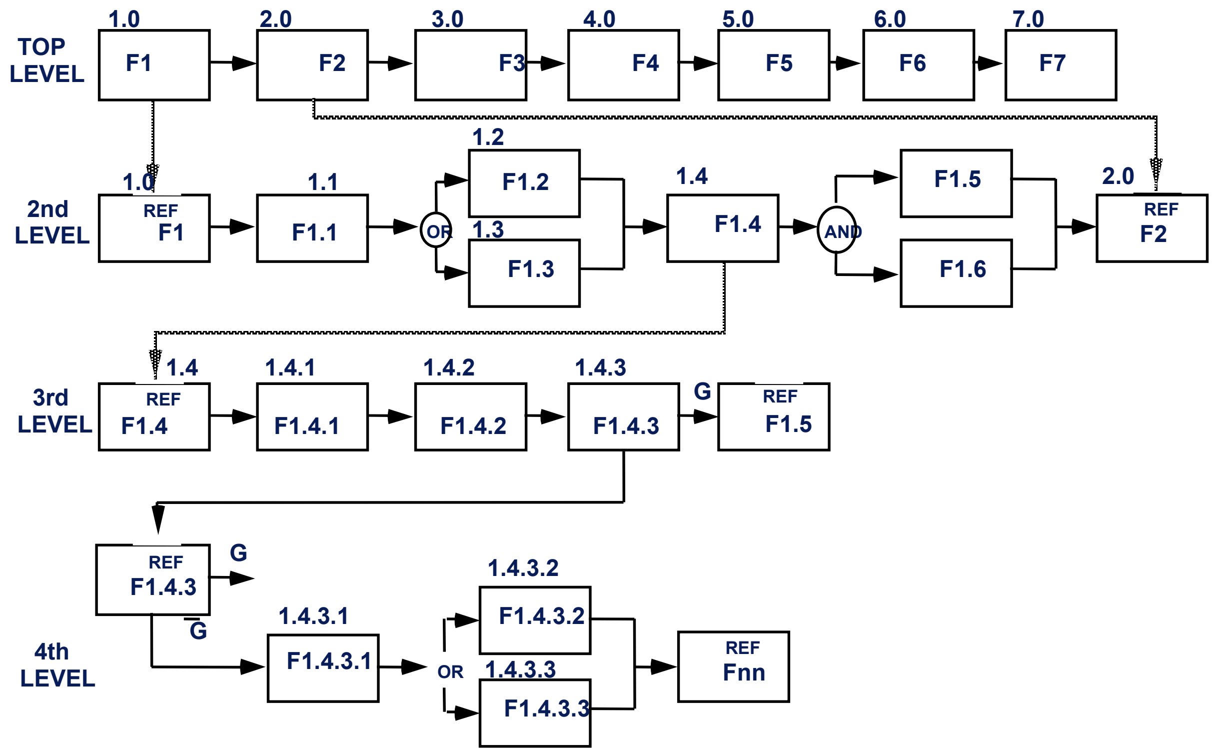


Figure D-1. Functional Flow Diagram Example

Multiple levels are shown in the above figure. Only the top level is complete. At lower levels only an example expansion of one function is shown. For example, at level 2, the Top Level Function, F1, is expanded into its 2nd level Functions, F1.1 through F1.6. At the third level, an example expansion of the 2nd level Function F1.4 is shown. Finally, at the 4th level, the Function F1.4.3 is expanded. Each level gives a different example of typical functional flow paths. Usually, only one or two levels is shown on one diagram to avoid confusion.

The information flow, content of each functional step, and timing details are not shown on the FFD. During functional analysis, interfaces are not shown on the FFD either. Later in the design process, when functions have been allocated to system elements, the FFDs will usually need to be revised extensively. Once the functions have been allocated to system elements, the element FFDs can be drawn with critical interactions between the elements shown on the FFDs. This can be very helpful in defining complex interfaces between system elements. System interface drawings (or specifications) between the elements will still be required to define all details.

D.2 N^2 CHARTS

The N2 chart is a systematic approach to identify, define, tabulate, design, and analyze functional and physical interfaces. N2 charts apply to systems interfaces and hardware and/or software interfaces. The N2 chart is a visual matrix, which requires the user to generate complete definitions of all the system interfaces in a rigid bi-directional, fixed framework. A basic N2 chart is illustrated in Figure D-2.

The system functions are placed on the chart diagonal. The remainder of the squares, in the N by N matrix, represents the interface inputs and outputs. Where a blank appears, there is no interface between the respective functions. Interfaces between functions flow in a clockwise direction. The entity being passed from function 1 to function 2 for example can be defined in the appropriate square. When a blank appears, there is no interface between the respective functions. When all functions have been compared to all other functions, then the chart is complete. If lower-level functions are identified in the process with corresponding lower-level interfaces, then they can be successively described in expanded or lower level diagrams. Sometimes characteristics of the entity passing between functions may be included in the box where the entity is identified. One of the main functions of the chart, besides interface identification, is to pinpoint areas where conflicts may arise between functions so that system integration later in the development cycle can proceed efficiently.

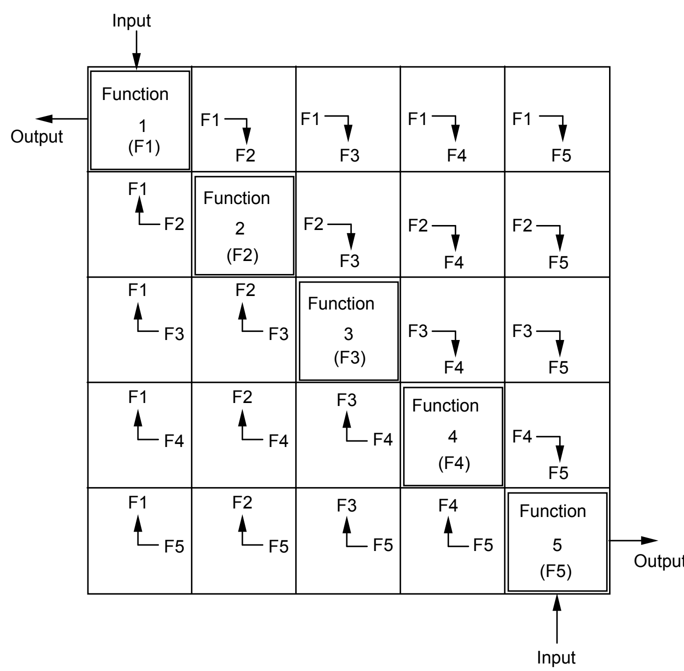


Figure D-2. N^2 Chart Definition

The N^2 chart has been used extensively to develop data interfaces, primarily in the software areas; however, it can be used to develop hardware interfaces. Data flows in a clockwise direction between functions; i.e., the symbol F1 --,> F2 indicates data flowing from function F1 to function F2; feedback is indicated by F2 ->,- F1. The data being transmitted can be defined in the appropriate squares. The N^2 chart can be taken down in successively lower levels to the hardware and software component functional level. In addition to defining the data that must be supplied across the interface, the N^2 chart can pinpoint areas where conflicts could arise.

References

Becker, Ofri, Ben-Asher, Joseph, and Ackerman, Ilya, *A Method for System Interface Reduction Using N^2 Charts*, Systems Engineering, The Journal of the International Council on Systems Engineering, Volume 3, Number 1, Wiley 2000.

Defense Systems Management College, "Systems Engineering Management Guide," Fort Belvoir, VA, 3 Oct 1983, page 6-5.

Lano, R. "The N^2 Chart," TRW Inc., 1977.

D.3 TIME LINE ANALYSIS

The FFD shows the sequential relationship among various functions, but it does not depict the actual duration, concurrent, or time overlap of the functions. Time line analysis adds the element of functional duration, and is used to support the development of design requirements for operation, test, and maintenance functions. It depicts in graphical form the concurrence, overlap, and sequential relationship of functions and related tasks and identifies time-critical functions. Time-critical functions are those that affect reaction time, down time, or availability. A generic example of what a time line analysis chart looks like is shown in Figure D-3.

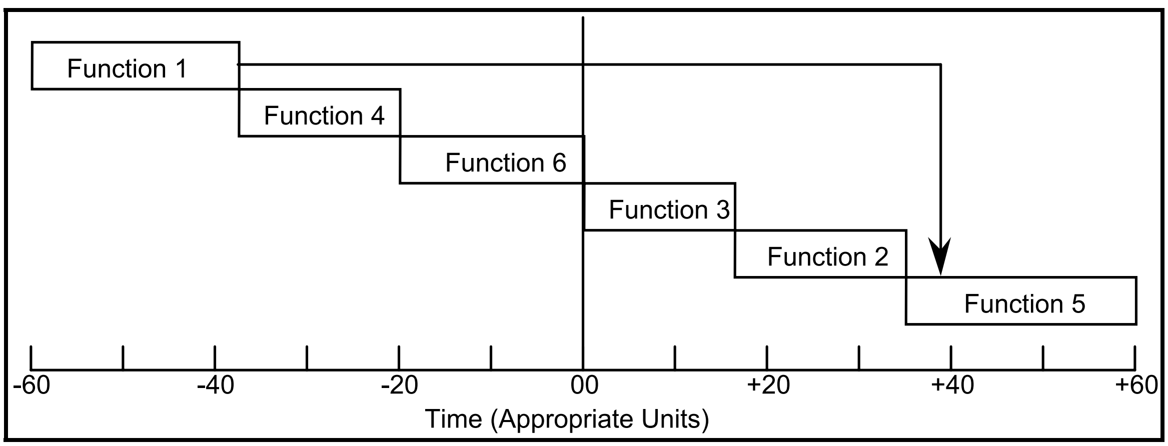


Figure D-3. Generic Example Of A Time Line Analysis Chart

D.4 REQUIREMENTS ALLOCATION

Performance requirements can be divided into allocable and non-allocable parameters. An example of the former, weight, is progressively divided at successively lower levels. An example of the latter is material and process standards, which are applied directly to all elements.

Allocable parameters can be divided into those that are allocated directly and those that are allocated indirectly. A fire control system pointing error is representative of directly allocated requirements in which the total pointing error is apportioned first to the various elements and then to subsystems and components. Indirectly allocated requirements are those that require an analysis to establish performance measures. An example of this would be the conversion of the mission requirements for aircraft target detection size and range into radiated power, pulse width, and timing stability which could then be used by the designer of the radar system in sizing his hardware. The top-level performance measures are used to derive lower-level subsystem requirements for configuring components. The process is documented for each requirement, identifying its source, and showing the allocation to the next lower level.

It is important to note that as a result of the system analysis and flow down, top-level functional requirements usually become lower-level performance requirements. For example:

- System - Transmit collected data in real time to remote ground site
- Segment - Provide wideband data link from spacecraft to relay
- Element - Provide 110 MHz link at 17.0 GHz
- Subsystem - Provide 10 MHz link at 17.0 GHz with 10 W effective radiated power for 20 minutes maximum per orbital revolution

In addition, support requirements from power, commands, and telemetry are developed and quantified. The most straightforward application of allocation is the direct apportioning of a value to its contributors. The resulting allocation for a specific area, such as pointing error, is usually referred to as a budget. The technical budget represents an apportionment of a performance parameter to several sources. This may be a top-down allocation, such as a pointing error budget.

D.5 FUNCTIONAL THREAD ANALYSIS

The purpose of this sub-section is to describe the use of stimulus-condition-response threads to control software development for specification, review and testing. This improves communication between system and software engineers and accuracy in requirements definition, review and verification. These threads can be used to control the software development process, including translation from system to software requirements, design verification, review of software test plans, and integration of software and system testing. The threads provide a way of enumerating the number of stimulus-response capabilities to be tested. Performance requirements can also be tied to them.

One of the biggest challenges facing Systems Engineers is the way in which they interface with the development of the software which implements the desired behavior of the system. Since the system behavior is primarily implemented in software, a critical issue in system development is: how should the Systems Engineers interact with the software engineers in order to ensure that the requirements for software are necessary, sufficient, and understandable? This is a question that must be addressed at the State of the Practice level; any approach must be inherently teachable to practicing Systems Engineers. Experience has shown that the approach of passing paper specifications between systems and software developers does not yield satisfactorily results.

How to perform Functional Thread Analysis

The central recommendation of this sub-section is that systems and software engineers should work together to identify the system level threads, and the subset of the threads which must be supported by the computer system. In this context, a "thread" consists of the system input, system output; a description of the transformations to be performed; and the conditions under which this transformation should occur. Such threads can be represented textually or graphically in a variety of ways, some of which are better than others and some of which are supported by tools.

Such threads satisfy all of the criteria for good communication between system and software developer:

a. The identification of a thread from input to output allows both system and software engineers to identify the subthread that should be allocated to the processing subsystem, and hence software;

b. The description of stimulus-condition response threads eliminates the ambiguities found in current specifications;

c. The description of threads is inherently understandable by both systems and software engineers, particularly if provided in some graphical format; and

d. The use of such threads aids both system and software designers in evaluating the impact of proposed changes.

The use of threads for systems and software is not a new idea. Reference 1 presents an overview of the history of this concept concluding that there is considerable reason to believe that the use of threads for specification and communication between systems and software engineers is feasible even if traditional specification techniques are used at the system level, threads can be used to validate these specs and then refined to show the allocation to the software.

In the steps that follow, it is assumed that the development of the software requirements is an evolutionary process, starting with allocation of processing requirements to a processing system, and ending with publication and review of the software requirements. Ignore for the moment the problems associated with the design of the distributed computing hardware system on which the software may reside, which is discussed in Reference 2.

Step 1. Deriving the System Level Threads for Embedded Systems

No matter how the system description is developed, even if it is no more than the identification of system functions for different modes of operation, at some point the system inputs and outputs must be identified in order to anchor the specification to reality.

This starts with the initial scenarios, which describe the system's intended operations. These can be rewritten into the form of stimulus-response threads.

To illustrate this, consider the ever popular Bank Automatic Teller Machine ATM System, which accepts ATM cards and enables customers to withdraw cash etc. Figure D-4 presents two top-level scenarios which describe the top level behavior of the ATM system when presented with an ATM card and a PIN. Two scenarios result: PIN is good, and PIN is bad.

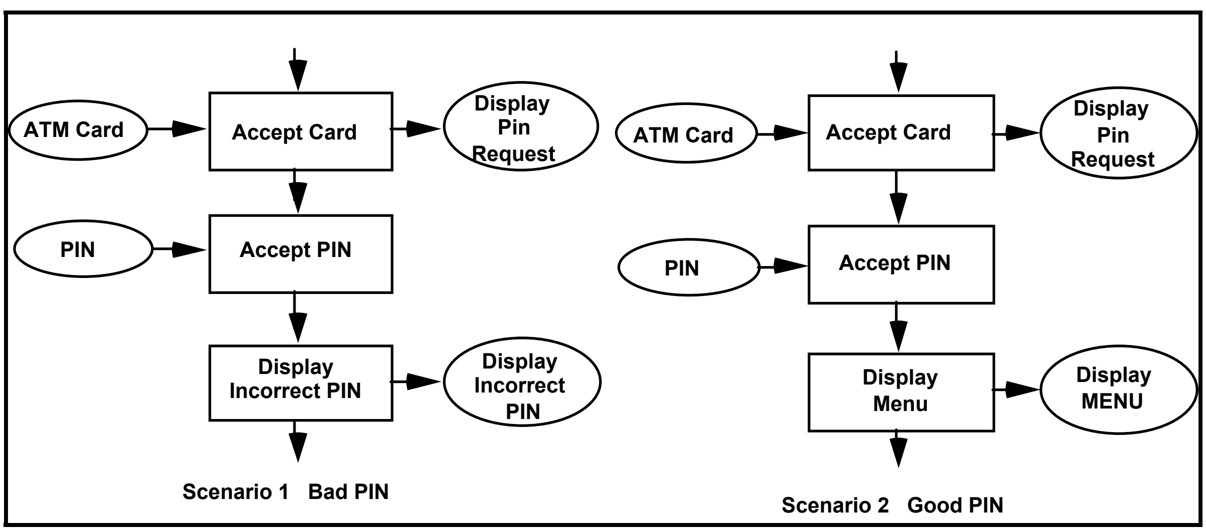


Figure D-4. Scenarios of Automated Teller Machine

From the scenarios or the integrated behavior, the stimulus response threads are identified. This set of threads can be specified in a number of notations. Figure D-5 presents the stimulus response threads in a functional format. Note that the conditions for each of the threads must be provided to avoid ambiguity. These conditions are a combination of two factors:

a) the “mode” of the system which determines which kind of input is expected and

b) the combination of values of the system state information and the contents of the input.

Thus, a correct PIN will yield a menu while an incorrect PIN will yield either a message to try again or a response of swallowing the card depending on the mode of the system. These conditions must be associated with the thread in order to make them testable. To show the conditions explicitly the "Accept PIN" function must be decomposed to show explicitly its input-output behavior under different conditions.

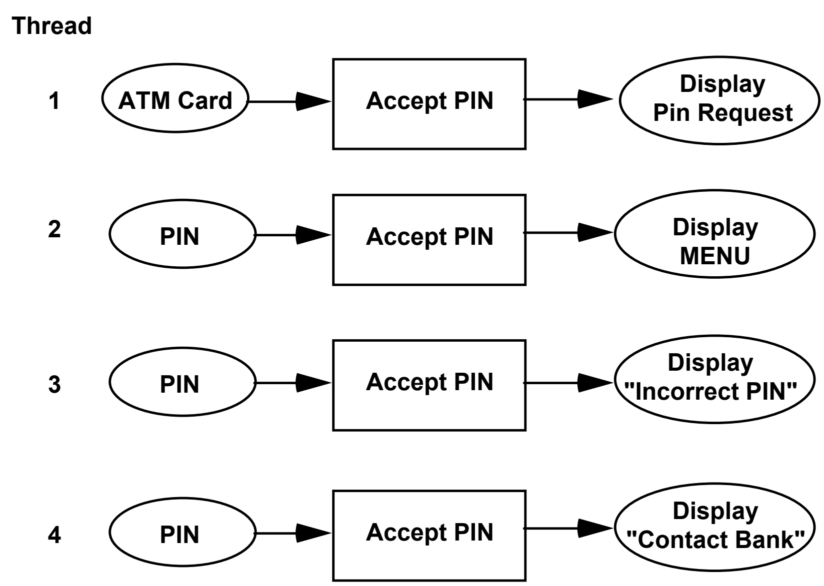


Figure D-5. Sample ATM Threads

Step 2. Allocating the Threads to the Computer Subsystem

If a function cannot be allocated uniquely to a component (e.g., the computer subsystem), it must be decomposed to the level where functions can be allocated to the components. This process is illustrated beginning with Figure D-6, where the threads are defined with their conditions and in Figure D-7, where the threads are defined in condition format. Then, in Figure D-8 where the system level function "accept card" is decomposed into functions to read the card, which is allocated to a card reader, and functions and conditions allocated to the computer.

Usually, most or all of the conditions are allocated to the computer system, with mechanical functions allocated to the other, less intelligent, components. Hence most of the system threads will yield a thread, with conditions, allocated to the computer subsystem; in turn, most of these are then allocated to the computer software, the computer hardware acting purely as the engine to be driven by the software. Thus, there is a direct traceable relationship between the system level, computer system level, and software level of requirements.

This clearly identifies the difference between the system and computer system threads. The system uses a Card Reader component to read the card, a Terminal Component to accept button push inputs from customers, and a Processor Component to provide the intelligence. Note that this results in the requirement for the computer system to perform its threads which translate "card info" and "PIN info" into various output displays.

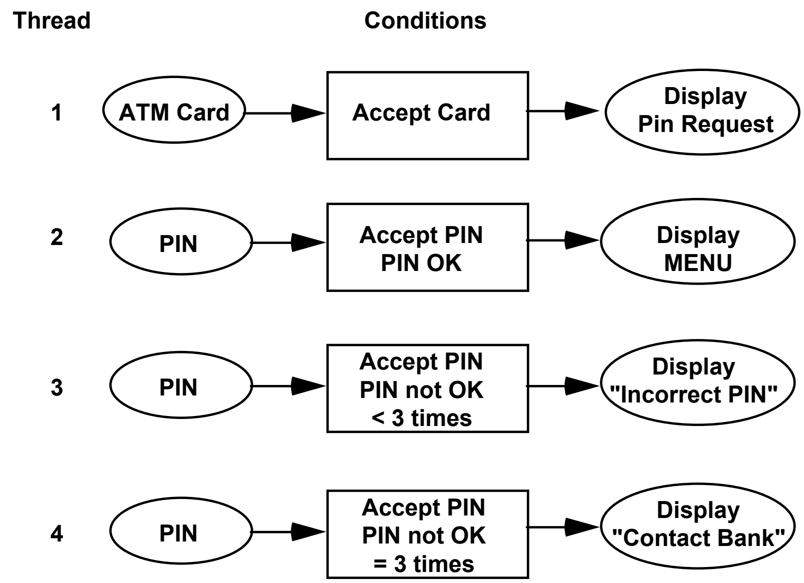


Figure D-6. Threads with Conditions

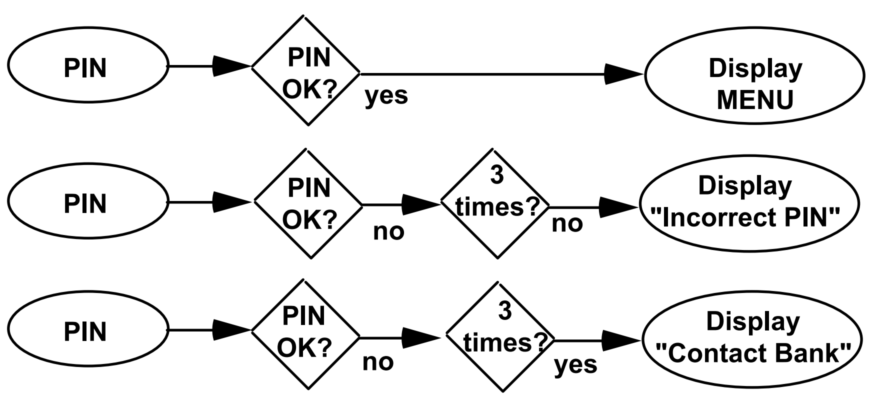


Figure D-7. Threads in Condition Format

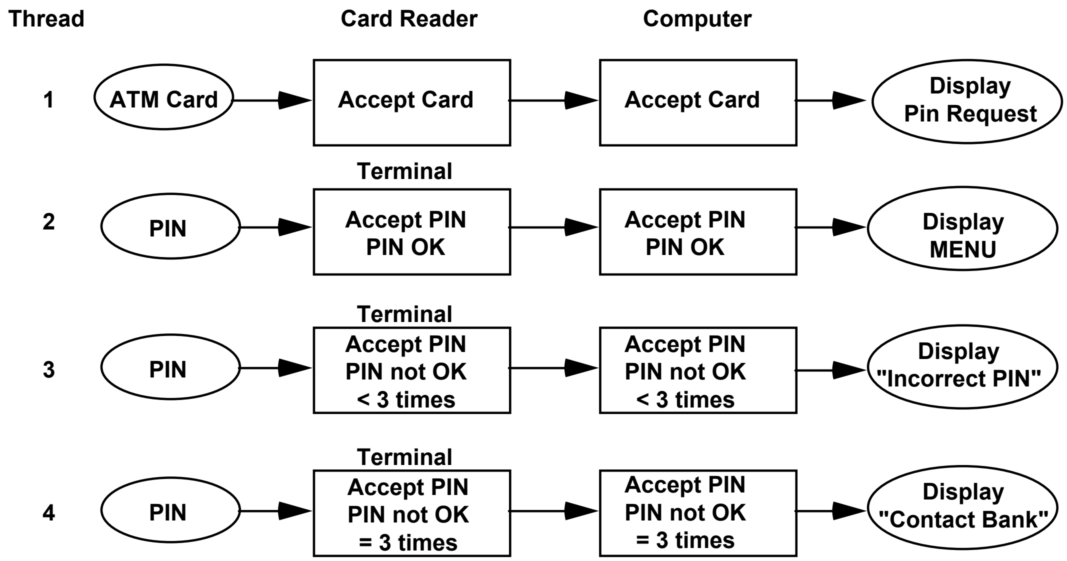


Figure D-8. Decomposed and Allocated Threads

__Step 3. Reviewing the Software Requirements and Design__

Regardless of how the software requirements are formatted, the systems and software engineers must be able to trace the computer system level threads (such as those illustrated above) through the document. If a thread cannot be so traced, then this represents an omission in the requirements. If additional threads are identified which do not deal with interface designs or computer system level fault detection/recovery, then such threads may represent "unnecessary processing", and perhaps should be omitted.

To illustrate this, Figure D-9 presents a rather simple software design in which a top level program "control" calls lower level units of code to carry out our operations.

Threads 1 through 4 then can be traced through this design, thus validating it. More complex examples are presented in Reference 3. The same approach works when an "Object Oriented Design" presents a number of "objects" implemented as independent software "processes" (e.g., supported by Ada).

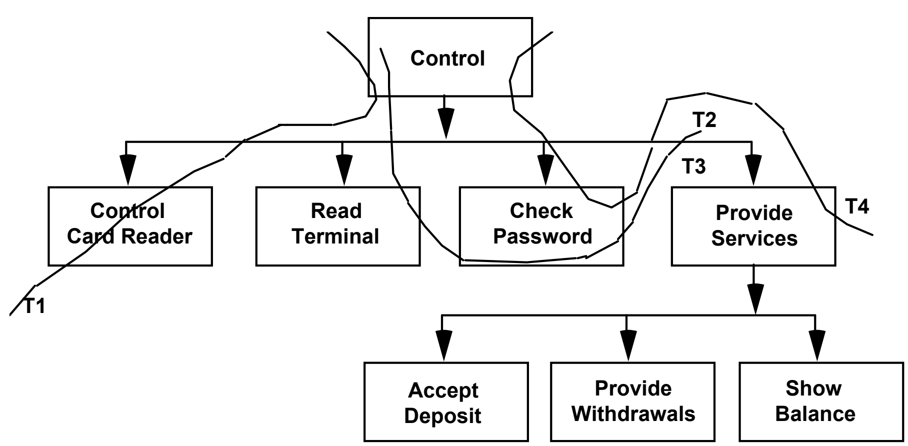


Figure D-9. Mapping S/W Threads onto a S/W Design

When software designs divide the overall software into CSCIs, CSCs, and CSUs, the above process of decomposing and allocating the system level threads onto the components is repeated for each level of component. Again, if a thread cannot be traced through, then this gap represents an omission in the design. The Systems Engineer should ensure that the tracing is done: from the allocated system requirements to the Software Requirements Review; and then through the CSCIs and CSCs and CSUs for the Software Preliminary and Critical Design Reviews.

If the software designers provide the tracing of computer system to software design threads as a part of the demonstration of satisfying the requirements, then the Systems Engineer need only verify the completeness of the traceability. Tools should strengthen the reliability of such traceability evaluation. If the software designers do not do so, then a joint team of system and software engineers should perform the tracing to verify the design in preparation for the design reviews. Again, the identification of the software level threads MUST be done in any event in order to provide a systematic test planning, so it represents no extra work, although it may represent an acceleration of the schedule for performing such work.

Step 4. Tracing the Threads to the Test Plans

Clearly, the collection of threads should be exercised by the collection of tests outlined in the test plan: at the software level, computer system level, and the system level. This can be represented by database and displayed in a cross reference matrix: system to software thread, software thread to software design threads, and threads to test cases at the various levels of integration. Tools can then ensure that every level of thread is tested somewhere.

It is noted that Deutsch recommends strongly that the software threads be used to drive the test planning process using the concept of "builds" of software. For system test, other components are added in, and the system test threads are tested. For the ATM example, the difference would be clear. In the software only test, the software would receive information in the format expected from the card reader; for the system test, the card reader component itself would be used as the source of the data when an ATM card is input.

This same approach can be used to construct the system level test plans in a way that exploits the early availability of computer software, which provides user oriented capabilities. Thus, an early build of software could be integrated with a card reader to perform test of Thread 1 through the system before the remainder of the software was developed. If the card reader were not available until later in the test cycle, then other threads could be tested first.

Notation

Several notations can be used for tracking the threads, but these usually divide into requirements-oriented and design-oriented notations. Requirements oriented notations describe the inputs and conditions and outputs, while the design notations describe the threads through the major design elements. Since both must eventually describe the same stimulus-condition-response information, their use is essentially equivalent (although, the design oriented notation is more useful for actually defining the builds of software).

References

1. Alford, Mack. *Strengthening the Systems/Software Engineering Interface for Real Time Systems*. NCOSE 1992 Proceedings.
2. Alford, Mack SREM. *Encyclopedia of Software Engineering*, Edited by J Marciniac. John Wiley & Sons, 1994.
3. Deutsch, Michael. *Verification and Validation: a Practical Approach*, 1985.

D.6 MODELING AND SIMULATION

Modeling and simulation are used during Functional Analysis/Allocation in order to verify the interpretation, definition, and/or viability of key functions or strings of functions.

To expedite discussion of ideas presented here, definitions of some potentially ambiguous terms are offered. Although these definitions may be debatable, they apply to subsequent discussions in this section.

Model - Any representation of a function or process, be it mathematical, physical, or descriptive.

Simulation - A computer program that represents the operation of a function or process to the degree of accuracy necessary to meet its purpose
.
System - A collection of hardware, software, and procedures that function together to meet an objective which no part of the system could meet alone. The perceived level of a system is unbounded where any system is simply one part of a larger system.

Fidelity - The degree to which a model realistically represents the system or process it is modeling. It is not necessarily synonymous with a model's level of detail or complexity.

The concept of modeling as a prerequisite to full-scale development is certainly not new. However, it has grown from a relatively minor activity to a scheduled effort on most programs. The US DoD has set up an Executive Council for Modeling and Simulation and a Defense Modeling and Simulation Office whose responsibilities include promoting modeling and simulation activities on applicable programs, advising on effective modeling practices, and coordinating and expanding modeling expertise for use in needed areas.[2]

In the commercial arena, modeling and simulation have been widely used for some time. The aircraft industry has taken modeling to a level in which aircraft designs are conceived, built, and flown with remarkable fidelity before a single step is taken on the factory floor. The model based design approach allows immediate transition into production once a design has been accepted. A similar approach is taken in the design of nuclear power plants. The common thread in these and other examples is the criticality of avoiding problems once the system is in operation. When catastrophic failures are possible and must be avoided, a development process using models as a core tool is intelligent. Modeling can play the same role in many other areas as well because the avoidance of operational failures and inappropriate systems is important in any application.

Applications throughout the System Lifecycle

The use of modeling is applicable at all stages of a system's life cycle. Whether used to validate concepts, designs, or test results, or to address risks at any point, models add value to the Systems Engineering process by providing information to those who need it in an economical and timely fashion.

Acquisition Planning

Models and computer simulations can be used to compare competing concepts/solutions to emerging problems and missions. For very complicated systems, acquisition strategies must be worked out several years in advance. Efforts such as Cost and Operational Effectiveness Analyses (COEA) and Acquisition Master Plans rely heavily on models and simulations to help determine which system concepts and options offer the greatest return for the money invested.

Concept Development

Models and simulations are often used to validate a system concept before full commitment is made to develop the system. Questions regarding satisfaction of requirements, extension of performance, improvement of reliability, reduction of cost, and the achievement of any other objectives must be addressed. Models and simulations can provide some of the answers at much less expense than by building a prototype. In some cases a well developed set of models can be directly transformed into or take the place of a prototype.

Performance Prediction

Simulation of parts of a system can help identify what requirements are not being met or will be difficult to meet. In this way, attention can be focused where it is most needed and the risk of not meeting requirements can be reduced. With successive predictions of chosen measures of performance, progress toward meeting requirements can be tracked and managed in support of a technical performance measurement (TPM) effort. Models offer flexibility to try out ideas and concept variations more rapidly and affordably at an early point in the life cycle.

Design Support

Models and computer simulations can directly support detailed design of either hardware or software and ease the transition from requirements development and system design. Techniques such as computer-aided design, design by simulation, and rapid prototyping are all examples in which a model can be transformed into an actual design with very few changes being needed. This concept can be extended to a level which significantly eases the effort of transitioning a system into production.

Test Validation

Models are needed to plan tests; predict outcomes; validate test results; rerun, analyze, and explain test results; and in cases where a contract calls for it, actually conduct tests. Models help prepare those involved in the tests by predicting outcomes, identifying problems a priori, and minimizing surprises in general. Activities supported by models include field exercises, design (prototype) tryouts, special experiments, qualification tests, and operational acceptance tests.

Operational Support

Models can support operational problems after a system has been fielded through independent case evaluation and examination of system responses in hard-to-repeat conditions. If a system experiences a fatal problem, a set of models may be the only available means for diagnosis and repair. Models can also support the investigation of system improvements and upgrades that are conceived after the system is in service.

Levels of Modeling

Models can be developed at many different levels. The levels represent differences in fidelity, intended purpose, types of resources, and commitment. Figure D-10 illustrates the levels as discussed in this paper and offers potential applications for each. It is important to develop and use models at the level which is appropriate to the objective and to the intended level of investment by those involved.

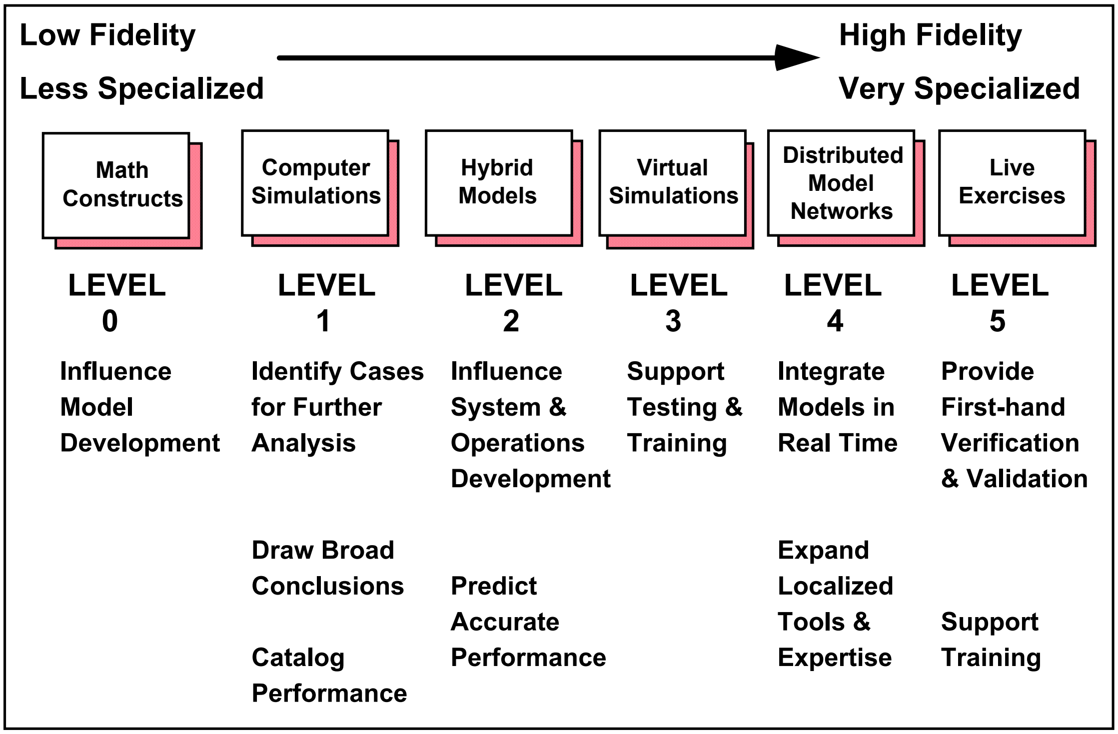


Figure D-10. Models Can Be Applied At Many Levels to Satisfy Different Purposes

<u>**Mathematical Constructs**</u>

Equations, state matrices, spreadsheets, and graphs are examples of mathematical models which represent a function or process in some deterministic form. Math constructs are defensible and universally understood; however, they often only provide roughly approximated solutions to the problem at hand. If the process being modeled is statistically random or overly complex, math constructs are likely to fall short. At the very least, they offer a good starting point in approaching a problem and understanding the major factors.

<u>**Computer or Constructive Simulations**</u>

When it is necessary to study the behavior of a system too complicated to be represented deterministically, a computer or constructive simulation may be the most economical approach. A computer program can be designed to include mathematical and logical processes which approximate a system's behavior within some bounded region. The program can then operate in time, process inputs, and be affected by external stimuli in a way similar to that expected for the real system. Computer simulations can exist at many levels (simplified top level to emulations) but usually still represent a simplified substitute for the real system.

<u>**System/Simulation Hybrids**</u>

When the accuracy of analysis and measurement is very important but a system is unavailable for extensive testing and experimentation, a system/simulation hybrid may be appropriate. As denoted by the term hybrid, this type of model integrates real aspects of a system (i.e., hardware, software, data) with other parts that are simulated to determine the behavior of the system as a whole. Terms such as "hardware-in-the-loop", "computer-in-the-loop" and "realistic-data-insertion" describe model setups which make use of a hybrid approach. The results obtained are more accurate and realistic than pure simulation, but do not require all the rigor of putting a real system into the field.

Virtual Simulations

Recent explosions in memory capacity, processing speed, and throughput make it possible to simulate a process or situation in real time with incredible detail and accuracy. Virtual reality and operational simulators represent a level of modeling which goes beyond mimicking behavior, wherein the users feel like they are actually operating a system in a real environment. At this level, the main purpose is training and exploring the aspects of operation and the man-machine interface. However, because they offer so much detail early in the process, virtual simulations can potentially go beyond training and actually replace current activities such as physical mock-up and prototyping.

Distributed Model Networks

Because models themselves are complex systems, organizations develop specialized capabilities which must then be integrated over long distances and cultural gaps. A relatively new development based on recent advances in computer network technology is the idea of distributed model networks. Model networks offer the potential to integrate many specialized models to represent a system more realistically than ever before. Existing models must be hooked together, which, of course, is no small task -- requiring a great deal of compatibility between the models. However, the potential exists for significant enhancement over stand-alone models. Extending the idea, live exercises can record and play back data through multiple, remotely located simulations in real time to accurately represent the behavior of an immensely complex system which has yet to be developed.

Live Exercises

At the extreme end of the scale of modeling, consider a real system that is put into a simulated scenario. A simulated scenario is still a step away from actual continuous use in which stimuli and upcoming situations are largely unknown. In live operations and exercises, a simulated scenario is well planned in order to prepare for all possibilities. Parts of the system which are to be scrutinized are instrumented, and events are designed to evoke specific responses and exercise specific system functions. Although the principles which apply come more from the best practices of system testing, live operations and exercises do share some of the same objectives as any other level of modeling

Development of a Model

The development of a model should reflect application of a Systems Engineering approach. That is, before attempting to build the model a significant effort should be given to determining the scope of the model, what purposes it will serve, who will use it, and even what its future applications and extensions might be. The process is iterative, implying that steps are revisited and refined, in some cases significantly, once enough detail has been revealed to require it. Figure D-11 illustrates the process.

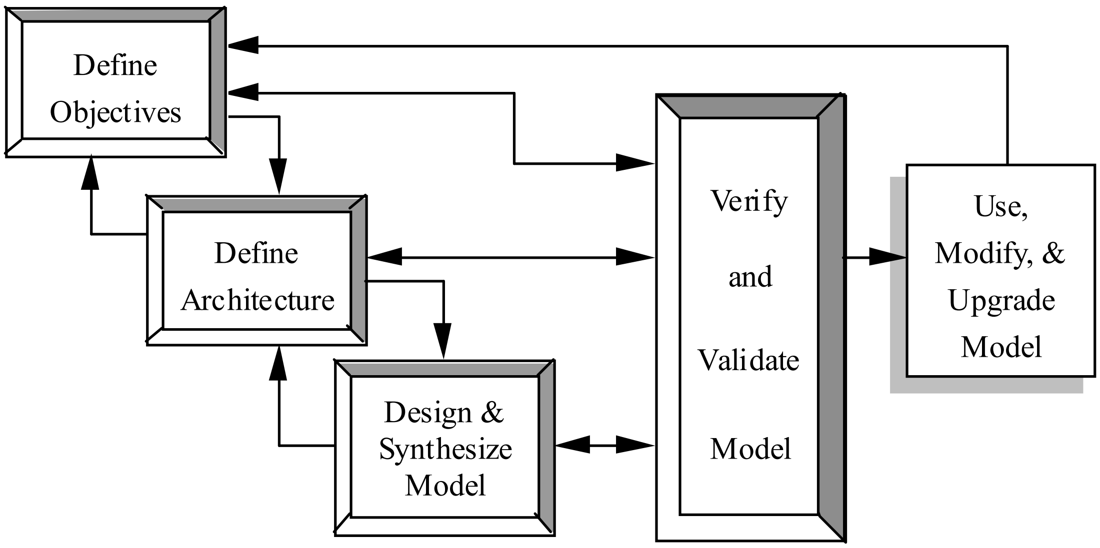


Figure D-11. Iterative Model Engineering Approach

Models should be developed in the same way as deliverable and complex systems are, following an iterative Systems Engineering approach

Definition of Objectives

A very important step all model developers must take is to limit the problem they are trying to solve and concisely describe what the model will be expected to do. This includes defining available inputs, expected outputs, desired run times, necessary functions, requirements for user friendliness, and level of fidelity expected. The list of objectives may also have to define the language and platform to be used. Depending on the situation, the user(s) and developer(s) may be the same or different people. This will influence how the model objectives are conceived. However, a developer who will also be a user should still seek outside review of the intended objectives from appropriate sources to ensure completeness and reasonable breadth of applicability. The objectives will form a baseline against which verification and validation will later be established.

Architecture Definition

As with any system, the model's architecture is defined to include major functions, modular partitioning, internal interfaces, external interfaces (including user interfaces), and data constructs. If not defined already, means of computer implementation (i.e., language, machine type, operating system, etc.) must be included with the architecture definition. It is also at this stage that consideration must be given to the input data to be used. Level of detail, formatting, and consistency must be appropriate to the stated objectives. There is no purpose served in building a model if suitable data can not be acquired to feed into it. Results from the model will depend on the quality of the data (at least as much as the model) and how compatible the data are with the model. As the architecture is developed, the definition of some of the objectives will be refined. Some objectives may not be practically feasible whereas others may be expanded because implementation is easier than expected. Architecture definition is reasonably complete when the architecture is expected to meet the objectives and those objectives are considered suitable to the overall needs for the model.

Design and Synthesis

Once a set of objectives and an architecture which can meet those objectives are well defined, the step of building the model can begin. In general this step will involve writing software code, but it may also involve interfacing computers, acquiring appropriate data, or setting up a laboratory environment with equipment appropriate to the implementation of the model architecture. In any case, this step is based on the defined architecture and model design provided. As with the other steps, design and synthesis may involve refinement of earlier steps. It will also involve some degree of test at levels appropriate to allow parts of the model to be integrated. Design and synthesis is complete when the model is in a form which can be reasonably tested against the established success criteria established when objectives are determined.

Verification and Validation

Verification is the confirmation that the model does what it was designed to do. Validation is the acceptance of the model objectives, architecture, design, and operation as suitable to the needs for the model. Verification should include tests for which the results are known (theoretical cases), qualification tests (tests to explore the model's limits and responses to out-of-bound data), and benchmarking against other similar models, if appropriate. Validation should judge the success of the model against the objectives defined earlier. If not done already, the users of the model must operate it successfully to provide the ultimate acceptance.

Modification and Upgrade

The true value of a model is its ability to be applied to problems for which it may not have been originally intended. Modularity, transportability, and flexibility of the model will determine how much of an effort will be required to modify the model to suit a new (but probably similar) purpose. Most models are derived to some degree from a previous model. The models that last the longest and reap benefits many times their own cost are those which can be easily adapted to new applications. This criterion should be a strong consideration during the development of a model.

Modeling Summary

Modeling in support of the development of large complex systems is a necessary activity, which should not be underestimated. With the development of high-performance computers, modeling is a more capable and economic tool for developing system designs and reducing risks than ever before. However, the development of models must follow a disciplined Systems Engineering approach.

Modeling has applications throughout the life cycle of a system. Models can be used long before a system is developed to predict performance, support design, and reduce risks. However, models can also remain useful after a system is built by supporting test activities, field troubleshooting, and future upgrades. Although models may represent a large initial investment, their value through an effort may well justify that investment.

The development and use of models must involve a systems approach to ensure they are consistent with their intended purpose and user environment. A model's structure should follow good software practices such as modularity and readability and the model should fulfill not only the purpose intended but also be flexible to potential modifications and expansions at a later time.

References

Hodapp, J., *Self Defense Spreadsheet Model Upgrades and Operations*, APL Internal Memorandum F3D-2-1547, Sep 14, 1992.

Hodapp, J., *Metamorphing: The Animation of Three-Dimensional Carpet Plots*, APL Internal Memorandum F3D-2-1530, May 31,1992.

Hyer, S., Johnston, J., Roe, C., *Combat System Effectiveness Modeling to Support the Development of Anti-Air Warfare Tactics*, Johns Hopkins APL Technical Digest, 16(1), 69-82, Jan 1995.

US DoD, "Modeling and Simulation Management", US DoD Directive 5000.59, Jan 4,1994.

D.7 REAL-TIME STRUCTURED ANALYSIS

Real-Time Structured Analysis is an alternative approach to *Functional Analysis/Allocation.* It has been applied most commonly to the design of software-intensive systems, but has applicability to other systems as well. The first steps in real-time structured analysis are to construct Data Flow Diagrams and a Data Dictionary.

Data/Control Flow Diagrams

A data/control flow diagram (D/CFD) is a graphical means for modeling the processes that transform data/control in a system. These diagrams model the work done by a system as a network of activities that accept and produce data/control messages. Alternatively, they can be used to model the system's network of activities as work done on a processor. Each successive level of D/CFD represents the internal model of the transformations contained in the previous level of D/CFDs.

DFDs are used to illustrate and document the functional decomposition of data throughout the system and as a means for defining all data transmissions and processing requirements, both hardware and software. The DFDs represents the system as a connected network showing data inputs, outputs, processing, and storage, and consists of four basic elements:

- Data flows, represented by vectors
- Processes represented by circles or ovals
- Data files, represented by parallel lines
- Data sources and external outputs, represented by rectangles

Transformation Specification

A transformation specification (TS) is a statement of the purpose and procedure that a given transformation of input data flow(s) into the output data flow(s) for a given functional primitive appearing on a data/control flow diagram. It defines the purpose and the processing performed by a data transformation. There is a transformation specification for each data transformation that is not further decomposed in a lower-level D/CFD.

Data Dictionary

A data dictionary (DD) provides definition of all system data representations defined in the models that binds the models together. It defines the data representations shown in the D/CFDs, STDs, and Entity Relationship Diagrams (ERDs). The DD also defines data items mentioned in the transformation specifications.

A DD is prepared that defines the content of each data item, table, and file in the system. Process specifications describe the capabilities that each process is required to provide. The specifications may be written in structured English and/or in the form of decision tables and decision trees. State diagrams graphically depict the legal states that the system may assume. Associated process descriptions specify the conditions that must be satisfied for the system to transition from one legal state to another legal state.

When working from a set of customer documents, a top-down approach is used to decompose customer defined processes. As each process is decomposed, so is the data. Only the data that a process requires to produce the specified outputs is documented in a data dictionary. Functional decomposition usually proceeds to a level where the requirements for each lower-level function can be stated on one page or less (this is called the primitive level). Interaction with the customer may be necessary to decompose and define data elements at lower levels. The resulting DFDs are analyzed to identify different processing states and to determine conditions for transitioning from one state to another state. Figure D-12 illustrates the application of DFDs and the top down decomposition process to produce a system model.

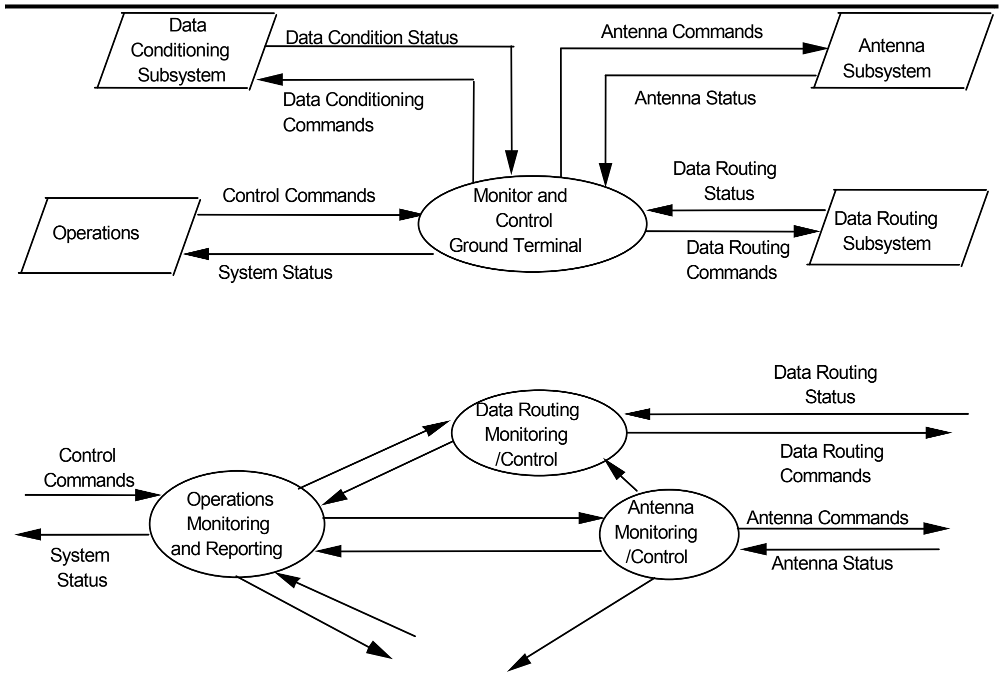


Figure D-12. Top-Down Requirements Decomposition

Building a system model by interviewing users usually starts with processes defined at the primitive level and data defined in forms and manual files. Figure D-13 illustrates part of model built from user interviews. The next step is to "logicalize" the data flows built from interviews and then collapse the lower-level functions into higher-level functions. Figure D-14 illustrates the "logicalized" version of the model built from interviews. The functions defined might collapse into the higher-level function "Control Parts Inventory".

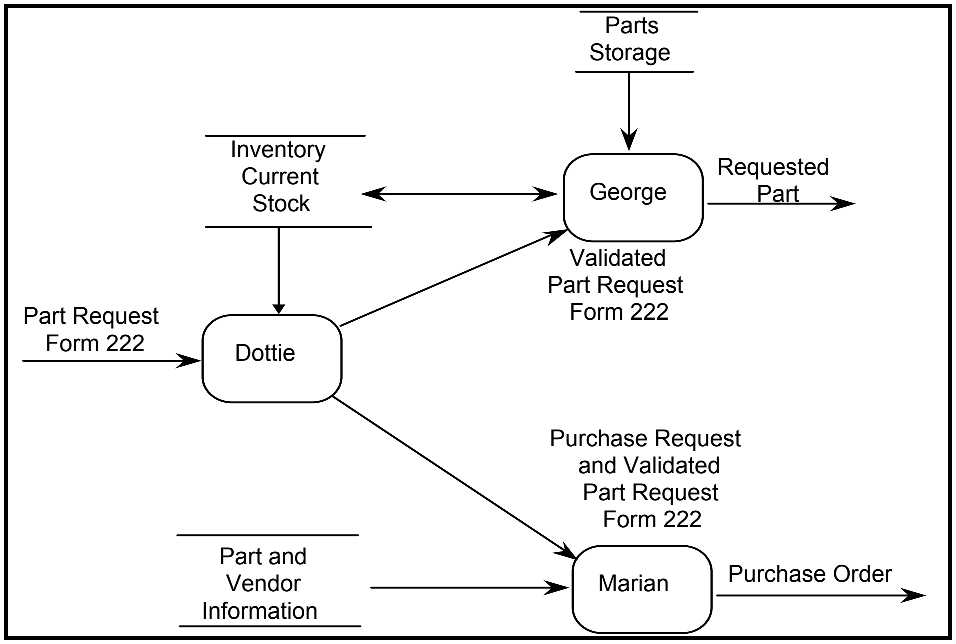


Figure D-13. Model Built from User Interviews

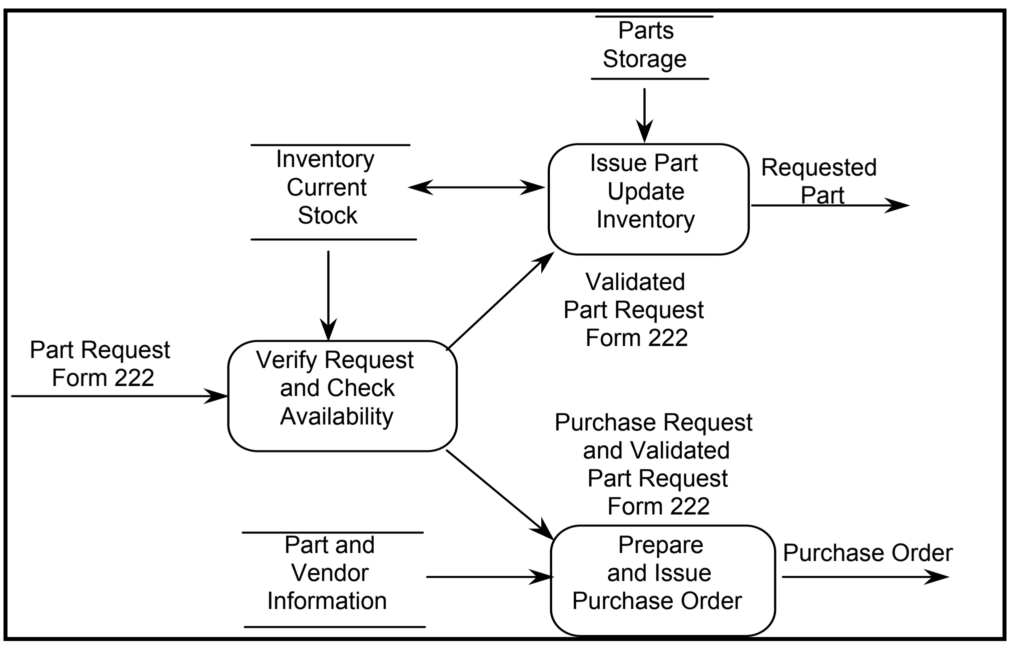


Figure D-14. "Logicalized" Model Built from User Interviews Entity Relationship Diagrams

An entity relationship diagram (ERD) is a graphical means of modeling the complexity of information the system requires about its environment. These diagrams generally are used to describe the relationship among the stored data. A level set of the entity relationship diagrams corresponds to each level of the data/control flow diagram. There is a level of ERDs for each level of D/CFD that shows multiple data stores.

<u>**State Transition Diagrams**</u>

After the DFDs and DD are complete, the next step is to identify the various states the system may assume and to produce diagrams depicting how the system transitions between states. A state transition diagram (STD) is a graphical means of modeling the dynamic behavior of a system. A state transition diagram describes the processing performed by a control transformation contained on a data/control flow diagram. It is a sequential state machine that graphically models the time dependent behavior of the control transformation.

A top-down approach should be used to identify various states of the system, working down through the subsystem. Figures D-15 and D-16 are examples of state transition diagrams for a system and an antenna subsystem.

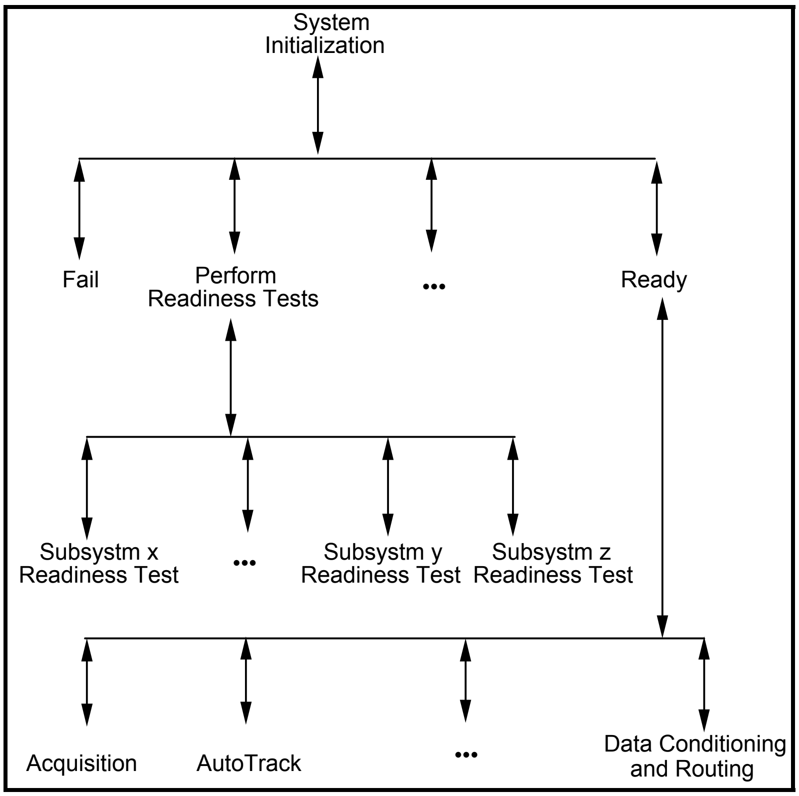


Figure D-15. Example of a System State Diagram

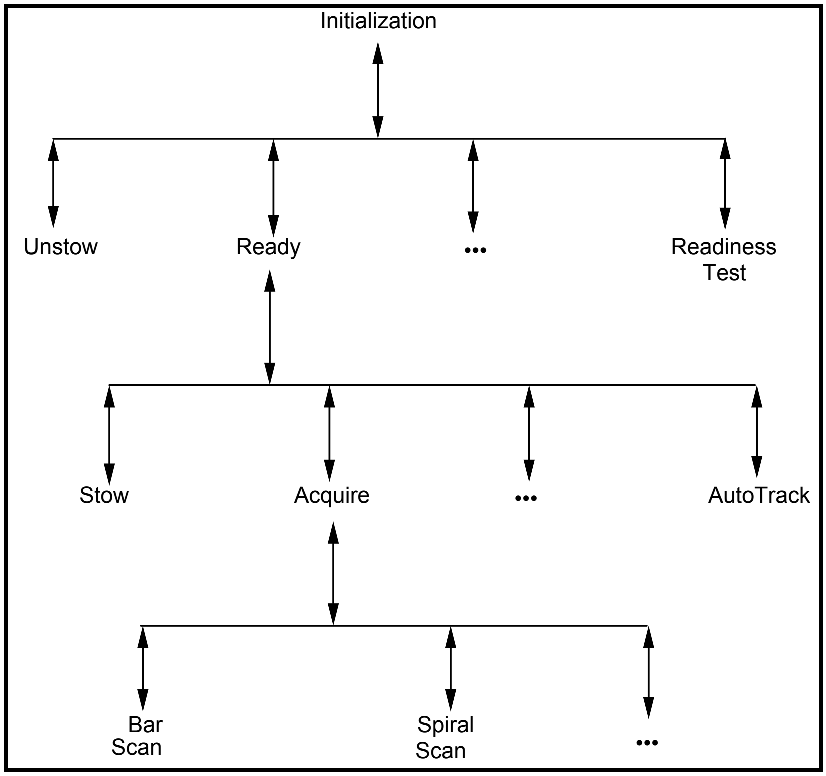


Figure D-16. Example of a State Diagram for an Antenna Subsystem

Deriving Requirements in Real-Time Structured Analysis

The contract Statement of Work (SOW), the customer specification document(s), DFDs, and state transition diagrams all help to provide the framework for developing the outline for the requirement specification. The SOW may specify a general outline to be used.

Customer specification documents identify major functions that serve as major paragraph headings for a specification. The functions identified in DFDs and states identified in state transition diagrams serve as subparagraph headings. The DFD functions become the paragraph headings for the requirement specification if the SOW did not specify a specific document outline and/or a customer specification document does not exist. The DFD function titles, when wrapped in a *shall* statement, become requirement statements within the specification. Process specifications at the primitive level need only to be wrapped with ***shalls*** to become requirement statements. Descriptions of how the system transitions from one state to another also become shall statements in the requirements specification.

D.8 OBJECT ORIENTED APPROACH TO SYSTEMS ENGINEERING

A model is an abstraction used to represent characteristics of a system for the purposes of understanding its complexity, communicating about its structure and behavior and designing the system, before it is built. In object oriented modeling, the fundamental construct is an object which combines both data structure and behavior in a single entity to represent the components of a system.

Other fundamental characteristics include classification, or the grouping of objects with similar data structure and behavior; inheritance or the sharing of properties and behavior among classes based on a hierarchical relationship and polymorphism in which the same operation may behave differently on different classes of objects.

An object-based approach to the lifecycle development of a system begins with a problem domain analysis phase where objects specific to the system domain are identified. The objects' static structure, properties and interactions with other objects are defined. Next, during the design phase, details are added to the domain model to describe and optimize the implementation. The focus here is architecture. Finally, the design model is implemented in a programming language and physical hardware components. This approach provides a seamless representation and description of the system from the problem domain analysis to design to implementation such that information is incrementally added as the models are evolved from one phase to the next so that no translation, restatement or reinterpretation is required, as shown in Figure D-17.

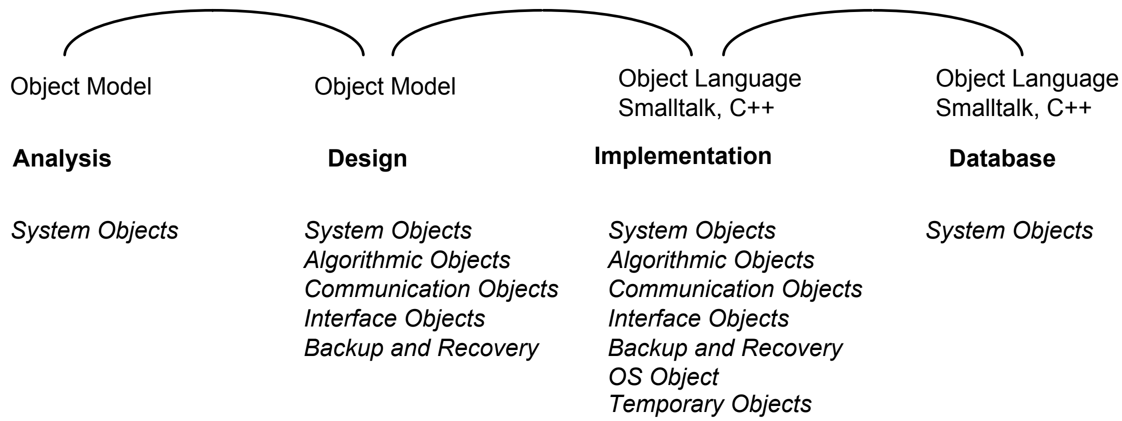


Figure D-17. Objects in the Lifecycle

Purpose

The purpose of this section is to describe an object-based approach to Systems Engineering. The emphasis is system analysis and design not programming issues. The greatest benefit of an object-based approach comes during the analysis and design phases where complex, conceptual issues can be clearly understood, represented and communicated to all, before it is built.

Object-based Approach

As discussed above, models are used to represent characteristics of the system as a means to understand, communicate and design the system before it is built. A good model has essential properties in common with the problem it represents and the nature of the properties it represents determines the use that can be made of the model. If temporal behavior is the fundamental characteristic of the system then a temporal, structured behavior model needs to be applied.

For complex system problems a number of different aspects need to be analyzed and designed, each of which is represented by a specific model. The different models permit different aspects to be investigated one at a time. These different modeling perspectives are incrementally constructed and integrated in a unified description (system model) to maintain a holistic system perspective from which the emergent properties of the system can be deduced and verified.

The system model emphasizes the interactions of the objects in the context of the system also including the objects in the environment. This is done with object semantics that represents the components of a system, their interconnections and their interactions when they are responding to the stimulus from the objects in the environment. These object semantics are partitioned into a static as well as dynamic modeling representation, describing the system's structure and behavior respectively. These modeling semantics are described in Figure D-18.

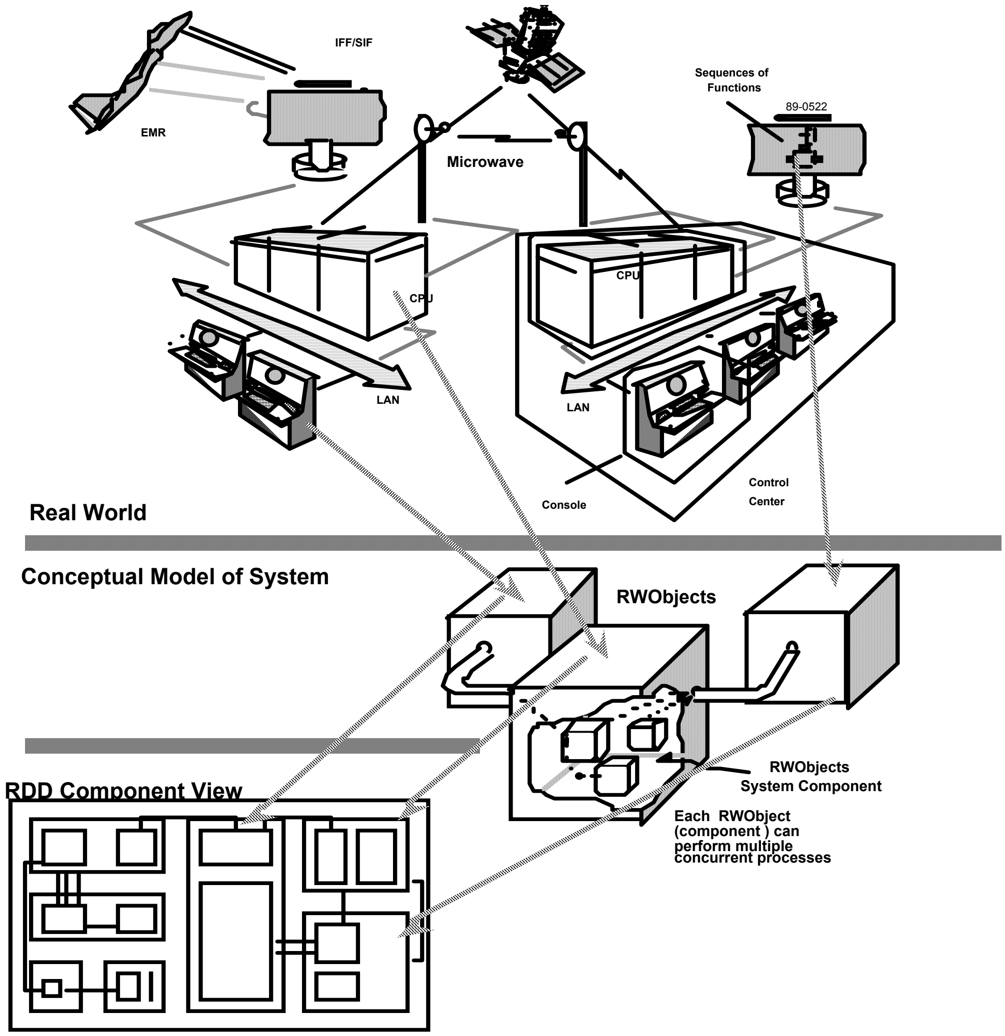


Figure D-18. The System Model

In this sense, the models embody the decisions made over the different steps of the lifecycle process. The Systems engineer develops the models as part of the decision making process and the models provide a trail of those decisions.

The models should support the evolution of the system design process as well as the iterative nature of the engineering in an environment where changes and enhancements to the models can be managed in a controlled manner.

Associated with the models are also the textual descriptions of requirement statements and constraints which in turn are structured and organized into groups that can be retrieved and manipulated for different purposes. These requirements and constraints are linked to the models and constitute the body of information that is documented in different specifications as milestone deliverables from the development process. Also, different types of issues are captured, leaving an audit trail of what happened and why.

Building System Models

Step 1. In this approach, first define the problem domain and identify the objects that participate in the closed system. It includes all of the objects in the environment and the components to be designed and constructed.

Step 2. Operational scenarios are developed that identify the interactions between the objects. The operational scenarios are snapshots of the system under certain specified conditions when the system is interacting with the objects in the environment. By focusing on the behavior of a single object in the absence of any conflicts or exceptions, the intended behavior can be identified. If this behavior is different for categories of objects, then the different behavior can be clearly defined and integrated to describe the overall behavior. The specification of the desired behavior of a single object can be used to structure the analysis of environmental exceptions, and additional functionality can be added to mitigate their effect, and to structure the analysis of interactions and interference between objects.

The object model together with the scenarios defines the desired and needed system capabilities (functional and behavioral aspects) under specified conditions. The integration of these functional and behavioral aspects yields the System Behavior Model that exhibits all the desired behavior. This conceptual behavior model is structured according to good rules (data encapsulation, placement of behavior in the object structure, strong cohesion, low coupling) for identifying stable system objects. In fact the objective is to build a conceptual model of the system objects that is a result of our understanding of the problem.

The conceptual system model, as a mapping of the problem will change only when the objects in the environment change. In this sense it is fairly stable over the life cycle as opposed to the allocated system solution that will change according to changing technology and architecture.

Step 3. The next step in the process is to search for a feasible architectural solution within given constraints. This implies that the conceptual system model is successively refined, partitioned and allocated to system objects that become progressively more implementation dependent. The object model is hence also used to describe the physical composition and characteristics of the final system products.

Step 4. Finally, the system functions allocated to communicating computer components are described as conditional sequences of operations that can be implemented by either hardware or software.

Summary

A system, such as the one in Figure D-18, can be considered as a collection of interacting objects (or components) that collaboratively achieve a common purpose. The focus on objects from the beginning of the development process ensures a strong coupling to the problem at hand. The use of models provides a means for understanding the problem and a way to investigate alternative solutions to the problem before the system is built.

The object-based Systems Engineering approach helps engineers manage complexity of the problem by abstracting knowledge and behavior and encapsulating them with objects in a manner that supports a separation of concerns. Finding these objects is the issue of structuring the knowledge and the expected behavior according to specified objectives and given constraints.

The object-based system model can serve as the foundation for the Systems Engineering process and provides a unified notation for hardware and software engineering.

The object-based Systems Engineering approach is somewhat different from other object oriented approaches in that it defines the system as a collection of interacting objects that need to work together to provide the expected solution to the defined problem and maintains that perspective throughout the development lifecycle.

In this approach the system objects are defined as a separate modeling concept that is connected in a traceable manner to the components in the design. Using this approach there are no a-priori assumptions of a particular implementation. Solutions based on both hardware and software benefit equally from the object centered approach. For complex system problems a number of different aspects need to be analyzed and designed, each of which is represented by a specific model. The different models permit different aspects to be investigated one at a time. These different modeling perspectives are incrementally constructed and integrated in a unified description (system model) to maintain a holistic system perspective from which the emergent properties of the system can be deduced and verified.

References

1. Alford, Mack; *SREM at the Age of Eight: The Distributed Computing Design System*; April 1985.
2. Bailin, S.C.; *An Object-Oriented Requirements Specification Method*; Communication of the ACM, May 1989.
3. Jacobsen, J. et al; *Object-Oriented Software Engineering;* Addison-Wesley, 1992.
4. Moanarchi, D. E. and Puhr, G. I.; *A Research Topology for Object-Oriented Analysis and Design*; Communication of the ACM, September 1992.
5. Rumbaugh, J., et al.; *Object-Oriented Modeling and Design;* Prentice Hall, 1991.
6. Tronstad, Y; Peterson, J.; Shreve, S.; *Development of a Systems Engineering Process Model Template*, Proceedings of the 3rd Annual International Symposium of NCOSE, July 1993.

D.9 ANALYTIC HIERARCHY PROCESS

The Analytic Hierarchy Process provides a comprehensive framework for dealing with the intuitive, the rational, and the irrational all at the same time in a trade-off. For complex trades or trades in which it is difficult to realistically set weighting the Analytic Hierarchy Process (AHP) is recommended. This process is described in detail in an article from IEEE Transactions on Engineering Management (August 1983).

As an example, the “Buying a House” example from the article is presented with elaboration on recommended approaches for handling the math. The criteria for buying a house are:

a. Size of the house
b. Location to bus lines
c. Neighborhood
d. Age of the house
e. Yard space
f. Modern facilities
g. General condition
h. Financing available

A scale of relative importance is used in making pairwise comparison, as shown in Table D-1.

Table D-6. Scale of Relative Importance

1	Equal Importance	Two activities contribute equally to the objective
3	Moderate Importance of one over the other	Experience and Judgement slightly favor one activity over another
5	Essential or Strong Importance	Experience and Judgement strongly favor one activity over another
7	Very Strong Importance	An activity is strongly favored and its dominance is demonstrated in practice
9	Absolute Importance	The evidence favoring one activity over another is the highest possible
2, 4, 6, 8	Intermediate Values between two adjacent judgements	
Recprocals of above non-zero numbers	If activity i has one of the above non-zero numbers assigned to it when compared with activity j, then j has the reciprocal value when compared to i.	

The pairwise comparisons for the criteria listed above are shown in Table D-2. The computations in Table D-2 were performed using an Excel spreadsheet. The comparison of relative importance of criteria is contained in columns 1 through 8. The 9th column computes the Nth root (8th root here) of the product of the values in each row. The 10th column contains the computed priority for each criteria. The computation is merely the value in column 9 divided by the sum of column 9 values (e.g., 0.175 = 2.053/11.742). The 9th row is merely the sum of the column values. The 10th row is computed by taking the sum value in the 9th row and multiplying by the respective criteria priority from the 10th column, e.g., 1.575 = 9.010 (0.175). Therefore lmax is merely the sum of the 10th row values.

CI = (lmax - n)/(n - 1) and CR = CI/(Constant from Table 4-11).

Table D-7. Pairwise Comparison of Criteria

	1	2	3	4	5	6	7	8	8th Root of Product	Priority Vector
1	1	5	3	7	6	6	0.333	0.25	2.053	0.175
2	0.2	1	0.333	5	3	3	0.2	0.143	0.736	0.063
3	0.333	3	1	6	3	4	6	0.2	1.746	0.149
4	0.143	0.2	0.167	1	0.333	0.25	0.143	0.125	0.227	0.019
5	0.167	0.333	0.333	3	1	0.5	0.2	0.167	0.418	0.036
6	0.167	0.333	0.25	4	2	1	0.2	0.167	0.497	0.042
7	3	5	0.167	7	5	5	1	0.5	1.961	0.167
8	4	7	5	8	6	6	2	1	4.105	0.350
□	9.010	21.87	10.25	41	26.33	25.75	10.08	2.551	11.742	1.000
□·PV	1.575	1.37	1.524	0.793	0.937	1.089	1.683	0.892	9.863	
l_{max}= 9.863 CI = 0.266 CR = 0.189										

Table D-8. Random Consistency

	n	1	2	3	4	5	6	7	8	9	10
Random Consistency		0	0	0.58	0.90	1.12	1.24	1.32	1.41	1.45	1.49

The value of CR should be less than 10 percent (up to 20 percent is tolerable) if you were consistent in making your importance judgments. If CR is too high, review your importance ratings.

With the priorities established for each criteria, then each alternative is rated (pairwise against each other alternative) using the same technique. The resulting priorities from this analysis give the rating of each alternative for each criteria. The ratings are then weighted by the criteria priority computed above to provide an overall favor selection.

COMPUTING THE WEIGHTED SUMS

Computation of the weighted sums is most conveniently done on an electronic spreadsheet, such as Excel. The example in Table D-4 illustrates what the spreadsheet might look like.

In this example, the clear winner is Alternative Solution 4. It did not get the best score on criteria 1 and 2, and it only tied on criterion 3. However, it scored ahead of all the alternatives on criteria 4 and 5. In all, it produced a weighted score that was 34 points higher than then second best--14% better, although it was 68 points below a perfect score.

Table D-9. Spreadsheet showing all parameters of the trade study

	Criterion 1		Criterion 2		Criterion 3		Criterion 4		Criterion 5		Maximum Score
Metric Value	**10**		**7**		**8**		**4**		**5**		**340**
	Raw Scr	Wtd Scr	Raw Scr	Wtd Scr	Raw Scr	Wtd Scr	Raw Scr	Wtd Scr	Raw Scr	Wtd Scr	Solution Score
Alternative Solution 1	8	80	9	63	5	40	5	20	7	35	238
Alternative Solution 2	7	70	5	35	8	64	5	20	8	40	229
Alternative Solution 3	6	60	7	49	7	56	4	16	9	45	226
Alternative Solution 4	7	70	8	56	8	64	8	32	10	50	272

D.10 DECISION ANALYSIS TECHNIQUE FOR RISK MANAGEMENT

Decision analysis is a method of identifying the best among a set of alternatives under uncertainty, using the possible outcomes of each alternative and their probabilities of occurrence to calculate the expected value of the outcome. The method makes use of Bayesian [1] subjective probabilities. Decision analysis can be applied to a wide-range of problems. Considerable interest has been given to applying decision analysis techniques to decisions from the areas of business and defense.

The first step in decision analysis is to create a decision "tree" diagram that represents the situation in question. Starting on the left with the initial decision point and proceeding to the right, the decision diagram must accurately represent each point where a decision is to be made and all the possible consequences of that decision. Figure 6-10 illustrates a decision tree for the following decision.

Assume there is a system that needs a particular function to be successful and that a choice needs to be made between two different alternatives.

Approach A represents developing a new design that, if successful, would be extremely competitive and would generate large profits from the system for the next five years or so. Approach A has been estimated to cost $6,000,000 to develop, but it is also believed that if the approach is successful, it will generate $20,000,000 in profits over the next five years.

Approach B represents modifying an older, existing design that is thought to be adequate for the near-term application, but is not thought to be competitive in the near future. Approach B will cost $50,000 to implement, but is felt to only be able to provide $10,000,000 in profits in the future since the design is reaching obsolescence.

Since there is uncertainty in most designs, both approaches may meet all of the near-term objectives, or meet a minimum set of objectives, or fail to meet even a minimum set of objectives. If an approach meets a minimum set of objectives, there is a possibility that additional funding would be made available for improvement or it may be used as is. If an approach does not meet the minimum objectives, there is a possibility that funding for improvement may be made available or the work on the approach may be cancelled. In this example, there is only one decision to be made.

The decision diagram is constructed by starting with a decision on the left side of the diagram and proceeding to the right with each possible outcome of each alternative. Figure D-19 is a decision diagram for our engineering risk example. The square on the left side of the diagram represents the decision. The circles represent outcome events and the branches radiating out from each represent a possible outcome. For example, the upper branch from the decision point represents the outcome if Approach A is chosen. The possible outcome paths radiating from the first outcome event represent the possible outcome of the new Approach A meeting its requirements. If Approach A does not meet all its objectives, the possibility for improvement funding exists.

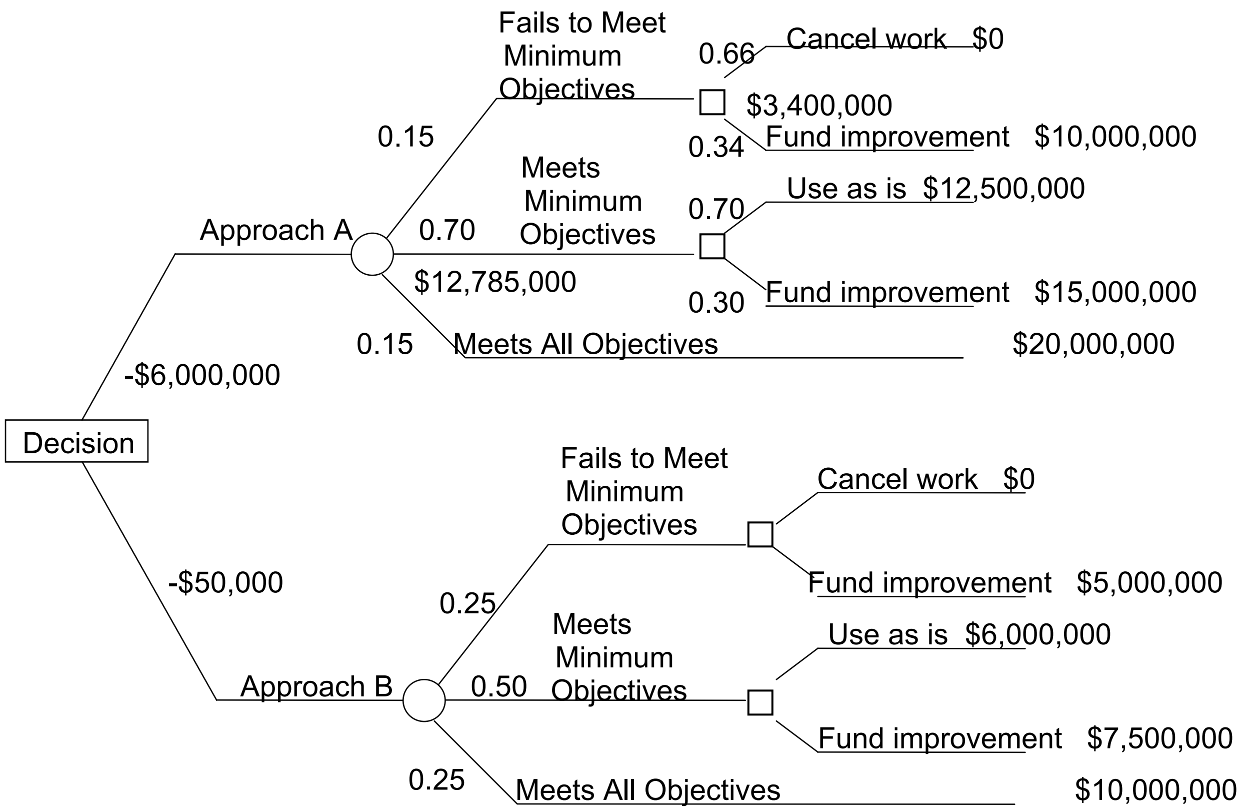


Figure D-19. Decision Diagram of Engineering Risk Example

The value of each outcome is an estimate of the profit to be generated minus any additional development costs. For Approach A, the additional development costs are \$15,000,000 if Approach A meets a minimum set of objectives and \$10,000,000 if it does not. Approach B will cost an additional \$7,500,000 if it meets a minimum set of objectives and \$5,000,000 if it does not. For basic decision analysis, all outcomes must be in the same units (usually dollars). For this example, the estimated development costs are shown on the branches leading from the decision and the possible outcomes are shown on the right side of Figure D-20.

Each outcome branch is assigned a probability of occurrence. In the example, Approach A, the new development, is thought to be more likely to meet, at least, a minimum set of objectives than Approach B, the old design. Note that the probabilities sum to 1.0 at each node.

Evaluation of a decision diagram proceeds from the right back to the initial decision on the left. The expected value (EV) at each outcome node is the sum of the product of the cost/value of each outcome and its probability. An expected value represents the average cost or value of the outcomes if the situation was evaluated an infinite number of times. In this case, the expected value of the outcome if Approach A is chosen and it meets a minimum set of its objectives is:

(consequence of using Approach A as is) x (probability of using Approach A as is)

+ (consequence of Approach A being funded) x (probability of Approach A being funded)

= ($12,500,000)(0.7) + ($15,000,000)(0.3) = $13,250,000.

Following this procedure, starting at the right edge, all the expected values for this example have been calculated for the given probabilities and outcome profits and losses. They are located beside each outcome circle of the decision diagram.

The best decision using decision analysis is the choice which has the highest valued branch. In this example, it is to use Approach A since the expected value of Approach A ($12,785,000) minus the Approach A development costs ($6,000,000) is greater than the expected value of Approach B ($6,500,000) minus the Approach B development costs ($50,000).

$12,785,000 - $6,000,000 = $6,785,000

$6,500,000 - $50,000 = $6,450,000

$6,785,000 > $6,450,000

Note that other decision situations can have negative expected values.

This technique can be extended to include multiple decision points and multiple outcomes as long as every possible outcome has a value and a probability of occurrence associated with it.

Methods/Techniques

Additional decision analytic techniques include:

a. Sensitivity analysis, which looks at the relationships between the outcomes and their probabilities to find how “sensitive” a decision point is to the relative numerical values.

b. Value of information methods, whereby expending some effort on data analysis and modeling can improve the optimum expected value.

c. Multi-attribute Utility Analysis (MAUA), which is a method that develops equivalencies between dissimilar units of measure.

Tools

There are many new tools available to support the risk management area. Please check on the INCOSE website for suggestions from the Tools Working Group database.

References

1. Bayes, the Reverend Thomas (1702 - 1761). “Essay Toward Solving a Problem in the Doctrine of Chances.” Philosophical Transactions of the Royal Society of London, 1763.

2. Raiffa, Howard, *Decision Analysis: Introductory Lectures on Choices Under Uncertainty.*, Addison-Wesley, New York, 1968.

3. Schlaiffer, Robert, *Analysis of Decisions Under Uncertainty*, McGraw-Hill Book Company, New York, 1969.

4. U.S. Department of Defense, Defense Advanced Research Project Agency, "Handbook for Decision Analysis", Prepared by Scott Barclay et al. Technical Report TR-77-6-30. September 1977.

References 2 and 3 are both excellent introductions to decision analysis. [Raiffa] has a more mathematical treatment than [Schlaiffer]. Often the relevant uncertainties in a decision problem are continuous in nature; however, decision trees require discrete probability distributions. Simple methods exist for making discrete continuous distributions.

Reference 4 contains an interesting case study that analyzes the decisions to be made by a captain whose warship is being approached by an unidentified airplane in a war situation. The approaching airplane could either be an attacking enemy plane or a damaged friendly plane trying to get close enough for the pilot to ditch and be rescued. The possible outcomes are analyzed as the basic scenario is modified.

THIS PAGE
INTENTIONALLY
LEFT BLANK

APPENDIX E - GLOSSARY AND DEFINITIONS

E- Introduction

One the key activities for the systems engineer is to develop a common language for the project/program. This enables the team members to have a common understanding of the terms and definition used for the development of the system. There are examples of projects where the terms used were not identified or defined sufficiently and projects have failed. The following is a Glossary and Definitions that can be used as a starting point for a systems development project/program. This is by no means a comprehensive list and on any given project within the domain environment that is not U.S. and/or DOD, Mil-Aero centric, the glossary will need to be added to and/or modified to the domain and culture.

E- Glossary and definitions:

Affinity Diagram. The Affinity Diagram, shown in Figure E-1, is a method to organize random, disparate ideas. Its a good place to start the creative process to solve a problem, address an issue, and to define requirements. The Affinity Diagram is a participative, consensus-building tool. The moderator starts the process by gathering an appropriate team and defining the problem to be addressed, e.g., "What are the issues in resolving customer complaints about long delivery times?" For the electric car program the issue could be, "How do we minimize customer inconvenience associated with the need to frequently recharge vehicle batteries?"

The process can be implemented with little yellow Post-Its™ on a large wall or 3 X 5 cards spread on a large table that is accessible to all. Participants write their ideas in approximately 3 to 7 words per card and place them randomly on the table or wall. Participants then seek to organize the ideas into logical groupings. This is done silently, without argument, to avoid domination of the group by a few individuals. (Sometimes those who are hesitant to speak out have the best ideas.)

A title card (header) is prepared for each group. Sometimes it can be drawn from one of the ideas suggested by the group, but usually it's a more all-encompassing, overview title prepared at the end of the process. During the organizing process redundant ideas can be discarded and overlapping ideas rephrased more distinctly on separate cards.

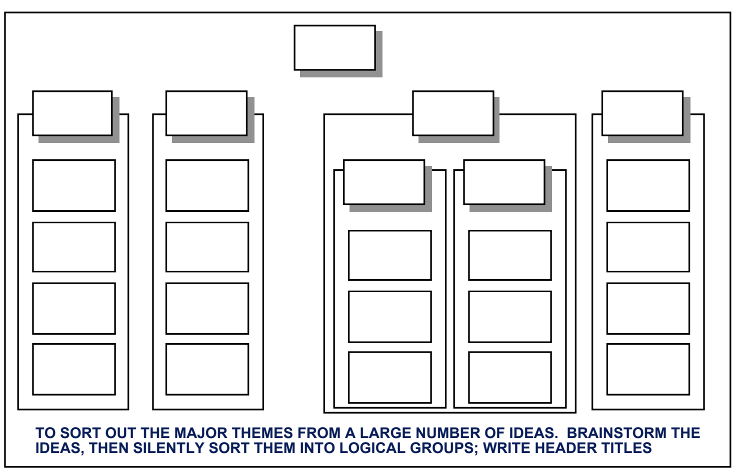


Figure E-1. Affinity Diagram

Allocated Baseline. The initially approved documentation describing a configuration item's (CI) functional, performance, interoperability, and interface requirements that are allocated from those of the system or a higher level CI; interface requirements with interfacing CIs; design constraints; derived requirements (functional and performance); and verification requirements and methods to demonstrate the achievement of those requirements and constraints. The allocated baseline is typically placed under government control during Engineering and Manufacturing Development (EMD). There is an allocated baseline for each configuration item.

Allocated Configuration Identification (ACI). Performance-oriented specifications governing the development of CIs, in which each specification:

a. Defines the functional characteristics that are allocated from those of the system or higher level CI;

b. Establishes the verification required to demonstrate achievement of its allocated functional characteristics;

c. Delineates necessary interface requirements with other associated CIs; and

d. Establishes design constraints, if any such as component standardization, use of inventory items, and integrated logistic support requirements. (MIL-STD-480B, Para 3.1.1)

Allocation.

1 The assignment of a requirement to a function.

2. The assignment of a system element to a requirement.

3. The division of a requirement, for example, weight, into parts and assignment of each part to a separate element.

Authentication. An act by the Government that results in the Government approving and taking control of a configuration baseline.

Best Practice. A best practice is a good practice that is available for use by other projects or for incorporation into the standard engineering process in order to improve development productivity or product quality. It is a relative term and usually indicates innovative or interesting business practices, which have been identified as contributing to improved performance, reduced cost or faster schedules. A good practice is a practice that has been used on at least one engineering development project and is considered to be worthy of consideration for use by other projects.

Capability. A measure of the system's ability to achieve the mission objectives, given that the system is dependable and suitable. Examples of capability measures are accuracy, range, payload, lethality, information rates, number of engagements, and destructiveness. Capability measures can be used as performance requirements, design constraints, and/or technical exit criteria. Capability is a systems engineering metric.

Compatibility. The capability of two or more items or components of equipment or material to exist or function in the same system or environment without mutual interference.

Computer Software Component (CSC).

1. A distinct part of a CSCI. CSCs may be further decomposed into other CSCs and CSUs. (DOD-STD-2167A, Para 3.8)

2. A functional or logical distinct part of a CSCI. CSCs may be top-level, or lower-level. (MIL-STD-483B, Para 5.1e)

Computer Software Configuration Item (CSCI).

1. See configuration item. (MIL-STD-483A, Para 5.1h; MIL-STD-490A, Para 1.4.4)

2. A configuration item for computer software. (MIL-STD-480B, Para3.1.8; DOD-STD-2167A, Para 3.9)

Computer Software Documentation.

1. Technical data or information, including computer listings and printouts, which documents the requirements, design, or details of the computer software; explains the capabilities and limitations of the software; or provides operating instructions for using or supporting computer software during the software's operational life. (MIL-STD-480B, Para 3.1.9; DOD-STD-2167A, Para 3.10)

2. Technical data, including computer listings and printouts, in human readable form which documents the design or details of computer software, explains the capabilities of the software, or provides operating instructions for using the software to obtain desired results from a computer. (MIL-STD-1456A, App A, Para 30.7)

Computer Software Unit (CSU). An element specified in the design of a CSC that is separately testable. (DOD-STD-2167A, Para 3.11)

Configuration Baseline. The configuration documentation formally designated by the Government at a specific time during a system's or configuration item's life cycle. Configuration baselines, plus approved changes from those baselines, constitute the current configuration documentation. There are three formally designated configuration baselines, namely the functional, allocated, and product baselines.

Configuration Item (CI). An aggregation of system elements that satisfies an end use function and is designated for separate configuration management.

Configuration Management (CM).

1. A discipline applying technical and administrative direction and surveillance to:
   a. Identify and document the functional and physical characteristics of CIs:
   b. Control changes to CIs and their related documentation; and
   c. Record and report change processing and implementation status. (MIL-STD-480B, Para 3.1.16)
2. A discipline applying technical and administrative direction and surveillance to:
   a. Identify and document the functional and physical characteristics of CIs:
   b. Audit the CIs to verify conformance to specifications, interface control documents, and other contract requirements;
   c. Control changes to CIs and their related documentation; and
   d. Record and report information needed to manage CIs effectively, including the status of proposed changes and the implementation status of approved changes. (MIL-STD-1456A, App A Para 30.15)

Configuration Management Plan (CMP). The CM plan defines the implementation (including policies and methods) of CM on a particular program/project. It may or may not impose contractor requirements depending on whether it is incorporated on the contract. (MIL-STD-483B, Para 5.1k)

Configuration Status Accounting (CSA).

1. The recording and reporting of information needed to manage configuration effectively, including:
   a. A listing of the approved configuration identification;
   b. The status of proposed changes, deviations, and waivers to the configuration;
   c. The implementation status of approved changes; and
   d. The configuration of all units of the CI in the operational inventory. (MIL-STD-480B, Para 3.1.17)
2. The recording and reporting of information needed to manage configuration effectively, including:
   a. A listing of the approved configuration identification;
   b. The status of proposed changes, deviations, and waivers to the configuration;
   c. The implementation status of approved changes. (MIL-STD-1456A, App A Para 30.16)

Context Diagram. The top level of a Data Flow Diagram which portrays all inputs and outputs of a given system element but shows no decomposition of the element.

Contract Change Proposal (CCP). A formal priced document also referred to as "task change proposal (TCP)" used to propose changes to the scope of work of the contract. It is differentiated from an *ECP* by the fact that it does not affect specification or drawing requirements. It may be used to propose changes to contractual plans, the SOW, CDRL, etc.

Contract Data Requirements List (CDRL), DD Form 1423. A form used as the sole list of data and information which the contractor will be obligated to deliver under the contract, with the exception of that data specifically required by standard Defense Acquisition Regulation (DAR) clauses. (MIL-STD-1388-1A, Para 20.)

Contract Work Breakdown Structure (CWBS). The CWBS is the complete WBS covering a particular contractor on a particular procurement. (MIL-HDBK-259(Navy), Para 3.5.2)

Cost Requirements. The financial thresholds and objectives expressed in terms of design-to-cost targets, research, development, test and evaluation (RDT&E), operating and support costs, and flyaway, weapon system, unit procurement, program acquisition, and life-cycle costs.

Cost Variance. Under the C/SCSC, the CV = BCWP - ACWP, or the Budgeted Cost of Work Performed (or Earned Value) minus the Actual Cost of Work Performed. A negative CV represents a cost overrun relative to the plan.

Critical Design Review (CDR). This review shall be conducted for each CI when detail design is essentially complete. The purpose of this review will be to:

a. Determine that the detail design of the CI under review satisfies the performance and engineering specialty requirements of the HWCI development specifications;
b. Establish the detail design compatibility among the CI and other items of equipment, facilities, computer software and personnel;
c. Assess CI risk areas (on a technical, cost, and schedule basis);
d. Assess the results of the producibility analyses conducted on system hardware; and
e. Review the preliminary hardware product specifications.

For CSCIs, this review will focus on the determination of the acceptability of the detailed design, performance, and test characteristics of the design solution, and on the adequacy of the operation and support documents. (MIL-STD-1521B, Para 3.5 and Appendix E)

Customers. Usage of this term has grown to include both internal and external customers. External customers are the procurers and users of system end items.

Customer Requirements. Statements of fact and assumptions that define the expectations of the system in terms of mission or objectives, environment, constraints, and measures of effectiveness. These requirements are defined from a validated needs statement (Mission Needs Statement), from acquisition and program decision documentation, and from mission analyses of each of the primary system life-cycle functions.

Data.

1. Recorded information, regardless of form or characteristics, including administrative, managerial, financial, scientific, technical, engineering, and logistics data, whether required to be delivered to the Government or retained by the contractor, as well as data developed by the Government. (MIL-STD-480B, Para 3.1.23)
2. Recorded information, regardless of form or method of the recording. (MIL-STD-961C, Para 3.8; MIL-HDBK-59A, App A, Para 30.4.1)
3. The raw materials from which a user extracts information. Data may include numbers, words, pictures, etc. (MIL-STD-1472D, Para 3.12)

Data Base. A set of related data, usually organized to permit efficient retrieval of specified subsets. In training simulation often used for environment models especially for visual and radar landmass simulation. (MIL-HDBK-220B)

Data Flow Diagram. Shows the interconnections for each of the behaviors that a system element must perform, including all inputs and outputs along with the data stores, that each behavior path must access.

Decision Data Base. The collection of data that provides the audit trail from initially stated needs and requirements to the current description of system products and processes. The repository of information used and generated, at the appropriate level for the acquisition phase, of the integrated requirements and flowdowns; interface constraints and requirements; functional and performance requirements: system concept; preliminary design and configuration alternatives; detailed design; verifications; decision criteria; trade study assessments; system, subsystem, and functional capability assessments; and other required documentation. It includes sets of schematic drawings, physical and mathematical models, computer simulations, layouts, detailed drawings, and similar configuration documentation and technical data, as appropriate, and:

a. Illustrates intrasystem, intersystem, and item interfaces;
b. Permits traceability between the elements at various levels of system detail;
c. Provides means for complete and comprehensive change control;
d. Includes the techniques and procedural data for development, manufacturing, verification, deployment, operation, support, training, and disposal;
e. Provides data to verify the adequacy of design development;
f. Provides data for trade-offs and assessments of an item's capability to satisfy objectives; and
g. Provides complete documentation of design to support progressive system development and subsequent iterations of the systems engineering process.

The database allows for presentation of data to reflect the interfunctional correlation and the interfaces between related primary system life-cycle functions (i.e. operations to support to training to manufacturing to deployment to development to verification).

Decomposition. The process of decomposing higher-level requirements into more-detailed constituent functions and associated performance levels and allocating those requirements to specific hardware, software, and support elements.

Dependability. A measure of the degree to which an item is operable and capable of performing its required function at any (random) time, given its suitability for the mission and whether the system will be available and operate when, as many times, and: long as needed. Examples of dependability measures are availability, interoperability, compatibility, reliability, repeatability, usage rates, vulnerability, survivability, penetrability, durability, mobility, flexibility, and reparability. Dependability measures can be used: performance requirements, design constraints, and/or technical exit criteria. Dependability is a systems engineering metric.

Deployment Function. The delivery tasks, actions, and activities to be performed and system elements required to initially transport, receive, process, assemble, install, test, checkout, train, operate and, as required, emplace, house, store, or field the system into a state of full operational capability.

Derived Requirements. Those characteristics typically identified during synthesis of preliminary product or process solutions and during related trade studies and verifications. They generally do not have a parent function and/or performance requirement but are necessary to have generated system elements accomplish their intended function.

Derivative System. A system which, by mandate, must retain major components of a prior system. For example, a derivative aircraft model may achieve increased range while retaining its fuselage or other major elements.

Design. (verb) The process of defining, selecting, and describing solutions to requirements in terms of products and processes. (noun) The product of the process of designing that describes the solution (either conceptual, preliminary, or detailed) of the system, system elements or system end-items.

Design Constraints. The boundary conditions within which the developer must remain while allocating performance requirements and/or synthesizing system elements. These design constraints may be externally imposed (e.g., safety, environmental) or internally imposed as a result of prior decisions which limit subsequent design alternatives. Examples of these constraints include: form, fit, function, interface, technology, material, standardization, cost, and time.

Design Parameters. Qualitative, quantitative, physical, and functional value characteristics that are inputs to the design process, for use in design tradeoffs, risk analyses, and development of a system that is responsive to system requirements. (MIL-STD-1388-1A, Para 20.)

Design Requirements. The “build to,” “code to,” and “buy to” requirements for products and “how to execute” requirements for processes. Design requirements are developed through synthesis of detailed design.

Development Function. The planning and execution of the definition, design, design implementation, integration, analyses, and control tasks, actions, and activities required to evolve the system from customer needs to system product and process solutions. Development applies to new developments, product improvements, and modifications, as well as any assessments needed to determine a preferred course of action for material solutions to identified needs, deficiencies, or problem reports.

Disposal Function. The tasks, actions, and activities to be performed and system elements required to ensure that disposal of decommissioned and destroyed or irreparable system end items complies with applicable classified and environmental regulations and directives. Also addresses the short and long term degradation to the environment and health hazards to humans and animals. The disposal function also includes recycling, material recovery, salvage for reutilization and disposal of by-products from development and production.

Effectiveness Analysis. An analytical approach used to determine how well a system performs in its intended utilization environment.

Engineering Change.

1. An alteration in the approved configuration identification of a CI under development, delivered or to be delivered. (MIL-STD-480B, Para 3.1.30)

a. **Class I engineering change.** See MIL-STD-480B, Para 5.1.

b. **Class II engineering change.** See MIL-STD-480B, Para 5.2.

2. An alteration in the configuration of a CI or item to be delivered, or under development, after formal establishment of its configuration identification. (DOD-STD-480B) (MIL-STD-1456A, App A Para 30.19)

Engineering Change Proposal (ECP). A proposed engineering change and the documentation by which the change is described, justified, and submitted to the procuring activity for approval or disapproval. (MIL-STD-480B, Para 3.1.33)

Engineering Change Proposal (ECP) class. See MIL-STD-480B.

Engineering Change Proposal (ECP) types.

1. A term covering the subdivision of ECPs on the basis of the completeness of the available information delineating and defining the engineering change. This will be identified as:

a. **Preliminary ECP.** See MIL-STD-480B, Para 5.1.4.1.

b. **Formal ECP.** See MIL-STD-480B, Para 5.1.4.2.

2. A term that includes both a proposed engineering change and the documentation by which the change is described and suggested. (DOD-STD-480) (MIL-STD-1456A, App A Para 30.20.2)

a. **Preliminary ECP (Type P)**. A type P ECP may be submitted to the Government for review prior to the availability of the information necessary to support a formal ECP. (DOD-STD-480) (MIL-STD-1456A, App A Para 30.20.2a)

b. **Formal ECP (Type F).** A type F ECP provides engineering information and other data in sufficient detail to support formal change approval and contractual authorization, and which may follow the submittal of a preliminary ECP or VECP. (DOD-STD-480) (MIL-STD-1456A, App A Para 30.20.2b)

Engineering Data.

1. Engineering documents such as drawings, associated lists, accompanying documents, manufacturer specifications, manufacturing planning documentation, and standards or other information prepared by a design activity and relating to the design, manufacturer, procurement, test, or inspection of hardware items or services, ad defined in DOD-STD-100 and DOD-D-1000. (MIL-STD-1521B, Para 3.15)

2. Any technical data (whether prepared by the government, contractor or vendor) relating to the specification, design, analysis, manufacture, acquisition, test, inspection, or maintenance of items or services. All information which contains authoritative engineering definition or guidance on material, constituent items, equipment or system practices, engineering methods and processes comprises engineering data. (MIL-HDBK-59A, App A, Para 30.4.3)

Environment. The natural environment (weather, climate, ocean conditions, terrain, vegetation, space conditions); combat environment (dust, fog, nuclear-chemical-biological); threat environment (effects of existing and potential threat systems to include electronic warfare and communications interception; operations environment (thermal, shock, vibration, power variations); transportation and storage environment; maintenance environment; test environments; manufacturing environments (critical process conditions, clean room, stress) and other environments (e.g. software engineering environment, electromagnetic) related to system utilization.

Environmental Requirements. The requirements that characterize the impact of the environment on the system/CI as well as the system/CI impact on the natural environment.

Enterprise Process Improvement Collaboration **(EPIC).** A collaboration of GTE Government Systems, Hughes Aircraft Company, Lockheed Martin, Loral (now part of Lockheed Martin), Software Productivity Consortium/Virginia Center of Excellence, Texas Instruments, and the Software Engineering Institute. This collaboration developed the System Engineering Capability Maturity Model (SE-CMM).

Evaluation.

1. The process of determining whether an item or activity meets specified criteria. (DOD-STD-2167A, Para 3.16)

2. A judgement expressed as a measure or ranking of trainee achievement, instructor performance, job performance, process, application, training material, and other factors. (MIL-STD-1379D, Para 3.38)

Evolutionary Acquisition. An adaptive and incremental strategy applicable to high technology and software intensive systems when requirements beyond a core capability can generally, but not specifically, be defined.

Exit Criteria. The specific accomplishments or conditions that must be satisfactorily demonstrated before an effort can progress further in the current acquisition phase or transition to the next acquisition phase. Technical exit criteria are used for SEMS events and for acquisition phase milestone reviews.

Fidelity. The degree to which a model realistically represents the system or process it is modeling. It is not necessarily synonymous with a model's level of detail or complexity.

Firmware. Computer processing instructions "burned in" to hardware, such as Programmable Read-Only Memory circuits (PROMs). Once the computer program instructions are burned in, they cannot be changed unless reprogrammable devices are used. Firmware replaces software in some applications, such as for computer operating systems.

Flowdown. The allocation of requirements down to successively lower level system elements.

Formal Qualification Review (FQR).

1. The test, inspection, or analytical process by which a group of CIs comprising the system is verified to have met specific contracting agency contractual performance requirements (specifications or equivalent). This review does not apply to hardware or software requirements verified at Functional Configuration Audit (FCA) for the individual CI. (MIL-STD-1521B, Para 3.9 and Appendix I)

2. A formal review, normally accomplished incrementally at the contracting facility, of test reports and test data generated during the formal qualification of a new group of CIs comprising a system to ensure that all tests required by Section 4 of the developmental specification(s) have been accomplished and that the system performs as required by Section 3. Usually held in conjunction with the FCA, it may be delayed until after the Physical Configuration Audit (PCA) if total system testing is required. See MIL-STD-1521B. (MIL-STD-483B, Para 5.10)

Formal Qualification Testing (FQT). A process that allows the contracting agency to determine whether a CI complies with the allocated requirements for that item.
(DOD-STD-2167A, Para 3.18)

Function. A task, action, or activity that must be performed to achieve a desired outcome.

Functional Analysis and Allocation. Examination of a defined function to identify all of the subfunctions necessary to the accomplishment of that function. The subfunctions are arrayed in a functional architecture to show their relationships and interfaces (internal and external). Upper-level performance requirements are flowed down and allocated to lower-level subfunctions.

Functional Architecture. The hierarchical arrangement of functions, their internal and external (external to the aggregation itself) functional interfaces and external physical interfaces, their respective functional and performance requirements, and the design constraints.

Functional Baseline. The initially approved documentation describing a system's or CI's functional, performance, interoperability, and interface requirements, and the verification required to demonstrate the achievement of those specified requirements. This baseline is normally placed under Government control during Demonstration and Validation.

Functional Configuration Audit (FCA).

1. A formal audit to validate that the development of a CI has been completed satisfactorily and that the CI has achieved the performance and functional characteristics specified in the functional or allocated configuration identification. In addition, the completed operation and support documents shall be reviewed. (MIL-STD-1521B, Para 3.7 and Appendix G)

2. The formal examination of functional characteristics of a CI, prior to acceptance, to verify that the item has achieved the performance specified in its functional or allocated configuration identification. (MIL-STD-480B, Para 3.1.11.1)

Functional Configuration Identification (FCI). The initial approved technical documentation for a CI which prescribes:

a. All necessary functional characteristics;
b. The verification required to demonstrate achievement of specified functional characteristics;
c. The necessary interface characteristics with associated CIs;
d. CI key functional characteristics and lower level CIs, if any; and
e. Design constraints, such as envelope dimensions, component standardization, use of inventory items and ILS policies. (MIL-STD-480B, Para 3.1.41)

Functional Requirement. The necessary task, action, or activity that must be accomplished. The initial set of top-level functions are the eight primary system life-cycle functions. Top-level functions are identified by requirements analysis and subdivided by functional analysis.

Functional Review. An incremental review conducted by a functional team, composed of representatives from the appropriate level of multi-disciplinary product teams, to address progress for a given function (e.g. support) across the system. Functional reviews are intended to provide across the system feedback, to determine and satisfy (via integration) functional planning requirements, and to identify and assess issues. Issues that arise during the review are resolved by the impacted multi-disciplinary product team(s). The multi-disciplinary product team(s) (not the functional team) implements any necessary corrective actions. If the issue is not resolved by the multidisciplinary product team(s), it is addressed at the next subsystem or interim system review.

Government Furnished Material (GFM). Material provided by the Government to the contractor or comparable Government production facility to be incorporated in, attached to, used with or in support of an end item to be delivered to the Government or ordering activity, or which may be consumed or expended in the performance of a contract. It includes, but is not limited to, raw and processed materials, parts, components, assemblies, tools and supplies. Material categorized as Government Furnished Equipment (GFE) and Government Furnished Aeronautical Equipment (GFAE) are included. (MIL-STD-1388-1A, Para 20.)

Hanger Queen. A low cost, semi-prototype test vehicle used by some aerospace companies for informal mechanical and electrical developmental testing of components to avoid wearing out or damaging expensive, possibly one-of-a-kind, flight vehicles.

Hardware Configuration Item (HWCI).

1. See configuration item. (MIL-STD-483B, ¶5.1q; MIL-STD-490A, Para 1.4.3)
2. A configuration item for hardware. (DOD-STD-2167A, Para 3.20)

Human Engineering. The area of human factors that applies scientific knowledge to achieve effective user-system integration. (MIL-H-46855B, Para 6.2.6)

Human Engineering Design Criteria. The summation of available knowledge which defines the nature and limits of human capabilities as they relate to the checkout, operation, maintenance and control of systems or equipment and which may be applied during engineering design to achieve optimum compatibility between equipment and human performance. (MIL-STD-1472D, Para 3.36)

Human Factors. A body of scientific facts about human characteristics. The term covers all biomedical and psycho-social 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-H-46855B, Para 6.2.7)

Independent Verification and Validation (IV&V). Verification and validation performed by a contractor or Government agency that is not responsible for developing the product or performing the activity being evaluated. IV&V is an activity that is conducted separately from the software development activities governed by this standard. (DOD-STD-2167A, Para 3.21)

Interface. The specifically defined physical or functional juncture between two or more configuration items. (MIL-STD-1456A, App A Para 30.22)

Interface Agreement. A document that describes the mutually agreeable configuration management practices and procedures for a given system or CI when more than one agency is designated design responsibility to perform management functions for items that interface with the configuration item. (MIL-STD-1456A, App A Para 30.23)

Interface Control.

1. The process of Identifying all functional and physical characteristics relevant to interfacing of two or more items provided by one or more organizations.(MIL-STD-480B, Para 3.1.43)

2. Interface control comprises the delineation of the procedures and documentation, both administrative and technical, contractually necessary for identification of functional and physical characteristics between two or more CIs which are provided by different contractors/Government agencies, and the resolution of problems thereto (MIL-STD-483B, Para 5.1r)
3. The delineation of the procedures and documentation, both administrative and technical, necessary for identification and management of functional and physical characteristics between two or more systems or CIs. (MIL-STD-1456A, App A Para 30.24)

Interface Control Working Group (ICWG).

1. For programs that encompass a system/CI design cycle, an ICWG normally is established to control interface activity between contractors or agencies, including the resolution of interface problems and documentation of interface agreements. (MIL-STD-483B, Para 5.1s)

2. For programs which encompass a system/CI/CSCI design cycle, and ICWG normally is established to control interface activity between the procuring activity, contractors and other agencies, including resolution of interface problems and documentation of interface agreements. See MIL-STD-483A. (MIL-STD-480B, Para 3.1.22)

Interface Requirement. The functional performance, electrical, environmental, human, and physical requirements and constraints that exist at a common boundary between two or more functions, system elements, configuration items, or system.

Interim System Review. A review conducted across the entire system to assess system development progress and to address issues not solved by a subsystem team. This type of review is conducted as often as necessary before formal major system level reviews (generally at least one would be held prior to initiating the first formal subsystem review).

Item. A non-specific term used to denote any product, including systems, subsystems, assemblies, subassemblies, units, sets, parts, accessories, computer programs, or computer software. In this standard, it also denotes any process that includes a series of actions, changes, or functions to achieve an end or result.

Lessons learned. A lesson learned is defined as any significant experience, observation, or insight that imparts beneficial knowledge relative to the performance of our work, Mission Success, or our products. A lesson learned occurs when a product or process element problem is fixed by a "one-time only" action that is insufficient to prevent its recurrence. Because of the problem's repetitive nature, the lesson learned will need to be flowed to the next similar milestone, event, product, or mission. Lessons learned are sometimes referred to as "knowledge reuse."

Positive lessons learned can provide inputs to determining best practices. Negative lessons learned, typically called mistakes, can indicate areas needing process improvement. We must understand why mistakes occurred in order to learn and avoid making the same mistakes the next time a similar situation occurs. When mistakes occur, it is important to analyze them to understand the root cause that caused the problem and the response originally used.

Life-Cycle Cost (LCC).

1. LCC is the sum total of the direct, indirect, non-recurring, recurring, and other related costs incurred, or estimated to be incurred in the design, research and development (R&D), investment, operation, maintenance, and support of a product over its live cycle, i.e., its anticipated useful life span. It is the total cost of the R&D, investment, O&S and, where applicable, disposal phases of the life cycle. All relevant costs should be included regardless of funding source or management control. (MIL-HDBK-259 (Navy), Para 3.3)

2. The sum total of the direct, indirect, non-recurring, recurring, and other related costs incurred, or estimated to be incurred, in the design, development, production (including manufacture and fabrication), acquisition, test and evaluation, acceptance, operation, maintenance, modernization, deactivation and support of a configuration item over its anticipated life span. (MIL-STD-480B, Para 3.1.48)

3. Includes all cost categories, both contract and in-house, and all related appropriations. It is the total cost to the government for a system over its full life, and includes the cost of development, procurement, operating, support, and, where applicable, disposal. (MIL-STD-1785, Para 3.11)

Life Cycle Cost Analysis. The identification, quantification, and qualification of LCC by segment with the purpose of establishing the cost interrelationships and the effect of each contributor to the total LCC. See Section 4.5.4 and cost element. (MIL-HDBK-259(Navy), Para 3.4.1)

Life Cycle Costing. Life cycle costing is the usage of LCC (or segments thereof) in various decisions associated with acquiring a product. (MIL-HDBK-259(Navy), Para 3.4)

Life Cycle Resources. All resources required for development, manufacturing, verification, deployment, operations, support, and disposal (including by-products) of an item throughout its life cycle. Also included are the resources required for training personnel in the operations and maintenance of an item throughout out its life cycle. These resources are measured in terms of:

a. Time (e.g., time required to develop and/or produce the item);
b. Dollars (e.g., RDT&E, production. operations and support);
c. Manpower (e.g., number of people required to develop, produce, support, and operate an item); and
d. Strategic materials.

Listserver. A mailing list server whose function is to distribute electronic mail (E-mail) to its users. Users on the network subscribe to the list server by having their E-mail address added to it.

Local Area Network (LAN). A communications network designed for a moderate size geographic area and characterized by moderate to high data transmission rates, low delay, and low bit error rates. (DIS PDU (draft), Para 3.20)

Logistic Support Analysis (LSA).

1. The selective application of scientific and engineering efforts undertaken during the acquisition process, as part of the system engineering and design process, to assist in complying with supportability and other ILS objectives. (MIL-STD-1388-1A, Para 20.)

2. LSA is a systems engineering and design process selectively applied during all life cycle phases of the system/equipment to help ensure supportability objectives are met. (MIL-STD-1785, Para 3.12)

Maintainability.

1. The measure of the ability of an item to be retained in or restored to 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. (MIL-STD-1388-1A, Para 20.)

2. A measure of the time or maintenance resources needed to keep an item operating or to restore it to operational status (or serviceable status). Maintainability may be expressed as the time to do maintenance (for example, maintenance downtime per sortie), as the staff required (for example, maintenance personnel per operational unit), or as the time to restore a system to operational status (for example, mean down time). (MIL-STD-1785, Para 3.13)

Maintenance. The physical act of preventing, determining, and correcting equipment or software faults. It includes all actions taken to retain system/equipment/product in a useful serviceable condition or to restore it to usefulness/serviceability. Maintenance include inspection, testing, servicing, classification as to serviceability, repair, rebuilding, and reclamation. (MIL-STD-1379D, Para 3.90)

Major Review. A formal design review or audit that constitutes a program milestone event. Prior to EMD the focus of these reviews is on system concepts, system-level requirements, and interface requirements. During END the focus is on system designs. Major reviews that are conducted incrementally consist of multiple formal functional, subsystem and interim system reviews to resolve issues and review progress; and a formal system level review to demonstrate that the system is ready for progressing to the next major event.

Management Plan. A program for the assignment, monitoring, and assessment of the personnel, materials, and resources dedicated to a specific mission, operation, or function. (MIL-STD-1379D, Para 3.91)

Manufacturing Function. The tasks, actions, and activities to be performed and system elements required for fabrication and assembly of engineering test models and brassboards and low-rate initial-production and full-rate production of system end items. It provides for definition of manufacturing methods and/or processes and for the fabrication, assembly, and checkout of component elements including test equipment, tooling, machinery, and manufacturing layouts.

Measure of Effectiveness (MOE). A metric used to quantify the performance of system products and processes in terms that describe the utility or value when executing customer missions. Systems engineering uses MOEs in a variety of ways including decision metrics, performance requirements, and in assessments of expected performance. MOEs can include cost effectiveness metrics.

Measure of Effectiveness Hierarchy. A top-down set of measures of effectiveness that establishes a relationship from customer needs, requirements and objectives to design criteria. The MOE hierarchy assists in the selection of requirements and in analytic estimates and verifications that product and process solutions satisfy customer needs, objectives, and requirements.

Metric. A composite (or calculated value) of one or more measures (e.g., software productivity = source lines of code/labor months). This term also describes a set of values that includes both metrics and measures.

Model Any representation of a function or process, be it mathematical, physical, or descriptive.

Need. A user related capability shortfall (such as those documented in a Mission Need Statement, field deficiency report, or engineering change proposal), or an opportunity to satisfy a capability requirement because of a new technology application or breakthrough, or to reduce costs. A statement of capability required for each supplier-related primary function, including disposal.

Non-developmental Item (NDI).

a. Any item of supply that is available in the commercial marketplace including COTS;
b. Any previously developed item of supply that is in use by a department or agency of the United States, a State or local government, or a foreign government with which the United States has a mutual defense cooperation agreement;
c. Any item of supply described in definition a or b, above, that requires only minor modification in order to meet the requirements of the procuring agency; or
d. Any item of supply that is currently being produced that does not meet the requirements of definition a., b., or c., above, solely because the item is not yet in use or is not yet available in the commercial marketplace.

Non-Developmental Software (NDS). Deliverable software that is not developed under the contract but is provided by the contractor, the Government or a third party. NDS may be referred to as reusable software, Government furnished software (GFS), or commercially available software, depending on its service. (DOD-STD-2167A, Para 3.22)

Object.

1. An abstraction of a physical entity that can (a) store inputs such as data, energy, mass, or parts and (b) perform sets of operations on inputs or stored parameters in response to a stimulus
2. The object may be the user/customer or a surrogate for the user if the user is a community of people.

Operational Effectiveness. An Operational Test & Evaluation (OT&E) metric that measures the overall degree of mission accomplishment of a system when used by representative personnel in the environment planned or expected (e.g., natural, electronic, threat) for operational employment of the system considering organization, doctrine, tactics, survivability, vulnerability, and threat (including countermeasures, initial nuclear weapons effects, nuclear, biological and chemical contamination threats). The operational system that is provided to users from the technical effort will be evaluated for operational effectiveness by a service OT&E agency. Also a useful metric for operational effectiveness assessments.

Operational Suitability. An OT&E metric that measures the degree to which a system can be placed satisfactorily in field use with consideration given to availability, compatibility, transportability, interoperability, reliability, wartime usage rates, maintainability, safety, human factors, manpower supportability, logistics supportability, natural environmental effects and impacts, documentation, and training requirements. The operational system that is provided to users from the technical effort will be evaluated for operational suitability by a service OT&E agency.

Operational Test and Evaluation (OT&E). Test and evaluation, initial operational test and evaluation, and follow-on OT&E conducted in as realistic and operational environment as possible to estimate the prospective system military utility, operational effectiveness, and operational suitability. In addition, OT&E provides information on organization, personnel requirements, doctrine, and tactics. Also, it may provide data to support or verify material in operating instructions, publications, and handbooks. (MIL-STD-1785, Para 3.15)

Operations Function. The tasks, actions, and activities to be performed and the system elements required to satisfy defined operational objectives and tasks in the peacetime and wartime environments planned or expected.

Performance. A quantitative measure characterizing a physical or functional attribute relating to the execution of a mission or function. Performance attributes include quantity (how many or how much), quality (how well), coverage (how much area, how far), timeliness (how responsive, how frequent), and readiness (availability, MTBF). Performance is an attribute for all system personnel, products and process including those for development, production, verification, deployment, operations, support, training, and disposal. Thus, supportability parameters, manufacturing process variability, reliability, and so forth, are all performance measures.

Performance Assessment. The instructor synthesizes all performance measurement information to assess trainee performance. The performance measures may be objective (machine generated information such as number of target hits) or subjective (information gathered through the instructor senses such as proper communication format used). (MIL-HDBK-220B)

Performance Requirement. The extent to which a mission or function must be executed, generally measured in terms of quantity, quality, coverage, timeliness or readiness. Performance requirements are initially defined through requirements analyses and trade studies using customer need, objective, and/or requirement statements. Performance requirements are defined for each identified customer (user and supplier) mission and for each primary function (and subfunction). Performance requirements are assigned to lower level system functions through top-down allocation, and are assigned to system elements, CIs and the system through synthesis.

Physical Architecture. The hierarchical arrangement of product and process solutions, their functional and performance requirements; their internal and external (external to the aggregation itself) functional and physical interfaces and requirements, and the physical constraints that form the basis of design requirements. The physical architecture provides the basis for system/CI baselines as a function of the acquisition phase. It documents one or more physical designs as required to 1) accomplish effectiveness analysis, risk analysis, and technology transition planning; 2) establish the feasibility of physically realizing the functional architecture; 3) identify manufacturing verification, support and training requirements; 4) document the configuration of prototypes and other test articles, and 5) define in increasing detail the solution to identified needs.

Physical Configuration Audit (PCA).

1. A technical examination of a designated CI to verify that the CI "as built" conforms to the technical documentation that defines the CI. (MIL-STD-1521B, Para 3.8 and Appendix H)

2. The formal examination of the "as built" configuration of a CI against its technical documentation to establish the CI's initial product configuration identification (PCI). (MIL-STD-480B, Para 3.1.11.2)

Preliminary Design Review (PDR). This review should be conducted for each CI or aggregate of CIs to:

a. Evaluate the progress, technical adequacy and risk resolution (on technical, cost and schedule basis) of the selected design approach;

b. Determine its compatibility with performance and engineering specialty requirements of the HWCI development specification;
c. Evaluate the degree of definition and assess the technical risk associated with the selected manufacturing methods/processes; and
d. Establish the existence and compatibility of the physical and functional interfaces among the CI and other items of equipment, facilities, computer software, and personnel.
e. Bring the requirements and configuration under configuration control.

For CSCIs, this review will focus on:

a. The evaluation of the progress, consistency, and technical adequacy of the selected top-level design and test approach;
b. Compatibility between software requirements and preliminary design; and
c. On the preliminary version of the operation and support documents. (MIL-STD-1521B, Para 3.4 and Appendix D)

Preplanned Product Improvement. Planned future improvement of developmental systems that defers capabilities associated with elements having significant risks or delays so that the system can be fielded while the deferred element is developed in a parallel or subsequent effort. Provisions, interfaces, and accessibility are integrated into the system design so that the deferred element can be incorporated in a cost effective manner when it becomes available.

Primary Functions. Those essential tasks, actions, or activities that must be accomplished to ensure that the system will satisfy customer needs from a system life-cycle perspective. The eight primary system life-cycle functions are development, manufacturing, verification, deployment, operations, support, training, and disposal.

Prime Mission Product (PMP). The operational product element of the Work Breakdown Structure.

Procuring Contracting Officer (PCO). An individual authorized to enter into contracts and agreements on behalf of the Government, including the issuance of contract modifications that authorize approved configuration changes. (MIL-STD-480B, Para 3.1.57)

Product Baseline. The initially approved documentation describing all of the necessary functional, performance, and physical requirements of the CI; the functional and physical requirements designated for production acceptance testing; and tests necessary for deployment, support, training, and disposal of the CI. This baseline normally includes product, process, and material specifications, engineering drawings, and other related data. In addition to the documentation, the product baseline of a configuration item may consist of the actual equipment and software.

The DoD normally places this baseline under control after completion of the physical configuration audit (PCA). There is a product baseline for each.

Production Readiness Review (PRR). This review is intended to determine the status of completion of the specific actions which must be satisfactorily accomplished prior to executing a production go-ahead decision. The review is accomplished in an incremental fashion during the FSD phase (EMD), usually two initial reviews, and one final review to assess the risk in exercising the production go-ahead decision. In its earlier stages, the PRR concerns such as the need for identifying high risk/low yield manufacturing processes or materials or the requirement for manufacturing development effort to satisfy design requirements. The reviews become more refined as the design matures, dealing with such concerns as production planning, facilities allocation, incorporation of producibility-oriented changes, identification and fabrication of tools/test equipment, long lead item acquisition, etc. Timing of the incremental PRRs is a function of program posture and is not specifically locked in to other reviews. (MIL-STD-1521B, Para 3.10 and Appendix K)

Program/project Work Breakdown Structure (PWBS). The PWBS is the complete WBS for the program or project covering the acquisition phase. It usually contains one or more contract work breakdown structure (CWBS) subsets.(MIL-HDBK-259 (Navy) ¶3.5.1)

Project Unique Identifier (PUID). An identification number assigned to each requirement that facilitates requirements management. This includes traceability from the parent requirement to sibling requirements.

Quality Function Deployment (QFD). Sometimes known as the "House of Quality" because these requirements correlation matrices resemble the elevation diagram of a house. A Japanese-developed technique for relating design requirements for "what", "how", "how much" and competitive "benchmarks" into a series of requirements flowdown charts for design and manufacturing. The Japanese used QFD instead of specifications. For further discussion, see Appendix A.

Reliability.

1. The duration or probability of failure-free performance under stated conditions. (MIL-STD-1388-1A, Para 20.)

2. The probability that an item can perform its intended *function* for a specified interval under stated conditions. (For non-redundant items, this is equivalent to definition (**1**). For redundant items, this is equivalent to mission reliability.). (MIL-STD-1388-1A, Para 20.)

Reliability and Maintainability (R&M) interface. Reliability and maintainability design parameters are a key factor in the design of affordable and supportable systems. R&M parameters provide inputs into the design and LSA process that quantitatively link system readiness to the ILS elements. One of the principal elements of ILS. (MIL-STD-1388-1A, Para 20.)

Requirements. Characteristics that identify the accomplishment levels needed to achieve specific objectives for a given set of conditions. Contractually binding technical requirements are stated in approved specifications.

Requirements Allocation Sheet. A method of documenting requirements allocation and associated rationale (see Figure E-2).

Requirements Analysis. The determination of system specific characteristics based on analyses of customer needs, requirements, and objectives; missions; projected utilization environments for people, products, and processes; constraints; and measures of effectiveness. Requirements analysis assists the customers in refining their requirements in concert with defining functional and performance requirements for the system's primary life cycle functions. It is a key link in establishing achievable requirements that satisfy needs.

REV.____

SYSTEM ____________ ELEMENT ____________ REQID ________ DATE ________

REQUIREMENT: ________________________________

RATIONALE: ________________________________

SOURCE/REF: ________________________________

ALLOCATED TO:(ELEMENT/SUBSYSTEM) REQID REQUIREMENT RATIONALE REF.

COMMENTS: ________________________________

APPROVALS:

PREPARER, ORG., TEL. ORGN. MGR., ORG., TEL. CONFIG. CONTROL BD. CHIEF ENGR.

Figure E-2. Requirements Allocation Sheet, Example

Research and Development (R&D) cost. The sum of all contract and in-house costs required to bring a product's development from concept to production including engineering design, analysis, development, test, evaluation, and management. Applicable DoD funds are:

a. Exploratory development (6.2 appropriation),
b. Advance development (6.3A appropriation),
c. Systems development (6.3B appropriation), and
d. Engineering development (6.4 and 6.5 appropriations).

Normally research (6.1 appropriation) is not acquisition related and therefore not usually considered part of the R&D cost. (MIL-HDBK-259(Navy), Para 3.2 and 5.3.1)

Risk. A measure of the uncertainty of attaining a goal, objective, or requirement pertaining to technical performance, cost, and schedule. Risk level is categorized by the probability of occurrence and the consequences of occurrence. Risk is assessed for program, product, and process aspects of the system. This includes the adverse consequences of process variability. The sources of risk include technical (e.g., feasibility, operability, producibility, testability, and systems effectiveness); cost (e.g., estimates, goals); schedule (e.g., technology/material availability, technical achievements, milestones); and programmatic (e.g., resources, contractual).

Risk Management. An organized, analytic process to identify what can go wrong, to quantify and assess associated risks, and to implement/control the appropriate approach for preventing or handling each risk identified.

Risk Management Plan. Description of the risk management program that describes the approach and activities for risk management. The technical risk management plan is an essential part of the SEMP.

Schedule Requirements. Progress characteristics imposed in terms of operational capability, production and surge rates, production and repair cycle times, or other development time constraints.

Schedule Variance. Under the C/SCSC it is: SV = BCWP - BCWS, or Budgeted Cost of Work Performed (or Earned Value) minus the Budgeted Cost of Work Scheduled, where SV is expressed in dollars. A negative SV indicates behind schedule. The approximate number of days of schedule variance can be determined from a plot of budgeted cost versus schedule.

Simulation.

1. Synthetically representing the characteristics of a real world system or situation. For example, in the Tactical Combat Training System (TCTS) context, the combat environment simulation represents selected characteristics of the behavior of
   (i) Physical or abstract entities including ships, aircraft, submarines, weapons, sensors, equipment;
   (ii) The behavior of physical environmental characteristics or phenomena including weather, thermals, sea states; and
   (iii) Combat-related characteristics and events including command and control decisions and interactions, responses to various events, and tactics.

   All the above may encompass interactions with human operators, real tactical systems/equipment, and other simulated entities. (AS 5721B, Para 6.2)

2. Synthetically representing the characteristics of a real world system or situation, typically by interfacing controls and displays (operational or simulated) and positions of the system with a computer, that solves a mathematical model of the real world system and situation. All or portions of the equipment may be simulated by solving mathematical models of the transfer functions in the simulation computer. It is a process of imitating one system with another. The simulation may encompass the interaction of the human operator with operational systems, the operating environment, and weapon platform. (MIL-HDBK-220B)

Software Development File (SDF). A repository for a collection of material pertinent to the development or support of software. Contents typically include (either directly or by reference) design considerations and constraints, design documentation and data, schedule, and status information, test requirements, test cases, test procedures and test results. (DOD-STD-2167A, Para 3.26).

Software Development Library (SDL). A controlled collection of software, documentation, and associated tools and procedures used to facilitate the orderly development and subsequent support of software. The SDL includes the Development Configuration as part of its contents. A software development library provides storage of and controlled access to software and documentation in human-readable form, machine-readable form, or both. The library may also contain management data pertinent to the software development project. (DOD-STD-2167A, Para 3.27).

Software Engineering Environment (SEE). The set of automated tools, firmware devices, and hardware necessary to perform the software engineering effort. The automated tools may include but are not limited to compilers, assemblers; linkers, loaders, operating system, debuggers, simulators, emulators, test tools, documentation tools, and data base management system(s). (DOD-STD-2167A, Para 3.28).

Software Specification Review (SSR). A review of the finalized CSCI requirements and operational concept. Conducted when CSCI requirements have been sufficiently defined to evaluate the contractor's responsiveness to and interpretation of the system, segment, or prime item level requirements. A successful SSR is predicated upon the contracting agency's determination that the SRS(s), IRS(s) and Operational Concept Document form a satisfactory basis for proceeding into preliminary software design. (MIL-STD-1521B Para 3.3 and Appendix C)

Source Documents. User's documents, which are a source of data eventually processed by the computer program, such as target lists, supple codes, parts list, maintenance forms, bills of lading, etc. (MIL-STD-1472D, Para 3.59).

Spares.

1. Spares are units or assemblies used for maintenance replacement purposes in end items of equipment. (MIL-STD-480B, Para 3.1.62)

2. Those support items that are an integral part of the end item or system that are coded as repairable. (MIL-STD-1388-1A, Para 20.)

Spares and Repair Parts. Spares are components or assemblies used in maintenance replacement purposes in major end items of equipment. Repair parts are those "bits and pieces," e.g., individual parts or non-repairable assemblies required for the repair of spares or major end items (DOD-STD-480). (MIL-STD-1456A, App A Para 30.27).

Specification. A document prepared to support acquisition and life cycle management that clearly and accurately describes essential technical requirements and verification procedures for items, materials, and services. When invoked by a contract it is legally enforceable and its requirements are contractually binding.

Specification Change Notice (SCN).

1. A document (DD Form 1696) used to propose, transmit, and record changes to a specification. In proposed form, prior to approval of a Class 1 engineering change, the SCN supplies proposed changes in the text of each page affected. In final approved form, the SCN summarizes the approved changes to the text of each page affected. (MIL-STD-480B, Para 3.1.52 and 5.6)

2. A document used to propose, transmit, and record changes to a specification. In proposed form, prior to approval, the SCN(P) supplies proposed changes in text of each page affected. (MIL-STD-1456A, App A Para 30.29)

Specification Tree (or Spec 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 products and processes that satisfy those needs.

Statement of Work (SOW). The non-specification work tasks to be completed by the contractor. The SOW is the part of a contract in which the systems engineering efforts, appropriately tailored, are defined.

Subcontractor. A person or business that contracts to provide some service or material necessary for the performance of another's contract. (MIL-STD-480B, Para 3.1.65)

Subsystem. A grouping of items satisfying a logical group of functions within a particular system.

Subsystem Review. An incremental review is held at the CI or aggregate of CI level to assess subsystem development risks, issues, and progress. It is conducted by an integrated, multi-disciplinary product team. Subsystem reviews can be formal (review of a single CI as part of PDR) or informal (a working group meeting assessing progress and actions required to meet future required accomplishments).

Sub-task. Activities (perceptions, decisions, and responses) that fill a portion f the immediate purpose within a task (for example, remove a lug nut). (MIL-STD-1379D, Para 3.137)

Suitability. A measure of the degree to which a system is appropriate for its intended use with respect to non-operational factors such as man-machine interface, training, safety, documentation, producibility, testability, transportability, maintainability, manpower availability, supportability, and disposability. The level of suitability determines whether the system is the right one to fill the customers' needs and requirements. Suitability measures can be used as performance requirements, design constraints, and/or technical exit criteria. Suitability is a systems engineering metric.

Suppliers. The development, manufacturing, verification, and deployment personnel that define, design, code, fabricate, assemble, integrate, verify, test, deliver, and/or install system end items, and safely dispose of the by-products of their activities.

Support Function. The tasks, actions, and activities to be performed and the system elements required to provide operations, maintenance, logistics (including training) and materiel management support. It provides for the definition of tasks, equipment, skills, personnel, facilities, materials, services, supplies, and procedures required to ensure the proper supply, storage, and maintenance of a system end item.

Survivability. The capability of a system to avoid or withstand a hostile environment without suffering an abortive impairment of its ability to accomplish its designated mission. Survivability includes nuclear survivability. (MIL-STD-480B, Para 3.1.67)

Synthesis. The translation of functions and requirements into possible solutions (resources and techniques) satisfying the basic input requirements. System element alternatives that satisfy allocated performance requirements are generated; preferred system element solutions that satisfy internal and external physical interfaces are selected; system concepts, preliminary designs and detailed designs are completed as a function of the development phase: and system elements are integrated into a physical architecture.

System. An interacting combination of elements to accomplish a defined objective. These include hardware, software, firmware, people, information, techniques, facilities, services, and other support elements.

System Architecture. The arrangement of elements and subsystems and the allocation of functions to them to meet system requirements.

System Design Review (SDR). This review shall be conducted to evaluate the optimization, correlation, completeness, and risks associated with the allocated technical requirements. Also included is a summary review of the system engineering process that produced the allocated technical requirements and of the engineering planning for the next phase of effort. Basic manufacturing considerations will be reviewed and planning for production engineering in subsequent phases will be addressed. This review will be conducted when the system definition effort has proceeded to the point where system characteristics are defined and the CIs are identified. (MIL-STD-1521B, Para 3.2 and Appendix B)

System Effectiveness. A quantitative measure of the extent to which a system can be expected to satisfy customer needs and requirements. System effectiveness is a function of suitability, dependability, and capability. System effectiveness is a systems engineering metric.

System Elements. The basic constituents (hardware, software, facilities, personnel, data, material, services, or techniques) that comprise a system and satisfy one or more requirements in the lowest levels of the functional architecture.

System End Item. A deployed system product and/or process that is ready for its intended use.

System Life Cycle. The period extending from inception of development activities, based on an identified need or objective, through decommissioning and disposal of the system.

Systems Analysis and Control. The assessment and control mechanisms, including performance based progress measurements, to:

a. Establish system effectiveness.
b. Balance cost, schedule, performance, and risk.
c. Control the system configuration.

Systems Engineering. An interdisciplinary approach and means to enable the realization of successful systems. [1] Systems engineering:

a. encompasses the scientific and engineering efforts related to the development, manufacturing, verification, deployment, operations, support, and disposal of system products and processes;
b. develops needed user training equipments, procedures, and data;
c. establishes and maintains configuration management of the system;
d. develops work breakdown structures and statements of work; and
e. provides information for management decision making.

Systems Engineering Detailed Schedule (SEDS). The detailed, task oriented schedule of the work efforts required to support the events and tasks identified in the SEMS. The SEDS is used to track day-to-day progress and includes the continual assessment of the technical parameters required to support each SEMS task/event.

Systems Engineering Management Plan (SEMP). A comprehensive document that describes how the fully integrated engineering effort will be managed and conducted.

Systems Engineering Master Schedule (SEMS). A compilation of key accomplishments, requiring successful completion to pass identified events. Accomplishments include major and critical tasks, activities, and demonstrations, with associated accomplishment criteria. Events include technical reviews and audits, demonstration milestones, and decision points. Successful completion is determined by the measurable criteria defined for each accomplishment. Examples of the criteria include completed work efforts and technical parameters used in TPM. Quantitative inputs into program decision points comes from the data associated with the accomplishment criteria.

---

[1] The INCOSE Board of Directors has approved this definition of Systems Engineering.

Systems Engineering Process. A comprehensive, iterative problem solving process that is used to:

a. transform validated customer needs and requirements into a life-cycle balanced solution set of system product and process designs,

b. generate information for decision makers, and

c. provide information for the next acquisition phase. The problem and success criteria are defined through requirements analysis, functional analysis/allocation, and systems analysis and control.

Alternative solutions, evaluation of those alternatives, selection of the best life-cycle balanced solution, and the description of the solution, through the design package are accomplished through synthesis and systems analysis and control.

System Requirements Review (SRR). The objective of this review is to ascertain the adequacy of the contractor's efforts in defining system requirements. It will be conducted with a significant portion of the system functional requirements has been established. (MIL-STD-1521B, Para 3.1 and Appendix A)

System Security Engineering (SSE). An element of systems engineering that applies scientific and engineering principles to identify security vulnerabilities and minimize or contain risks associated with these vulnerabilities. It uses mathematical, physical, and related scientific disciplines, and the principles and methods of engineering design and analysis to specify, predict, and evaluate the vulnerability of the system to security threats. (MIL-STD-1785, Para 3.21)

System Security Management Plan (SSMP). A formal document that fully describes the planned security tasks required to meet system security requirements, including organizational responsibilities, methods for accomplishment, milestones, depth of effort, and integration with other program engineering, design and management activities and related systems. (MIL-STD-1785, Para 3.23)

Tailoring.

1. The process by which specific requirements (sections, paragraphs, or sentences) of the selected specifications, standards, and related documents are evaluated, to determine the extent to which each requirement is most suitable for a specific material acquisition and the modification of these requirements, where necessary, to ensure that each tailored document invoked states only the minimum needs of the Government (DOD 4120.3-M) (MIL-STD-1456A, App A Para 30.33)

2. The process by which specific requirements (sections, paragraphs, or sentences) of the specifications, standards, and related documents are evaluated to determine the extent to which each requirement is 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. (see MIL-HDBK-248B and 4.2.1). The tailoring of data product specifications and DIDs shall be limited to the exclusion of information requirement provisions. (MIL-STD-961C, Para 3.41 and 4.2.1)

3. The process by which the individual requirements (sections, paragraphs, or sentences) of the selected specifications and standards are evaluated to determine the extent to which each requirement is most suitable for a specific material acquisition and the modification of these requirements, where necessary, to assure that each tailored document invoked states only the minimum needs of the Government. Tailoring is not a license to specify a zero LSA program, and must conform to provisions of existing regulations governing LSA programs. (MIL-STD-1388-1A, Para 20.)

Task.

1. A single unit of specific work behavior with clear beginning and ending points and directly observable or otherwise measurable process, frequently but not always resulting in a product that can be evaluated for quality, quantity, accuracy, or fitness in the work environment. (MIL-STD-1379D, Para 3.142; MIL-STD-1388-1A, Para 20.)

2. A task is performed for its own sake, that is, it is not dependent upon other tasks, although it may fall in a sequence with other tasks in a duty of job array. (MIL-STD-1379D, Para 3.142)

3. Formed in clusters which make up duties. (MIL-HDBK-220B)

4. A task is the lowest level of behavior in a job that describes the performance of a meaningful function in the job under consideration. (MIL-STD-1388-1A, Para 20.; MIL-HDBK-220B)

Task Analysis.

1. A process of reviewing job content and context as it pertains to an emerging equipment design, to classify units of work (duties/primary skills and tasks/discrete skills) within a job. The process provides a procedure for isolating each unique unit of work and for describing each unit until accomplished. (MIL-STD-1388-1A, Para 20.)

2. 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, rather than a sequential operation of the equipment. It is a part of systems engineering analysis where systems engineering is required. (MIL-H-46855B, Para 6.2.5)

Task Description. Verbal description, in column, outline, decision table, or time-line format that describes the required job behavior at the highest level of generality. Intended to provide an overview of total performance. (MIL-STD-1379D, Para 3.143)

Technical Data. The recorded information (regardless of the form or method of recording) of a scientific or technical nature (including computer software documentation) relating to products and processes. Technical data is required to define and document an engineering design, product configuration, or process description (sufficient to allow duplication of the original item) and is used to support engineering, manufacturing, logistics, and sustaining engineering.

Technical Data Package. The engineering drawings, associated lists, process descriptions, and other documents that define system product and process physical geometry; material composition; performance characteristics; and manufacture, assembly, and acceptance test procedures.

Technical Effort. Any activity that influences system design.

Technical Objectives. The “target” values for the development effort, when insufficient data is available for stating binding technical requirements. Also can be used to define capabilities beyond established technical requirements when opportunities have been identified for substantial increases in effectiveness, decreases in cost, or additional flexibility. Includes cost, schedule, and performance attributes deemed important.

Technical Parameters (TPs). A selected subset of the system's technical metrics tracked in TPM. Critical technical parameters are identified from risk analyses and contract specification or incentivization, and are designated by management. Example of Technical Parameters include:

a. Specification Requirements.
b. Metrics associated with technical objectives and other key decision metrics used to guide and control progressive development.
c. Design to cost requirements.
d. Parameters identified in the acquisition program baseline or user requirements documentation.

Technical Performance Measurement (TPM). The continuing verification of the degree of anticipated and actual achievement of technical parameters. TPM is used to identify and flag the importance of a design deficiency that might jeopardize meeting a system level requirement that has been determined to be critical. Measured values that fall outside an established tolerance band require proper corrective actions to be taken by management. Relevant terms and relationships are illustrated in Figure E-3.

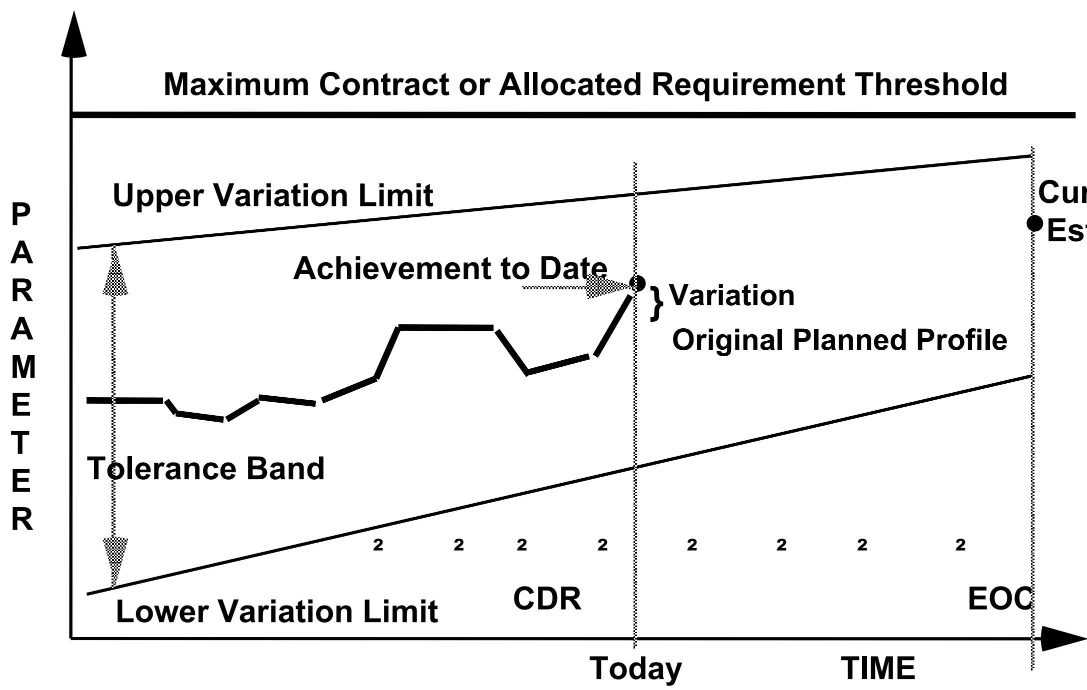


Figure E-3. Technical Performance Measurement Profile Illustration

a. **Achievement to Date.** Measured progress or estimate of progress plotted and compared with the planned progress at designated milestone dates.

b. **Current Estimate.** The value of a technical parameter that is predicted to be achieved with existing resources by the End of Contract (EOC).

c. **Milestone.** Time period when a TPM evaluation is accomplished. Evaluations are made to support technical reviews, significant test events, and cost reporting intervals.
d. **Planned Value.** Predicted value of the technical parameter for the time of measurement based on the planned profile.

e. **Planned Profile.** Profile representing the projected time-phased demonstration of a technical parameter requirement.

f. **Tolerance Band.** Management alert limits placed each side of the planned profile to indicate the envelop or degree of variation allowed. The tolerance band represents the projected level of estimating error.

g. **Threshold**. The limiting acceptable value of a technical parameter; usually a contractual performance requirement.

h. **Variation**. Difference between the planned value of the technical parameter and the achievement-to-date value derived from analysis, test, or demonstration

Technical Reviews. A series of systems engineering activities by which the technical progress of a program is assessed relative to its technical or contractual requirements. Conducted at logical transition points in the development effort to reduce risk by identifying and correcting problems/issues resulting from the work completed before the program is disrupted or delayed. Provide a method for the contractor and Government to determine that the development of a system and/or configuration item and its documentation, have met contract requirements. Includes incremental reviews (functional, subsystem, and interim system) and major system level technical reviews.

TEMPEST. Government requirements on the electromagnetic emissions from electronic boxes, buildings, etc. to minimize the probability of data intercept by unauthorized parties, e.g., the building is TEMPEST qualified.

Test. Any device/technique used to measure the performance, skill level, and knowledge of an individual. (MIL-STD-1379D, Para 3.145)

Test Readiness Review (TRR). A review conducted for each CSCI to determine whether the software test procedures are complete and to assure that the contractor is prepared for formal CSCI testing. Software test procedures are evaluated for compliance with software test plans and descriptions, and for adequacy in accomplishing test requirements. At TRR, the contracting agency also reviews the results of informal software testing and any updates to the operation and support documents. A successful TRR is predicated on the contracting agency's determination that the software test procedures and informal test results from a satisfactory basis for proceeding into formal CSCI testing. (MIL-STD-1521B, Para 3.6 and Appendix F)

Time Requirements. Factors critical to achieving required functional capabilities that are dependent on accomplishing a given action within an opportunity window (e.g., a target is vulnerable to attack only for a certain amount of time). Frequently defined for mission success, safety, system resource availability, and production and manufacturing capabilities.

Time Line Analysis. Analytical task conducted to determine the time sequencing between two or more events. Examples of time lines include:

a. A schedule line showing key dates and planned events
b. A mission flight path identifying when and where planned changes in course and velocity take place
c. A portion of an engagement profile detailing time based position changes between a weapon and its target.

Training Function. The tasks, actions, and activities to be performed and system elements required to achieve and maintain the knowledge and skill levels necessary to efficiently and effectively perform operations and support functions.

Tree Diagram. A chart used to break out tasks, requirements, or functions, etc., into elements of increasing detail, as shown in Figure E-4.

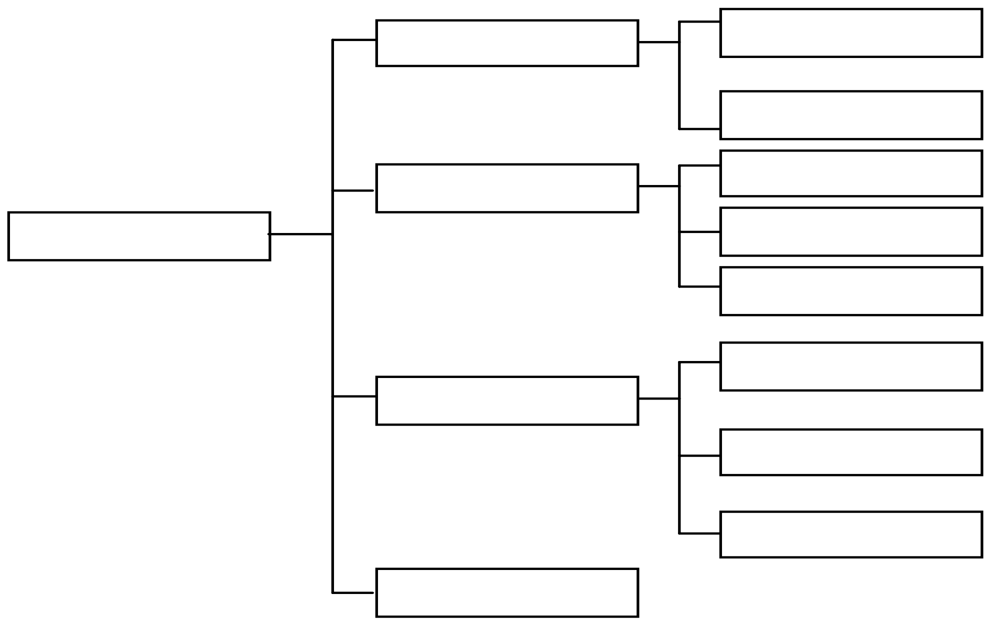


Figure E-4. Tree Diagram

Usenet. A facet of the Internet that provides asynchronous text discussion on many topics -- separated into newsgroups.

Users. The operators and supporters of system end items, and the trainers that train the operations and support personnel. Users execute the operations, support, training, and disposal functions associated with system end items.

Validation.

1. The determination that the requirements for a product are sufficiently correct and complete. (ARP 4754, 1996)

2. The effort required of the contractor or preparing activity, during which the technical data product is tested for technical adequacy, accuracy, and compliance with the provisions of the specifications and other contractual requirements. Validation is accomplished by comparing the data product with the actual systems or equipment for which the data product was prepared. Validation is normally conducted at the preparing activity or vender's facility. In extenuating circumstances, validation may be conducted at an alternative site. (MIL-HDBK-59A, App A, Para 30.8.6)

3. The process of evaluating software to determine compliance with specified requirements. (DOD-STD-2167A, Para 3.32)

4. The process by which the curriculum materials and instruction media materials are reviewed by the contractor for instructional accuracy and adequacy, suitability for presentation and effectiveness in providing for the trainee's accomplishment of the learning objectives. Validation is normally accomplished in tryouts with a representative target population. The materials are revised as necessary, as a result of the validation process. (MIL-STD-1379D, Para 3.167)

Vendor. A manufacturer or supplier of an item. (MIL-STD-480B, Para 3.1.73)

Venn Diagram. A graphical technique for displaying functional interrelationships, as shown in Figure E-5. below.

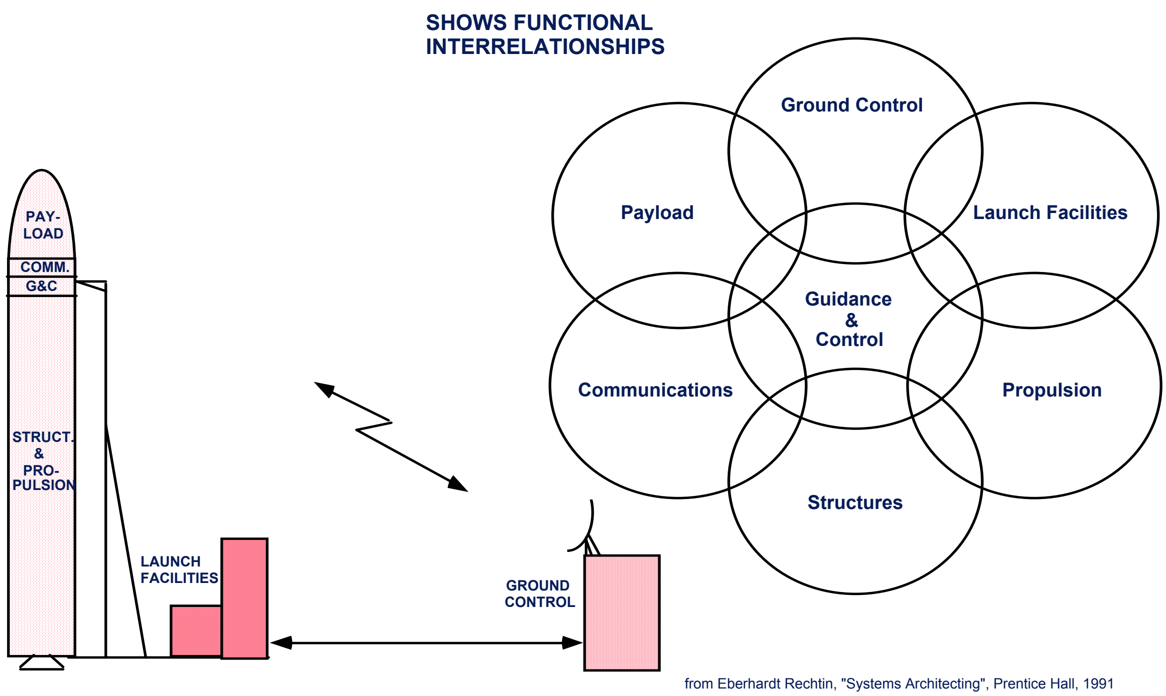


Figure E-5. Venn Diagram, Example

Verification Function. The tasks, actions, and activities to be performed and the system elements required to evaluate progress and effectiveness of evolving system products and processes and to measure specification compliance. Analysis (including simulation), demonstration, test, and inspection are verification approaches used to provide information to evaluate: risk, compliance with specification requirements, product and process capabilities, proof of concept, and warranty compliance. Included are technology verification, manufacturing process proofing, quality assurance and acceptance, and development test and evaluation (DT&E).

Work Breakdown Structure (WBS). A product-oriented family tree composed of hardware, software, services, data, and facilities which result from systems engineering efforts during the development and production of a defense materiel item, and which completely defines the program. Displays and defines the product(s) to be developed or produced, and relates the elements of work to be accomplished to each other and to the end product. Provides structure for guiding multi-disciplinary team assignment and cost tracking and control.

APPENDIX F - ACRONYM LIST

ACI	Allocated Configuration Identification
ACWP	Actual Cost of Work Performed
AHP	Analytic Hierarchy Process
A&IT	Analysis and Integration Team
ANSI	American National Standards Institute
APL	Applied Physics Laboratory (at Johns Hopkins University)
ARP	Aerospace Recommended Practice (SAE designation for aerospace standards)
ATE	Auxiliary Test Equipment; Automatic Test Equipment
ATM	Automated Teller Machine
BCE	Baseline Cost Estimate
BCWP	Budgeted Cost of Work Performed (equals Earned Value in C/SCSC)
BCWS	Budgeted Cost of Work Scheduled
BER	Bit Error Rate
BIT	Built-In Test capability
BPS	Bits Per Second
CAIV	Cost As an Independent Variable
CAWG	Capability Assessment Working Group
C/SCSC	Cost/Schedule Control System Criteria
CBSE	Computer Based System Engineering
CCB	Configuration/Change Control Board
CCP	Contract Change Proposal
CDR	Critical Design Review
CDRL	Contract Data Requirements List
CE	Concept Exploration or Concurrent Engineering
CED	Concept Exploration and Definition (program phase)
CET	Concurrent Engineering Team
CER	Cost Estimating Relationship
CF	Consequence of Failure
CFD	Control Flow Diagram
CI	Configuration Item
CM	Configuration Management
CMM	Capability Maturity Model
CMO	Configuration Management Officer
CMP	Configuration Management Plan
COB	Close of Business
COEA	Cost and Operational Effectiveness Analyses
COTS	Commercial Off-The-Shelf
COMPUSEC	Computer Security
CPU	Central Processor Unit
CSA	Configuration Status Accounting
CSC	Computer Software Component
CSCI	Computer Software Configuration Item
CSM	Center for Systems Management (in Cupertino California)

CSU	Computer Software Unit
CWBS	Contract Work Breakdown Structure
DAA	Designated Accrediting Authority
D/CFD	Data/Control Flow Diagram
DCR	Design Concept Review
DD	Data Dictionary
DEM/VAL	Demonstration and Validation program phase
DFD	Data Flow Diagram
DID	Data Item Description
DMP	Data Management Plan
DMVDOSTD	Development, Manufacturing, Verification, Deployment, Operations, Support, Training, and Disposal
DoD	Department of Defense
DSMC	Defense System Management College
DSS	Decision Support Software
DT&E	Development Test and Evaluation
DTU	Design Test Unit
DTUPC	Design-To-Unit-Production-Cost
D/V	Demonstration/Validation
EAC	Estimate at Complete
ECP	Engineering Change Proposal
EIA	Environmental impact analysis
EIA	Electronic Industries Alliance
EIS	Environmental Impact Statement
EMD	Engineering & Manufacturing Development
EMI/EMC	Electromagnetic Interference/Electromagnetic Compatibility
EMR	Electromagnetic Radiation
EOC	End of Contract
EPA	Environmental Protection Agency
EPIC	Enterprise Process Improvement Collaboration
ERB	Engineering Review Board
ERD	Entity Relationship Diagram
ERP	Effective Radiated Power
ERM	Environmental Risk Management
ETC	Estimate to Complete
EV	Earned Value or Expected Value
EVMS	Earned Value Measurement System
FAR	Federal Acquisition Regulations
FCA	Functional Configuration Audit
FCI	Functional Configuration Identification
FMECA	Failure Modes, Effects and Criticality Analysis
FFD	Functional Flow Diagrams
FQR	Formal Qualification Review
FQT	Formal Qualification Testing
F&R	Functions and Responsibilities
FTP	File Transfer Protocol
G&A	General and Administrative (expenses)

GFAE	Government Furnished Aeronautical Equipment
GFE	Government Furnished Equipment
GFM	Government Furnished Material
HAR	Hazard Analysis Report
HCDE	Human Centered Design Environment
HDN	Human Engineering Design Notes
HE	Human Engineering
HEDD	Human Engineering Design Document
HEPP	Human Engineering Program Plan
HHA	Health Hazard Analysis
HRI	Hazard Risk Index
HSE	Human Systems Engineering
HSI	Human Systems Integration
HWCI	Hardware Configuration Item
ICAM	Integrated Computer-Aided Manufacturing
ICD	Interface Control Document or Interface Control Drawing
ICI	Integrated Communication Infrastructure
ICWG	Interface Control Working Group
IDI	Internal Data Item
IDEF	Integrated DEFinition, and ICAM DEFinition
IEEE	Institute of Electrical and Electronics Engineers
IFS	In Flight Safety or Interface Specification
IFWG	Interface Working Group
ILS	Integrated Logistics Support
INCOSE	International Council On System Engineering (formerly NCOSE)
IPD/CE	Integrated Product Development/Concurrent Engineering
IPDR	Internal Preliminary Design Review
IPDT	Integrated Product Development Team
IPO	Input-Process-Output
IPPD	Integrated Product & Process Development
IPPT	Integrated Product and Process Teams
IRS	Interface Requirements Specification (software)
IV&V	Independent Verification and Validation
JMSNS	Justification for Major System New Start
KFA	Key Focus Area
LAN	Local Area Network
LCC	Life Cycle Cost
LSA	Logistic Support Analysis
MAUA	Multi-attribute Utility Analysis
MDT	Multidisciplinary Team
MEU	Maximum Expected Utility
MIME	Multipurpose Internet Mail Extension
MNS	Mission Need Statements
MOE	Measure of Effectiveness
MS&T	Manufacturing Science and Technology Program
MTBF	Mean-Time-Between-Failures
MTTR	Mean-Time-To-Repair
MWG	MANPRINT Working Group

NCAT	National Center for Advanced Technologies
NCOSE	National Council on Systems Engineering
NDE	Non-Developmental Equipment
NDI	Non-Development Item
NDS	Non-Developmental Software
NEPA	National Environmental Protection Act
OCD	Operational Concept Document
O&S	Operations and Support program phase
OS&HA	Operating and Support Hazard Analysis
OT&E	Operational Test and Evaluation
PA	Product Assurance or Process Areas
PCA	Physical Configuration Audit
PCO	Procuring Contract Officer
PCR	Process Compliance Review
PDCA	Plan-Do-Check-Act (in Shewhart Cycle for Continuous Improvement)
PD&RR	Program Definition and Risk Reduction
PDR	Preliminary Design Review
PDT	Product Development Team
PERT	Program Evaluation and Review Technique
PF	Probability of Failure
PFD&OS	Production, Fielding/Deployment, & Operational Support
PHA	Preliminary Hazard Analysis
PHL	Preliminary Hazard List
PIT	Product Integration Team
PMD	Program Management Directive
PMO	Program Management Office
PMP	Prime Mission Product
POA	Plan Of Attack
PRM	Program Risk Management
PROM	Programmable Read-Only Memory
PRR	Production Readiness Review
RSSI	Received Signal Strength Intensity
PTO	Project Team Organization
PTPO	Project Team Personnel Organization
PUID	Project Unique Identifier
PWBS	Program/Project Work Breakdown Structure
QFD	Quality Function Deployment
QPR	Quarterly Progress Review
RAM	Random Access Memory
R&D	Research and Development
R&M	Reliability and Maintainability
RDTE	Research, Development, Test and Evaluation
RFP	Request For Proposal
RMPP	Risk Management Program Plan
ROM	Read-Only Memory
RTM	Requirements Traceability Matrix

SAE	Society of Automotive Engineers
SAM	SE-CMM Assessment Method
SAR	Safety Analysis Report
SCE	Software Capability Evaluation
SCN	Specification Change Notice
SDF	Software Development File
SDL	Software Development Library
SDN	System Design Notebook
SDR	System Design Review or Software Design Review
SE	Systems Engineering
SECA	Systems Engineering Capability Assessment
SECAM	System Engineering Capability Assessment Model (INCOSE version)
SECM	Systems Engineering Capability Model (EIA/IS 731 Version)
SE-CMM	System Engineering Capability Maturity Model (SEI version)
SEDS	System Engineering Detailed Schedule
SEE	Software Engineering Environment
SEI	Software Engineering Institute (at Carnegie Mellon University)
SE&I	System Engineering and Integration
SE&IT	System Engineering and Integration Team
SEMP	System Engineering Management Plan
SEMS	System Engineering Master Schedule
SFR	System Functional Review
SI	System Integration
SON	Statement of Operational Need
SOO	Statement of Operational Objectives
SOP	Standard Operating Procedure
SORD	System Operational Requirements Document
SOW	Statement of Work
SPA	Software Process Assessment
SRD	System Requirements Document
SRL	System Requirements Letter
SRR	System Requirements Review
SSD	Space Systems Division
SSE	System Security Engineering
SSMP	System Security Management Plan
SSPP	System Safety Program Plan
SSR	System Specification Review
SSS	System/Segment Specification
SSWG	System Safety Working Groups
STD	State Transition Diagram or Standard
STS	Space Transportation System (NASA's Space Shuttle)
SV	Schedule Variance
SVR	System Verification Review
SYSPG	System Engineering Process Group
T&E	Test and Evaluation
TBD	To Be Determined
TBR	To Be Reviewed or To Be Resolved
TBS	To Be Supplied
TCPI	To Complete Performance Index
TCTS	Tactical Combat Training System

TEMP	Test and Evaluation Master Plan
TIM	Technical Interchange Meeting
TPM	Technical Performance Measurement
TP	Technical Parameter
TPO	Team Program Office
TOR	Technical Operational Requirements
TRD	Technical Requirements Document
TRR	Test Readiness Review
TS	Transformation Specification
VCRM	Verification Cross-Reference Matrix
VECP	Value Engineering Change Proposal
WBS	Work Breakdown Structure
WMP	Waste Management Plan

COMMENT FORM

Reviewed document:	**(GUIDE)**	SE Handbook version 2
Name of submitter (first name & last name):		John Doe III
Date Submitted:		21-Aug-2010
Contact info (email address)		john.doe@anywhere.com
Type of submission (individual/ group):		group
Group name and number of contributors (if applic.):		INCOSE XYZ WG
		Please read examples carefully before providing your comments (and delete the examples provided.)

Com-menter's Name	**Com-ment Seq-uence No.**	**Cate gory (TH, TL, E, G)**	**Section Number (eg, 3.4.2.1, no alpha)**	**Specific Reference (e.g., Paragraph, Line, Figure, Table)**	**Issue, comment and rationale *(rationale must make comment clearly evident and supportable)***	**Proposed Changed/New Text -- *MANDATORY ENTRY* -- *(must be substantial to increase the odds of acceptance)***	**Importance Rating (R = Required, I = Important, T = Think About for future version)**
John Doe III	1	E	6.3.2	Paragraph three	Is the inclusion of the spiral model in the incremental life cycle stray text? The spiral model is more often associated with the evolutionary model (6.3.3)	(delete third paragraph)	
John Doe III	2	TH	A5.2.e	first line	Find a different term for reviewing requirements to assure goodness: this is not requirements validation. Call the activity review, or ?	Each technical requirement statement should be reviewed to ensure that it exhibits the following quality attributes:	R

John Doe III	4	TH	A.5.5	This section wants validation to be completed before integration. Usually validation is completed after integration. If this is written as intended, then more amplification is needed to clarify why validation should precede integration. These sound like they are notes for A.5.8; the validation notes section, and belong in that section. This section should address some notes tied directly to integration. (See the SAE TBD WG or INCOSE's Jane Smith for some further thoughts on integration.)	Clarify section. Address integration issues in the Integration Notes section, and put validation issues in the Validation Notes section.	R

Submit comments to SE Handbook Working Group chair. Current SEH WG chair will be listed at: http://www.incose.org/techcomm.html

If this fails, comments may be sent to: info@incose.org (the INCOSE main office), which can relay to the appropriate WG, if so requested in the comment cover page.

THIS PAGE
INTENTIONALLY
LEFT BLANK

(BACK COVER)

International Council on Systems Engineering
SE Handbook Working Group