3

SYSTEM DESIGN REQUIREMENTS

This chapter addresses total system design, which can be defined as "the systematic activity necessary, beginning with the identification of the user need, to the selling and delivery of a product that will satisfy that need—an activity that encompasses product, process, people, and organization."[1] System design requirements evolve from the initial identification of a consumer/user need and are developed through the accomplishment of a feasibility analysis, the definition of system operational requirements and the maintenance concept, the development and prioritization of technical performance measures (TPMs), the accomplishment of a functional analysis, and the top-down allocation of requirements to the depth necessary. Given these basic requirements, the design process encompasses the activities of synthesis, analysis, and design optimization through the accomplishment of trade-off studies, and ultimately leads to the detail definition of the system configuration down to the component level. This is a continuous and iterative activity with the appropriate feedback, as illustrated in Figure 1.12. With the system configuration initially defined, the next step is *assessment* or *validation* through an ongoing test and evaluation effort, and any subsequent modification and refinement as necessary.

Figure 3.1 shows all of the basic functions in the system life cycle; *design* activities are included within each of the blocks in the figure. The *design process* is continuous, commencing with the system-level configuration being developed (conceptual design), the subsystem-level next (preliminary system design), and, ultimately, the component level (detail design and development). At the same time, there are design requirements for the production/construction capability, the maintenance and support infrastructure, and the system retirement and material recycling capability (refer to Figure 1.10). Subsequent to the detailed definition of the system and all of its elements, there is an on-

[1]This definition conveys the same thought, but is slightly modified from that included in S. Pugh, *Total Design: Integrated Methods for Successful Product Engineering*, (Reading, MA: Addison-Wesley, 1990).

going data collection, analysis, and evaluation effort that is critical to ensure that the proposed design configuration will, indeed, meet the customer's requirements. If the initially specified requirements are not met, then there may be some changes required for corrective action and the system configuration will be modified accordingly.

The design process is "closed-loop" in a sense, as there are *development* activities, *evaluation and assessment* activities, and *redesign* activities as necessary. There is a general perception, adopted by many, that the design is *complete* when the system configuration is initially developed, independent of any subsequent evaluation and assessment activity. If this is the case, how does one really know whether the initially specified requirements have been met in a satisfactory manner? In any event, the design process must include all three of the aforementioned activity areas if it is to be complete.

The design process occurs as a result of a team effort, combining the necessary expertise from various organizations and technical specialties in the right place and in a timely manner. Depending on the system in question (and its mission objectives), there may be electrical requirements, mechanical requirements, structural requirements, material requirements, hydraulic requirements, reliability requirements, maintainability requirements, producibility and manufacturing requirements, environmental requirements, supportability requirements, quality requirements, and so on. These requirements will vary somewhat as one progresses through the steps in Figure 3.1. In the early stages, the design team may include only a few selected individuals with the appropriate system-level design experience. Later, additional personnel may be brought into the process as needs dictate—not too many or too few, but the proper mix of expertise at the time needed. Thus, the makeup of the design team will likely vary somewhat as the overall system development process evolves.

One of the objectives in system engineering is to assume a leadership role to ensure the proper and timely selection and integration of the required design disciplines, combined in such a manner as to enable the development of a system that will respond to the consumer/user need in a cost-effective manner. Not only will a typical program include designers representing different disciplines and with a wide variety of backgrounds and experiences, but the specific requirements for these areas of expertise will shift from one program phase to the next. Moreover, given the current emphasis relative to increased supplier participation on an international and global basis, there may a need to involve a number of suppliers from different nations as members of the design team.

Chapter 2 set the stage in defining the overall development process and the role of system engineering within its context. It is now appropriate to address some of the specifics relative to design requirements. The purpose of this chapter is to cover these requirements through the development of *specifications*, and then review some of the details as they pertain to individual design disciplines. An introduction to a select sample of disciplines is included, some commonalities are noted, and the importance of design integration through the application of system engineering methods is highlighted.[2]

[2]It should be emphasized that no attempt has been made to cover all of the disciplines that may be required in the design of a system. Only a few are identified, with the intent of highlighting those areas that are not always properly addressed as part of a typical project activity.

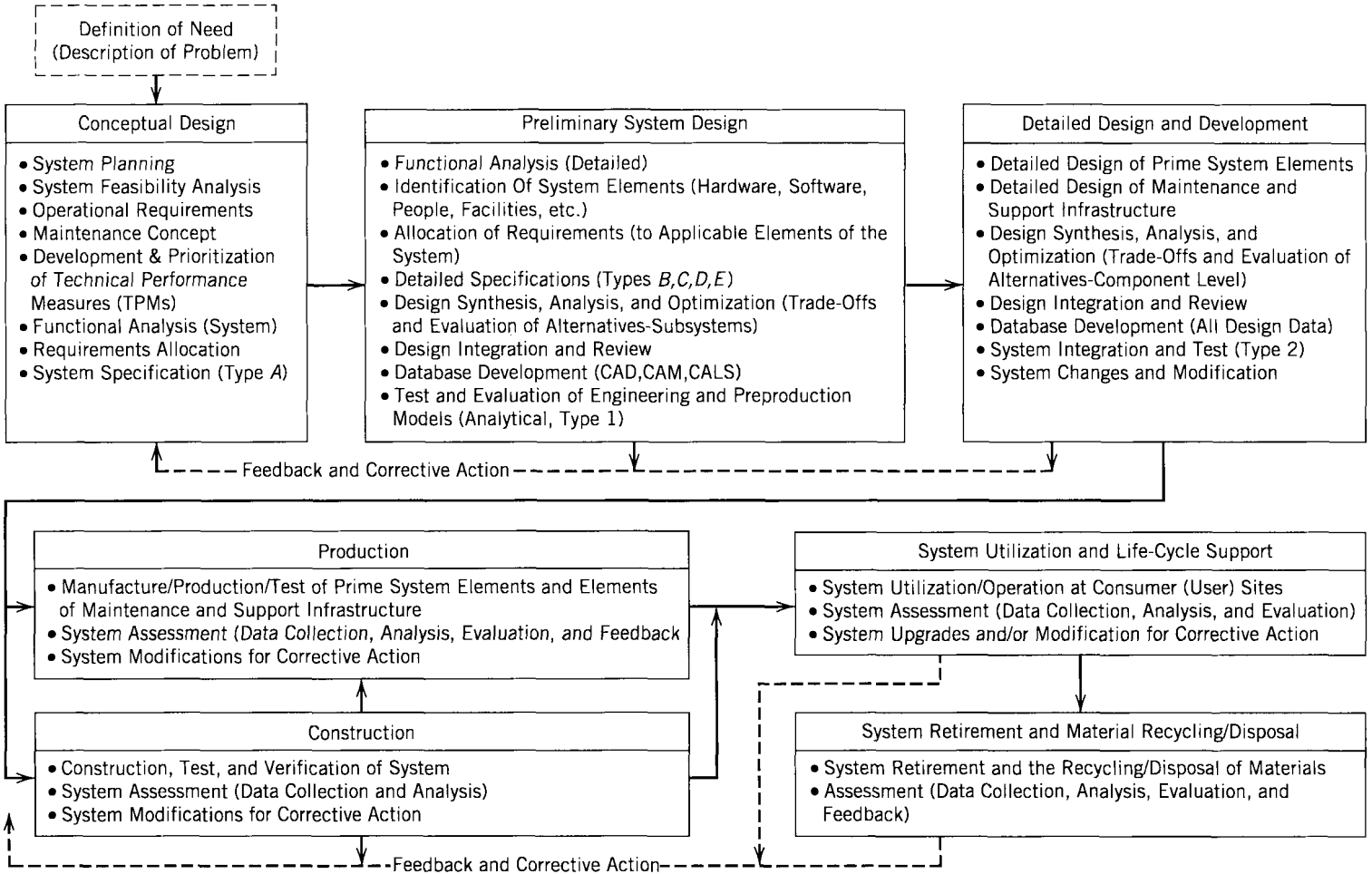

Figure 3.1 The major steps of system design and development.

3.1 DEVELOPMENT OF DESIGN REQUIREMENTS AND "DESIGN-TO" CRITERIA

As indicated in Figure 3.1, system *design requirements* evolve from the activities that occur throughout the conceptual design phase: definition of operational requirements, the maintenance concept, and the development and prioritization of the applicable technical performance measures (TPMs). As one determines these requirements for any given system, it will be necessary to define and allocate the appropriate *design-to criteria* for its different elements, as described in Chapter 2 (Section 2.8.2). As shown in Figure 2.21, the system may be partitioned and broken down into its elements, which may include various combinations of equipment, software, people, facilities, data, information, and so on. The question is, What specific quantitative and qualitative design-to criteria should be applied to the equipment, to the software, to the human element of the system, to the facilities, to the support infrastructure, to the information network, and to various combinations of these? An example of such requirements, as applied and allocated to equipment, is presented in Figure 2.23. However, the other elements of the system must be addressed as well.

Although these requirements (and design-to criteria) will vary from one system to the next, depending on the mission and the functions to be performed by the system, they may be defined through a combination of statements which, in turn, must be included in the applicable specification(s). A limited example follows:

1. The operational availability (Ao) for the system shall be greater than 98%.
2. The system software shall be designed such that a failure and its "cause" can be determined, with 99 percent accuracy, within 10 seconds from the time of its occurrence.
3. The system mean time between maintenance (MTBM) shall be 1000 hours or greater.
4. The mean maintenance downtime (MDT) for the system shall be 4 hours or less.
5. The equipment reliability mean time between failure (MTBF) shall be at least 3500 hours.
6. The mean corrective maintenance time ($\bar{M}ct$) for software shall be 30 minutes or less.
7. The response time for the logistics and maintenance support infrastructure shall not exceed 24 hours.
8. The processing of a purchase order for the acquisition of a needed system component shall not exceed 1 hour.
9. The turnaround time for maintenance shall be 24 hours or less.
10. The data access time (i.e., time to search for and acquire a desired element of data), from the management information network, shall be 15 minutes or less.
11. The human operator error rate shall not exceed 1% per month, based on the operational scenarios completed by the system in accomplishing its mission.

12. The utilization of the facility, in direct support of system operations and maintenance, shall be 85%, or greater.
13. The unit life-cycle cost for the system shall not exceed x dollars, based on a 10-year planned life cycle.
14. The production facility shall be capable of producing x products, in y time, and at a z unit cost, with a defect rate not to exceed 1%.
15. The system shall be designed such that 95% of the material that makes up the system is recycleable.
16. The process time for removing an obsolete item from the operational inventory shall not exceed 12 hours, the cost per item processed shall be x dollars or less, and the resulting environmental degradation shall be zero.

The challenge, from a system engineering perspective, is to integrate and to view these requirements in the context of the whole. Although each specific requirement may be valid when addressed individually, there may be some conflicting issues that will require adjustment when they are considered in the context of the overall system. As an aid, it may be worthwhile to identify the different requirements and how they apply to the forward and reverse flow processes shown in Figure 1.20. This, in turn, may lead to the development of some lower-level and supporting requirements. In addition, a hierarchy of requirements should be developed and applied to the various elements of the system, broken down into its components as illustrated in Figure 2.21. This process is somewhat iterative and may be accomplished in several steps. The ultimate objective is to develop such requirements for the system and its elements for inclusion in the appropriate specification (see Figure 2.24).

3.2 DEVELOPMENT OF SPECIFICATIONS

The initial definition of system requirements is projected through a combination of formal specifications and planning documentation. Specifications basically cover the *technical* requirements for system design, and planning documentation includes all *management* requirements necessary to fulfill program objectives. The combination of specifications and plans is considered the basis for all future program engineering and management decisions.

The scope and depth of such documentation depend on the nature, size, and complexity of the system. In addition, the extent to which new design is feasible (where extra guidance and controls are desired), versus the selection of an "off-the-shelf" capability, will dictate the amount of documentation necessary. For small and relatively simple items, the technical specification and program planning requirements may be included in a single document. On the other hand, for large-scale systems there may be a significant assemblage of documentation. In either case, the amount of documentation must be tailored to the need as dictated by the degree of technical and management controls necessary to accomplish program objectives.

In dealing with large systems, there are numerous elements that must be covered by specifications. Some components of the system may require an extensive amount of research-and-development effort, whereas other components are procured directly from existing supplier inventories. In regard to new items, some are developed by the major producer of the system and others are developed by suppliers remotely located in various parts of the world. In manufacturing, certain components may be produced in multiple quantities using conventional methods, whereas a special process may be required to produce other items. There may be a variety of specifications necessary to provide the guidance and controls associated with the development of the system and its components.

In preparing and applying specifications, there may be different classifications, depending on the type and nature of the item being designed or purchased off the shelf, as noted and illustrated in Figure 3.2:[3]

1. *System Specification* (Type "A"): Includes the technical, performance, operational, and support characteristics for the system as an entity. It includes the allocation of requirements to functional areas, and it defines the various functional area interfaces. The information derived from the feasibility analysis, operational requirements, maintenance concept, and functional analysis is covered (refer to Sections 2.3 to 2.8).

2. *Development Specification* (Type "B"): Includes the technical requirements for any item below the system level where research, design, and development are accomplished. This may cover an equipment item, assembly, computer program, facility, critical item of support, and so on. Each specification must include the performance, effectiveness, and support characteristics that are required in the evolving of the design from the system level and down.

3. *Product Specification* (Type "C"): Includes the technical requirements for any item below the top system level that is currently in the inventory and can be procured off the shelf. This may cover standard system components (equipment, assemblies, units, cables), a specific computer program, a spare part, a tool, and so on.

4. *Process Specification* (Type "D"): Includes the technical requirements that cover a service performed on any component of the system (e.g., machining, bending, welding, plating, heat treating, sanding, marking, packing, and processing).

5. *Material Specification* (Type "E"): Includes the technical requirements that pertain to raw materials, mixtures (e.g., paints, chemical compounds), and/or semifabricated materials (e.g., electrical cable, piping) that are used in the fabrication of a product.

The preparation of specifications is a key engineering activity. The System Specification (Type "A") is prepared during the conceptual design phase. Development and

[3]These specification classifications were originally taken from MIL-STD-490, "Specification Practices," Department of Defense, Washington, DC (latest revision).

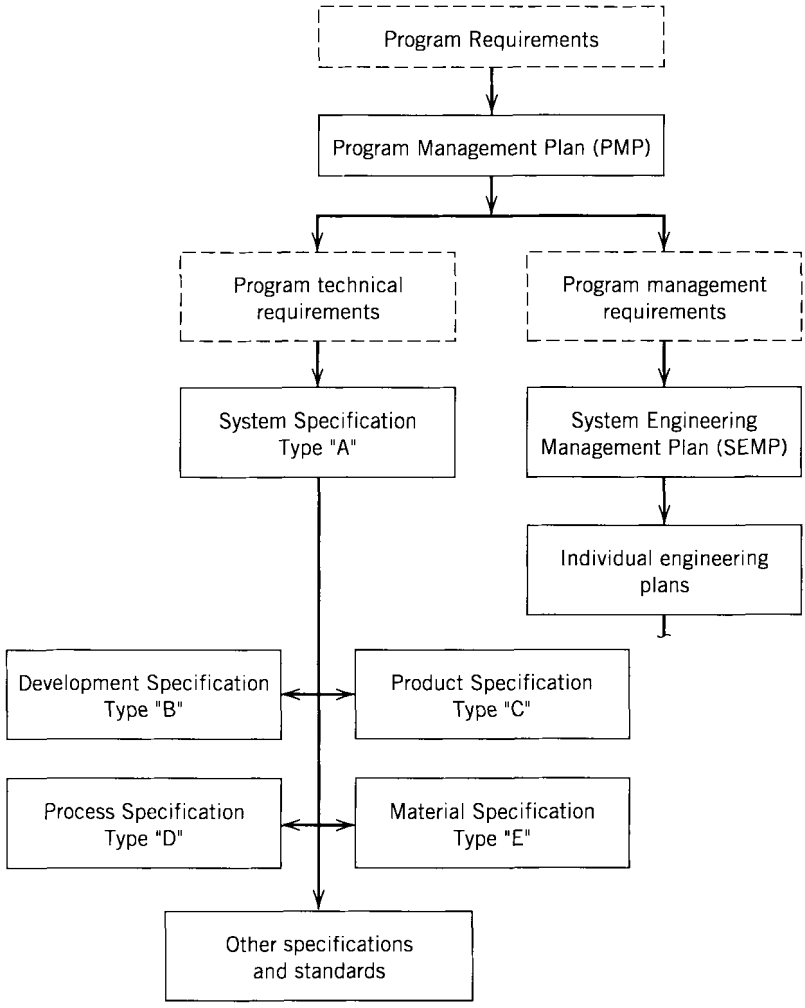

Figure 3.2 Hierarchy of technical specifications.

Product Specifications are based on the results of "make-or-buy" decisions and are generally prepared during preliminary design.[4] Process and Material Specifications are more generally oriented to production and/or construction activities, and are usually prepared during the detail design and development phase. The relative timing of these specifications, in terms of program scheduling, is illustrated in Figures 1.12 and 1.26.

[4]The results of make-or-buy decisions determine whether an item is to be designed and manufactured within the producer's facility or purchased from an outside source. Economic factors and scheduling requirements, combined with the availability of sources of supply, are prime considerations in the decision-making process. Make-or-buy (outsourcing) decisions are covered further in Chapter 6 (Section 6.3).

For large-scale systems, involving a wide mix of component suppliers, it is likely that many specifications will be generated and applied at varying stages in the system design and development process. In reviewing past experiences associated with different programs, it is clear that the generation and application of many different specifications on an independent basis has resulted in conflicts (i.e., contradictions relative to design criteria), as well as questions pertaining to which specification takes precedence in the event of conflict. In addition, there has been a tendency to specify not only the "WHATs" but the "HOWs" as well. This, in turn, can lead to a costly result. Specifications should be prepared to cover *performance* requirements, or *what is required* versus *how to do it.*

To help resolve the precedence problem, a "documentation tree" (or "specification tree") should be prepared, showing the hierarchy of specifications (and plans) from the top-level System Specification and down. In the process of requirements allocation discussed in Section 2.8, it is necessary to establish requirements at the system level first, and then allocate these requirements down to the various components of the system. In developing specifications that dictate the design requirements for the various system components, it is essential that a good comprehensive System Specification be developed first, and this specification then supplemented with the generation of good Development, Product, Process, and/or Material Specifications as required.

In Figure 2.21 (Chapter 2), a preliminary hierarchy of system components is shown as a basis for the allocation of requirements. Figure 3.3 shows a variation of this hierarchy, converted into the form of a specification tree. Basically, the System Specification is the top technical document for design. Other specifications supplement the System Specification to varying degrees. Further, an order of precedence must be established to provide guidance as to which specification governs in the event of possible conflict.[5]

With the System Specification (Type "A") being the prime document for *technical* guidance, it is appropriate that the responsibility for its preparation and implementation be assigned as a system engineering task. Care must be taken to ensure that all significant design requirements, applicable at the system level, are included. The requirements must be *integrated,* and meaningful technical performance measures (TPMs) must be identified. TPMs include those quantitative characteristics of the system that are initially specified, then reflected in the follow-on design, and later used as measures against which the system is evaluated/validated (e.g., speed, range, accuracy, size, weight, capacity, MTBM, MDT, maintenance labor hours per operating hour (MLH/OH), and cost).[6]

An example of the format for a System Specification is presented in Figure 3.4. The specification should include a description of the system (system architecture), its

[5]Throughout this section there has been reference to physical "documents" and "specifications." It should be noted that with today's EC technology, this same information may be presented in the form of a database, disk, or via some other electronic means.

[6]The system-level TPM requirements, developed through the implementation of a QFD analysis (or equivalent), as discussed in Section 2.6, should be included in the System Specification (Type "A").

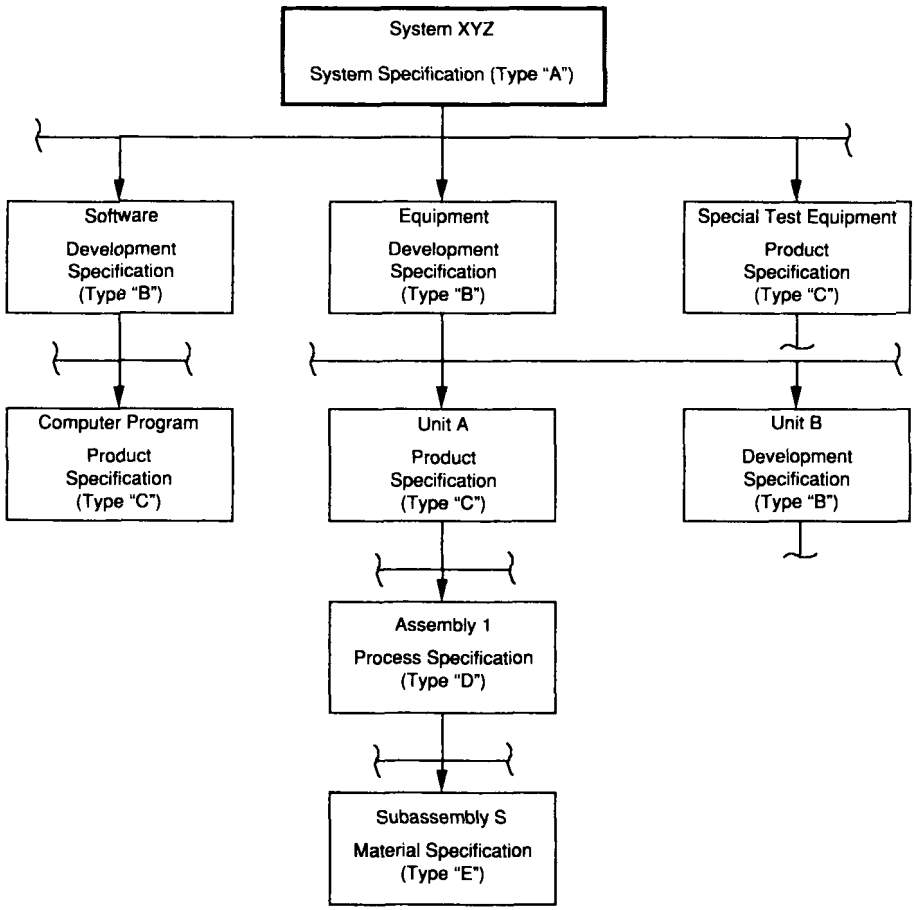

Figure 3.3 Sample specification tree (partial).

major characteristics, some general criteria for design and assembly, major data requirements, logistics and producibility considerations, test and evaluation requirements, maintenance and support requirements, quality assurance provisions, and major customer service activities. Although there may be some variations in format from one program to the next, the intent is to provide a description of the "functional baseline" for the system. This, in turn, constitutes the framework used in the preparation of subordinate specifications (e.g., *Development, Product, Process,* and *Material* Specifications) by the responsible designer(s) for the numerous components of the system, and it serves as the major technical reference for all program planning documentation.

Finally, the preparation of a good system specification is highly dependent on the abilities of those accomplishing the task relative to their thorough understanding of the system in total, its intended mission, its components and their interrelationships,

System Specification

1.0 Scope
2.0 Applicable Documents
3.0 Requirements
- 3.1 General Description of System Architecture
  - 3.1.1 System Operational Requirements
  - 3.1.2 Maintenance Concept
  - 3.1.3 Tecnical Performance Measures (TPMs) ✔
  - 3.1.4 Functional Analysis (System Level)
  - 3.1.5 Allocation of Requirements ✔
  - 3.1.6 Functional Interfaces
  - 3.1.7 Environmental Requirements ✔
- 3.2 System Characteristics
  - 3.2.1 Performance Characteristics
  - 3.2.2 Physical Characteristics
  - 3.2.3 Effectiveness Requirements
  - 3.2.4 Reliability ✔
  - 3.2.5 Maintainability
  - 3.2.6 Usability (Human Factors)
  - 3.2.7 Supportability
  - 3.2.8 Transportability/Mobility
  - 3.2.9 Producibility
  - 3.2.10 Disposability ✔
- 3.3 Design and Construction
  - 3.3.1 CAD/CAM/CALS Requirements
  - 3.3.2 Materials, Processes, and Parts
  - 3.3.3 Hardware ✔
  - 3.3.4 Software
  - 3.3.5 Electromagnetic Radiation
  - 3.3.6 Interchangeability
  - 3.3.7 Flexibility/Robustness
  - 3.3.8 Workmanship
  - 3.3.9 Safety
  - 3.3.10 Security
- 3.4 Design Data and Database Requirements
- 3.5 Logistics
  - 3.5.1 Supply Chain Requirements
  - 3.5.2 Spares, Repair Parts, and Inventory Requirements
  - 3.5.3 Test and Support Equipment
  - 3.5.4 Personnel and Training
  - 3.5.5 Packaging, Handling, Storage, and Transportation
  - 3.5.6 Facilities and Utilities
  - 3.5.7 Technical Data/Information
  - 3.5.8 Computer Resources (Software)
- 3.6 Interoperability
- 3.7 Affordability

4.0 Test and Evaluation
5.0 Maintenance and Support (Life Cycle)
6.0 Quality Assurance
7.0 Customer Service

Figure 3.4 Example of a System Specification (Type "A") format.

the various design disciplines required and their interfaces, and so on. It is not sufficient to merely prepare a series of individual write-ups covering each discipline, stapled together and submitted as a specification. This type of output usually results in contradictions, confusion, and inefficiencies throughout all subsequent phases of system design and development. Further, there are likely to be a significant number of errors and inconsistencies in the lower-level specifications if a good baseline is not established from the beginning. Without a good technical baseline, many of the design decisions made later will be suspect. Thus, the realization of a good comprehensive and highly integrated specification is critical from the beginning.[7]

3.3 THE INTEGRATION OF SYSTEM DESIGN ACTIVITIES

Based on the System Specification, there may be a variety of design-to requirements, such as those illustrated in Figure 3.5. These requirements may be mutually supportive by nature, or there may be some inherent conflicts in goals. These goals are viewed in terms of relative importance (refer to Figure 2.10), and design optimization is accomplished through trade-off studies with the objective of establishing a mutually satisfactory approach.

In response to the specification and the established goals, certain categories of engineering expertise are identified as being necessary for the design and development of the system in question. These categories, and associated levels of effort, depend on the nature and complexity of the system and the size of the project. For relatively small systems/products such as a radio, an electrical household appliance, or an automobile, the quantity and variety of engineering expertise may be limited. On the other hand, there are many large-scale systems that require the combined input of specialists representing a wide variety of engineering disciplines. Two examples of relatively large projects are noted:

1. *Commercial aircraft system:* Aeronautical engineers determine aircraft performance requirements and design the overall airframe structure. Electrical engineers design the aircraft power distribution system and ground power requirements. Electronic engineers of various types are responsible for the development of subsystems such as radar, navigation, communications, and data recording and handling. Mechanical engineers are called on in the areas of mechanical structures, linkages, pneumatics, and hydraulics. Metallurgists are needed in the selection and application of

[7]There have been a number of instances in which the System Specification was prepared in a hurry, was incomplete, and included very little in terms of specific and meaningful design criteria. The motivation for such was driven by several factors, expressed as follows: (1) "We don't want to be too specific, and be committed, in the System Specification, as we may want to introduce a lot of design changes just before we go into production," or (2) "We are already behind in schedule, we need to get going with the design, and we will worry about the System Specification later." The ultimate result: confusion, inconsistencies at the subsystem level and below, and a lot of last-minute costly design changes to correct problems that were induced in the process.

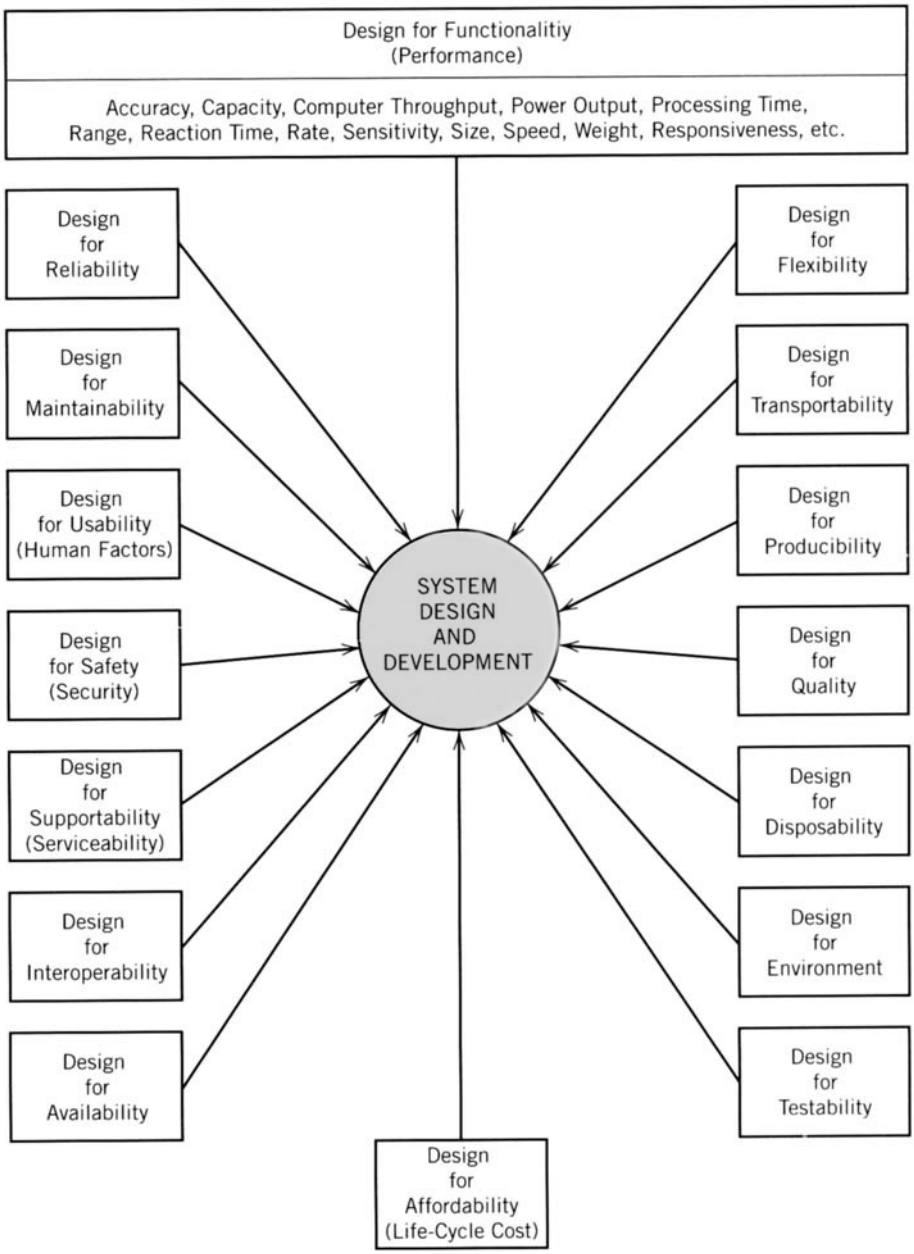

Figure 3.5 System design requirements.

materials for the aircraft structure. Reliability and maintainability engineers are concerned with availability, mean time between failure (MTBF), mean corrective maintenance time ($\overline{M}ct$), maintenance labor hours per operating hour (MLH/OH), and system logistic support. Human factors engineers are interested in the man–machine functions, cockpit and cabin layout, and design of the various operator control panels. System engineering is concerned with the overall development of the airplane as a system and ensuring the proper integration of the numerous aircraft subsystems. Industrial engineers of various types are directly involved in the production of the aircraft itself and its many components. Test engineers are required to evaluate the system to ensure conformance to consumer requirements. Other engineering specialties are employed on a task-by-task basis.

2. *Ground mass-transit system:* Civil engineers are required for the layout and/or design of railroad tracks, tunnels, bridges, cables, and facilities. Electrical engineers are involved in the design of automatic train control provisions, traction power, substations and power distribution, automatic fare collection, digital data systems, and so on. Mechanical engineers are necessary in the design of passenger vehicles and related mechanical equipment. Architectural engineers may provide support in the construction of passenger terminals. Reliability and maintainability engineers would likely be involved in the design for system availability and the incorporation of supportability characteristics. Human factors engineers are involved in the aspects of lighting, comfort ventilation, boarding access, handicapped accommodations, audio boarding instructions, and comfort stations. Industrial engineers will deal with the production aspects of passenger vehicles and vehicle components. Test engineers will evaluate the system to ensure that all performance, effectiveness, and system support requirements are met. Engineers in the planning and marketing areas will be required to keep the public informed and to promote the technical aspects of the system (i.e., keep the politicians and local citizens happy). Additional engineering specialists of various categories will be required to perform specific project-related tasks on an as-required basis.

Although these examples may not necessarily be all-inclusive, it is apparent that many different engineering disciplines are directly involved. Some of the more traditional disciplines such as electrical engineering and mechanical engineering are fractionated and broken down into specific job-oriented classifications. Engineering requirements for many large projects may include hundreds of individuals (or more, in the case of an aircraft development project or the equivalent) with varied backgrounds assigned to perform engineering functions. These engineers, forming a part of a larger organization, must not only be able to communicate with each other, but must be conversant with such supplemental activities as purchasing, accounting, manufacturing, and legal.

When considering personnel loading for large projects, there are likely to be some fluctuations. Depending on the functions to be performed, some engineers will be assigned to a given project until system development is completed, some will be assigned through the completion of production, and others will be brought in for a short

term to perform specific tasks. The requirements, in terms of needed engineering expertise, will also vary from phase to phase because the areas of emphasis change as system development progresses. During the early phases of advance planning and conceptual design, individuals with a broad systems-oriented background are needed in greater quantities than detailed design specialists, whereas the reverse may be true during the detail design and development phase. In any event, the needs for engineering will change as the system evolves through its planned life cycle.

An additional characteristic associated with large projects pertains to the split in the design engineering workload between the major system producer and the suppliers of system components. A great deal of development, evaluation, production, and support of system components is accomplished at supplier facilities located throughout the world (the percentage of supplier activity in terms of total acquisition cost often ranges up to 75%; refer to Chapter 6). In other words, the major producer who is ultimately responsible for the development, integration, production, and support of the total system as an entity is greatly dependent on the results of engineering activities being accomplished at numerous disperse locations.

The project environment for the design and development of a large number of systems today is highly "dynamic." There are many individuals with different specialties and backgrounds, rotating "on" and "off" a program at varying times. The need for good communications is essential, as well as a good understanding of the numerous interfaces that exist. The electrical design engineer needs to understand his or her interface relationships with the mechanical designer, the structural engineer, the reliability engineer, and/or the human factors specialist. The logistics engineer needs to understand the design process and the responsibilities of the electronics engineer. To acquire the necessary design integration, within the context of system engineering, requires this understanding and appreciation, along with good communications.

To enable a better understanding of design requirements overall (with the objective of further promoting the integration process), a few design disciplines have been selected for the purpose of providing additional emphasis. Initially, it is important to address the area of *software engineering,* inasmuch as a large part of the makeup of a system today involves the utilization of software. Next, it is important to address reliability, maintainability, human factors, safety, security, producibility, serviceability and supportability, logistics, disposability, environmental quality, value/cost engineering, and a few related disciplines. These areas, by themselves, certainly do not represent the total spectrum of design activity. However, in the past some of these particular requirements have not been adequately addressed or reflected in many of the systems developed, perhaps because of a lack of understanding and appreciation for these areas as they apply to design. As a result, these disciplines (and others) have been addressed independently through specifications and standards, but have not been well integrated into the mainstream design effort. Further, when addressing the individual requirements for each of the disciplines presented, one will find some commonalities. Through a review of these requirements, it is hoped that an even better appreciation of the need for total design integration will occur.

3.4 SELECTED DESIGN ENGINEERING DISCIPLINES

3.4.1 Software Engineering[8]

Software engineering deals with the *process* of bringing software into being. In the system engineering process shown in Figure 1.12, system-level requirements are defined and the accomplishment of functional analysis and allocation leads to the identification of "software" requirements (see Figure 1.13). From this point on, the emphasis may shift to the development and integration of the required software, in conjunction with the development of hardware, facilities, data/information, people requirements, and so on. It must be reemphasized that software is *not* a system in itself, but a major element of a system and must be treated on an *integrated* basis along with the other elements.

Clearly, with the advent of today's technology, software has become a major ingredient in the makeup of many systems currently being designed and developed. There are estimates that 50 to 80% of the elements of many large systems constitute software. Thus, there has been a great deal of activity in this area, which, in turn, has led to the development of the "waterfall model" (Figure 1.16), "spiral model" (Figure 1.17), "Vee model" (Figure 1.18), and others. Although the nomenclature may vary from one application to the next, the basic overall objectives are similar. The main objective has been to develop a process that will facilitate the design and development of software, which has become a complex issue and a significant challenge in the design of major systems today.

However, in spite of the availability of these recommended guidance-related models, many software developers today view "software processes" as being unduly rigid, restrictive, and inefficient and would much prefer to progress at their own speed, without any restrictions and independent of the activities associated with the development of the other elements of the system. This, in turn, has resulted in many problems, numerous program/project failures, and high costs.[9]

As a result of these and related experiences, it is believed that adherence to a good and sound "process" for the development of software (within the context of the over-

[8]The object herein is to present an overview of *software* and the acquisition of such, and to relate its importance to the system engineering process. Three good text references are (1) B. W. Boehm, *Software Engineering Economics*, (Upper Saddle River, NJ: Prentice-Hall, 1981); (2) R. S. Pressman, *Software Engineering: A Practitioner's Approach*, 4th ed. (New York: McGraw-Hill, 1996); and (3) A. P. Sage and J. D. Palmer, *Software Systems Engineering* (New York: John Wiley & Sons, Inc., 1990). In addition, it is recommended that the reader review the monthly publication *CrossTalk—The Journal of Defense Software Engineering*, published by the Software Technology Support Center (STSC), Hill AFB, Ogden, UT. See Appendix A for additional references.

[9]Two good references in this area are (1) "Five Ways to Destroy a Development Project—By Not Adhering to Basic Software-Engineering and Management Principles, Any Project Can Run Aground," by David R. Lindstrom, *IEEE Software: Lessons Learned,* IEEE Computer Society (September 1993). The five ways include "inadequate system engineering, improper requirements management, improper hardware sizing, selection of inappropriate methodologies, and ineffective metrics program"; and (2) "Seven Characteristics of Dysfunctional Software Projects," by M. W. Evans, A. M. Abela, and T. Beltz, *CrossTalk—The Journal of Defense Software Engineering* Vol. 15, no. 4, Software Technology Support Center (STSC), Hill AFB, Ogden, UT, April 2002. One of the seven causes is "failure to implement effective software processes."

all system engineering process) is essential and must be implemented. As a start, it may be worthwhile to refer to the seven steps identified in the section entitled "The Power of Process" in S. McConnell's book, *Software Project Survival Guide:*[10]

1. Committing all requirements to writing;
2. Using a systematic procedure to control additions and changes to the software requirements;
3. Conducting systematic technical reviews of all requirements, designs, and source codes;
4. Developing a systematic Quality Assurance Plan in the very early stages of the project that includes a test plan, review plan, and defect tracking plan;
5. Creating an implementation plan that defines the order in which the software's functional components will be developed and integrated;
6. Using automated source code control; and
7. Revising cost and schedule estimates as each major milestone is achieved. Milestones include the completion of requirements analysis, architecture, and detailed design as well as the completion of each implementation stage.

The critical issue pertains to the adherence of a *disciplined* approach in the development of software.

As a basis for further discussion, Figure 3.6 (which is a simplification of Figure 1.13) is included to show the major hardware–software steps and interfaces. The software development cycle may be described as follows:

1. *Software planning:* A description of software requirements as defined through the system engineering process, problem statement, project vision, definition of project scope, sponsorship, proposed targets (schedules), development strategy and acquisition process, organizational approach and personnel strategies, procurement and purchasing philosophy, supplier requirements and interfaces, cost and budgetary requirements, configuration control, measures and measurement control, areas of risk and risk management, and related management issues, should be included in the *software plan.*

The software plan should be prepared early in the life cycle and at the time when software requirements are first identified, and the plan must be closely integrated with the System Engineering Management Plan (SEMP).[11]

2. *Requirements analysis:* A definition of system requirements (i.e., operational requirements, maintenance concept, technical performance measures, functional analysis and allocation), an identification of software operational and maintenance functions, top-level software functional block diagrams, performance factors, speci-

[10]S. McConnell, *Software Project Survival Guide* (Redman, WA: Microsoft Press, 1998), 20–23.

[11]J. Cooper, et al., *Software Acquisition Capability Maturity Mode (SA-CMM), Version 1.02 (CMU/SEI-99-TR-002, ADA 362667).* (Pittsburgh, PA: Software Engineering Institute (SEI), Carnegie Mellon University, 1999). This model was developed for the purpose of aiding in the acquisition of software.

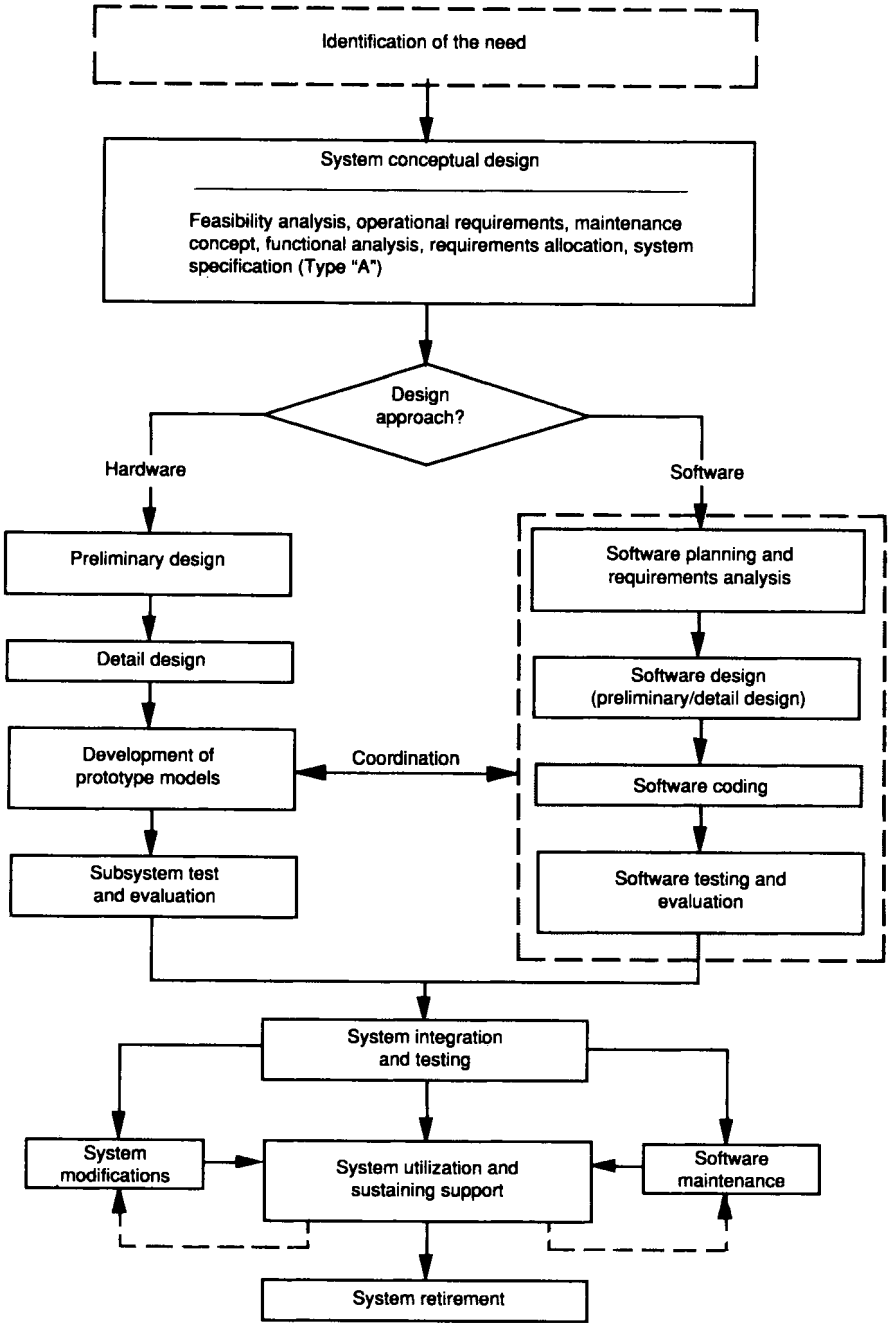

Figure 3.6 Software-hardware steps and interfaces.

fic software design criteria/constraints, and so forth, are included. There must be a top-down "traceability" of requirements, and the specific software requirements must be traceable back to one or more system-level functions.[12] Specific design requirements for software should address such attributes (expressed in quantitative terms where possible) as availability, efficiency, flexibility, integrity, interoperability, reliability, maintainability, portability, usability and reusability, testability, and robustness.[13]

3. *Software design:* The development of a software hierarchical structure is accomplished in preliminary design to establish the basic relationships between the functional elements of the software, software modules, and coupling interface requirements. Information/data flow diagrams are developed further, and database requirements are defined. In detail design, the functional flow diagrams are broken down into detailed flowcharts, and the requirements for program design language(s) and application of the appropriate design tools are identified. The key issue here is to thoroughly document (i.e., establish a baseline for) the design, and the evolvement from one configuration to the next should changes be incorporated.[14]

4. *Software coding:* The preparation and writing of programs using structured code, the appropriate code format and code documentation, debugging and error-checking procedures, and so on, are accomplished. When incorporating different software packages into the system design, one needs to ensure code compatibility across the board.

5. *Software test and evaluation:* A "verification" of software design is accomplished to ensure that each of the various individual software products fulfills its specific purpose. The verification process often constitutes a step-by-step approach to testing one program, followed by the testing of another program, given the results of the first, and so on. The verification process is initially iterative, using a *rapid prototyping* approach throughout the preliminary and detail design phases. Later, the software may be verified as part of the overall system test and evaluation process.[15]

6. *Software maintenance:* The maintenance of software may be broken down into two basic categories. The first has to do with *corrective* maintenance and the fixing

[12]It should be noted that software development often assumes an *object-oriented* approach whereby "objects" are identified from the bottom up and software modules are developed around these objects. Although this approach may be highly beneficial in many respects, problems often occur when the software developer fails to address the aspects of *functionality* from the top down. Software modules are developed, but when combined and integrated with other elements of the system, there is often a "mismatch." Thus, although the object-oriented approach is certainly feasible, the results must be compatible with the functions defined through the process described in Section 2.7, and the various software modules should fit within some functional block (refer to Figures 2.11–2.17).

[13]K. E. Wiegers, *Software Requirements: Practical Techniques for Gathering and Managing Requirements Throughout the Product Development Cycle* (Redmond, WA: Microsoft Press, 1999) pp. 196–198. This is an excellent text for coverage of software requirements in general.

[14]Properly "documenting" the design baseline is very often an unpopular requirement for the designer, as it is perceived to take too much time and to inhibit progress. On the other hand, the absence of such documentation can be very costly, particularly in attempting to trace requirements back from the beginning, adding capability and making changes to the software, and so on.

[15]The concept of *rapid prototyping* is discussed further in Chapter 4.

of problems that have resulted from a failure (or defect). Defects may occur as a result errors in initially defining requirements, errors in design, errors in coding, errors in documentation, and/or errors resulting from bad fixes associated with earlier problems.[16] The second category includes the process of upgrading the system and the incorporation of improvements in the design, and this is often referred to as *adaptive* and/or *perfective* maintenance. Care must be taken to ensure that such maintenance does not cause the introduction of more problems in the process.

In summary, it should be emphasized that software requirements are growing in terms of system applications, and the costs of software development and maintenance continue to increase at a rapid rate. Much of this cost is due to the lack of adhering to a good disciplined "process" approach in software development and not integrating the requirements for such with the other elements of the system concurrently. Hence, it is essential that these requirements be appropriately *integrated* and *controlled* through the system engineering process.

3.4.2 Reliability Engineering[17]

Reliability, in a generic sense, can be defined as "the probability that a system or product will perform in a satisfactory manner for a given period of time when used under specified operating conditions." The *probability* factor relates to the number of times that one can expect an event to occur in a total number of trials. A probability of 95%, for example, means that (on average) a system will perform properly 95 out of 100 times, or that 95 of 100 items will perform properly.

The aspect of *satisfactory performance* relates to the system's ability to perform its mission. A combination of qualitative and quantitative factors defining the functions that a system is to accomplish, usually presented in the context of the System Specification, is included. These factors are defined under system operational requirements, described in Section 2.4.

The element of *time* is most significant because it represents the measure against which the degree of performance can be rated. A system may be designed to perform under certain conditions, but for how long? Of particular interest is the ability to predict the probability of a system surviving for a designated period of time without failure. Other time-related measures are mean time between failure (MTBF), mean time to failure (MTTF), mean cycles between failure (MCBF), and failure rate (λ).

The fourth key element in the reliability definition, *specified operating conditions,*

[16]*Software Cost Estimation in 2002,* by C. Jones, *CrossTalk—The Journal of Defense Software Engineering* 15 (6), Software Technology Support Center (STSC), Hill AFB, Ogden, UT, June 2002. According to the author, the "average number of software errors in the United States is about five per function point."

[17]The intent herein is to provide an introductory overview of reliability engineering, in terms of both definitions and program requirements, but not to cover the subject in depth. However, it is highly recommended that the subject area be pursued further. Three good references are (1) D. Kececiogly, *Reliability Engineering Handbook,* 2 vols. (Upper Saddle River, NJ: Prentice-Hall, 1991); (2) W. G. Ireson and C. F. Coombs, (eds.), *Handbook of Reliability Engineering and Management* (New York: McGraw-Hill, 1988); and (3) J. Knezevic, *Reliability, Maintainability, and Supportability* (New York: McGraw-Hill, 1993). Additional references on reliability are included in the bibliography in Appendix A.

pertains to the environment in which the system will operate. Environmental requirements are based on the anticipated mission scenarios (or profiles), and appropriate considerations for reliability must include temperature cycling, humidity, vibration and shock, sand and dust, salt spray, and so on. Such considerations must address not only the conditions when the system is operating and in a "dynamic" state, but also the conditions of the system during the accomplishment of maintenance activities, when the system (or components thereof) is being transported from one location to another, and/or when the system is in the storage mode. Experience indicates that the transportation, handling, maintenance, and storage modes are often more critical from a reliability standpoint than the environmental conditions during the periods of actual system utilization.

This definition of reliability is rather basic, and it can be applied to almost any type of system. However, there are instances in which it may be more appropriate to define reliability in terms of some specific mission scenario. In such cases, reliability can be defined as "the probability that a system will perform a designated mission in a satisfactory manner." This definition may, of course, imply the accomplishment of maintenance activities, as long as it does not interfere with the successful completion of the mission. The aspect of maintenance is covered more extensively in subsequent sections of this text.

In applying reliability requirements to a specific system, one needs to relate these requirements in terms of some quantitative measure (or a combination of several figures of merit). The basic reliability function, $R(t)$, may be stated as

$$R(t) = 1 - \mathrm{F}(t) \tag{3.1}$$

where $R(t)$ is the probability of success and $F(t)$ is the probability that the system will fail by time t. $F(t)$ represents the failure distribution function.

When dealing with failure distributions, one often assumes average failure rates and attempts to predict the expected (or average) number of failures in a given period of time. To assist in this prediction, the Poisson distribution (which is somewhat analogous to the binomial distribution) can be applied. This distribution is generally expressed as

$$P(x,t) = \frac{(\lambda t)^x e^{-\lambda t}}{x!} \tag{3.2}$$

where λ represents the average failure rate, t is the operating time, and x is the observed number of failures.

This distribution states that if an average failure rate (λ) for an item is known, then it is possible to calculate the probability, $P(x,t)$, of observing 0, 1 2, 3, . . . , n number of failures when the item is operating for a designated period of time, t. Thus, the Poisson expression may be broken down into a number of terms:

$$1 = e^{-\lambda t} + (\lambda t)e^{-\lambda t} + \frac{(\lambda t)^2 e^{-\lambda t}}{2!} + \frac{(\lambda t)^3 e^{-\lambda t}}{3!} + \ldots + \frac{(\lambda t)^n e^{-\lambda t}}{n!} \tag{3.3}$$

where $e^{-\lambda t}$ represents the probability of zero failures occurring in time t, $(\lambda t)e^{-\lambda t}$ is the probability that one (1) failure will occur, and so on.

In addressing the reliability objective, dealing with the probability of success, the first term in the Poisson expression is of significance. This term, representing the "exponential" distribution, is often assumed as the basis for specifying, predicting, and later measuring the reliability of a system.[18] In other words,

$$R = e^{-\lambda t} = e^{-t/M} \tag{3.4}$$

where M is the MTBF. If an item has a constant failure rate, the reliability of that item at its mean life is approximately 0.37, or there is a 37% chance that the item will survive its mean life without failure.

Figure 3.7 presents the traditional exponential reliability curve. The basic underlying assumption is that the failure rate is constant. When dealing with failure rates, it is necessary to view them in terms of both time and life-cycle activity. Figure 3.8 presents some typical failure-rate curve relationships. Although somewhat "puristic" in nature, the illustrations are included to support additional discussion of reliability.

In Figure 3.8, the "bathtub" curve will vary somewhat, depending on the type of equipment (whether electronic or mechanical), the degree of system/equipment maturity (new design or production versus state of the art), and so on. Usually, there is an initial "break-in" or "infant mortality" period in which a certain amount of "debugging" or "burn-in" is required in order to reach a stabilized condition. Design and/or manufacturing defects often occur, maintenance is accomplished, and corrective action is taken to resolve any outstanding problems. Subsequently, when stability is acquired, the failure rate is relatively constant until a point in time when components begin to wear out, causing an ever-increasing failure rate as time goes on.

The curves presented in Figure 3.8 may also be highly influenced by individual program activities. For example, it is not uncommon for a customer to demand that a system (or components thereof) be delivered earlier than initially scheduled. In responding, the producer may eliminate certain essential quality checks in the production process in order to "get the equipment/software out the door." This usually leads to more initial defects, the consumption of more resources for maintenance and support than initially anticipated, and a higher degree of customer dissatisfaction in the long term. In Figure 3.8, the system becomes operational at the early stages of the bathtub curve before stabilization is attained.

In the world of software, failures may be related to calendar time, processor time, the number of transactions per period of time, the number of faults per module of code, and so on. Expectations are usually based on an operational profile and criticality to the mission. Thus, an accurate description of the mission scenario(s) is

[18]It should be noted that many of the assumptions herein are based on an average, or *constant,* failure rate. Although this assumption sometimes simplifies the reliability calculation process, there are instances in which failure rates are constantly changing. In these situations, it may be more appropriate to assume a Weibull distribution (or equivalent) in lieu of the negative exponential.

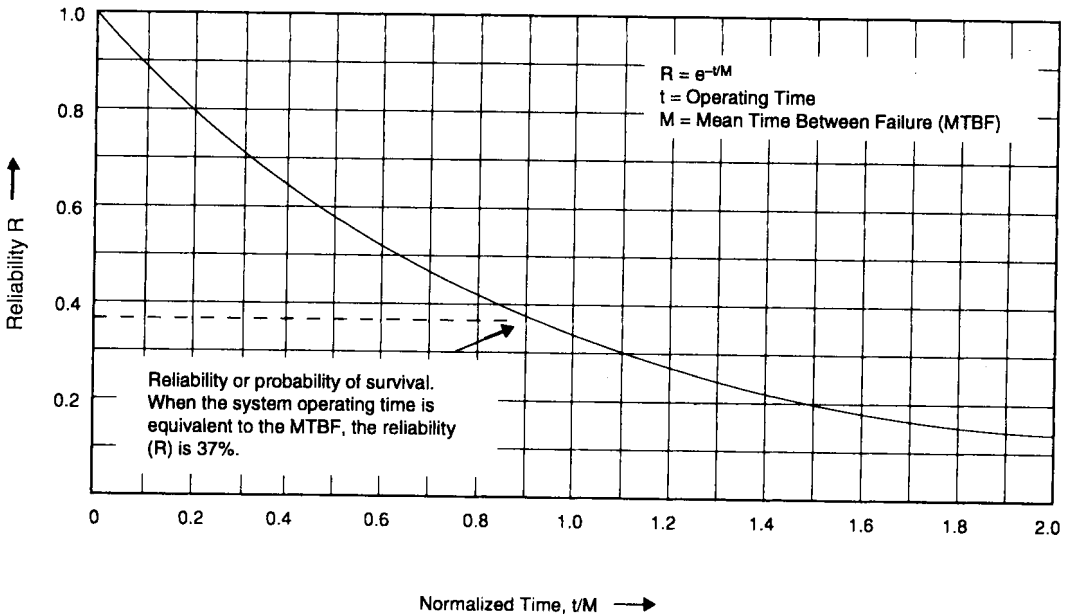

Figure 3.7 Traditional reliability exponential function.

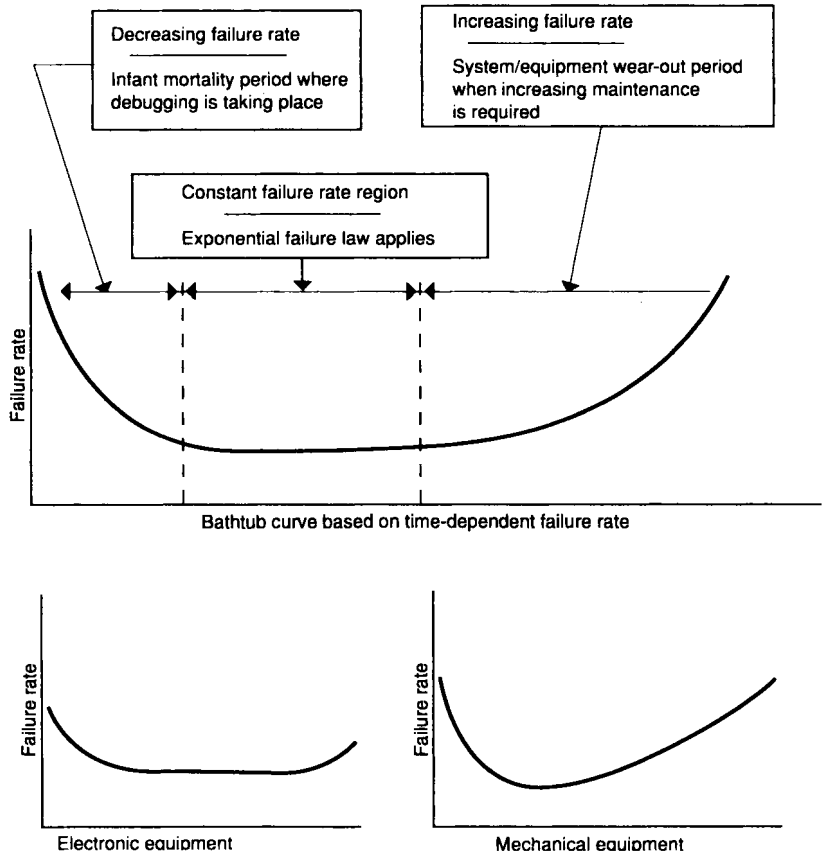

Figure 3.8 Typical failure-rate curve relationships.

required. As the system evolves from the design and development stage to the operational utilization phase, the ongoing maintenance of software often becomes a major issue. Whereas the failure rate of equipment generally assumes the profiles in Figure 3.8, the maintenance of software often has a negative effect on the overall system reliability. The performance of software maintenance on a continuing basis, along with the incorporation of system changes in general, usually impacts the overall failure rate, as shown in Figure 3.9. When a change or modification is incorporated, "bugs" are usually introduced and it takes a while for these to be worked out of the system.

As indicated in Figure 3.8 and Equations (3.2) through (3.4), the failure rate constitutes the number of failures occurring during a specified interval of time, or

$$\lambda = \frac{\text{number of failures}}{\text{total operating hours}} \tag{3.5}$$

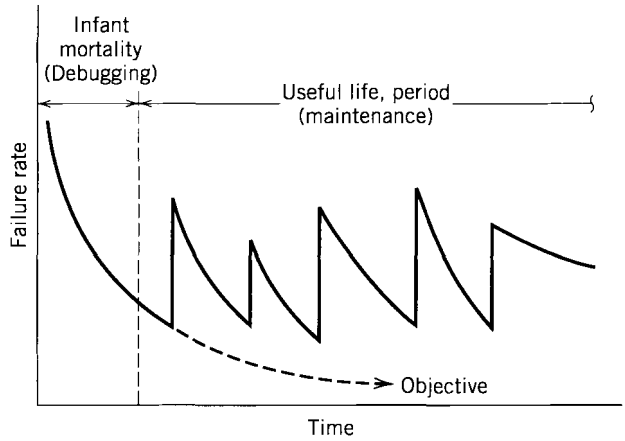

Figure 3.9 Failure-rate curve with maintenance (software application).

More specifically, the failure rate may be expressed in terms of failures per hour, failures per million hours, or percent failures per 1000 hours.[19]

In addition, when defining failures in a pure reliability sense, this refers to "primary" or "catastrophic" failures; that is, instances when the system is not operating in accordance with specification requirements because of an actual component failure stemming from an overstressed condition. A component failure may, in turn, cause other components to fail through a chain reaction of events. Thus, we need to consider both primary catastrophic failures and secondary failures, sometimes known as dependent failures.[20]

With the identification of reliability measures, it is now appropriate to show some applications. System components are functionally related to each other through a series relationship, a parallel relationship, or a combination of these. Figure 3.10 illustrates some examples.

Figure 3.10(A) shows a series network. All components must operate in a satisfactory manner if the system is to function properly. The reliability, or the probability of success, of the system is the product of the reliabilities for the individual components and is expressed as

$$R_s = (R_A)(R_B)(R_C) \tag{3.6}$$

[19]This definition applies primarily to operating equipment. Failure rates may also be expressed in terms of failures per cycle of operation, errors per page of documentation, errors per operator or maintenance task, failures per module of software, and so on.

[20]The overall frequency of unscheduled maintenance considers primary failures, secondary failures, manufacturing defects, operator-induced failures, maintenance-induced failures, defects due to handling, and so on. From a systems engineering perspective, this overall frequency factor needs to be addressed, and is discussed further in Section 3.4.3.

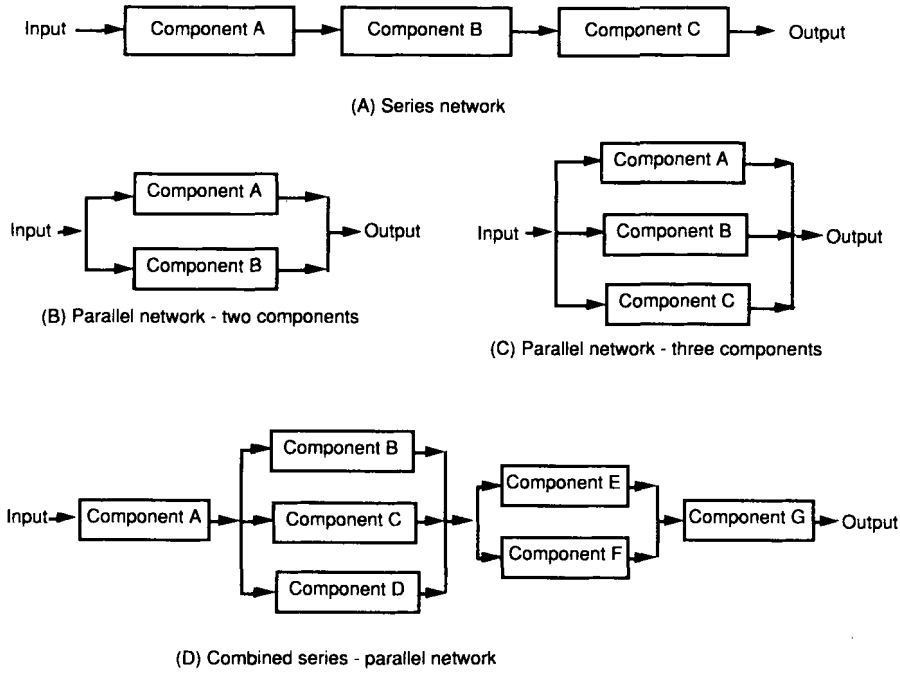

Figure 3.10 Reliability component relationships.

If the system operation is to be related to a specific time period, then, by substituting Equation (3.4) into Equation (3.6), the overall reliability of the series network is

$$R_s = e^{-(\lambda_A + \lambda_B + \lambda_C)t} \tag{3.7}$$

Figure 3.10(B) illustrates a parallel redundant network with two components. The system will function if either "A" or "B," or both, are working. The reliability expression for this network is

$$R_s = R_A + R_B - (R_A)(R_B) \tag{3.8}$$

Now consider a network with three components in parallel, as shown in Figure 3.10(C). For the system to fail, all three components must fail individually. The reliability of the network is

$$R_s = 1 - (1 - R_A)(1 - R_B)(1 - R_C) \tag{3.9}$$

In the event that all three components are identical, the reliability expression in Equation (3.9) can be simplified to

$$R_s = 1 - (1 - R)^3$$

For a system with n components, the expression becomes

$$R_s = 1 - (1 - R)^n \tag{3.10}$$

Incorporating redundancy in design helps to improve system reliability. The effects of redundancy on design, presented in a simple generic sense, is illustrated in Figure 3.11. One can also determine the degree of reliability improvement through redundancy by developing some mathematical examples using Equations (3.8) and (3.9).

Redundancy can be applied in design at different hierarchical indenture levels of the system. At the subsystem level, it may be appropriate to incorporate parallel functional capabilities, so that the system will continue to operate if one path fails to function properly. The flight control capability (incorporating electronic, digital, and mechanical alternatives) in an aircraft is an example where there are alternate paths in case of a failure in any one. At the detailed piece-part level, redundancy may be incorporated to improve the reliability of critical functions, particularly in areas where the accomplishment of maintenance is not feasible. For example, in the design of many electronic circuit boards, redundancy is often built in for the purpose of improving reliability when the accomplishment of maintenance is not practical.

The application of redundancy in design is a key area for evaluation. Although redundancy per se does improve reliability, the incorporation of extra components in a design requires additional space, and the costs are higher. This leads to a number of questions: Is redundancy really required in terms of criticality relative to system operation and the accomplishment of the mission? At what level should redundancy be incorporated? What type of redundancy should be considered (active or standby)?

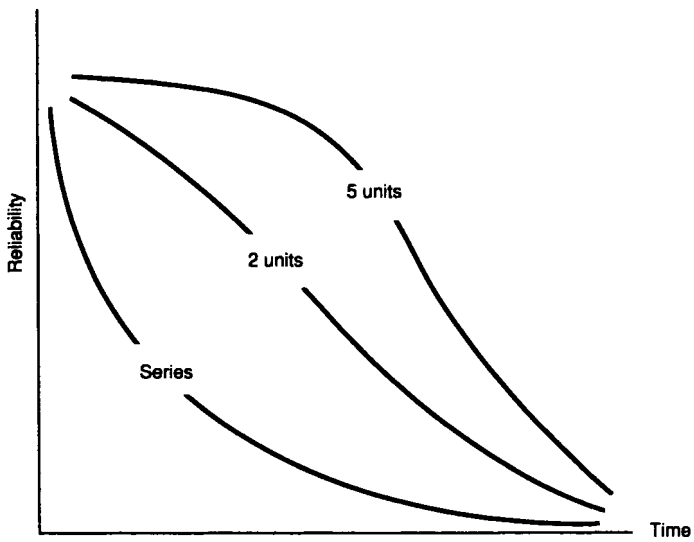

Figure 3.11 Effects of redundancy on reliability in design.

Should maintainability provisions be considered? Are there any alternative methods for improving reliability (e.g., improved part selection, part derating)? In essence, there are many interesting and related concerns that require further investigation.

Given an introduction to series and parallel networks, the next step is to combine these, as shown in Figure 3.10(D). The reliability expression for this network can be derived by applying Equations (3.6), (3.8), and (3.9). Thus,

$$R_s = (R_A)[1 - (1 - R_B)(1 - R_C)(1 - R_D)][R_E + R_F - (R_E)(R_F](R_G) \tag{3.11}$$

In evaluating combined series-parallel networks, like the one illustrated in Figure 3.10(D), the analyst should first evaluate the parallel redundant elements to obtain the unit reliability, and then combine the unit(s) with other elements of the system in a series format. Overall system reliability is determined by calculating the product of all series reliabilities.

Through various applications of series-parallel networks, a system reliability block diagram can be developed for use in reliability allocation, modeling and analyses, predictions, and so on. The reliability block diagram is derived directly from the system functional analysis described in Section 2.7 (refer to Figures 2.11 to 2.16) and is expanded downward, as illustrated in Figure 3.12. Figure 3.13 shows an expanded block diagram. The block diagram describes the system reliability in terms of various component relationships.

The material presented in this section is obviously not intended to be a comprehensive text on the subject of reliability, but enough information is included to provide the reader with some overall knowledge of key terms, definitions, and the prime objectives associated with the discipline. Basically, the subject of reliability is being presented as one of the many disciplines requiring consideration within the overall context of system engineering. A general familiarization of the subject area is necessary, as well as an understanding of some of the activities that are usually undertaken in the performance of a typical reliability program. Having covered some key terms and definitions, it is now appropriate to describe related program activities.[21]

In implementing a reliability program for a typical large-scale system, the tasks identified in Figure 3.14 are generally applicable. Although there are variations from one program to the next, the performance of these tasks in terms of overall program phasing is assumed to be in accordance with Figure 3.15. The major program phases, and system-level activities, are derived from the baseline presented in Figure 1.12 (Chapter 1).

In Figure 3.14, the reliability tasks listed can be categorized in three basic areas: (1) program planning, management, and control (Tasks 1–5), (2) design and analysis (Tasks 6–16), and (3) test and evaluation (Tasks 17–20). The first category of tasks must be closely integrated with system engineering activities and reflected in the System Engineering Management Plan (SEMP). The second group of tasks constitutes

[21]Although specific reliability tasks should be tailored to the system and the associated program needs, the tasks listed in Figure 3.14 are assumed to be typical for the purposes of discussion.

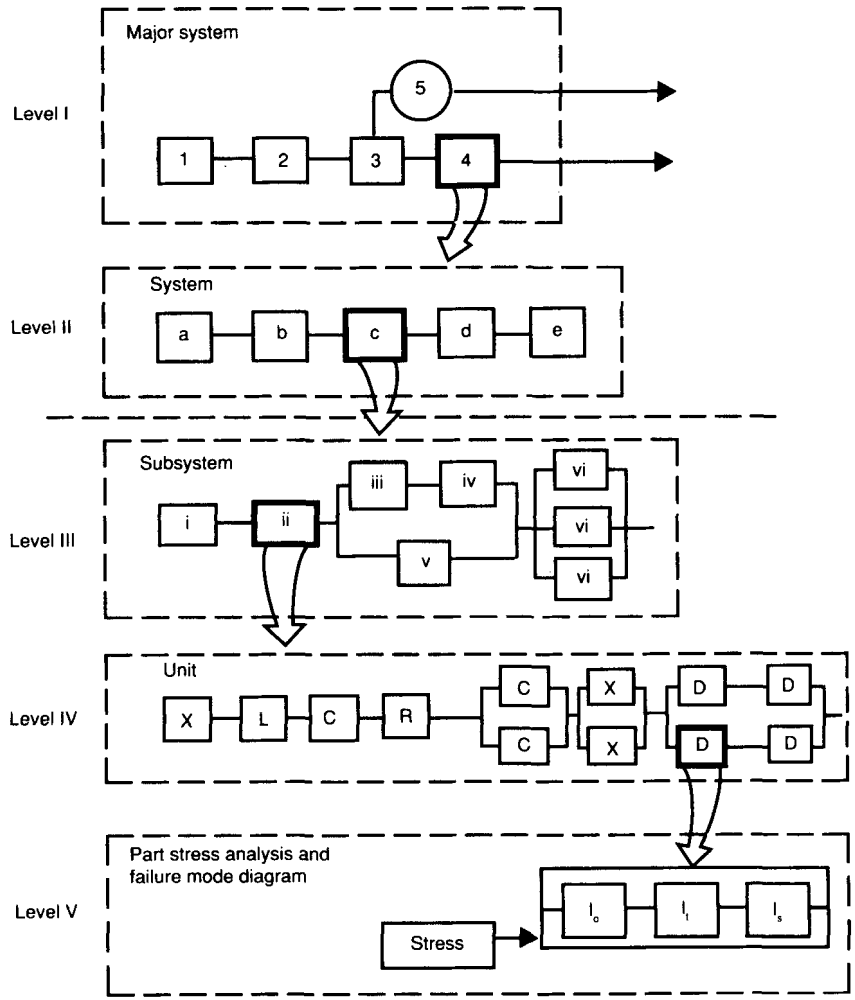

Figure 3.12 Progressive expansion of the reliability block diagram. *Source:* MIL-HDBK-338, Military Handbook, *Electronic Reliability Design Handbook* (Washington, DC: Department of Defense).

tools used in support of the mainstream design engineering effort, in response to the reliability requirements included in the System Specification and the program plan. The third group, involving reliability testing, must be integrated with system-level testing activities and covered in the Test and Evaluation Master Plan (TEMP). Although these tasks are primarily in response to reliability program requirements, there are many interfaces with basic design functions and with other supporting disciplines such as maintainability and logistic support.

Although brief task descriptions are included in Figure 3.14, some additional comments covering a select few are noted for the purposes of emphasis.

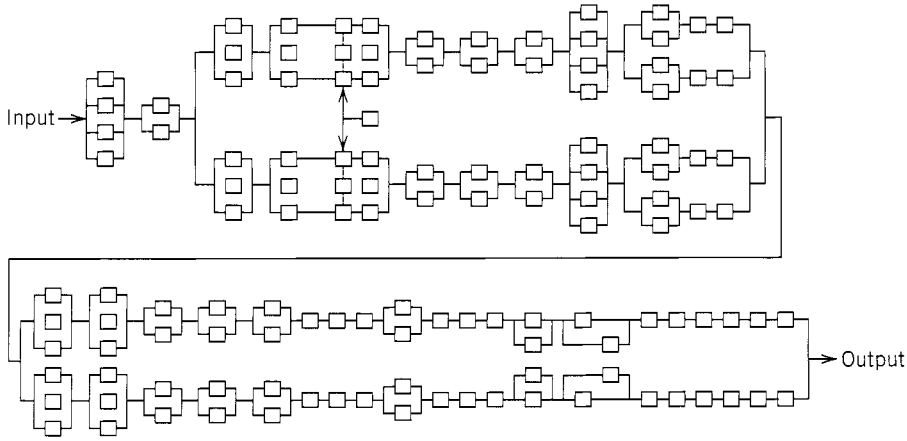

Figure 3.13 Expanded reliability block diagram of system.

1. *Reliability program plan:* Although the requirements for a reliability program may specify a separate and independent effort, it is *essential* that the program plan be developed as part of, or in conjunction with, the SEMP. Organizational interfaces, task inputs-outputs, schedules, and so on, must be directly supportive of system engineering activities. In addition, reliability activities must be closely integrated with maintainability and logistic support functions and must be included in the respective plans for these program areas (which also should be tied directly into the SEMP). The SEMP is introduced in Section 1.4 (refer to Figure 1.26) and is described further in Chapter 6.

2. *Reliability modeling:* This task, along with several others (e.g., allocation, prediction, stress/strength analysis, tolerance analysis), depends on the development of a good reliability block diagram (refer to Figure 3.13). The block diagram should evolve directly from, and support, the system functional analysis and associated functional flow diagrams (refer to Section 2.7). Further, the reliability block diagram is used for analyses and predictions, the results of which are provided as a major input to maintainability, human factors, logistics, and safety analyses. The reliability block diagram represents a major link in a long series of events and must be developed in conjunction with these other activities.

3. *Failure mode, effect, and criticality analysis* (FMECA): The FMECA is a tool that has many different applications. Not only is it an excellent design tool for determining cause-and-effect relationships and identifying weak links, but it is useful in maintainability for the development of diagnostic routines. It is also required in the accomplishment of supportability analysis (SA) relative to the identification of both corrective and preventive maintenance requirements. The FMECA constitutes a major input to the reliability-centered maintenance (RCM) program. It is used to supplement both the fault-tree analysis and the hazard analysis accomplished in a system safety program. The FMECA is a critical activity, must be accomplished in a timely manner (early during preliminary design and subsequently updated on an iterative basis), and must be directly tied into these other activities. Figure 3.16 points to the application of the FMECA to a package handling system, and Case Study B.1, Appendix B, describes the FMECA process.

Program Task	Task Description and Application
1. Reliability Program Plan	To develop a reliability program that identifies, integrates, and assists in the implementation of all management tasks applicable in fulfilling reliability program requirements. This plan includes a description of the reliability organization, organizational interfaces, a listing of tasks, task schedules and milestones, applicable policies and procedures, and projected resource requirements. This plan must tie directly into the System Engineering Management Plan (SEMP).
2. Review and control of suppliers of subcontractors	To establish initial reliability requirements and to accomplish the necessary program review, evaluation, feedback, and control of component supplier/subcontractor program activities. Supplier program plans are developed in response to the requirements of the overall Reliability Program Plan for the system.
3. Reliability program reviews	To conduct periodic program and design reviews at designated milestones; e.g., conceptual design review, system design reviews, equipment/software design reviews, and critical design review. The objective is to ensure that reliability requirements will be achieved.
4. Failure reporting, analysis, and corrective-action system (FRACAS)	To establish a closed-loop failure reporting system, procedures for analysis and for determining the cause(s) of failures, and documentation for recording the corrective action initiated.
5. Failure review board (FRB)	To establish a formal review board to review significant or critical failures, failure trends, corrective-action status, and to assure that adequate actions are being taken in a timely manner to resolve any outstanding problems.
6. Reliability modeling	To develop a reliability model for making initial numerical allocations, and for subsequent estimates to evaluate system/component reliability. As design progresses, a reliability block diagram is developed and used as a basis for accomplishing periodic reliability predictions. The reliability block diagram should evolve directly from the system functional flow block diagram.
7. Reliability allocation	To allocate, or apportion, top system-level requirements to lower idenfure levels of the system (e.g., subsystem, unit, assembly). This is accoplished to the depth necessary to provide specific criteria as an input to design.
8. Reliability prediction	To estimate the reliability of a system (or components thereof) based on a given design configuration. This is accomplished periodically throughout the system design and development processs to determine whether the initially specified system requiremtns are likely to be met given the proposed design at that time.
9. Failure mode, effect, and criticality analysis (FMECA)	To identify potential design weaknesses through a systematic analysis approach considering all possible ways in which a component can fail (the modes of failure), the possible causes for each failure, the likely frequency of occurrence, the criticality of failure, the effects of each failure on system operation (and on various system components), and any corrective action that should be initiated to prevent (or reduce the probability of) the potential problem from occuring in the future. Refer to Case Study B.1, Appendix B.
10. Fault-tree analysis (FTA)	To determine system design weaknesses using a deductive approach involving the graphical enumeration and analysis of different ways in which a system failure can occur. Refer to Case Study B.2, Appendix B.
11. Reliability-centered maintenance (RCM)	To identify alternatives and determine the best overall program for preventive maintenance using life-cycle criteria. Refer to Case Study B.3, Appendix B.
12. Sneak circuit analysis (SCA)	To identify possible latent paths that could cause the occurance of unwanted functions, assuming that all components are functioning properly in the beginning.
13. Electronic parts/circuits tolerance analysis	To examine the effects of parts/circuits electrical tolerances, specified over a range of operations (performance, temperature, etc.), on system reliability. The objective is to asses part drift characteristics, possible tolerance buildup, and identify design weaknesses.
14. Parts program	To establish a procedure for controlling the selection and use of standard and nonstandard parts.
15. Reliability critical items	To identify components requiring "special attention" because of their complexity, their relatively short life, and/or their use in new state-of-the-art technology application. Critical items usually require special maintenance/logistic support provisions.
16. Effects of testing, storage, handling, packaging, transportation, and maintenance	To determine the effects of these activities (i.e., handling, transportation, etc.) on system, or component, reliability.
17. Environmental stress screening	To plan and implement a program where the system (or components thereof) is tested using various environmental stresses; e.g., thermal or temperature cycling, vibration and shock, burn-in, X-ray, etc. The objective is to stimulate potential relevant failures early in the life cycle.
18. Reliability development/ growth testing	To plan and implement a 'test-analyze-and-fix' procedure whereby system/component weaknesses can be identified, modifications can be incorporated, and reliability growth can be realized as the system development process evolves. This is an iterative activity, and involves performance testing, environmental testing, accelerated testing, and so on.
19. Reliability qualification test	To plan and implement a program where sequential testing is accomplished, using a preproduction prototype and considering statistical "accept" and "reject" criteria, to measure the reliability MTBF of the system. This occurs prior to entering production.
20. Production reliability acceptance test	To plan and implement a program where testing is accoplished, on a sampling basis, throughout the production process to ensure that degradation has not occured as a result of that process

Figure 3.14 Reliability engineering program tasks.

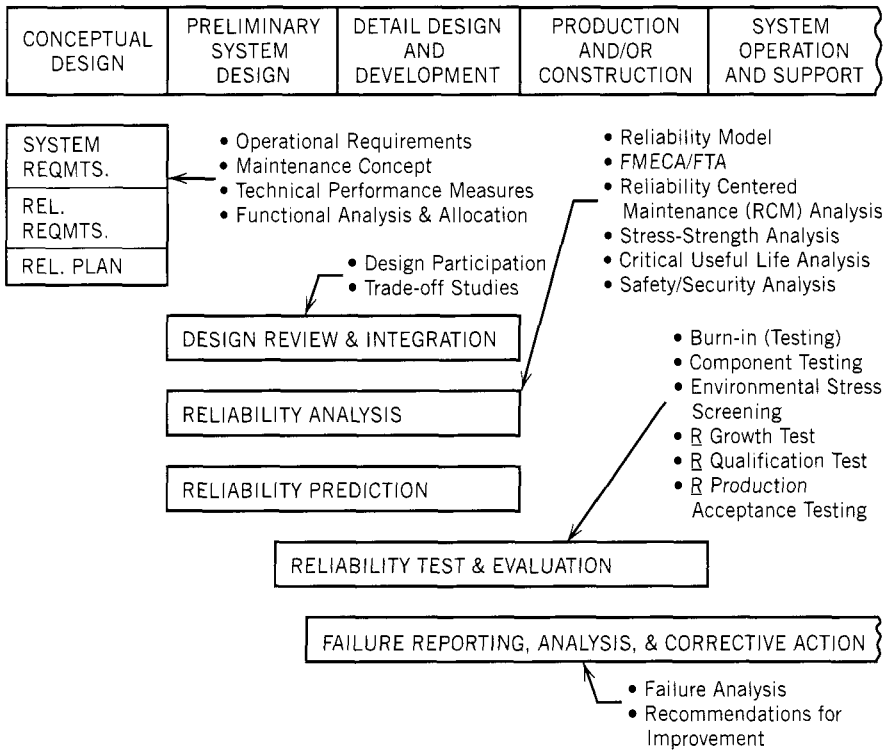

Figure 3.15 Reliability tasks in the system life cycle.

4. *Fault-tree analysis* (FTA): The FTA is a deductive approach involving the graphical enumeration and analysis of different ways in which a particular system failure can occur and the probability of its occurrence. A separate fault tree may be developed for every critical failure mode or undesired top-level event. Attention is focused on this top-level event and the first-tier causes associated with it. Each of these causes is next investigated for its causes, and so on. The FTA is narrower in focus than the FMECA and does not require as much input data. Case Study B.2, Appendix B, describes the FTA and its application.

5. *Reliability-centered maintenance* (RCM) analysis: The RCM analysis includes an evaluation of the system/process, in terms of the life cycle, to determine the best overall program for preventive (scheduled) maintenance. Emphasis is on the establishment of a cost-effective preventive maintenance program based on reliability information derived from the FMECA (i.e., failure modes, effects, frequency, criticality, and compensation through preventive maintenance). Case Study B.3, Appendix B, describes the RCM analysis and its application.

6. *Failure reporting, analysis, and corrective-action system* (FRACAS): Although this is identified as a reliability program task designed to address recommendations for corrective action as a result of catastrophic failures, the overall task objective re-

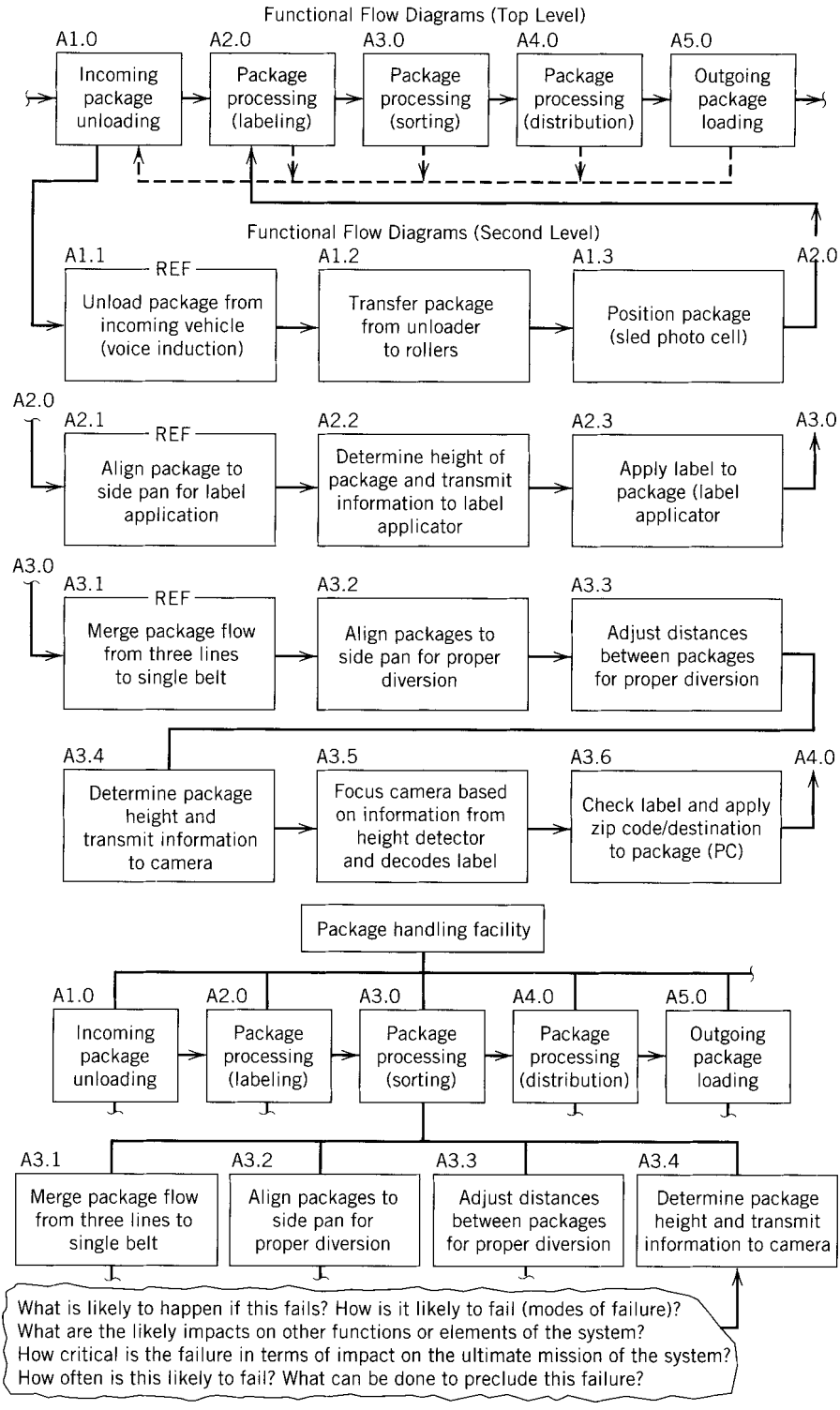

Figure 3.16 Application of FMECA to a package handling system.

lates closely to the system engineering feedback and control loop. Often, as problems arise and corrective action is initiated, the events that take place and the results are not adequately documented. Although it is important to respond to the "short-term" needs (i.e., correct outstanding problems in an expeditious manner), it is also important to provide some "long-term memory" through good reporting and documentation. This task should be tied directly with the system engineering reporting, feedback, and control process.

7. *Reliability, qualification testing:* This task, usually accomplished as part of Type 2 testing, should be defined in the context of the *total* system test and evaluation effort (refer to Figure 2.32). The specific requirements will depend on the system complexity, the degree of design definition, the nature of the mission that the system is expected to accomplish, and the TPMs (and their priorities) established for the system. In addition, for this and any other individual test, there are certain expectations and opportunities for gathering information. For instance, the objective of environmental qualification testing is to determine whether the system will perform in a specified environment. In performing this test, it may be possible to gather some reliability information by observing system operating times, failures, and so on. This, in turn, may permit a reduction in subsequent reliability testing. A second example pertains to the gathering of maintainability data during the performance of formal reliability testing. As failures occur during the test, maintenance actions can be evaluated in terms of elapsed times and resource requirements. This, in turn, may allow for some reductions in both maintainability and supportability test and evaluation efforts. In other words, there are numerous possibilities for reducing costs (while still gathering the necessary information) through the accomplishment of an integrated testing approach. Thus, reliability testing must be viewed in the context of the overall system test effort, and the requirements for this must be covered in the TEMP.

In summary, the tasks identified in Figure 3.14 are generally performed in response to some detailed specification or program requirement. For many programs, these are completed on a relatively independent basis. Yet, the interfaces are many, and there are some excellent possibilities for task integration, resulting in reduced costs. Throughout this text, opportunities for integration are discussed further. The intent of this section is to provide an introduction to the requirements associated with most reliability programs.

3.4.3 Maintainability Engineering[22]

Maintainability is an inherent characteristic of system design that pertains to the ease, accuracy, safety, and economy in the performance of maintenance actions. It

[22]The objective is to provide an introductory overview of maintainability engineering, including a few definitions and program requirements, but not to cover the subject in depth. However, for more information, two good references are (1) *Maintainability Toolkit: A Practical Guide for Designing and Developing Maintainable Products and Systems,* Reliability Analysis Center (RAC), 201 Mill Street, Rome, NY 13440, 2000; and (2) B. S. Blanchard, D. Verma, and E. L. Peterson, *Maintainability: A Key to Effective Serviceability and Maintenance Management* (New York: John Wiley & Sons Inc., 1995). Additional references are included in Appendix A.

deals with component packaging, diagnostics, part standardization, accessibility, interchangeability, mounting and labeling, and so on. A system should be designed so that it can be maintained without large investments of time and resources (e.g., personnel, materials, test equipment, facilities, data) and at minimum cost, while still fulfilling its designated mission. Maintainability is the *ability* of an item to be maintained, whereas maintenance constitutes those actions taken to restore an item to (or retain an item in) a specified operating condition. Maintainability is a design parameter, whereas maintenance is the result of design.

Maintainability, defined in the broadest sense, can be measured in terms of a combination of maintenance times, personnel labor hours, maintenance frequency factors, maintenance cost, and related logistic support factors. There is no single measure that will address *all* issues. For instance, an objective may be to shorten the elapsed time for accomplishing maintenance by adding more personnel (and possibly with greater skills). Although such an action may reduce the time requirement, it may cause an increase in personnel requirements and a resultant increase in life-cycle cost. Further, it may be desirable to reduce the frequency of unscheduled maintenance by adding the requirements for more scheduled maintenance. If this is done, there may be an increase in the overall frequency of maintenance and the life-cycle cost may increase as well. In essence, these factors (as applicable) must be addressed on a collective basis, as well as being considered in conjunction with the reliability measures discussed in Section 3.4.2.

One of the most commonly used measures of maintainability is the aspect of "time." In Figure 3.17, the overall time spectrum can be broken down into different applications. "Uptime" pertains to the elapsed time applicable to the system when in operational use, or when in a standby or ready state awaiting for use. On the other hand, "downtime" refers to the total elapsed time required, when the system is not operational, to accomplish corrective maintenance and/or preventive maintenance. These categories of maintenance are defined as follows:

1. *Corrective maintenance:* The unscheduled actions, initiated as a result of failure (or a perceived failure), that are necessary to *restore* a system to its required level of performance. Such activities may include troubleshooting, disassembly, repair, remove and replace, reassembly, alignment and adjustment, checkout, and so on. In addition, this includes all software maintenance that is not initially planned—for example, *adaptive* maintenance, *perfective* maintenance, and so on.
2. *Preventive maintenance:* The scheduled actions necessary to *retain* a system at a specified level of performance. This may include periodic inspections, servicing, calibration, condition monitoring, and/or the replacement of designated critical items.

In Figure 3.17, total maintenance downtime (MDT) is the elapsed time required to repair and restore a system to full operational status and/or to retain a system in that condition. MDT can be broken down into the following components: