12

ENGINEERING DESIGN

12.1 IMPLEMENTING THE SYSTEM BUILDING BLOCKS

The engineering design phase is that part of the development of a new system that is concerned with designing all the component parts so that they will fit together as an operating whole that meets the system operational requirements. It is an intensive and highly organized effort, focused on designing components that are reliable, maintainable, and safe under all conditions to which the system is likely to be subjected, and that are producible within established cost and schedule goals. While the general design approach required to meet the above objectives presumably has been established in previous phases, the engineering design phase is where detailed internal and external interfaces are established and the design is first fully implemented in hardware and software.

It was noted in Chapter 10 that during the advanced development phase, any previously unproven components should be further developed to the point where all significant issues regarding their functional and physical performance have been resolved. However, experience in developing complex new systems has shown that some

Systems Engineering Principles and Practice, Second Edition. Alexander Kossiakoff, William N. Sweet, Samuel J. Seymour, and Steven M. Biemer

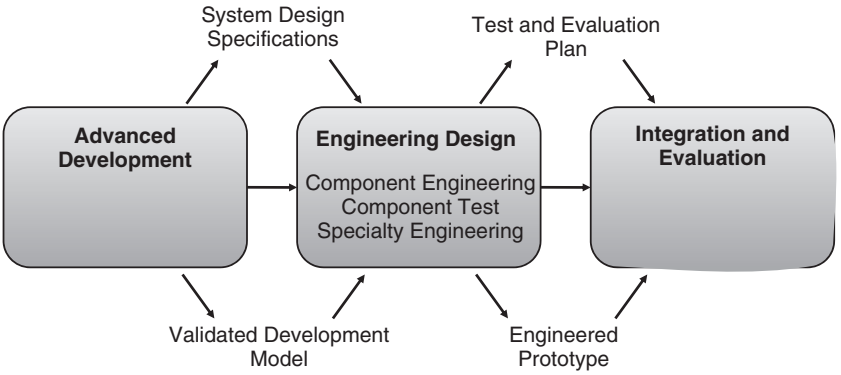

Figure 12.1. Engineering design phase in a system life cycle.

"unknown unknowns" (unk-unks) almost always escape detection until later, revealing themselves during component design and integration. Such eventualities should therefore be anticipated in contingency planning for the engineering design phase.

Place of the Engineering Design Phase in the System Life Cycle

As shown in Figure 12.1, the place of the engineering design phase in the systems engineering life cycle follows the advanced development phase and precedes the integration and evaluation phases. Its inputs from the advanced development phase are seen to be system design specifications and a validated development model of the system. Other inputs, not shown, include applicable commercial components and parts, and the design tools and test facilities that will be employed during this phase. Its outputs to the integration and evaluation phase are detailed test and evaluation plans and a complete set of fully engineered and tested components. Program management planning documents, such as the work breakdown structure (WBS) and the systems engineering management plan (SEMP), as well as the test and evaluation master plan (TEMP), or their equivalents, are utilized and updated in this process. Figure 12.2 shows that the integration and evaluation phase usually begins well before the end of engineering design to accommodate test planning, test equipment design, and related activities.

Design Materialization Status

The change in system materialization status during the engineering design phase is schematically shown in Table 12.1. It is seen that the actions "visualize," "define," and "validate" in previous phases are replaced by the more decisive terms "design," "make," and "test," representing implementation decisions rather than tentative proposals. This is characteristic of the fact that in this phase, the conceptual and developmental results of the previous phases finally come together in a unified and detailed system design.

At the beginning of the engineering design phase, the design maturity of different components is likely to vary significantly; and these variations will be reflected in dif-

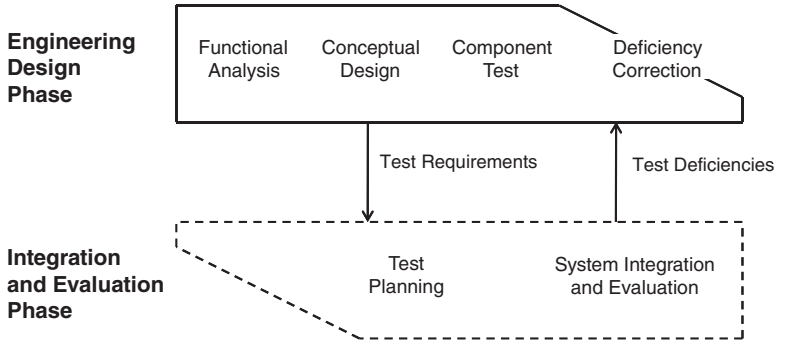

Figure 12.2. Engineering design phase in relation to integration and evaluation.

ferences in component materialization status. For example, some components that were derived from a predecessor system may have been fully engineered and tested in substantially the same configuration as that selected for the new system, while others that utilize new technology or innovative functionality may have been brought only to the stage of experimental prototypes. However, by the end of the engineering design phase, such initial variations in component engineering status must be eliminated and all components fully "materialized" in terms of detailed hardware and software design and construction.

A primary effort in this phase is the definition of the interfaces and interactions among internal components and with external entities. Experience has shown that aggressive technical leadership by systems engineering is essential for the expeditious resolution of any interface incompatibilities that are brought to light during engineering design.

Systems Engineering Method in Engineering Design

The principal activities in each of the four steps in the systems engineering method (see Chapter 4) during engineering design are briefly stated below and are illustrated in Figure 12.3. Steps 3 and 4 will constitute the bulk of the effort in this phase.

1. *Requirements Analysis.* Typical activities include
   - analyzing system design requirements for consistency and completeness and
   - identifying requirements for all external and internal interactions and interfaces.
2. *Functional Analysis and Design (Functional Definition).* Typical activities include
   - analyzing component interactions and interfaces and identifying design, integration, and test issues;
   - analyzing detailed user interaction modes; and
   - designing and prototyping user interfaces.

TABLE 12.1. Status of System Materialization at the Engineering Design Phase

Phase	*Concept development*			*Engineering development*		
Level	Needs analysis	Concept exploration	Concept definition	Advanced development	Engineering design	Integration and evaluation
System	Define system capabilities and effectiveness	Identify, explore, and synthesize concepts	Define selected concept with specifications	Validate concept		Test and evaluate
Subsystem		Define requirements and ensure feasibility	Define functional and physical architecture	Validate subsystems		Integrate and test
Component			Allocate functions to components	Define specifications	Design and test	Integrate and test
Subcomponent		Visualize		Allocate functions to subcomponents	Design	
Part					Make or buy	

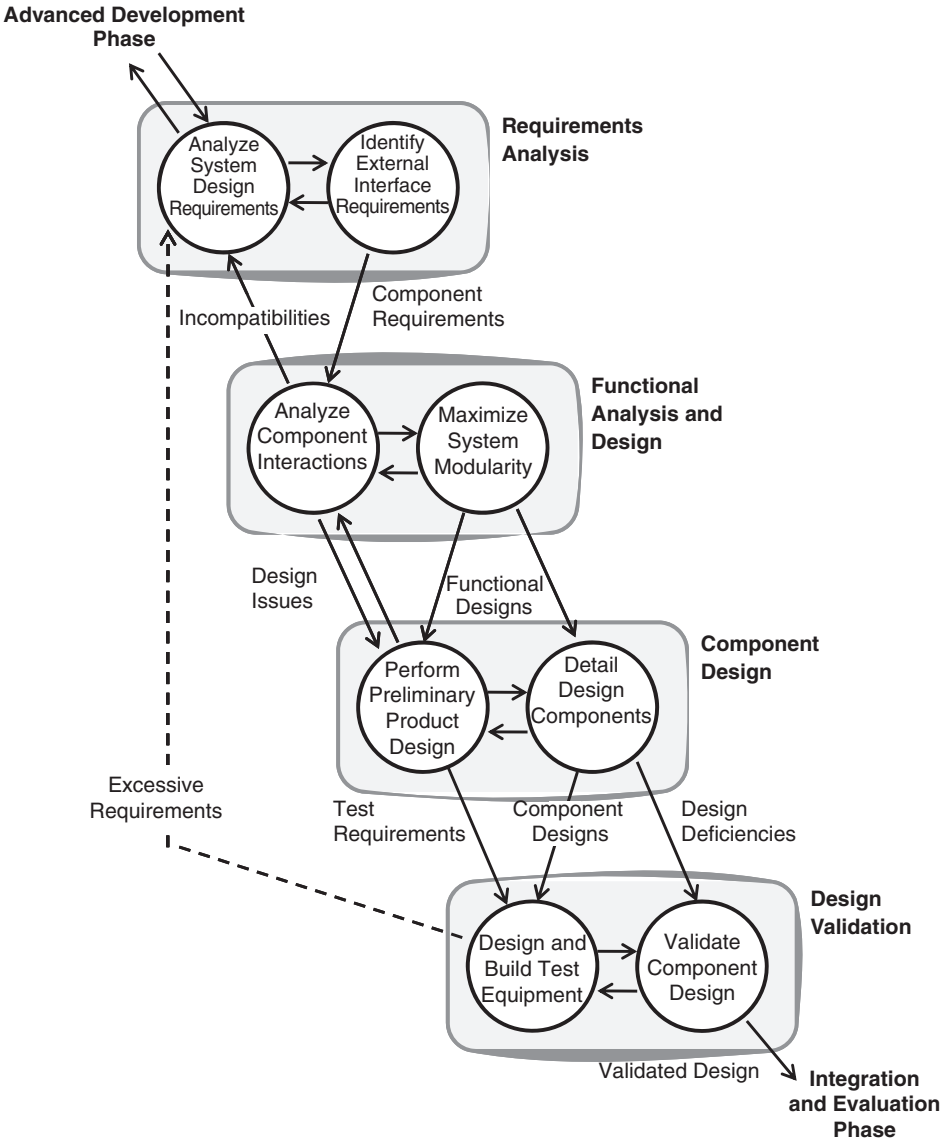

Figure 12.3. Engineering design phase flow diagram.

3. *Component Design (Physical Definition).* Typical activities include
   - laying out preliminary designs of all hardware and software components and interfaces,
   - implementing detailed hardware designs and software code after review, and
   - building prototype versions of engineered components.

4. *Design Validation.* Typical activities include
   - conducting test and evaluation of engineered components with respect to function, interfaces, reliability, and producibility;
   - correcting deficiencies; and
   - documenting product design.

12.2 REQUIREMENTS ANALYSIS

In the advanced development phase, the system functional specifications were translated into a set of system design specifications that defined the design approach selected and validated as fully addressing the system operational objectives. As in previous phases of the development process, these specifications must now be analyzed again for relevance, completeness, and consistency to constitute a sound basis for full-scale engineering. In particular, the analysis must consider any changes occurring due to the passage of time or external events.

System Design Requirements

It will be recalled that the focus of the advanced development phase was on those system components that required further maturation in terms of analysis, design, development, and/or testing to demonstrate fully their validity. These are the components that represent the greatest development risks, and hence their design approach must be carefully analyzed to ensure that the residual risks have been reduced to manageable levels. For example, components with initially ill-defined external interface descriptions must be reexamined to resolve any remaining uncertainties.

Components that were identified as involving some risk, but not to the extent of requiring special development effort, and previously proven components that will be required to perform at higher levels or in more stressful environments must be particularly scrutinized at this stage. The results of these analyses should be inputs to the planning of risk management during engineering design. (See section on risk management in Chapter 5.)

External System Interface Requirements

Since the whole system has not been physically assembled in previous phases, it is likely that the design of its interfaces with the environment has been considered less than rigorously. Hence, a comprehensive analysis of system-level environmental interfaces must be carried out prior to the initiation of engineering design.

User Interfaces. As noted previously, the functional interactions and physical interfaces of the system with the user(s) are not only often critical but also difficult to define adequately. This situation is aggravated by the fact that potential users of a new system do not really know how they can best operate it before they first physically interact with it. Thus, except for very simple human–machine interfaces, a prototype

model of the user consoles, displays, and controls should be constructed at the earliest practicable time to enable the user(s) to examine various responses to system inputs and to experiment with alternative interface designs. If this has not been done adequately in the advanced development phase, it must be done early in engineering design.

The user interfaces related to system maintenance involve fault isolation, component replacement, logistics, and a host of related issues. Interface design is often given only cursory attention prior to the engineering design phase, an omission that is likely to lead to the need for a significant redesign of previously defined component interfaces.

Environmental Interfaces. In defining external interfaces subject to shock, vibration, extreme temperatures, and other potentially damaging environments, it is essential to again consider all stages of a system's life, including production, shipment, storage, installation, operation, and maintenance, and to anticipate all of the interactions with the environment during each step. Interface elements such as seals, joints, radiation shields, insulators, shock mounts, and so on, should be reviewed and redefined if necessary to ensure their adequacy in the final design. Some of the above subjects are treated more fully in a later section of this chapter, which discusses interface design.

Assembly and Installation Requirements

In addition to the usual design requirements, the system design must also take into consideration all special requirements for system assembly and installation at the operational site. This is especially important for large systems that must be shipped in sections. An example is a shipboard system, subsystems of which are to be installed below decks in an existing ship. In this case, the size of hatches and passageways will dictate the largest object that can pass through. System installation aboard aircraft is another example. Even buildings have load and size limits on freight elevators. In any case, when on-site assembly is required, the system design must consider where the system will be "cut" and how it will be reassembled. If physical mating is implemented by bolts, for example, then the location and size of these fasteners must take into account the size and position of the wrenches needed for assembly. Many developers have been embarrassed when they realize there is not enough elbowroom to perform a prescribed assembly procedure.

Another on-site problem can occur when the assembly process is found to be difficult and slow to perform. A classic example concerns a suspended walkway that was installed in a large Midwestern hotel lobby. During a large evening dance party, a number of people were dancing on the elevated walkway, causing it to collapse with attendant loss of life. The investigation of this accident revealed that a design change had been made at the assembly site because the originally specified long, threaded supporting rods were difficult to install. A so-called trivial design change was made to permit easier assembly, but it increased the load on the rod structure by a factor of two. The fault was attributed to those involved in the design change. However, it can be argued that if the original designers had given more attention to the difficult assembly process, this problem and the resulting accident might not have occurred.

Risk Mitigation

As in the previous chapters, a necessary step in the planning of the development and engineering process is the consideration of program risk. In the advanced development phase, risk assessment was used to refer to the process of identifying components that required further maturation to eliminate or greatly reduce the potential engineering problems inherent in the application of new technology or complex functionality. By the beginning of the engineering design phase, those risks should have been resolved through further development. This in turn should have reduced the remaining program risks to a level that could be tolerated through the application of risk management, a process that identifies and seeks to mitigate (abate or minimize) the likelihood and impact of residual risks. Methods for mitigating risks are discussed briefly in the section on component design (Section 12.4) and in greater detail in Chapter 5.

Critical Design Requirements

To the extent that previous analysis has shown that a particular requirement places undue stress on the engineering design, this is the last opportunity seriously to explore the possibility of its relaxation and thus to reduce the risk of an unsuccessful design.

12.3 FUNCTIONAL ANALYSIS AND DESIGN

The principal focus of the engineering design phase is on the design of the system components. Insofar as the functional definition of the components is concerned, it may be assumed that the primary allocation of functions has been accomplished in previous phases, but that the definition of their mutual interactions has not been finalized. A primary objective of the functional analysis and design step is to definitize the interactions of components with one another and with the system environment in such a way as to maximize their mutual independence, and thus to facilitate their acquisition, integration, and maintenance and the ease of future system upgrading.

This section stresses three important areas of functional analysis and design:

1. *Modular Configuration:* simplifying interactions among system components and with the environment
2. *Software Design:* defining a modular software architecture
3. *User Interfaces:* defining and demonstrating effective human–machine interfaces.

Modular Configuration

The single most important objective of the functional analysis and design step in the engineering design phase is to define the boundaries between the components and subsystems so as to minimize their interactions (i.e., their dependence on one another). This is essential to ensure that

1. each component can be specified, developed, designed, manufactured, and tested as a self-contained unit;
2. when assembled with the other components, a component will perform its functions properly and without further adjustment;
3. a faulty component can be replaced directly by an equivalent interchangeable component; and
4. a component can be upgraded internally without affecting the design of other components.

A system design with the above characteristics is referred to as "modular" or "sectionalized." These characteristics apply to both hardware and software components. They depend on physical as well as functional interactions, but the latter are fundamental and must be defined before the physical interfaces can be established.

Functional Elements. The system functional elements defined in Chapter 3 are examples of highly modular system building blocks. These building blocks were selected using three criteria:

1. *Significance:* each functional element performs a distinct and significant function, typically involving several elementary functions.
2. *Singularity:* each functional element falls largely within the technical scope of a single engineering discipline.
3. *Commonality:* the function performed by each element is found in a wide variety of system types.

Each of the functional elements was seen to be the functional embodiment of a type of component element (see Table 3.3), which is a commonly occurring building block of modern systems. Their characteristic of "commonality" results from the fact that each is highly modular in function and construction. It follows that the functional elements of a new system should use standard building blocks whenever practicable.

Software Design

As noted previously, the development and engineering of software components are sufficiently different from that of hardware components that a separate chapter is devoted to the special systems engineering problems and solutions of software (Chapter 11). The paragraphs below contain a few selected subjects relevant to this chapter.

Prototype Software. The previous chapters noted that the extensive use of software throughout most modern complex systems usually makes it necessary to design and test many software components in prototype form during the advanced development phase. Common instances of this are found in embedded real-time programs and user interfaces. The existence of such prototype software at the beginning

of the engineering design phase presents the problem of whether or not to reuse it in the engineered system and, if so, just how it should be adapted for this purpose.

Redoing the prototype software from scratch can be extremely costly. However, its reuse requires careful assessment and revision, where necessary. The following conditions are necessary for successful reuse:

1. The prototype software must be of high quality, that is, designed and built to the same standards that are established for the engineered version (except perhaps for the degree of formal reviews and documentation).
2. Changes in requirements must be limited.
3. The software should be either functionally complete or compatible with directly related software.

Given the above conditions, modern computer-aided software engineering (CASE) tools are available to facilitate the necessary analysis, modification, and documentation to integrate the prototype software into the engineered system.

Software Methodologies. Chapter 11 identifies many of the key aspects of software engineering that are of direct interest to systems engineers. Two principal methodologies are used in software analysis and design: *structured analysis* and *design and object-oriented analyses and design*. The former and more mature methodology is organized around functional units generally called procedures or functions and is assembled in modules or packages. In good structured design, data values are passed between procedures by means of calling parameters, with a minimum of externally addressed (global) data. Object-oriented analysis (OOA) and object-oriented design (OOD) are more recent methods of software system development and are widely believed to be inherently superior in managing complexity, which is a critical problem in all large, information-rich systems. Using object-oriented methods in developing hardware and combined hardware/software systems has become more commonplace and is usually referred to as object-oriented systems engineering (OOSE). The particulars of this method will be described below in a separate section. Accordingly, today's systems engineers need to know the basic elements and capabilities of these methodologies in order to evaluate their appropriate place in system development.

User Interface Design

Among the most critical elements in complex systems are those concerned with the control of the system by the user—analogous to the steering wheel, accelerator, shift lever, and brakes in an automobile. In system terminology, those elements are collectively referred to as the "user interface." Their criticality is due to the essential role they play in the effective operation of most systems, and to the inherent problem of matching a specific system design to the widely variable characteristics of the many different human operators who will use the system during its lifetime. If several individuals operate different parts of the system simultaneously, their mutual interactions present additional design issues.

The principal elements involved in user control include

1. *Displays:* presentations provided to the user containing information on system status to indicate need for possible user action. They may be dials, words, numbers, or graphics appearing on a display screen, or a printout, sound, or other signals.
2. *User Reaction:* user's interpretation of the display based on knowledge about system operation and control, and consequent decisions on the action to be taken.
3. *User Command:* user's action to cause the system to change its state or behavior to that desired. It may be movement of a control lever, selection of an item from a displayed menu, a typed command, or another form of signal to which the system is designed to respond.
4. *Command Actuator:* device designed to translate the user's action into a system response. This may be a direct mechanical or electrical link or, in automated systems, a computer that interprets the user command and activates the appropriate response devices.

The design of a user–system interface is truly a multidisciplinary problem, as the above list implies, and hence is the domain of systems engineering. Even human factors engineering, considered to be a discipline in itself, is actually fragmented into specialties in terms of its sensory and cognitive aspects. While much research has been carried out, quantitative data on which to base engineering design are sparse. Thus, each new system presents problems peculiar to itself and often requires experimentation to define its interface requirements.

The increasing use of computer automation in modern systems has brought with it the computer-driven display and controls as the preferred user interface medium. A computer interface has the facility to display information in a form processed to give the user a clearer and better organized picture of the system status, to simplify the decision process, and to offer more simple and easier modes of control.

Chapter 11 contains a brief description of computer control modes, graphical user interfaces (GUIs), and advanced modes of user–computer interactions, such as voice control and visual reality.

Functional System Design Diagrams. As the components of a complex system are integrated, it becomes increasingly important to establish system-wide representations of the functional system architecture to ensure its understanding by all those concerned with designing the interacting system elements. Functional diagrams are discussed in more detail in Chapter 8.

12.4 COMPONENT DESIGN

The object of the component design step of the engineering design phase is to implement the functional designs of system elements as engineered hardware and

software components with compatible and testable interfaces. During this phase, the system components that do not already exist as engineered items are designed, built, and tested as units, to be integrated into subsystems and then assembled into an engineering prototype in the integration and evaluation phase. The associated engineering effort during this phase is more intense than at any other time during the system life cycle. During the design of any complex new system, unexpected problems inevitably occur; their timely resolution depends on quick and decisive action. This high level of activity, and the potential impact of any unforeseen problems on the successful conduct of the program, tends to place a severe stress on systems engineering during this period.

In the development of major defense and space systems, the engineering design effort is performed in two steps: designated preliminary design and detailed design, respectively. Although preliminary design is typically started under systems architecting, many official programs continue to establish a subphase where the initial architecture is translated into a preliminary design. Each step is followed by a formal design review by the customer before the succeeding step is authorized. The purpose of this highly controlled process is to ensure very thorough preparation prior to commitment to the costly full-scale implementation of the design into hardware and software. This general methodology, without some of its formality, may be applied to any system development.

The level of system subdivision on which the above design process is focused is called a "configuration item" (CI). This level corresponds most closely to that referred to here as "component." It should be noted that in common engineering parlance, the term component is used much more loosely than in this book and sometimes is applied to lower-level system elements, which are identified here as subcomponents. CIs and "configuration baselines" are discussed further in the section on configuration management (CM) (Section 9.6).

Preliminary Design

The objective of preliminary design is to demonstrate that the chosen system design conforms to system performance and design specifications and can be produced by existing methods within established constraints of cost and schedule. It thereafter provides a framework for the next step, detailed design. The bulk of the functional design effort, as described in the previous section, is properly a part of preliminary design.

Typical products of preliminary design include

- design and interface specifications (B specs);
- supporting design and effectiveness trade studies;
- mock-ups, models, and breadboards;
- interface design;
- software top-level design;
- development, integration, and verification test plans; and
- engineering specialty studies (RMA, producibility, logistic support, etc.).

Major systems engineering input and review is essential for all of the above items. Of particular importance is the manner in which the functional modules defined in the functional design process are implemented in hardware and software. Often this requires detailed adjustments in the boundaries between components to ensure that physical interfaces, as well as functional interactions, are as simple as practicable. To the extent that the advanced development phase has not resolved all significant risks, further analyses, simulations, and experiments may have to be conducted to support the preliminary design process.

Preliminary Design Review (PDR). In government programs, the PDR is normally conducted by the acquisition agency to certify the completion of the preliminary design. For major commercial programs, company management acts in the role of the customer. The process is frequently led or supported by a commercial or nonprofit systems engineering organization. The review may last for a few or many days and may require several follow-on sessions if additional engineering is found to be required.

The issues on which PDR is usually centered include major (e.g., subsystem and external) interfaces, risk areas, long-lead items, and system-level trade-off studies. Design requirements and specifications, test plans, and logistics support plans are reviewed. Systems engineering is central to the PDR process and must be prepared to deal with any questions that may arise in the above areas.

Prior to the formal PDR, the development team should arrange for an internal review to ensure that the material to be presented is suitable and adequate. The preparation, organization, and qualification of the review process is critical. This is no less important for commercial systems, even though the review process may be less formal, because success of the development is critically dependent on the quality of design at this stage.

The completion of preliminary design corresponds to the establishment of the allocated baseline system configuration (see Section 12.6).

Detailed Design

The objective of detailed design is to produce a complete description of the end items constituting the total system. For large systems, a massive engineering effort is required to produce all the necessary plans, specifications, drawings, and other documentation necessary to justify the decision to begin fabrication. The amount of effort to produce a detailed design of a particular component depends on its "maturity," that is, its degree of previously proven design. For newly developed components, it is usually necessary to build prototypes and to test them under simulated operating conditions to demonstrate that their engineering design is valid.

Typical products of detailed design include

- draft C, D, and E specs (production specifications);
- subsystem detailed engineering drawings;
- prototype hardware;

- interface control drawings;
- configuration control plan;
- detailed test plans and procedures;
- quality assurance plan; and
- detailed integrated logistic support plans.

Systems engineering inputs are especially important to the interface designs and test plans. Where necessary, detailed analysis, simulation, component tests, and prototyping must be performed to resolve risk areas.

Critical Design Review (CDR). The general procedures for the CDR of the products of detailed design are similar to those for the PDR. The CDR is usually more extensive and may be conducted separately for hardware and software CIs. The CDR examines drawings, schematics, data flow diagrams, test and logistic supply plans, and so on, to ensure their soundness and adequacy. The issues addressed in the CDR are partly predicated on those identified as critical in the PDR and are therefore scheduled for further review in light of the additional analysis, simulations, breadboard or brassboard, or prototype tests conducted after the PDR.

As in the case of PDR, systems engineering plays a crucial role in this process, especially in the review of interfaces and plans for integration and testing. Similarly, internal reviews are necessary prior to the official CDR to ensure that unresolved issues do not arise in the formal sessions. But if they do, systems engineering is usually assigned the responsibility of resolving the issues as quickly as possible.

The completion of detailed design results in the product baseline (see Section 12.6).

Computer-Aided Design (CAD)

The microelectronic revolution has profoundly changed the process of hardware component design and fabrication. It has enabled the development and production of increasingly complex systems without corresponding increases in cost and degradations in reliability. The introduction of CAD of mechanical components has completely changed how such components are designed and built. Even more dramatic has been the explosive development of electronics in the form of microelectronic chips of enormous capacity and power, and their principal product, digital computing.

Mechanical Components. CAD permits the detailed design of complex mechanical shapes to be performed by an engineer at a computer workstation without making conventional drawings or models. The design takes form in the computer database and can be examined in any position, at any scale of magnification, and in any cross section. The same database can be used for calculating stresses, weights, positions relative to other components, and other relevant information. When the design is completed, the data can be transformed into fabrication instructions and transferred to digi-

tally driven machines for computer-aided manufacture (CAM) of exact replicas of the design. It can also generate production documentation in whatever form may be required.

One of the dramatic impacts this technology has had on the design and manufacturing process is that once a part has been correctly designed and built, all subsequent copies will also be correct within the tolerances of the production machines. An equally major impact is on the ease of integrating mechanical components with one another. Since the physical interfaces of components can be specified precisely in three dimensions, two adjacent components made to a common interface specification will match exactly when brought together. Today, a complex microwave antenna can be designed, fabricated, and assembled into a finely tuned device without the months of cut-and-try testing that used to characterize antenna design. This technique also largely eliminates the need for the elaborate jigs and fixtures previously used to make the parts fit a given pattern, or specially built gages or other inspection devices to check whether or not the parts conform to the established tolerances.

Electronic Components. The design of most electronic components has been revolutionized by modern technology even more than that of mechanical components. Processing is almost entirely digital, using standard memory chips and processors. All parts, such as circuit cards, card cages, connectors, equipment racks, and so on, are purchased to strict standards. All physical interfaces fit because they are made to standards. Further, in digital circuits, voltages are low; there is little heat generation; and electrical interfaces are digital streams. Inputs and outputs can be generated and analyzed using computer-based test equipment.

Most circuits are assembled on standard circuit cards and are interconnected by programmed machines. Instead of being composed of individual resistors, capacitors, transistors, and so on, most circuit functions are frequently incorporated in circuit chips. The design and fabrication of chips represents a still higher level of automation than that of circuit boards. The progressive miniaturization of the basic components (e.g., transistors, diodes, and capacitors) and of their interconnections has resulted in a doubling of component density and operating speed every 18 months (Moore's law) since the early 1980s—a trend that has not yet diminished. However, the cost of creating an assembly line for a complex new chip has progressively mounted into hundreds of millions of dollars, restricting the number of companies capable of competing in the production of large memory and processing chips. On the other hand, making smaller customized chips is not prohibitive in cost and offers the advantages of high reliability at affordable prices.

Components that handle high power and high voltage, such as transmitters and power supplies, generally do not lend themselves to the above technology, and for the most part must still be custom built and designed with great care to avoid reliability problems (see later section).

Systems Engineering Considerations. To the systems engineer, these developments are vital because of their critical impact on component cost, reliability, and

often design feasibility. Thus, systems engineers need to have first-hand knowledge of the available automated tools, their capabilities and limitations, and their effect on component performance, quality, and cost. This knowledge is essential in judging whether or not the estimated performance and cost of proposed components are realistic, and whether their design takes adequate advantage of such tools.

It is also important that systems engineers be aware of the rate of improvement of automated tools for design and manufacture, to better estimate their capabilities at the time they will be needed later in the system development cycle. This is also important in anticipating competitive developments, and hence the likely effective life of the system prior to the onset of obsolescence.

Example: The Boeing 777. The development of the Boeing 777 airliner has received a great deal of publicity as a pioneer in large-scale automated design and manufacture. It was claimed by Boeing to be the first major aircraft that was designed and manufactured without one or more stages of prototype ground and flight testing. This achievement was made possible mainly because of four factors: (1) the use of automated design and manufacture for all parts of the aircraft structure, (2) the high level of knowledge of aerodynamics and structures of aircraft obtained through years of development and experimentation, (3) the application of computer-based analysis tools, and (4) highly integrated and committed engineering teams. Thus, aircraft body panels were designed and built directly from computer-based design data and fit together perfectly when assembled. This approach was used for the entire airplane body and associated structures.

It should be noted that the 777 engines, whether built by Pratt and Whitney, General Electric, or Rolls Royce, were thoroughly ground tested before delivery because the degree of knowledge and predictability for jet engines is not at the level of that for airframes. Also, the 777 design did not embody radical departures from previous aircraft experience. Thus, the development cycle of the 777 as a total system did not depart as widely from the traditional sequence as it may have appeared to. However, it was a major milestone and a dramatic illustration of the power of automation in certain modern systems.

Reliability

The reliability of a system is the probability that the system will perform its functions correctly for a specified period of time under specified conditions. Thus, the total reliability (P_R) of a system is the probability that every component on which its function depends functions correctly. Formally, reliability is defined as one minus the failure distribution function of a system or component:

$$R(t) = 1 - F(t) = \int_t^\infty f(t)dt,$$

where $F(t)$ is the failure distribution function and $f(t)$ is the probability density function of $F(t)$. $f(t)$ can follow any number of known probability distributions. A common representation for a failure function is the exponential distribution

$$\text{pdf: } f_X(x) = \begin{cases} \lambda e^{-\lambda} & \text{if } x \geq 0 \\ 0 & \text{otherwise} \end{cases}; \quad \text{cdf: } F_X(x) = \begin{cases} 1 - e^{-\lambda} & \text{if } x \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

$$\text{Expectation: } E[X] = \frac{1}{\lambda}; \quad \text{variance: } \text{Var}[X] = \frac{1}{\lambda^2}.$$

This distribution is used quite extensively for common component reliability approximations, such as those relating to electrical and mechanical devices. An advantage of using the exponential distribution is its various properties relating to reliability:

$$f(t) = \frac{1}{\theta} e^{-t/\theta}, \quad \text{where } \theta = \text{mean life and } t \text{ is the time period of interest;}$$

$$\lambda - \frac{1}{\theta}, \quad \text{where } \lambda = \text{failure rate; and}$$

$$R = e^{-t/M}, \quad \text{where } R = \text{reliability of the system}$$

$$M = \text{MTBF.}$$

MTBF is "mean time between failure" and is explained below. By using the exponential distribution, we can calculate individual reliabilities fairly easily and perform simple mathematics to obtain reliability approximations, described below.

Calculating the probability depends on the configuration of the individual system components. If the components are arranged in a series, each one depending on the operation of the others, the total system probability is equal to the product of the reliabilities of each component (Pr):

$$P_R = \text{Pr}_1 \times \text{Pr}_2 \ldots \text{Pr}_n.$$

For example, if a system consisting of 10 critical components in series is required to have a reliability of 99%, then the average reliability of each component must be at least 99.9%.

If a system contains components that are configured in parallel, representing redundancy in operations, a different equation is used. For example, if two components are operating in parallel, the overall reliability of the system is

$$P_R = \text{Pr}_1 + \text{Pr}_2 - (\text{Pr}_1 \times \text{Pr}_2).$$

For pure parallel components, such as the example above, at least one component operating would allow the total system to operate effectively. Redundancy is discussed further below.

In most cases, a system consists of both parallel and series components. Keep in mind that for both examples above, time is considered integral to the definition of probability. Pr_i would be defined and calculated from the failure distribution function, which contains t. For the exponential distribution, Pr_i would be expressed as $1/e^{-t/M}$.

For systems that must operate continuously, it is common to express their reliability in terms of the MTBF. In the 10-component system just mentioned, if the system MTBF must be 1000 hours, the component MTBF must average 10,000 hours. From these

considerations, it is evident that the components of a complex system must meet extremely stringent reliability standards.

Since system failures almost always occur at the level of components or below, the main responsibility for a reliable design rests on design specialists who understand the details of how components and their subcomponents and parts work and are manufactured. However, the difficulty of achieving a given level of reliability differs widely among the various components. For example, components composed largely of integrated microcircuits can be expected to be very reliable, whereas power supplies and other high-voltage components are much more highly stressed and therefore require a greater fraction of the overall reliability "budget." Accordingly, it is necessary to allocate the allowable reliability requirements among the various components so as to balance, insofar as practicable, the burden of achieving the necessary reliabilities among the components. This allocation is a particular systems engineering responsibility and must be based on a comprehensive analysis of reliability records of components of similar functionality and construction.

A number of specific reliability issues must not be left entirely to the discretion of the component designers; these issues should not only be examined at formal reviews but should also be subject to oversight throughout the design process. Such issues include

1. *External Interfaces:* Surfaces exposed to the environment must be protected from corrosion, leakage, radiation, structural damage, thermal stress, and other potential hazards.
2. *Component Mounting:* Systems subjected to shock or vibration during operation or transport must have suitable shock mountings for fragile components.
3. *Temperature and Pressure:* Systems subjected to extremes of temperature and pressure must provide protective controls at either the system or component level.
4. *Contamination:* Components susceptible to dust or other contaminants must be assembled under clean room conditions and sealed if necessary.
5. *High-Voltage Components:* Components using high voltage, such as power supplies, require special provisions to avoid short circuits or arcing.
6. *Workmanship:* Parts requiring precise workmanship should be designed for easy inspection to detect defects that could lead to failures in operation.
7. *Potential Hazards:* Components that may present operating hazards if not properly made or used should be designed to have large reliability margins. These include rocket components, pyrotechnics, hazardous chemicals, high-pressure containers, and so on.

Software Reliability. Software does not break, short-circuit, wear out, or otherwise fail from causes similar to those that lead to most hardware failures. Nevertheless, complex systems do fail due to malfunctioning software as often as and sometimes even more often than from hardware faults. Anyone whose computer keyboard has "locked up," or who has tried to buy an airline ticket when the "computer is down" has

experienced this phenomenon. With systems increasingly dependent on complex software, its reliability is becoming ever more crucial.

Software operating failures occur due to imperfect code, that is, computer program deficiencies that allow the occurrence of unintended conditions, causing the system to produce erroneous outputs, or in extreme cases to abort ("crash"). Examples of conditions that cause such events are infinite loops (repeated sequences that do not always terminate, thereby causing the system to hang up), overflows of memory space allocated to data arrays (which cause excess data to overwrite instruction space, producing "garbage" instructions), and mishandling of external interrupts (which cause losses or errors of input or output).

As described in Chapter 11, there is no possibility of finding all the deficiencies in complex code by inspection, nor is it practical to devise sufficiently exacting tests to discover all possible faults. The most effective means of producing reliable software is to employ experienced software designers and testers in combination with disciplined software design procedures, such as

1. highly modular program architecture,
2. disciplined programming language with controlled data manipulation,
3. disciplined coding conventions requiring extensive comments,
4. design reviews and code "walk-throughs,"
5. prototyping of all critical interfaces,
6. formal CM,
7. independent verification and validation, and
8. endurance testing to eliminate "infant mortality"

Redundancy. Complex systems that must operate extremely reliably, such as air traffic control systems, telephone networks, power grids, and passenger aircraft, require the use of redundant or backup subsystems or components to achieve the required levels of uninterrupted operation. If a power grid line is struck by lightning, its load is switched to other lines with a minimum disruption of service. If an aircraft landing gear's motors fail, it can be cranked down manually. Air traffic control has several levels of backups to maintain safe (though degraded) operation in case of failure of the primary system.

The equation for calculating the reliability of parallel components was presented above. Another perspective on parallel component reliability is to understand that the failure probability is a product of the failure probabilities of the individual system modes. Since the reliability (P_R) is one minus the failure probability (P_F), the reliability of a system with two redundant (parallel) subsystems is

$$P_R = 1 - (P_{F1} \times P_{F2}).$$

The reader is encouraged to prove to himself that the two reliability equations presented are indeed equivalent. As an example, if a system reliability of 99.9 is required for a system, and a critical subsystem cannot be designed to have a reliability better than

99%, providing a backup subsystem of equal reliability will raise the effective reliability of the parallel subsystem to 99.99% ($1 - 0.01 \times 0.01 = 0.9999$).

Systems that must reconfigure themselves by automatically switching over to a backup component in place of a failed one must also incorporate appropriate failure sensors and switching logic. A common example is the operation of an uninterruptible power supply for a computer, which automatically switches to a battery power supply in the event of an interruption in external power. Telephone networks switch paths automatically not only when a link fails but also when one becomes overloaded. An inherent problem with such automatic switching systems is that the additional sensors and switches add further complexity and are themselves subject to failure. Another is that complex automatic reconfiguration systems may overreact to an unexpected set of conditions by a catastrophic crash of the whole system. Such events have occurred in a number of multistate power grid blackouts and telephone outages. Automatically reconfigurable systems require extremely comprehensive systems engineering analysis, simulation, and testing under all conceivable conditions. When this has been expertly done, as in the manned space program, unprecedented levels of reliability have been achieved.

Techniques to Increase Reliability. Several techniques exist to increase, or even maximize, reliability within a system design. Several have been discussed already:

- *System Modularity.* Increase the modularity of system components to achieve loose coupling among components. This will minimize the number of components that are in series and thus could cause a system failure.
- *Redundancy.* Increase component redundancy either with parallel operating components or through the use of switches that automatically transfer control and operations to backup components.
- *Multiple Functional Paths.* A technique to increase reliability without necessarily adding redundant components involves including functional multiple paths within the system design. This is sometimes known as "channels of operation."
- *Derating Components. Derating* refers to the technique of using a component under stress conditions considerably below the rated performance value to achieve a reliability margin in the design.

Several methods and formal techniques exist to analyze failure modes, effects, and mitigation strategies. Five common techniques (not described here) are failure mode, effects, and criticality analysis (FMECA), fault tree analysis, critical useful life analysis, stress–strength analysis, and reliability growth analysis. The reader is encouraged to explore any or all of the techniques as effective analyses strategies.

Maintainability

The maintainability of a system is a measure of the ease of accomplishing the functions required to maintain the system in a fully operable condition. System maintenance takes

two forms: (1) repair if a system fails during operation and (2) scheduled periodic servicing including testing to detect and repair failures that occur during standby. High maintainability requires that the system components and their physical configurations be designed with an explicit and detailed knowledge of how these functions will be carried out.

Since to repair a system failure it is first necessary to identify the location and nature of the fault, that is, to carry out a failure diagnosis, system design should provide for means to make diagnosis easy and quick. In case repair is needed, the design must be dovetailed with logistic support plans to ensure that components or component parts that may fail will be stocked and will be replaceable in minimum time.

Unlike hardware faults, replacement of the failed component is not an option for software because software failures result from faults in the code. Instead, the error in the code must be identified and the code modified. This must be done with great care and the change in configuration documented. To prevent the same fault from causing failures in other units of the system, the correction must be incorporated in their programs. Thus, software maintenance can be a critical function.

A measure of system maintainability during operation is the mean time to repair/restore (MTTR). The "time to repair" is the sum of the time to detect and diagnose the fault, the time to secure any necessary replacement parts, and the time to effect the replacement or repair. The "time to restore" also includes the time required to restore the system to full operation and to confirm its operational readiness.

Built-In Test Equipment (BITE). A direct means for reducing the MTTR of a system is to incorporate auxiliary sensors that detect the occurrence of faults that would render the system inoperable or ineffective when called upon, then to signal an operator that repairs are required, and indicate the location of the fault. Such built-in equipment effectively eliminates the time to detect the fault and focuses the diagnosis on a specific function. Examples of such built-in fault detection and signaling devices are present in most modern automobiles, which sense and signal any faulty indications of air bag or antilock brake status, low oil level, or low battery voltage, and so on. In controlling complex systems, such as in aircraft controls, power plant operations, and hospital intensive care units, such devices are absolutely vital. In automatically reconfigurable systems (see section on redundancy, above) the built-in sensors provide signals to automatic controls rather than to a system operator.

The use of BITE presents two important system-level problems. First, it adds to the total complexity of the system and hence to potential failures and cost. Second, it is itself capable of false indications, which can in turn impact system effectiveness. Only when these problems are examined in detail can a good balance be struck between not enough and too much system self-testing. Systems engineering bears the principal responsibility for achieving such a balance.

Design for Maintainability. The issues that must be addressed to ensure a maintainable system design begin at the system level and range all the way down to component parts. They include

1. *Modular System Architecture:* A high degree of system modularity (self-contained components with simple interfaces) is absolutely vital to all three forms of maintenance (repair of operational failures, periodic maintenance, and system upgrading).
2. *Replaceable Units:* Because it is often impractical to repair a failed part in place, the unit that contains the part must be replaced by an identical spare unit. Such units must be accessible, simply and safely replaceable, and part of the logistic support supply.
3. *Test Points and Functions:* To identify the location of a failure to a specific replaceable unit, there must be a hierarchy of test points and functions that permits a short sequence of tests to converge on the failed unit.

To achieve the above, there must be an emphasis on design for maintainability throughout the system definition, development, and engineering design process. In addition to the design, comprehensive documentation and training are essential.

Availability

An important measure of the operational value of a system that does not operate continuously is referred to as system availability, that is, the probability that it will perform its function correctly when called upon. Availability can be expressed as a simple function of system reliability and maintainability for relatively short repair times and low failure rates:

$$P_A = 1 - \frac{\text{MTTR}}{\text{MTBF}},$$

where

P_A = probability that the system will perform when called upon;
MTBF = mean time between failure; and
MTTR = mean time to restore.

This formula shows that system maintainability is just as critical as reliability and emphasizes the importance of rapid failure detection, diagnosis, and repair or parts replacement. It also points to the importance of logistic support to ensure the immediate availability of necessary replacement parts.

Producibility

For systems that are produced in large quantities, such as commercial aircraft, automotive vehicles, or computer systems, reducing the costs associated with the manufacturing process is a major design objective. The characteristic that denotes relative system production costs is called "producibility." The issue of producibility is almost wholly

associated with hardware components since the cost of replicating software is only that of the medium in which it is stored.

Design for producibility is the primary province of the design specialist. However, systems engineers need to be sufficiently knowledgeable about manufacturing processes and other production cost issues to recognize characteristics that may inflate costs and to guide design accordingly. Such understanding is necessary for the systems engineer to achieve an optimum balance between system performance (including reliability), schedule (timeliness), and cost (affordability).

Some of the measures that are used to enhance producibility are

1. maximum use of commercially available parts, subcomponents, and even components (referred to as commercial off-the-shelf ["COTS"] items); this also reduces development cost;
2. setting dimensional tolerances of mechanical parts well within the normal precision of production machinery;
3. design of subassemblies for automatic manufacture and testing;
4. maximum use of stampings, castings, and other forms suitable for high-rate production;
5. use of easily formed or machined materials;
6. maximum standardization of subassemblies, for example, circuit boards, cages, and so on; and
7. maximum use of digital versus analog circuitry.

As noted in previous chapters, the objective of producibility, along with other specialty engineering features, should be introduced into the system design process early in the life cycle. However, the application of producibility to specific design features occurs largely in the engineering design phase as part of the design process. Chapter 14 is devoted to the subject of production and its systems engineering content.

Risk Management

Many of the methods of risk mitigation listed in Chapter 5 are pertinent to the component design step in the engineering design phase. Components containing residual risk factors must be subjected to special technical and management oversight, including analysis and testing to ensure the early discovery and resolution of any design problems. Where the acceptability of a given design requires testing under operational conditions, as in the case of user interfaces, rapid prototyping and user feedback may be in order. In exceptional circumstances, where the risk inherent in the chosen approach remains unacceptably high, it may be necessary to initiate a backup effort to engineer a more conservative replacement in case the problems with the first line design cannot be resolved when the design must be frozen. Alternatively, it may be wise to seek relaxation of stringent requirements that would produce only marginal gains in system effectiveness. All of the above measures require systems engineering leadership.

12.5 DESIGN VALIDATION

Design validation proceeds at various levels throughout the engineering development stage of the system life cycle. This section focuses on the validation of the physical implementation of the component system building blocks.

Test Planning

Planning the testing of components to validate their design and construction is an essential part of the overall test and evaluation plan. It covers two types of tests: development testing during the component design process and unit qualification testing to ensure that the final production design meets specifications.

Component test planning must be done during the early part of the engineering design phase for several reasons. First, the required test equipment is often complex and requires a time to design and build comparable to that required for the system components themselves. Second, the cost of test tools usually represents a very significant fraction of the system development costs and must be provided for in the total cost equation. Third, test planning must involve design engineers, test engineers, and systems engineers in a *team effort*, often across organizational and sometimes across contractual lines. From these detailed plans, test procedures are derived for all phases of the test operations.

As in system-level test planning, systems engineering must play a major role in the development of component test plans, that is, what should be tested, at what stage in the development, to what degree of accuracy, what data should be obtained, and so on. An important systems engineering contribution is to ensure that component features that were identified as potential risks are subjected to tests to confirm their elimination or mitigation.

Component Fabrication

In the previous sections, the design process has been discussed in terms of its objectives and has been related to design decisions defined in terms of drawings, schematics, specifications, and other forms of design representation as expressed on paper and in computer data. To determine the degree to which a design will actually result in the desired component performance, and whether or not the component will properly interface with the others, it is necessary to convert its design to a physical entity and to test it. This requires that hardware elements be fabricated and individual software components be coded. Prior to fabrication, reviews are held between the designers and fabrication personnel to assure that what has been designed is within the capabilities of the facility that has to build it.

The implementation process is seldom unidirectional (i.e., noniterative). Design deficiencies are often discovered and corrected during implementation, even before testing, especially in hardware components. Even though CAD has greatly reduced the probability of dimensional and other incompatibilities, it has to be anticipated that some changes will need to be made in the design to achieve a successful functioning product.

At this stage of component engineering, the tools that are to be used in production (such as computer-driven, metal-forming machines and automatic assembly devices) are seldom available for use, so that initial fabrication must often be carried out using manually operated machines and hand assembly. It is important, however, that a realistic experimental replica of the fabrication process be employed for any component parts that are to be built using unconventional manufacturing processes. This is essential to ensure that the transition to production tooling will not invalidate the results of the prototyping process. Involving the production people during sign-off, prior to the time the article reaches the manufacturing facility, will greatly expedite production.

In the case of complex electronic circuits, significant alterations in the initially fabricated model are to be expected before a completely suitable design is finally achieved. Accordingly, it has been customary to first construct and test these circuits in a more open "breadboard" or "brassboard" form (with rudimentary packaging constraints) so as to facilitate circuit changes before packaging the component in its final form. However, with modern automated tools, it is often more efficient to go directly to a packaged configuration, even though this may dictate the fabrication of several such packages before a suitable design is finally achieved.

Development Testing

The objective of engineering development testing is different from production acceptance testing in that the latter is mainly concerned with whether the component should be accepted or rejected, while the former must not only quantify each discrepancy but must also help diagnose its source. It should be anticipated that design discrepancies will be found and design changes will be needed in order to comply with requirements. Thus, component testing is very much a part of the development process. Changes at this point must be introduced via an "engineering change notice" agreed to by all cognizant parties to avoid chaotic, noncoordinated change.

Development testing is concerned with validating the basic design of the component, focusing on its performance, especially on features that are critical to its operation within the system or that represent characteristics that are highly stressed, newly developed, or are expected to operate at levels beyond those commonly attained in previous devices of this type. These tests also focus on the features of the design that are subject to severe environmental conditions, such as shock, vibration, external radiation, and so on.

For components subject to wear, such as those containing moving parts, development tests can also include endurance testing, usually performed under accelerated conditions to simulate years of wear in a matter of months.

Reliability and Maintainability Data. Whereas during development components may not be built from the identical parts used in the production article, it is good practice to begin collecting reliability statistics as early as possible by recording all failures during operation and test and by identifying their source. This will reduce the likelihood of incipient failures carrying on into the production article. This is particularly important where the number of units to be built is too small to collect adequate

statistical samples of production components. Involvement of quality assurance engineers in this process is essential.

Development testing must also examine the adequacy and accessibility of test points for providing failure diagnosis during system maintenance. If maintenance of the system will require disassembling the component and replacing subcomponents such as circuit boards, this feature must also be evaluated.

Test Operations. Component development tests are part of the design process and are usually conducted within the design group by a team headed by the lead design engineer and composed of members of the design team as well as other staff experienced in testing the type of component under development. The team should be intimately familiar with the use of test tools and special test facilities that may be required. The validity and adequacy of the test setup and analysis procedures should be overseen by systems engineering.

An important lesson that systems engineers (and test engineers) must learn is that the apparent failure of a component to meet some test objective may not be due to a defective design but rather due to a deficiency in the test equipment or test procedure. This is especially true when a component is first tested in a newly designed test setup. The need for testing the test equipment occurs all too frequently. This is a direct result of the difficulty of ensuring perfect compatibility between two or more interacting and interfacing components, whether they are system elements or test equipment units (hardware or software). Thus, a period of preliminary testing should be scheduled to properly integrate a new component with its test equipment, and unit testing should not begin until all the test bugs have been eliminated.

Change Control. It will be recalled that after the CDR, the detailed design of a complex system is frozen and placed under formal CM (see Section 9.6). This means that thereafter, any proposed design change requires justification, evaluation, and formal approval, usually from a "configuration control board" or an equivalent. Such approval is usually granted only on the basis of a written engineering change request containing a precise definition of the nature of the deficiency revealed by the test process and a thorough analysis of the impact of the proposed change on system performance, cost, and schedule. The request should also contain trade-offs of alternative remedies, including possible relaxation of requirements, and an in-depth assessment of risks and costs associated with making (and not making) the change. This formal process is not intended to prevent changes but to ensure that they are introduced in an orderly and documented manner.

Qualification Testing

Testing a productionized component ("first-unit" testing) prior to its delivery to the integration facility is very much like the acceptance testing of units off the production line. Qualification tests are usually more limited than development tests, but are frequently more quantitative, being concerned with the exact conformance of the unit to interface tolerances so that it will fit exactly with mating system components.

Accordingly, equipment used for this purpose should be much like production test equipment. Qualification tests are generally more severe than the conditions to which the article is subjected in operational use.

The validation of the design of an individual system component can be rigorously accomplished only by inserting it into an environment identical to that in which it will operate as part of the total system. In the case of complex components, it is seldom practicable to reproduce exactly its environment. Therefore, a test setup that closely approximates this situation has to be used.

The problem is made more difficult by the fact that components are almost always developed and built by different engineering groups, often by independent contractors. In the case of software programs, the designers may be from the same company but generally do not understand each other's designs in detail. The system developer thus has the problem of ensuring that the component designers test their products to the identical standards to be used during system integration. The critical point, of course, is that each component's interfaces must be designed to fit exactly with their connecting components and with the environment.

Tolerances. The specification of component interfaces to ensure fit and interchangeability involves the assignment of tolerances to each dimension or other interface parameters. Tolerances represent the positive and negative deviation from a nominal parameter value to ensure a proper fit. The assignment of tolerances requires striking a balance between ease of manufacturing on one hand and assurance of satisfactory fit and performance on the other. Whenever either producibility or reliability is significantly affected, the systems engineer needs to enter the process of setting the preferred balance.

Computer-Aided Tools. The widespread use of CAD and CAM has greatly simplified the above problems in many types of equipment. With these tools, component specifications can be converted into a digital form and can be directly used in their design. The CAD database can be shared electronically between the system developer and the component designer and producer. The same data can be used to automate test equipment.

In the area of electronic equipment, the widespread use of standard commercial parts, from chips to boards to cabinets to connectors, has made interfacing much easier than with custom-built components. These developments have produced economies in test and integration, as well as in component costs. Miniaturization has resulted in a greater number of functions being performed on a circuit board, or encapsulated in a circuit chip, thereby minimizing interconnections and numbers of boards.

Test Operations. Component qualification tests are performed to ensure that the final production component design meets all of its requirements as part of the overall system. Hence, they are much more formal than development tests and are conducted by the test organization, sometimes with oversight by the system contractor. Design engineering supports the test operations, especially during test equipment checkout and data analysis.

Test Tools

A set of test tools for verifying the performance and compatibility of a system component must be designed to provide an appropriate set of inputs and to compare the resulting outputs with those prescribed in the specifications. In effect, they constitute a simulator, which models the physical and functional environments of the component, both external and internal to the system, and measures all significant interactions and interfaces. Functionally, such a simulator may be as complex as the component that it is designed to test, and its development usually requires a comparable level of analysis and engineering effort. Moreover, the assessment of a component's adherence to specified parameter tolerance values usually requires the test equipment precision to be several times better than the allowable variations in component parameters. This requirement sometimes calls for precision greater than that readily available, involving a special effort to develop the necessary capability.

Development test tools often may be available or may be adaptable from other programs. In addition, standard measuring instruments, such as signal generators, spectrum analyzers, displays, and so on, are readily available in a form that can be incorporated as part of a computer-driven test setup. On the other hand, highly specialized and complex components, such as a jet engine, may require the provision of dedicated and extensively instrumented test facilities to be used to support testing during component development and sometimes also during production.

In any event, such special tools as are required to support design and testing during component development must be designed and built early in the engineering design phase. Moreover, since similar tools will also be needed to test these same components during production, efforts should be made to assure that the design and construction of engineering and production test equipment are closely coordinated and mutually supporting. To keep the cost of such test tools within acceptable bounds, significant systems engineering effort is usually needed to support the planning and definition of their design and performance requirements.

Role of Systems Engineering

From the above discussion, it should be evident that systems engineering plays an essential part in the component validation process. Systems engineers should define the overall test plan, specify what parameters should be tested and to what accuracy, how to diagnose discrepancies, and how the test results should be analyzed. Systems engineering must also lead the change initiation and control process. The proper balance between "undertesting" and "overtesting" requires knowledge of the system impact of each test, including overall cost. This, in turn, depends on a first-hand knowledge of the interactions of the component with other parts of the system and with its environment.

12.6 CM

The development of a complex new system has been seen to be resolvable into a series of steps or phases in which each of the characteristics of the system is defined in terms

of successively more specific system requirements and specifications. The systems engineering process that maintains the continuity and integrity of the system design throughout these phases of system development is called "CM."

The CM process generally begins incrementally during the concept exploration phase, which first defines the selected top-level system configuration in terms of functional requirements after a process of trade-offs among alternative system concepts. It then progresses throughout the phases of the engineering development stage, culminating in system production specifications. The CM process is described more fully in this chapter because the intensity and importance of CM is greatest during the engineering design phase. The terminology of formal CM includes two basic elements, CIs and configuration baselines. Each of these is briefly described below.

CIs

A CI is a system element that is the basis of describing and formally controlling the design of a system. In early phases of system definition, it may be at the level of a subsystem. In later phases, it usually corresponds to that of a component in the hierarchy defined in this book (see Chapter 3). Like the component, the CI is considered as a basic building block of the system, designed and built by a single organization, whose characteristics and interfaces to other building blocks must be defined and controlled to ensure its proper operation within the system as a whole. It is customary to distinguish between hardware configuration items (HWCIs) and computer software configuration items (CSCIs) because of the basically different processes used in defining and controlling their designs.

Configuration Baselines

An important concept in the management of the evolving system design during the system life cycle is that of configuration baselines. The most widely used forms are called functional, allocated, and product baselines. Table 12.2 shows the phase in which each is usually defined, the type of specification that describes it, and the primary characteristics that are specified.

TABLE 12.2. Configuration Baselines

Baseline	Phase defined	Type of specification	Characteristics	Element specified
Functional	Concept definition	A	Functional specifications	System
Allocated	Engineering design	B	Development specifications	Configuration item
Product	Engineering design	C, D, E	Product, process specifications	Configuration item

The functional baseline describes the system functional specifications as they are derived from system performance requirements during the concept definition phase and serves as an input to the advanced development phase.

The allocated baseline is defined during the engineering design phase as the allocations of functions to system components (CIs) are validated by analyses and tests. The resulting development specification defines the performance specifications for each CI, as well as the technical approaches developed to satisfy the specified objective.

The product baseline is established during the engineering design phase in terms of detailed design specifications. It consists of product, process, and material specifications and engineering drawings.

Interface Management

It has been stressed throughout this book that the definition and management of the interfaces and interactions of the system's building blocks with one another and with the system environment is a vital systems engineering function. This function is embodied in the concept of CM, irrespective of whether or not it is formally defined in terms of CIs and baselines as described above. It is therefore incumbent on project management with the aid of systems engineering to organize the necessary people and procedures to carry out this function.

A primary condition for the effective definition and management of a given interface is to ensure the involvement of all key persons and organizations responsible for the designs of the CIs. This is generally accomplished by means of interface configuration working groups (ICWGs), or their equivalents, whose members have the technical knowledge and authority to represent their organizations in negotiating a complete, compatible, and readily achievable definition of the respective interfaces. In large systems, formal sign-off procedures have been found to be necessary to ensure commitment of all parties to the agreed-upon interface coordination documents (ICDs). The form of these documents is a function of the type of interface being documented, but during the engineering design phase, it must be sufficiently specific in terms of data and drawings to specify completely the interface conditions, so that the individual component developers may design and test their products independently.

Change Control

Change is vital to the development of a new and advanced system, especially to take advantage of evolving technology to achieve a sufficient advance in system capability to provide a long useful life. Thus, during the formative stages of system development, it is desirable to maintain sufficient design flexibility to accommodate relevant technological opportunities. The price of such flexibility is that each change inevitably affects related system elements and often requires a series of adaptations extending far beyond the initial area of interest. Thus, a great deal of systems engineering analysis, test, and evaluation is required to manage the system evolution process.

The effort and cost associated with accommodating changes increases rapidly as the design matures. By the time the system design is formulated in detail during the

engineering design phase, the search for opportunities for further enhancement can no longer be sustained. Accordingly, the system design is frozen, and formal change control procedures are imposed to deal with necessary modifications, such as those required by incompatibilities, external changes, or unexpected design deficiencies. This usually happens after successful completion of the CDR or its equivalent.

It is customary to categorize proposed changes as class I, or class changes have system- or program-level impact, such as cost, schedule, major interfaces, safety, performance, reliability, and so on. Formal change control of system-level changes is usually exercised by a designated group composed of senior engineers with recognized technical and management expertise capable of making judgments among performance, cost, and schedule. For large programs, this group is called a change control board. It is of necessity led by systems engineering but usually reports at the topmost program level.

12.7 SUMMARY

Implementing the System Building Blocks

The objectives of the engineering design phase are to design system components to performance, cost, and schedule requirements. This phase also establishes consistent internal and external interfaces.

Engineering design culminates in materialization of components of a new system focused on the final design of the system building blocks. Activities constituting engineering design are

- *Requirements Analysis:* identifying all interfaces and interactions,
- *Functional Analysis and Design:* focusing on modular configuration,
- *Component Design:* designing and prototyping all components, and
- *Design Validation:* testing and evaluating system components.

Requirements Analysis

External system interface requirements are especially important at this point in development. User interfaces and environmental interactions require particular attention.

Functional Analysis and Design

Functional design stresses three areas:

- *Modular Configuration:* simplified interactions
- *Software Design:* modular architecture
- *User Interfaces:* effective human interaction.

Modular partitioning groups “tightly bound” functions together into “loosely bound” modules.

Component Design

Major defense and space systems engineering is performed in two steps: preliminary design followed by a PDR detailed design followed by a CDR.

The engineering design process is focused on CIs. These are substantially equivalent to components as defined in this book.

A preliminary design has the objective to demonstrate that chosen designs conform to system performance and design requirements that can be produced within cost and schedule goals. The PDR centers on major interfaces, risk areas, long-lead items, and system-level trade studies.

A detailed design has the objective to produce a complete description of the end items (CIs) constituting the total system. The CDR examines drawings, plans, and so on, for soundness and adequacy. Within the detailed design, CAD has revolutionized hardware implementation—mechanical component design can now be analyzed and designed in software. Digital electronics is miniaturized, standardized, and does not need breadboarding. The Boeing 777 development illustrates the power of automated engineering.

Reliability must be designed at the component level where interfaces, environment, and workmanship are vulnerable areas. Additionally, software must be built to exacting standards and prototyped. Where extreme reliability is required, it is typically achieved by redundancy. Measuring reliability usually includes the MTBF.

Maintainability requires rapid fault detection diagnosis and repair. MTTR is used as a typical measure of maintainability. BITE is used to detect and diagnose faults.

Availability measures the probability of the system being ready when called in: availability increases with MTBF and decreases with MTTR.

Producibility measures the ease of production of system components and benefits from use of commercial components, digital circuitry, and broad tolerances.

Design Validation

Test planning must be done early since test equipment requires extensive time to design and build. Additionally, test costs must be allocated early to ensure sufficient resources. Finally, test planning is a team effort.

Development testing is part of the design process and should start accumulating reliability statistics on failures. These test failures are often due to test equipment or procedures and should be planned for since changes after CDR are subject to formal CM.

Qualification testing validates component release to integration and focuses on component interfaces. Regardless of the testing phase, test tools must be consistent with the system integration process.

CM

CM is a systems engineering process that maintains the continuity and integrity of system design. Configuration baselines defined in major system developments include

- *Functional Baseline:* system functional specifications,
- *Allocation Baseline:* system development specifications, and
- *Product Baseline:* product, process, and material specifications.

The CI is a system element used to describe and formally control system design.

PROBLEMS

12.1 In spite of the effort devoted to develop critical system components during advanced development, unknown unknowns can be expected to appear during engineering design. Discuss what contingency actions a systems engineer should take in anticipation of these "unk-unks." Your answer should include the consideration of the potential impact on cost, schedule, personnel assignments, and test procedures. If you have knowledge of a real-life example from your work, you may use that as the basis for your discussion.

12.2 External system interfaces are especially important during engineering design. Using the design of a new subway system as an example, list six types of external interfaces that will require critical attention. Explain your answer.

12.3 Modular or sectionalized system design is a fundamental characteristic of good system design practice. Using a passenger automobile as an example, discuss its main subsystems from the standpoint of modularity. Describe those that are modular and those that are not. For the latter, state how and why you think they depart from modular design.

12.4 A PDR is an important event during engineering design and the systems engineer has a key role during this review. Assume you (the systems engineer) have been given the assignment to be the principal presenter for an important PDR. Discuss what specific actions you would take to prepare for this meeting. How would you prepare for items that could be considered controversial?

12.5 The personal laptop computer is a product that has proven to be very reliable in spite of the fact that it has many interfaces, is operated by a variety of people, operates nearly continuously, and includes a number of internal moving parts (e.g., floppy disk drive, hard drive, and CD-ROM drive). It is a portable device that operates in a wide range of environments (temperature, shock, vibration, etc.). List six design features or characteristics that contribute to the laptop reliability. For each item in your list, estimate the contributions this item has on the overall computer cost. A ranking of high, medium, and low is sufficient.

12.6 There are six methods of dealing with program risks listed in the section labeled "Risk Management Methods." For four of these six methods, give two examples of situations where that method could be used for risk reduction and explain how.

12.7 Design changes are vital to the development of new and advanced systems, especially to take advantage of evolving technology. Thus, during system development, some degree of design flexibility must be maintained. However, design changes come with a price that increases as the design matures. Assuming you are the systems engineer for the development of a new commercial jet aircraft, give two types of design changes you would support in each of the early part, middle part, and late part of the engineering design phase.

FURTHER READING

C. Alexander. *The Timeless Way of Building*. Oxford University Press, 1979.

A. B. Badiru. *Handbook of Industrial and Systems Engineering*. CRC Press, 2006, Chapters 8, 9, 11, and 15.

B. Blanchard and W. Fabrycky. *System Engineering and Analysis*, Fourth Edition. Prentice Hall, 2006, Chapters 4 and 5.

F. P. Brooks, Jr. *The Mythical Man Month—Essays on Software Engineering*. Addison-Wesley, 1995.

G. E. Dieter and L. C. Schmidt. *Engineering Design*, Fourth Edition. McGraw-Hill, 2009, Chapters 1, 2, 9, 13, and 14.

C. E. Ebeling. *An Introduction to Reliability and Maintainability Engineering*. Waveland Press, Inc., 2005, Chapters 1, 2, 5, 8, 9, and 11.

H. Eisner. *Computer-Aided Systems Engineering*. Prentice Hall, 1988, Chapters 14 and 15.

B. Hyman. *Fundamentals of Engineering Design*, Second Edition. Prentice Hall, 2003, Chapters 1, 5, 6, and 10.

J. A. Lacy. *Systems Engineering Management: Achieving Total Quality, Part II*. McGraw Hill, 1992.

P. D. T. O'Connor. *Practical Reliability Engineering*, Fourth Edition. John Wiley & Sons, Inc., 2008, Chapters 1, 2, and 7.

R. S. Pressman. *Software Engineering: A Practitioner's Approach*, McGraw Hill, 1982.

E. Rechtin. *Systems Architecting: Creating and Building Complex Systems*, Prentice Hall, 1991, Chapter 6.

N. B. Reilly. *Successful Systems for Engineers and Managers*, Van Nostrand Reinhold, 1993, Chapters 8–10.

A. P. Sage. *Systems Engineering*, McGraw Hill, 1992, Chapter 6.

R. M. Shinners. *A Guide for Systems Engineering and Management*, Lexington Books, 1989, Chapter 3.

Systems Engineering Fundamentals. Defense Acquisition University (DAU Press), 2001, Chapters 6 and 10.

Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities, INCOSE-TP-2003-002-03.2, Section 4. International Council on Systems Engineering, 2010.