Part **2**

Design and Integration

Chapter **6**

Requirements and Defining the Design Problem

6.1 INTRODUCTION

Requirements are the cornerstone of the systems engineering process: Stakeholders' requirements provide operational statements by the stakeholders concerning their needs; derived requirements enable the engineers of systems to partition the design problem into components that can be worked in parallel while maintaining design control through the requirements partition and the interfaces between the components; derived requirements enable the verification of the configuration items and components during the qualification activity during development; and stakeholders' requirements provide the means for validating the system's design during qualification.

Requirements do not just show up on the systems engineer's desk. Obtaining "good" requirements is critical to the successful engineering of a system [Blum, 1992, pp. 68–81; Davis, 2005, pp. 3–39]. The systems engineer must work hard with the stakeholders of the system to develop the requirements. Fortunately, there is a tried and true method with some valuable modeling techniques that can be used in this effort.

There are few references that provide a coherent view of the systems engineering process for developing stakeholders' requirements for a system, including a definition of how these requirements might be usefully characterized to aid the generation process. Grady [1993] provides an excellent discussion of what requirements are, how requirements should be written one at a time and in documents, and how requirements should be allocated. Faulk et al.

The Engineering Design of Systems: Models and Methods, Second Edition. By Dennis M. Buede

[1992] describe a software engineering method for real-time requirements that has many of the characteristics that are important. Crowe et al. [1996] adapt the method of Faulk et al. [1992] to software-intensive systems; however this adaptation is incomplete because software engineering assumes systems engineers are their interface to the stakeholders. However, no reference found by the author discusses systematically how requirements should be developed and how such constructs as the operational concept, prototyping, objectives hierarchy, and external systems diagram can be used in this process. This chapter (an expansion of Buede [1997]) defines such a process that is consistent with most systems engineering practice.

This chapter begins by discussing what requirements are. Definitions that are key to putting a system in its context with external systems and the environment are provided next. Section 6.4 defines the process or method by which requirements are developed. A discussion of various categories of requirements found in the literature of systems engineering are then discussed, followed by the partition of requirements that will be used in this book. The proposed outline for a stakeholders' requirements document that addresses all phases of the system's life cycle is provided in Section 6.7. The literature on requirements has proposed a number of characteristics that define either a sound individual requirement or a set of sound requirements; these characteristics of sound requirements are given in Section 6.8. The convention for writing requirements is discussed in Section 6.9.

Sections 6.10 to 6.13 describe in detail the portions of the process for developing requirements: defining the operational concept for each phase of the system's life cycle, creating an external systems diagram for each phase of the life cycle, establishing an objectives hierarchy for each phase of the life cycle, and conducting prototyping and usability testing to analyze the potential requirements in each phase of the life cycle. Section 6.14 provides a detailed discussion of the four segments of the requirements partition for each phase of the life cycle: the input/output requirements, the system-wide and technology requirements, the trade-off requirements, and the qualification requirements. Finally, the issue of managing requirements during the development of a system is discussed.

The focus of this chapter is the method for defining requirements for a system and all of the systems associated with each phase of the system's life cycle. There are seven activities associated with this method: developing the operational concept; defining the system boundary; developing an objectives hierarchy; developing, analyzing, and refining the requirements (including prototyping and usability testing); ensuring requirements feasibility; defining the qualification system requirements; and obtaining approval of the requirements.

Several models are introduced to support the process for defining requirements. A qualitative model, an input/output trace, is described for defining a scenario that is part of the system's operational concept. An application of IDEF0 (Integrated Definition for Function Modeling) modeling is described

for defining the process of a system's interaction with other (external) systems; this external system diagram defines all of the inputs and outputs associated with the system. A hierarchical decomposition of the objectives for a system is another example of a qualitative model used in this requirements definition process.

The exit criterion for this initial activity in the engineering of a system is the approval of the requirements document by the stakeholders. Often the engineers of a system are focused on obtaining this approval as quickly as possible, often without defining all of the requirements suggested in this chapter. The trade-off and qualification requirements are missing from most requirements documents. The contention of this chapter is that the real exit criterion of the requirements definition process is the approval by the stakeholders of the acceptance plan for the system. If the acceptance plan is affirmed, then all of the other portions of the requirements document are presumed to he defined in acceptable detail.

6.2 REQUIREMENTS

Many authors have defined the term *requirement*. The list below provides several definitions that highlight key concepts (the italics are the author's).

Sailor [1990]: identifiable capabilities expressed as *performance measurables* of functions that the system must possess to meet the mission objectives.

MIL-STD 499B [Military Standard, 1993]: identifies the *accomplishment levels* needed to achieve specific objectives.

Chambers and Manos [1992]: the attributes of the final design that must be a part of any *acceptable solution to the design problem.*

Grady [1993]: an *essential attribute* for a system or an element of a system, *coupled by a relation statement with value and units information for the attribute*.

Davis [2005]: an *externally observable characteristic of a desired system.*

The requirements for a system set up standards and measurement tools for judging the success of the system design. These requirements should be viewed hierarchically. At the top are mission-level requirements that establish how the stakeholders will benefit by introducing the system in question into the supersystem of the system. These *mission requirements* relate to objectives of the stakeholders that are defined in the context of the supersystem, not the system itself. For example, Boeing identified two primary mission requirements when starting on the Boeing 777 commercial aircraft: trip cost per seat and total trip cost. Each airline company that purchases a 777 is the meta-system that most influences an aircraft company during the development phase.

Stakeholders' requirements are developed next in the context of these mission requirements and should focus on the boundary of the system. If the stakeholders' requirements are defined internally to the system, the risk of having design statements embedded in the requirements goes up substantially. A major emphasis of this chapter is that the stakeholders' requirements should be as design independent as possible. Boeing's stakeholders' requirements for the 777 included such topics as liftable weight of the aircraft at specified conditions, the empty weight of the aircraft, the drag force on the aircraft for certain specified flight conditions, and the fuel consumption of the aircraft at certain specified flight conditions.

As discussed in Chapter 1 *system requirements* are a translation (or derivation) of the stakeholders' requirements into engineering terminology. Once this translation occurs, the derivation process of requirements continues. Recall from Chapter 1 that the goal of the design process is to create a system specification that can be developed into specifications for the system's components, which are then segmented into specifications for the system configuration items (CIs). As a result the design process creates two hierarchies of requirements as shown in Figure 6.1.

The stakeholders' requirements are produced in conjunction with the stakeholders of the system, based upon the operational needs of these stakeholders. Some systems engineers believe the systems engineering process begins when the Stakeholders' Requirements Document (StkhldrsRD) arrives; however the position taken here and supported by Pragmatic Principle 1 [De Foe, 1993] of the International Council on Systems Engineering (INCOSE) is that the systems engineers must be involved with the stakeholders to have any hope of producing a useful StkhldrsRD; note italicized items. In fact, the process described in this chapter is focused on methods and models for developing a valid and complete StkhldrsRD.

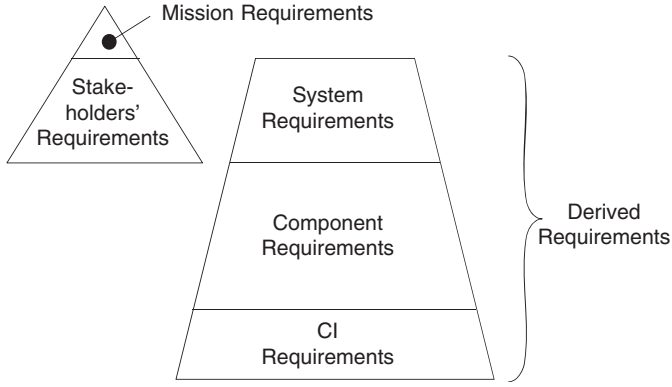

FIGURE 6.1 Requirements hierarchies.

The Systems Requirements Document (SysRD), which is derived from the StkhldrsRD, is a translation from the language of stakeholders to the language of engineers. The system's requirements are traced directly from the stakeholders' requirements.

Note the term *stakeholder* is used in the above discussion in place of the more common term *user*. This is to emphasize the fact that there are usually multiple categories of users of a system: owner and/or bill payer, developer, producer or manufacturer, tester, deployer, trainer, operator, user, victim, maintainer, sustainer, product improver, and decommissioner. Each stakeholder has a significantly different perspective of the system and the system's requirements. If one perspective is singled out as the only appropriate one, the developers of the system will miss key information, and the system will be viewed negatively or as a failure from the other perspectives.

The systems engineering process for creating a system design is decision rich. That is, the systems engineer is searching via a great deal of analysis and experience to find a very good (optimum is usually not possible to determine) solution that satisfies all of the mandatory requirements of the stakeholders and delivers as much performance as possible within the guidelines of cost and schedule.

This *search process* involves making many decisions about the system's physical character (or resources) and allocations of functions to resources that are usually only revisited if absolutely necessary. This search process occurs as the top-down onion-peeling process of systems engineering occurs. Figure 6.1 shows derived requirements at the component level (which may be several layers of the onion) and the CI (or bottom) level. Chapters 7 through 10 will describe this process of architecture development and creation of appropriate derived requirements, supported by analysis and judgment. To continue the story of the Boeing 777, Boeing created requirements for a major subsystem of the 777 — the engine. These derived requirements for the engine included the weight of the engine (derived from the weight of the empty aircraft), the thrust of the engine at specified conditions (derived from the liftable weight of the aircraft), the drag of the engine at specified conditions (derived from the drag of the aircraft), and the fuel consumption of the engine at specified conditions (derived from the fuel consumption of the aircraft).

A major impediment to this design process being successful is the overconstraint of the solution space by the stakeholders' requirements. The systems engineers job is to work with the stakeholders to define the stakeholders' requirements so as to make sure that there is significant design freedom within these requirements and that many feasible designs exist. Stakeholders and (all too often) engineers are willing to constrain the requirements space very tightly without fully understanding or appreciating the potential value of the design options that they are eliminating. The stakeholders' requirements process defined in this chapter takes explicit account of this need to have and define a large tradable region in design space for the systems engineers to search with quantitative techniques utilizing the priorities of the stakeholders.

Pragmatic Principle 1 [DeFoe, 1993] Know the Problem, the Customer, and the Consumer

1. *Become the "customer/consumer advocate/surrogate" throughout the development and fielding of the solution.*
2. *Begin with a validated customer (buyer) need — the problem.*
3. *State the problem in solution-independent terms.*
4. *Know the customer's (or buyer's) mission or business objectives.*
5. *Do not assume that the original statement of the problem is necessarily the best, or even the right one.*
6. *When confronted with the customer's need, consider what smaller objective(s) is/are key to satisfying the need, and from what larger purpose or mission the need drives; that is, find at the beginning the right level of problem to solve.*
7. *Determine customer priorities (performance, cost, schedule, risk, etc.).*
8. *Probe the customer for new product ideas, product problem/shortfalls, identification of problem fixes.*
9. *Work with the customer to identify the consumer (user) groups that will be affected by the system.*
10. *Use a systematic method for identifying the needs and solution preferences of each customer group.*
11. *Don't depend on written specifications and statements of work. Face-to-face sessions with the different customer/consumer groups are necessary.*
12. *State as much of each need in quantified terms as possible. However, important needs for which no accurate or quantified measure exists still must be explicitly addressed.*
13. *Clarify each need by identifying the power and limitations of current and projected technology relative to the customer's larger purpose, the environment, and ways of doing business.*

6.3 DEFINITIONS

Before discussing the process for developing stakeholders' requirements, the definitions presented in Chapter 2 are reviewed.

A *system* is a set of components (subsystems, segments) acting together to achieve a set of common objectives via the accomplishment of a set of tasks.

A *system task* or *function* is a set of functions that must be performed to achieve a specific objective.

A human-designed system is (a) a specially defined set of segments (hardware, software, physical entities, humans, facilities) acting as planned (b) via a

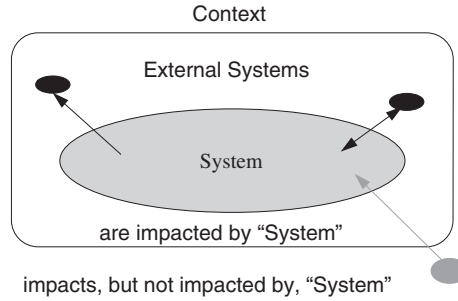

FIGURE 6.2 Depiction of the system, external systems, and context.

set of interfaces, which are designed to connect the components, (c) to achieve a common mission or fundamental objective (i.e., a set of specially defined objectives), (d) subject to a set of constraints, (e) through the accomplishment of a predetermined set of functions.

The external *systems* [Levis, 1993] of a system are a set of entities that interact with the system via the system's external interfaces. Note in Figure 6.2, the external systems can impact the system and the system does impact the external systems. The system's inputs may flow from these external systems or from the context, but all of the system's outputs flow to these external systems. The external systems, many or all of which may be legacy (existing) systems, play a major role in establishing the stakeholders' requirements.

The *context* [Levis, 1993] of a system is a set of entities that can impact the system but cannot be impacted by the system. The entities in the system's context are responsible for some of the system's requirements. See Figure 6.2. Wieringa [1995] uses the phrase "universe of discourse" to label the context and external systems that part of the world about which the system registers data and controls behavior.

6.4 STAKEHOLDERS' REQUIREMENTS DEVELOPMENT: DEFINING THE DESIGN PROBLEM

Developing a good and complete set of requirements is very difficult. First, we have to figure out what topics we should be writing requirements about. These topics for the system-level requirements should all be at the same level of granularity, a level of granularity that is consistent with the system-level and not the meta-system or subsystems. To facilitate defining these topics we will introduce the concepts of an operational concept, external systems diagram, and objectives hierarchy.

After we determine what the topics of the requirements conversation are going to be, we can start writing specific requirements. Now we have to determine what we want to say in that requirement. What is the threshold we are going to set for the minimum level of acceptable achievement? Here we will talk about prototyping, analysis, elicitation, and usability testing.

Next the requirements should be analyzed to determine that at least one feasible solution exists. A common problem is that we have defined thousands of requirements and together they are so constraining that there is no solution with enough performance at a low enough cost and a quick enough schedule. Often it is very difficult to determine that there is a feasible solution so this step is skipped. Typically the selected design proves to be insufficient for 5 to 20 requirements, meaning it was not a feasible solution. Late in the design process systems engineers are confronted with the problem of should we search for a new design or accept the fact the current design cannot meet all of the requirements.

The last step before approval should be defining qualification or test requirements that are appropriate for the level of requirements being defined. When defining system-level requirements these qualification requirements should address how will system-level verification and validation be done.

So the seven functions of this stakeholders' requirements development process are:

1. Develop operational concept
2. Define system boundary with external systems diagram
3. Develop system objectives hierarchy
4. Develop, analyze, and refine requirements (stakeholders' and system)
5. Ensure requirements feasibility
6. Define the qualification system requirements
7. Obtain approval of system documentation

These seven functions are shown in an IDEF0 diagram in Figure 6.3. This diagram is taken from the IDEF0 model of the process for engineering a system in Appendix B. To define this process fully, the first three functions must be defined in meaningful terms to justify their presence and provide explicit inputs to the fourth function. The last three functions are important but follow-on from the development of the StkhldrsRD. The resource that performs these functions is the systems engineering team; this resource is not shown in Figure 6.3 to improve the readability of the IDEF0 diagram.

The operational concept is prepared from the perspective of the stakeholders of the system and describes how these stakeholders expect the system to fit into their world that contains a number of external systems and has a certain context. The objectives of each stakeholder group are suggested here. The operational concept defines the system and external systems in very general terms (often as a block diagram) and establishes a use case diagram and the

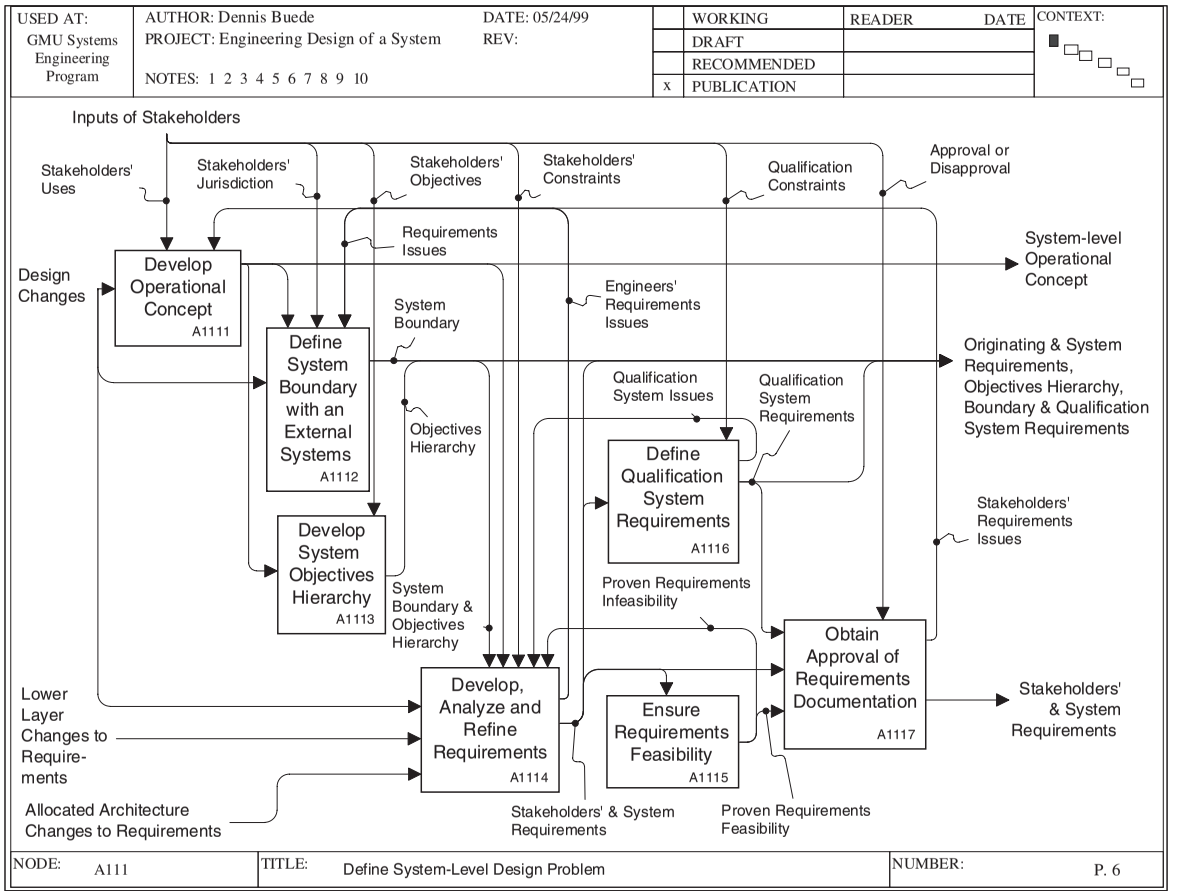

FIGURE 6.3 IDEF0 diagram of the system-level design process.

associated usage scenarios as sequence diagrams. These usage scenarios describe ways in which the stakeholders will use the system as well as interactions between the system and other systems. These scenarios define inputs to and outputs of the system. In addition, the operational concept includes the mission requirements for the system.

The second step, creating the external systems diagram, makes the boundaries between the system and external systems clear, leaving no doubt in anyone's mind where the system starts and stops. The development of this diagram and the explication of the system's boundaries are nearly always harder than most people expect. As part of defining the system's boundaries, all of the inputs to and outputs of the system are established, as well as the external system or context with which each input and output is associated.

The third step clarifies the objectives of the stakeholder groups and formulates a coherent set of objectives for the system. Again, the output of this step looks like it could have been created in a few hours, but generally takes days if not weeks. Each objective is part of the value system of one or more stakeholders for determining their satisfaction with the system. Naturally these objectives conflict with each other in the sense that gaining value on one objective (e.g., availability) means it will be necessary to give up value on another objective (e.g., cost).

The creation of the stakeholders' requirements, followed by the translation of these requirements into system requirements, is the fourth step. The stakeholders' requirements are created by an analysis of the operational concept for system functions, an exhaustive examination of the system's inputs and outputs, the specification of interfaces of the external systems with which the system must interact, a thorough examination of the system's context and operational concept for system-wide and technology constraints, a detailed discussion with the stakeholders to understand their willingness to trade-off a wide range of non-mandatory but desirable system features, and the complete specification of qualification requirements needed to verify and validate the system's capabilities from the stakeholders perspectives. Often a simulation model that depicts some or all of the interaction between the system and one or more other external systems is developed. These simulation models often address timing issues, specific performance issues, reliability or availability, safety and security, or quality of inputs and outputs. Cost analyses of a system should be done with the context in which the system is going to operate in mind. An important tool used during requirements development is prototyping, the development of replicas of the parts of the system. For user interfaces this prototyping is particularly important because users often do not know what is possible with new technology or how they might use this new technology effectively. For prototyping of user interfaces to be effective some form of usability testing is commonly used to determine how the users function with the prototype.

Before proceeding too far into the design process, these requirements must be examined to ensure that a feasible design exists that meets the requirements.

For example, building a supersonic transport aircraft that has a production cost of $1000 is not possible. While this simple, exaggerated example illustrates the problem, in practice the development of hundreds, or even thousands, of requirements makes the test for feasibility quite difficult.

The sixth step is the development of requirements for the qualification system needed to verify and validate the resulting system. This involves the development of input/output requirements for the qualification system, as well as system-wide requirements. Trade-off requirements are also needed for the qualification system. Finally, the qualification system must also be qualified.

Finally, the stakeholders must approve the requirements documents. This approval process works best when the stakeholders are actively involved in and understand the previous steps.

Before defining and discussing requirements, noting that requirements must be developed for each phase of the system's life-cycle is important. The life-cycle phases used in this book are:

1. Development (design and integration)
2. Manufacturing or production
3. Deployment
4. Training
5. Operations, maintenance, and support
6. Refinement
7. Retirement

There is a strong correlation between the stakeholders and the life-cycle phases. These seven functions should be applied to each stakeholder group and phase of the system's life cycle. Note that some of these phases may not be relevant for some systems. Most of the discussion from here on out will focus on the operations, maintenance, and support phase, but keep in mind that all phases of the life cycle should be addressed. Table 6.1 discusses who is involved in this requirements generation process and what their roles are.

6.5 REQUIREMENTS CATEGORIES

Many authors have categorized requirements. Here are some of the often-discussed categories:

1. *Specification Level Stakeholders', Derived, Implied and Emergent*: Stakeholders' requirements, derived from operational needs, are those top-level statements defined in language that is understandable to the stakeholders, leaving substantial room for design flexibility. Stakeholders' requirements should define the essence of the stakeholders' needs sufficiently clearly for

TABLE 6.1 Roles and Responsibilities during Requirements Generation

Who has the right to have a stakeholders' requirement?	Any individual/organization with a need involved in the development (design and qualification), production, deployment, training, operation, maintenance, support, refinement, decommissioning of **and payment** for the system.
What does one call a requirer?	Customer or stakeholder
Who must respond to the requirer(s) having a requirement and how?	System's **requirements team**, a collection of *stakeholders* and *systems engineers*. Response is acceptance, request for clarification, or rejection.
By what criteria does the Systems Requirements Team respond?	This team establishes the external systems diagram and fundamental objectives hierarchy of the system, and then determines if the requirement fits within the scope of the system's boundary and fundamental objective. Stakeholders' requirements also have to be assessed for the proper level of abstraction. A requirement should not be too strategic (mission-oriented) or means (or solution) oriented.
How does one know that the requirement is "right?"	There is no right or wrong, only acceptable or unacceptable at this time. Over time, some of the stakeholders' requirements will change.
How are these requirements conveyed to the people who get involved once a requirer has enunciated a requirement?	The system's requirements team documents the collection of stakeholders' requirements. This stakeholders' requirements document (StkhldrsRD) is distributed to the stakeholders and systems engineers. Included in this document is a discussion of the operational concept of the system and the external systems and context associated with the system, that is, how each stakeholder expects to interact with the system. By reviewing the stakeholers' requirements document each stakeholder can see how the requirement s/he suggested fits into the envisioned operation of the system, and can judge whether this vision makes sense from her/his perspective.
What does the Systems Requirements Team do next?	The system's stakeholders' requirements team remains active throughout the

(*Continued*)

TABLE 6.1. Continued

	system's life cycle. During design there will be many occasions when the system's stakeholders' requirements must be reviewed and modified. These occasions will diminish in frequency once the system is deployed, but the requirements process is still critical as requirements changes and system modifications are envisioned, agreed to, developed and fielded.

the stakeholders to be completely satisfied with whatever system results from the systems engineering process. Derived requirements are those requirements defined by the systems engineering team in engineering terms during the design process. Derived requirements are needed to complete the design to sufficient detail for the specification to be delivered to the design teams responsible for the physical configuration items of the system. *Implied* requirements are those requirements not specifically identified in the StkhldrsRD but that can be inferred based upon information in the StkhldrsRD. *Emergent* requirements are those requirements that are not even hinted at in the StkhldrsRD but whose presence is made known by stakeholders later in the systems engineering process. These last two sets of requirements are to be avoided if possible by a sound and systematic stakeholders' requirements development process.

2. *Performance Requirements Versus Constraints. Performance requirements*: define on some index that establishes a range of acceptable performance from a minimum acceptable threshold to a design goal. Constraints simply rule out certain possible designs; for example, the system must be painted a specific shade of green. A performance requirement defines a desired direction of performance; for an elevator system (which is used throughout this book as an example), a performance requirement might be to "minimize passengers' waiting time during peak periods." For any performance requirement there must also be a minimum acceptable performance constraint or threshold associated with the index, beyond which designs with such poor performance are not feasible (e.g., average passengers' waiting time during peak periods shall be less than 35 seconds). Often there is also a maximum threshold or goal on the performance index that states the stakeholders do not noticeably value performance beyond this point (e.g., average passengers' waiting time during peak periods need not be less than 27 seconds).
3. *Application — System Versus Program*: System requirements relate to characteristics of the system's performance (in the broadest sense). Program

requirements relate to the first life-cycle phase of the systems engineering process and usually address the treatment of the cost and schedule for this phase. Program requirements relate either to the programmatic tasks that must be performed, programmatic trade offs among cost and schedule, and programmatic products associated with the systems engineering process (e.g., the Up & Down Elevator Corporation shall own full rights to the design data of the elevator).

4. *Functional, Interface, or System-wide Requirements*: Functional requirements relate to specific functions (at any level of abstraction) that the system must perform while transforming inputs into outputs. As a result, a functional requirement is a requirement that can be associated with one or more of the system's outputs. Interface requirements are usually constraints that define the reception of inputs and transmission of outputs between the system and the system's environment. System-wide requirements (often called "-ilities") are characteristics of the entire system; examples include availability, reliability, maintainability, durability, supportability, safety, trainability, testability, extensibility (growth potential), and affordability (e.g., operating cost).

6.6 REQUIREMENTS PARTITION

There is great value in having a structure for various types of requirements. If the requirements are listed in random order in a requirements document, it is nearly impossible to be sure that a given requirement is not addressed multiple times in that single requirements document. It is also difficult to find a specific requirement in a large document. There are other benefits of a requirements structure, especially if the structure is a partition. A partition is a structure that has subcategories that are mutually exclusive, meaning a requirement can only be put in one category. A partition also needs to be exhaustive, meaning every requirement has some category that is appropriate for it. By creating such a partition, it is easy to review the partition to ensure that there as many requirements in that category as expected and every requirement in the category is appropriate for that category.

The partition that is introduced here has both a vertical spectrum and a horizontal spectrum. The vertical spectrum was introduced in Figure 6.1, which shows two vertical levels of requirements written for the stakeholders and three or more levels of derived requirements written for the engineers. The horizontal spectrum addresses the life cycle as well as categories of requirements within each phase of the life cycle. The life-cycle steps or phases include development, production, operations, etc.; recall Figure 6.1. The categories of requirements within each phase of the life cycle are discussed next.

Wymore [1993] identifies six types of system design requirements: input/output, technology and system-wide, performance trade-off, cost trade-off,

cost–performance trade-off, and test. These six types of requirements are condensed into four categories: input/output, technology and system-wide, trade-off, and qualification (test). From a concurrent engineering perspective each requirements category should be used to address the relevant system (e.g., development system, manufacturing system) in each phase of the system's life cycle (development, production, deployment, training, operation and maintenance, refinement and retirement). Table 6.2 provides examples of various types of requirements; these examples have been collected from a wide variety of sources.

1. *Input/output requirements*: include sets of acceptable inputs and outputs, trajectories of inputs to and outputs from the system, interface constraints imposed by the external systems, and eligibility functions that match system inputs with system outputs for the life-cycle phase of interest. Clearly there are a number of requirements in this category during the operations phase of the life cycle. However, the system may have inputs and outputs in all portions of the system's life cycle (e.g., training stimulations, standardized internal interfaces for product improvement); if so, the requirements for these activities would be found in this category in the appropriate life-cycle phase. This category is partitioned into four subsets: (a) inputs, (b) outputs, (c) external interface constraints, and (d) functional requirements. Input requirements state what inputs the system must receive and any performance or constraint aspects of each. Output requirements state what outputs the system must produce and any performance aspects; Table 6.2 provides an extensive list of possible performance issues for the outputs of any system, segmented by quality, quantity, and timeliness. External interface requirements deal with limitations placed upon the receipt of inputs and transmission of outputs by the interfaces of the external systems; see Table 6.2. Functional requirements can be endless unless organized; the functional requirements proposed here are the two to seven functions that are the first-level decomposition of the system's function.

The very strong position being taken here is that the input and output requirements are the key to defining the needs of the stakeholders in terms that they can understand. Stakeholders in each phase of the system's life cycle can relate to quantity, quality, and timing aspects of the outputs delivered by the system under question and the ability to deal with quantity, quality, and timing of inputs. The engineers of the system develop the system's functions during the design process. This development of a functional architecture (see Chapter 7) is a very valuable means for dealing with the complexity of the engineering problem. But the stakeholders should not care a whit about the functions being performed by the system as long as they are happy with the characteristics of the inputs being consumed and the outputs being produced by the system. The concept of having a major section of requirements devoted to the functions of the system is misguided and guaranteed not to elicit the needs of the stakeholders.

TABLE 6.2 Exemplary Requirement Dimensions

Requirements Category	Exemplary Requirement Dimensions
Input or Output Performance	Quality of an output Accuracy (or precision) Correctness (or confidence, error rate) Security (or perishability, survivability) Quantity of an output Intensity, Size, or Distance Number per unit time (throughput, velocity) Coverage (area or volume served by outputs) Timing of outputs Response time (timeliness, time to create an output) Update frequency Availability
Undesired or Unexpected Inputs	Unexpected or undesired inputs and appropriate response Bounds on expected inputs and appropriate response
Interface Constraint	Required format of an input or output as defined by the interface Timing constraint associated with an interface Physical form or fit of an interface
Suitability or Quality Issues of the System	Usability Weight of the system Form (volume) and fit (dimensions) of the system Survivability of the system Availability, reliability, maintainability of the system Supportability of the system Safety of the system Security Trainability of the system Testability of the system Extensibility (expected changes/growth potential) of the system
Costs for Various Life Cycle Phases	Affordability (or operating and maintenance cost) of the system Development cost Production cost (manufacturability) of the system Deployment and training costs of the system Decommissioning cost of the system
Schedule for Various Life Cycle Phases	Development period Manufacturing time for each unit Training time to reach proficiency by category of user Deployment period Durability (or operational life) of the system

2. *Technology and system-wide requirements*: consist of constraints and performance index thresholds (e.g., the length of the operational life for the system, the cost of the system in various life-cycle phases, and the system's availability) that are placed upon the physical resources of the system. Many of the requirements from each phase of the system's life cycle are found in this category because these requirements specifically relate to the physical manifestation of the system. This category can be partitioned into four subsets: (a) technology, (b) suitability and quality issues, (c) cost for the relevant system (e.g., development cost, operational cost), and (d) schedule for the relevant life-cycle phase (e.g., development time period, operational life of the system).
3. *Trade-off requirements*: are algorithms for comparing any two alternate designs on the aggregation of cost and performance objectives. These algorithms can be divided into (a) performance trade offs, (b) cost trade offs, and (c) cost–performance trade offs. The performance trade-off algorithm defines how the relative performance of any two alternate designs can be compared in terms of the system's performance objectives. These performance objectives are defined within the input/output and non-cost system-wide requirements. The performance trade-off algorithm specifically defines how the performance parameters are to be compared to each other. The cost trade-off algorithm defines how the relative cost of any two alternate designs can be compared across all cost parameters (life-cycle phases) of interest to the stakeholders. Note dollars spent at different times may not be comparable by present value computations when there are different bill payers at different times. Finally, the cost–performance trade offs define how performance objectives should be traded with cost objectives.

These trade-off algorithms could be based upon many different mathematical logics; indeed many have been proposed. The strong position taken in this book is that these trade-off algorithms must be based upon the value preferences of the stakeholders. Decision analysis provides a normative basis for these preference judgments and algorithms, as described in detail in Chapter 13. For applications of these decision analysis techniques (value curves and swing weights) see Buede and Bresnick [2007], Buede and Choisser [1992], Daniels et al. [2001], Ross et al. [2004], Thurston and Carnahan [1993], Walton and Hastings [2004].

The ideal approach for quantifying the trade-off preferences of the stakeholders would be to obtain these preferences as statements of "willingness-to-pay" (in terms of money for development effort) for enhanced performance and decreased cost in each of the other life-cycle phases. To make these statements of "willingness-to-pay" operationally meaningful, the appropriate contractual arrangements must be established that would permit the transfer of payments based upon the stated payment preferences. In addition, a warranty system must be established that requires the developers to stand behind their developmental phase claims of performance attainment during the remaining phases of the system's life cycle. For example, if a

performance claim made during the development phase is not achieved during the operational phase, the developer would have to make a warranty payment to the stakeholders. Although this entire approach is known and obviously will work, the approach has never been used to the author's knowledge. In fact, users are quite cynical about the performance claims made by developers during the development phase.

4. *System qualification requirements*: address the needs to qualify the system as being designed right, the right system, and an acceptable system. There are four primary elements:
    a. Observance: state which qualification data for each input/output and system-wide requirement will be obtained by (i) demonstration, (ii) analysis and simulation, (iii) inspection, or (iv) instrumented test.
    b. Verification Plan: state how the qualification data will be used to determine that the real system conforms to the design that was developed.
    c. Validation Plan: state how the qualification data will be used to determine that the real system complies with the stakeholders' performance, cost, and trade-off requirements.
    d. Acceptance Plan: state how the qualification data will be used to determine that the real system is acceptable to the stakeholders.

Note the qualification requirements associated with the first objective define the basis for the requirements for the suite of qualification systems (e.g., simulations, instrumented test equipment) needed for the system under development. Having technology/system-wide requirements that limit the flexibility to develop new test equipment is common.

This requirements' partition provides a solid basis and set of guidelines for guaranteeing that the system's requirements are complete, consistent, unique, comparable, and modifiable. (These terms will be defined a little later.) Success is not certain with this basis and guidelines but is greatly enhanced over current industry practice.

Figure 6.4 traces the origins of the performance requirements to the objectives hierarchy by showing that the objectives hierarchy defines the performance parameter requiring nonpoint requirements. These performance parameters can fall within the categories of input, output, "-ilities," cost, and schedule requirements. The thresholds and goals for these tradable requirements are defined as part of the input, output, "-ilities," cost, and schedule requirements. The algorithms that define the tradable space over these performance parameters are documented in the performance, cost, and cost–performance trade-off requirements. The performance, cost, and cost–performance trade-off requirements combine to define the iso-value lines in the tradable space; these iso-value lines will be the basis for all design trade offs.

If every set of requirements contained the information defined by Wymore [*1993*], *there would be far fewer problems in system development efforts*. Very few

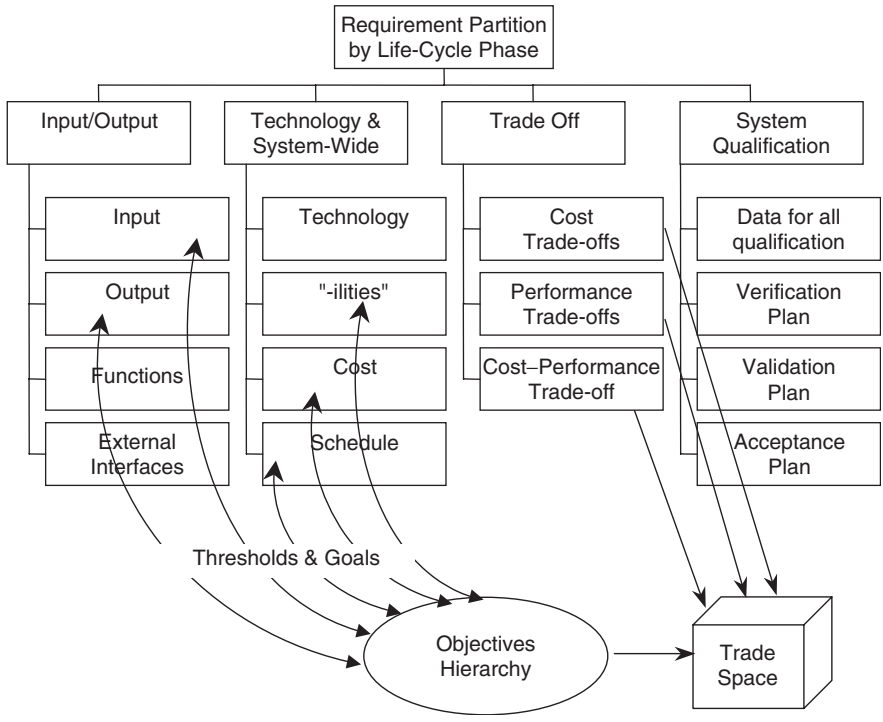

FIGURE 6.4 Objectives hierarchy, requirements partition, and trade space.

requirements documents contain performance, cost, and cost–performance trade-off requirements as defined by Wymore. These elements should be defined in the stakeholders' requirements document from the stakeholders' perspective; otherwise the systems engineers must guess at the ultimate trade offs of the stakeholders; the ability of engineers to do a complete and effective job of guessing iso-value trade offs is questionable at best.

6.7 STAKEHOLDERS' REQUIREMENTS DOCUMENT (StkhldrsRD)

The format for an StkhldrsRD (Fig. 6.5) should include sections for a brief overview of the system, references to relevant documents from which the stakeholders' requirements have been traced, and the requirements. The requirements should be organized by life-cycle phase. Within each life-cycle phase requirements from the four segments of the above taxonomy should be developed. The life-cycle phases are being called out explicitly to highlight the criticality of the concurrent engineering nature of the design problem.

Stakeholders' Requirements Document

1.0 System Overview
2.0 Applicable Documents
3.0 Requirements
- 3.1 Development Phase (Programmatic) Requirements
  - 3.1.1 Input/Output Requirements for Development
  - ...
  - 3.1.4 Qualification Requirement for Development
- 3.2 Manufacturing Phase Requirements
- ...
- 3.3 Deployment Phase Requirements
- ...
- 3.4 Training Phase (if present) Requirements
- ...
- 3.5 Operational Phase Requirements
  - 3.5.1 Input/Output Requirements for Operations
    - 3.5.1.1 Input Requirements for Operations
    - 3.5.1.2 Output Requirements for Operations
    - 3.5.1.3 External Interface Requirements for Operations
    - 3.5.1.4 Functional Requirements for Operations
  - 3.5.2 System-wide/Technology Requirements for Operations
  - 3.5.3 Trade-off Requirement for Operations
  - 3.5.4 Qualification Requirement for Operations
- 3.6 System Improvement/Upgrade Phase Requirements
- ...
- 3.7 Retirement Phase Requirements
- ...
- 3.8 Overall Trade-Off Requirement

Appendix A. Operational Concepts by Phase
Appendix B. External System Diagrams by Phase

FIGURE 6.5 Outline of stakeholders' requirements document.

The designs of the life-cycle systems needed to obtain an operational system are not that straightforward. Requirements in one phase of the life cycle will often have a major impact on the design of a system in another phase. For example, a requirement that the manufacturing system be operational by a specified date precludes many interesting designs of the operational system. This interaction of requirements and design options across life-cycle phases is a major contributing factor to failure in the real world; in addition, this interaction makes the concept of formulating the design problem as an optimization problem nonsensical to practitioners. Rather, the segregation of requirements by life-cycle phase is meant to aid in attaining the desired attributes (e.g., complete, consistent) of requirements discussed in Table 6.3 of the next section.

Given the organization of the StkhldrsRD shown in Figure 6.5, an overall tradeoff requirement (Section 3.8 of the StkhldrsRD) that addresses

comparisons across life-cycle phases is needed to enable coherent evaluations of design options.

6.8 CHARACTERISTICS OF SOUND REQUIREMENTS

A number of authors [Frantz, 1993; Davis, 1993; Mar, 1994] have developed various numbers of attributes for requirements. The literature is not in total agreement about the meaning of these attributes. Table 6.3 is the result of a detailed examination of the literature. The characteristics are divided into those that are related to individual requirements and those relevant to groups of requirements.

In any systems engineering effort, as many correct requirements must be developed as possible; these correct requirements should be verifiable. In addition, as many incorrect requirements should he eliminated as possible. In summary, the requirements document should contain a complete, consistent, comparable, design independent, modifiable, and attainable statement of the design problem.

6.9 WRITING REQUIREMENTS

Certain procedures have been developed [Grady, 1993; Hooks, 1994] for writing requirements. These procedures guide requirements writers toward the achievement of the above attributes. First, a set of terms has been developed. Specifically, a statement of a requirement includes the use of the word "shall" to indicate the limiting nature of a requirement; statements of fact use "will"; and goals use "should." The requirements statement shall include a subject (the relevant life-cycle system), the word "shall," a relation statement (e.g., less than or equal to), and the minimum acceptable threshold with units. Data clarifying the terms in the requirement can also be added. Examples of appropriate grammar are:

> The system shall provide the customer a receipt at the end of each transaction. The receipt shall contain Bank Name, Account Number, Date and Time of Day, Type of Transaction, Account Balance at the end of the Transaction, and Automatic Teller Location Code Number.
> The system shall stop the flow of liquid hydrogen in 0.5 seconds or less. The liquid stopping time is measured from the time the control signal for stopping is received until the flow through reaches zero.

It is important to avoid compound predicates and negative predicates:

> The system shall fit …, weigh …, cost … (this causes traceability problems).
> The system shall not … (attempt to turn this into a positive statement of what the system shall do).

TABLE 6.3 Attributes of Requirements

Individual Requirement Attributes

1) unambiguous – every requirement has only one interpretation
2) understandable – the interpretation of each requirement is clear to those selected to review the requirement
3) correct – the requirement states something required of the system, as judged by the stakeholders
4) concise – no unnecessary information is included in the requirement
5) traced – each stakeholders' requirement is traced to some document or statement of the stakeholders
6) traceable – each derived requirement must be traceable to a higher level requirement via some unique name or number
7) design independent – each requirement does not specify a particular solution or a portion of a particular solution
8) verifiable – a finite, cost-effective process can be defined to check that the requirement has been attained

Attributes of the Set of Requirements

9) unique – requirement(s) is(are) not overlapping or redundant with other requirements
10) complete – (a) everything the system is required to do throughout the system's life cycle is included, (b) responses to all possible (realizable) inputs throughout the system's life cycle are defined, (c) the document is defined clearly and self-contained, and (d) there are no "to be defined" (TBD) or to be reviewed (TBR) statements; completeness is a desired property but cannot be proven at the time of requirements development, or perhaps ever
11) consistent – (a) internal – no two subsets of requirements conflict and (b) external – no subset of requirements conflicts with external documents from which the requirements are traced
12) comparable – the relative necessity of the requirements is included
13) modifiable – changes to the requirements can be made easily, consistently (free of redundancy) and completely
14) attainable – solutions exist within performance, cost and schedule constraints
15) organized – grouped according to a hierarchical set of concepts, such as life cycle and categories.

Similarly, the "and/or" colloquialism is inappropriate because "and/or" provides the designer with a choice; be specific about whether you mean "and" or "or." The requirement should not start with an "If …"statement. Conditions under which the requirement is true should be placed at the end of the requirement.

Ambiguous terms are a plague on requirements. Common verbs that are not specific enough include "maximize" and "minimize" because the system is

seldom operating in an environment in which optimization is possible. "Accommodate" is another example of a vague verb. Adjectives are a major source of ambiguity; examples include "adaptable," "adequate," "easy," "flexible," "rapid," "robust," "sufficient," "supportable," and "user-friendly."

Requirements should start with the system of interest, be followed by a verb phrase starting with the word "shall", be followed by an object that describes an input, output, etc., and end (if necessary) with conditions under which the previous was true. Examples include:

The development system shall receive inputs from stakeholders. (Input requirement)

The manufacturing system shall have a scrap page rate that is less than $x\%$. The design goal is $0.7x\%$. (Output requirement)

The deployment system shall accept boxes of x ft^3 or less. The design goal is $0.5 \times ft^3$. (Input requirement)

The training system shall complete training in x hours per student or less. The design goal is $0.9x$ hours. (Output requirement)

The operational system shall have an operational life of x years or more. The design goal is $2x$ years. (System-wide schedule requirement)

The refinement system shall be compatible with the following new technologies (x, y, z) for the central processing unit. (Input requirement)

The retirement system shall retire units for less than $\$x$ each. (System-wide cost requirement)

6.10 OPERATIONAL CONCEPT

An *operational concept* [Lano, 1990a] is a vision for what the system is (in general terms), a statement of mission requirements, and a description of how the system will be used. Hooks and Farry [2001] describes the operational concept as a "day in the life of your product." This operational concept is an opportunity to create a vision that is shared among all of the stakeholders for the really major interactions of people and things with the system of interest. The shared vision is from the perspective of the system's stakeholders, addressing how the system will be developed, produced, deployed, trained, operated and maintained, refined, and retired to overcome some operational problem and achieve the stakeholders' operational needs and objectives. The development of the operational concept serves the purpose of obtaining consensus in the written language of the stakeholders about what needs the system will satisfy and the ways in which the system will be used. Remember that there is a system for each phase of the system's life cycle and that an operational concept is needed for each of the systems. By describing how the system will be used, the operational concept is providing substantial

(but incomplete) information about the system's interaction with other systems and the context of the system.

Figure 6.6 shows the three primary choices that were considered by the National Aeronautics and Space Administration (NASA) engineers in determining an operational concept for landing on the moon during the 1960s [Brooks et al., 1979; Murray and Cox, 1989]. The NASA engineers called these concepts modes and started out favoring the direct ascent from Earth to moon and back to Earth.

However, calculations concerning the thrust required for this concept quickly proved that the concept was infeasible. As a result the second and third concepts (Earth rendezvous and lunar rendezvous) were defined and explored in detail. Werner von Braun had previously developed the concept of staged rockets for lifting payloads into Earth orbit; with staged rockets the weight that is no longer relevant can be shed. The same concept applied to Earth and lunar rendezvous. Many teams conducted calculations and simulations of these two concepts over several years, focusing primarily on cost (using energy as a surrogate) and safety. The final results estimated that the lunar orbit rendezvous concept was almost $1.5 billion cheaper and had a 6- to 8-month shorter timeline for landing on the moon. There was some controversy at the end about which was safest; many engineers felt they were about equal with respect to safety, each having different strengths and weaknesses.

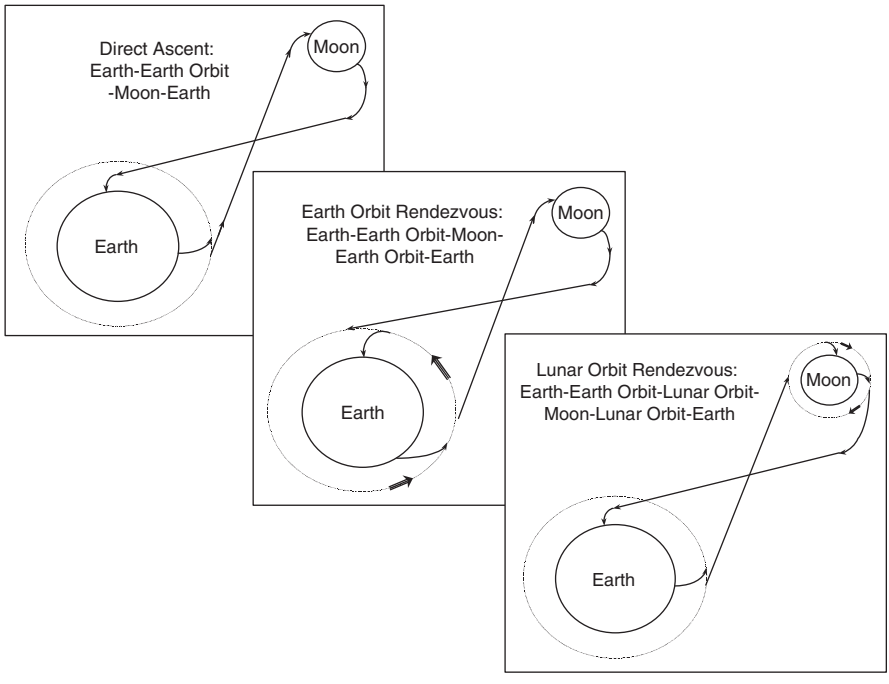

FIGURE 6.6 Alternate operational concepts for Apollo's moon landing.

The operational concept includes a collection of scenarios as described in a use case diagram (see Fig. 3.1). One or more scenarios are needed for each group of stakeholders in each relevant phase of the system's life cycle. The use case diagram is used to provide a "big picture" of how the individual scenarios relate to each other in defining how the system is to be employed. Each scenario addresses one way that a particular stakeholder(s) will want to use, deploy, and fix the system; the scenario defines how the system will respond to inputs from other systems in order to produce a desired output. Included in each scenario are the relevant inputs to and outputs from the system and the other systems that are responsible for those inputs and outputs. The scenario should not describe how the system is processing inputs to produce outputs. Rather, each scenario should focus on the exchange of inputs and outputs by the system with other systems. It is critical that this shared vision be consistent with the collection of scenarios comprising the operational concept.

Hunger [1995] uses the phrase "mission analysis" for the development of the operational concept. The collection of scenarios in the operational concept includes sortie missions (or scenarios) and life missions, both from the perspective of the stakeholders. Sortie missions are scenarios that describe how the system will be used during the operational phase, capturing the reasons the system has for existing. The life missions address the nonoperational, life-cycle aspects of the system, resulting in scenarios for each life-cycle phase and some that cross life-cycle phases. Hunger has suggested using time lines to better define these system scenarios (or sorties as he calls them).

The mission requirements of the system are the key statements of the needs of the stakeholders in the context of the stakeholders and other systems with which the system interoperates. These mission requirements are stated in terms of the measures relevant to enabling the stakeholders to meet some missions important to the stakeholders. For example, a major mission requirement for the Apollo moon landing was "bringing the astronauts home alive." Within the elevator case study the output requirements were divided into average wait for service and average transit time. The mission requirement would be average time from request for service until service was completed.

In software engineering, Jacobson [1992, 1995] proposed the creation of use cases to capture the interactions between people (users) of the software system, as well as among other systems; users and external systems are called actors. The concept of use cases was embraced so thoroughly by many software engineers that Cockburn [1997a,b] documents 18 different definitions of a use case. These definitions of use cases vary along four dimensions: purpose, contents, plurality, and structure. Cockburn [1997a,b] adopts the same definition for each of the four dimensions that Jacobson put forth. The purpose of use cases is to support the development of requirements; the contents are consistent prose; the plurality is that each use case contains multiple scenarios (as defined in this book for the operational concept); and the structure of the use cases is semiformal. A use case is developed around a specific goal; goal is synonymous with desired output of the system. The use case contains one main

scenario and as many variations around that scenario as are meaningful. For our elevator system, variations may relate to the types of people using the elevator system, for example, blind people, deaf people, small children, people in wheelchairs. So far a collection of use cases is very consistent with a collection of scenarios as defined for the operational concept. However, a number of authors [Jacobson, 1995; Eriksson and Penker, 1998] illustrate the use case with statements of functions that the system and actors are performing, rather than the flow of information and physical entities between the system and the actors. As stressed so far in this chapter, the focus during the development should be on defining requirements related to inputs and outputs of the system and not on the functions of the system and functional requirements. There is quite a bit of confusion and sloppiness in discussions of use cases on this issue; several of the authors [Cockburn, 1997a,b; Eriksson and Penker, 1998] are really clear that the system should be treated as a black box with no visibility into functions, yet the functions show up in the discussion and diagrams documenting the use cases [see Jacobson, 1995; Eriksson and Penker, 1998].

The emphasis in this book has been on defining all aspects of the life-cycle system. Consistent with Hunger's [1995] concept for sortie and life missions, the engineers for a system should develop scenarios for the system of interest in every phase of the life cycle. There should be scenarios and mission requirements for the development, manufacturing, training, deployment, refinement, and retirement phases unless one or more of these phases is not relevant.

To generate these scenarios, start with the key stakeholder, the operator/user, and generate a number of simple scenarios. Then scenario generation is expanded to other stakeholders while staying simple. Finally, complexity is added to all scenarios for each stakeholder, explicitly addressing atypical weather situations, failure modes of external systems that are relevant, and identifying key failure modes, constraints, standards, and external system interfaces that the system should address *in every phase of the life cycle*. In all scenarios the focus should be on *what* the stakeholders and external systems do and *not on how* the systems accomplish their tasks. The system of interest should be viewed as a black box; that is, the system's internals are blacked out, leaving only the inputs and outputs to the system. Table 6.4 shows sample operational concept scenarios for an elevator.

There are some common operating scenarios for nearly every system:

- Initialization of the system
- Normal steady state operation in standard operating modes of the system for all possible contexts (environments) in which the system may be placed (e.g., extreme cold, outer space)
- Extremes of operations due to high and low peaks of the external systems in each standard operating mode in each context
- Standard maintenance modes of the system

TABLE 6.4 Sample Operational Concept Scenarios for an Elevator

1) Passengers (including mobility, visually and hearing challenged) request up service, receive feedback that their request was accepted, receive input that the elevator car is approaching and then that an entry opportunity is available, enter elevator car, request floor, receive feedback that their request was accepted, receive feedback that door is closing, receive feedback about what floor at which elevator is stopping, receive feedback that an exit opportunity is available, and exit elevator with no physical impediments.
2) Passengers are receiving transportation in the elevator system when a fire breaks out in the building; building alarm system sends signal to elevator system to stop elevator cars at the nearest floor, provide exit opportunity, and sound a fire alarm. Passengers leave elevator cars. Elevator cars are reactivated by special access available to maintenance personnel after the building is re-opened.
3) Passengers are entering (exiting) an elevator car when doors start to shut; passengers can stop doors from shutting and continue to enter (exit).
4) Elevator car stops functioning. Passengers in the elevator car push an emergency alarm that notifies building personnel to come and help them. Passengers use a phone system in the elevator car to call a centralized service center and report the problem to the people that answer. Elevator maintenance personnel arrive and create an exit opportunity.
5) Too many passengers enter an elevator car and the weight of passengers in the elevator car exceeds a preset safety limit; the elevator car signals a capacity problem and provides prolonged exit opportunity until some passengers exit the car.
6) Maintain a comfortable environment in the elevator by sensing the temperature in the elevator car that is based upon heat loss/gain of the passengers and the building and then supplying the necessary heat loss/gain to keep the passengers comfortable.
7) A maintenance person needs to repair an individual car; the maintenance person places the elevator system in "partial maintenance" mode so that the other cars can continue to pick up passengers while the car(s) in question is (are) being diagnosed, repaired, and tested. After completion the maintenance person places the elevator system in "full operation" mode.
8) Electric power is transferred to the elevator from the building.

- Standard resupply modes of the system
- Reaction to failure modes of other systems
- Failure modes due to internal problems, providing as much graceful degradation of the meta-system as possible
- Shutdown of the system
- Termination (phase out) of the system

The total number of scenarios for a common (relatively simple) system would be 25 to 50.

The SysML modeling technique called a *sequence diagram* (formerly called an input/output trace in the first edition of this book) can be used to make

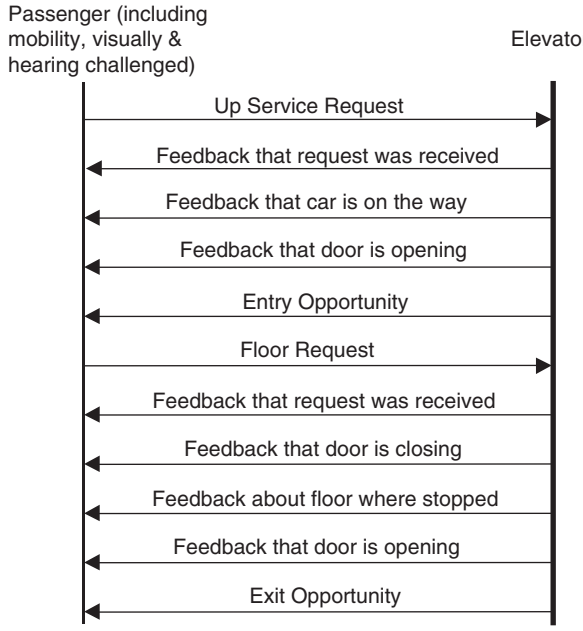

FIGURE 6.7 Sequence diagram of first elevator scenario.

the description of each scenario as explicit as possible. A sequence diagram (see Fig. 6.7) has a time line associated with each major actor (our system and other systems) in the scenario. The systems involved are listed across the top of the diagram with the time lines running vertically down the page under each of the systems. Time moves from top to bottom in an input/output trace; the system of concern is highlighted with a bold label and heavier line. Interactions involving the movement of data, horizontal arcs from the originating system to the receiving system, designate energy or matter among systems. A label is shown just above each arc to describe the data or item being conveyed. Double-headed arcs are permissible to represent dialog in a compact manner. Having two or more arcs in quick succession is also common to illustrate that the same item is being transmitted from one system to multiple systems or multiple systems are potentially transmitting the same item to one system. Figure 6.7 shows the first of these scenarios documented as an input/output trace diagram. See the elevator case study on the author's web site for more examples.

The purpose of these sequence diagrams is to be more explicit than written text can be about the systems involved with a specific focus on the time-based interaction of systems and the transmission of data and items. Compare the sequence diagram in Figure 6.7 to the first scenario in Table 6.4. These sequence diagrams are not meant to be exact representations of dynamic interaction. An interval time scale is not being represented; rather time is ordinal — any arc that

is above another happens earlier, but there is no indication as to how large the time interval is.

The *shared vision, mission requirements*, and *the use case diagram with sequence diagrams for the scenarios* define the system's mission and provide the first hints as to the boundary of the system. The external systems are defined in the scenarios, also defining the inputs and outputs of the system. The system's inputs and outputs cross this boundary, defining the input/output requirements of the system and the external interfaces. The mission requirements suggest the fundamental objectives (objectives hierarchy of the stakeholders). This objectives hierarchy becomes the basis of the system's performance requirements. Finally, the first-level decomposition of the system's function can be suggested by examining the operational concept. Thus the operational concept also leads to the functional requirements.

Recall that multiple systems are being developed concurrently, one for each phase of the life cycle and a qualification system for each of those systems. Each of these systems should have an operational concept.

The American Institute of Aeronautics and Astronautics (AIAA) and the Institute of Electrical and Electronics Engineers (IEEE) have standards documents for the Concept of Operations and Operational Concept for the interested reader.

6.11 EXTERNAL SYSTEMS DIAGRAM

The single, largest issue in defining a new system is where to draw the system's boundaries; see Figure 6.2. Everything within the boundaries of the system is open to change and subject to the requirements, and nothing outside of the boundaries can be changed, leading to many of the system's constraint requirements. The *external systems' diagram* is the model of the interaction of the system with other (external) systems in the relevant contexts, thus providing a definition of the system's boundary in terms of the system's inputs and outputs.

Who is responsible for drawing these boundaries? All of the stakeholders have a say in drawing these boundaries. However, there are substantial cost and schedule implications so the procurer of the system typically has a major input. Nonetheless, all of the stakeholders should be prepared to discuss the impact upon them of various boundary-drawing options. The systems engineer is responsible for guiding this boundary-drawing process to a conclusion that the stakeholders understand and accept. The systems engineer uses these boundaries to establish and maintain control of the system's interfaces.

The system's boundaries need to be drawn early in the systems engineering process because so much else in the design phase is dependent upon them. As is discussed next, the fundamental objectives or measures of effectiveness of the system need to be focused just beyond the external interfaces of the system. The operational concept relies upon knowing where the boundaries are for each

stakeholder. The interface requirements capture the implications of the boundaries on the system design.

Many graphical modeling techniques (e.g., IDEF0, N^2 charts, data flow diagrams, EFFBDs) can be used to define the system boundary; see Davis, [1990]. See Chapter 12 for a discussion of these techniques. IDEF0 is used in this chapter to illustrate external systems diagrams in terms of the elevator. The boundary for the elevator is defined so as to exclude the passenger, the maintenance personnel, and the building.

First, the purpose and viewpoint are defined:

Purpose: Explicitly define the system's boundary and needed interfaces
Viewpoint: Systems Engineering Team

Next the mechanisms or external systems are established, followed by the functions of these systems. The system and external system come directly from the input/output traces of the scenarios in the operational concept:

Mechanism (System/External System)	System Function
1. Elevator — the system	Provide elevator services
2. Passengers	Request and use elevator services
3. Maintenance personnel	Maintain elevator operations
4. Building	Provide structural support

Now the inputs, controls, and outputs of these functions are developed to finish the external system diagram. Recall that as part of this analysis of the elevator boundaries the focus is on the context or environment of the elevator, and these key variables are shown in the diagram as controls. See Figure 6.8.

The above discussion has focused on an external systems diagram for the operational phase of the system, in which the system is interacting with the system's users and other systems. External systems diagrams can and should be developed for every phase of the system's life cycle.

In addition to the usual syntax and semantics requirements of IDEF0 diagrams, an external systems diagram introduces several new constraints for the diagram to be valid. First, all of the *outputs of the system's function* (the elevator in this case) have to go to at least one of the external systems' functions on the page and *cannot exit the diagram*. If the output did exit the page, there would be an external system that was not included in the diagram, invalidating the purpose of the effort. Similarly, *each of the external systems must receive at least one output of our system*; otherwise, the system should be part of the context. In some cases part of the context could be shown on the external systems diagram to emphasize the importance of a particular input to the system.

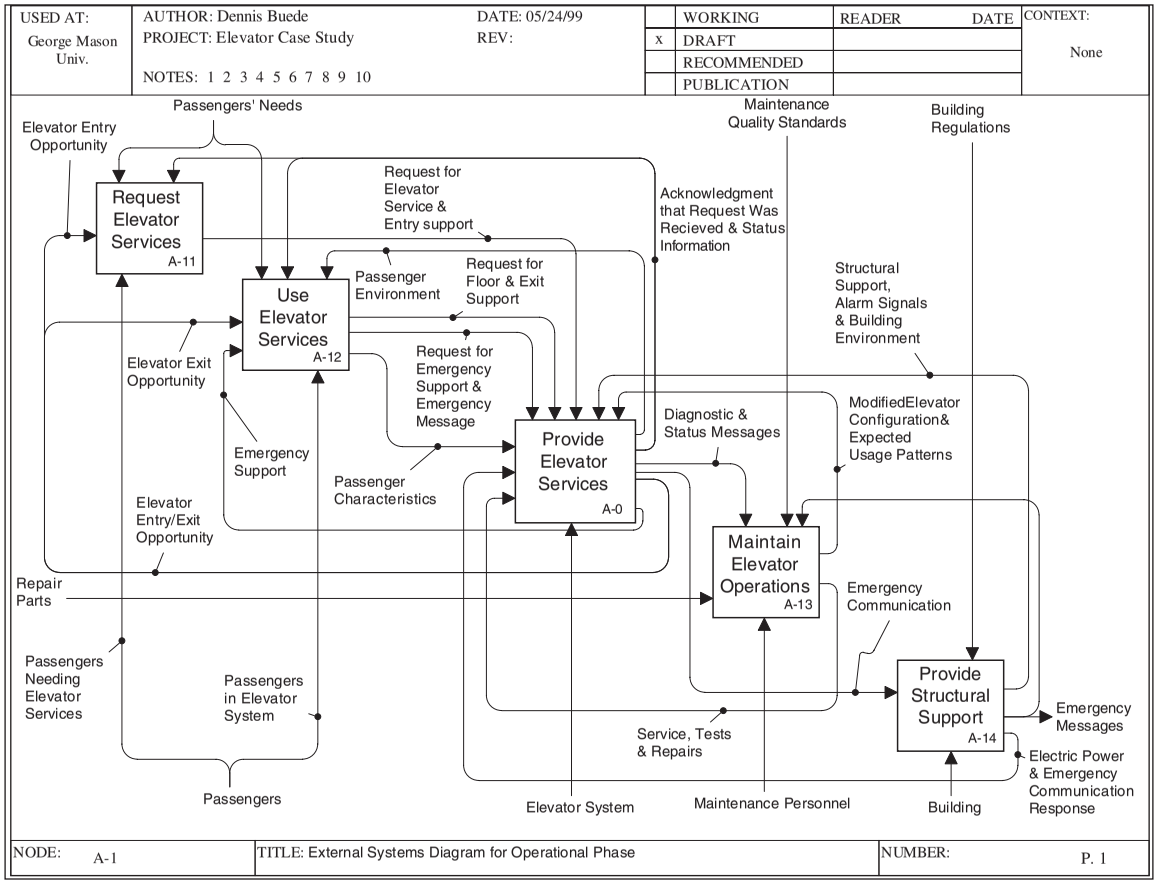

FIGURE 6.8 External systems' diagram for operational use of an elevator.

6.12 OBJECTIVES HIERARCHY FOR PERFORMANCE REQUIREMENTS

Traditionally, systems engineers have used the terms measure of effectiveness (MOE) and measure of performance (MOP), some times called a figure of merit (FOM). A *measure of effectiveness* describes how well a system carries out a task or set of tasks within a specific context; an MOE is measured outside the system for a defined environment and state of the context variables and is used to define mission requirements. Note that the further outside the system that the MOE measurement process is established, the more influence the external systems have on the measurement, yielding less sensitivity in the measurement process for evaluating the effectiveness of the system. The MOE or MOEs that were used to define the mission requirements can be divided into additional MOEs for a given system, often one for each major output of the system.

An MOP (or FOM) describes a specific system property or attribute for a given environment and context; an MOP is measured within the system. There are many possible and relevant MOPs for a specific system output; examples include accuracy, timeliness, distance, throughput, workload, and time to complete. Usually only a few of these MOPs matter for each output. The MOPs form the basis of stakeholders' requirements when they address outputs. The MOPs that address the performance of system components [e.g., chip speed of the central processing unit (CPU)] are completely inappropriate for use as requirements because they address how to achieve the stakeholders' needs, not how well to meet these needs.

Since the systems engineering design process is decision rich, introducing some concepts from decision analysis is important. Value-focused thinking [Keeney, 1992] emphasizes the proper structuring of decisions in terms of a fundamental objective. The *fundamental objective* is the aggregation of the essential set of objectives that summarizes the current decision context and is yet relevant to the evaluation of the options under consideration. Generally, this fundamental objective can be subdivided into value objectives that more meaningfully define the fundamental objective, thereby forming a *fundamental objectives hierarchy* or value structure. Keeney [1992] distinguishes this hierarchy from a *means–ends objectives network*, which relates means or "how to" variables (the design options and context) to the fundamental objective.

The *objectives hierarchy* of a system is the hierarchy of objectives that are important to the system's stakeholders in a value sense; that is, the stakeholders would (should) be willing to pay to obtain increased performance (or decreased cost) in any one of these objectives. Means objectives should not be part of this objectives hierarchy. These means objectives describe physical ways to achieve improvements in the fundamental objectives. Means objectives often contain the variables used in simulation models to estimate the system's performance on the fundamental objectives. If there is some scientific relationship among a set of variables in the objectives hierarchy, then these objectives are very likely (but not definitely) means objectives and should be removed. Carrying the decomposition of the fundamental objectives too far is a mistake.

The process that Keeney [1992] describes for defining this situation-based fundamental objectives hierarchy is consistent with INCOSE Pragmatic Principle 2 (as shown in italics) and involves working from both ends, by generalizing means–ends objectives and operationalizing strategic objectives. Means–ends objectives are ways to achieve the fundamental objective. Strategic objectives are beyond the time horizon and immediate control of options associated with the current system design decision situation. As an example, one of the set of fundamental objectives for the operation of a new elevator (see Fig. 6.9) would be "minimize passenger time in the system." The set of fundamental objectives define value trade offs among the stakeholders of the elevator system. A strategic objective would be to "improve the working environment in the building"; there are too many other factors beyond the elevator that will determine whether this objective is met for the objective to be a fundamental objective. A means–ends objective would be to "use a fuzzy logic controller"; this statement addresses a means for achieving an objective.

Next, the fundamental *objectives hierarchy* is developed by defining the natural subsets of the fundamental objective. Keeney gives the following example of a fundamental objectives hierarchy: maximize safety (the fundamental objective) is disaggregated into minimize loss of life, minimize serious injuries, and minimize minor injuries. The trade offs among these objectives clearly entail one's values, and only one's values. This subdivision is contrasted with a means–ends breakout of maximize safety that starts with minimize accidents and maximize the use of safety features on vehicles, both of which are means oriented and involve outcomes for which value trade offs are difficult. Figure 6.9 provides the fundamental objectives hierarchy for the operation of the elevator.

Pragmatic Principle 2 [DeFoe, 1993] Use Effectiveness Criteria Based on Needs to Make System Decisions

1. *Select criteria that have demonstrable links to customer/consumer needs and system requirements.*
   a. *Operational criteria: mission success, technical performance*
   b. *Program criteria: cost, schedule, quality, risk*
   c. *Integrated logistics support (ILS) criteria: failure rate, maintainability, serviceability*
2. *Maintain a* "need-*based" balance among the often-conflicting criteria.*
3. *Select criteria that are measurable (objective and quantifiable) and express them in well-*known, *easily understood units. However, important criteria for which no measure seems to exist still must be explicitly addressed.*
4. *Use trade offs to show the customer the performance, cost, schedule, and risk impacts of requirements and solutions variations.*

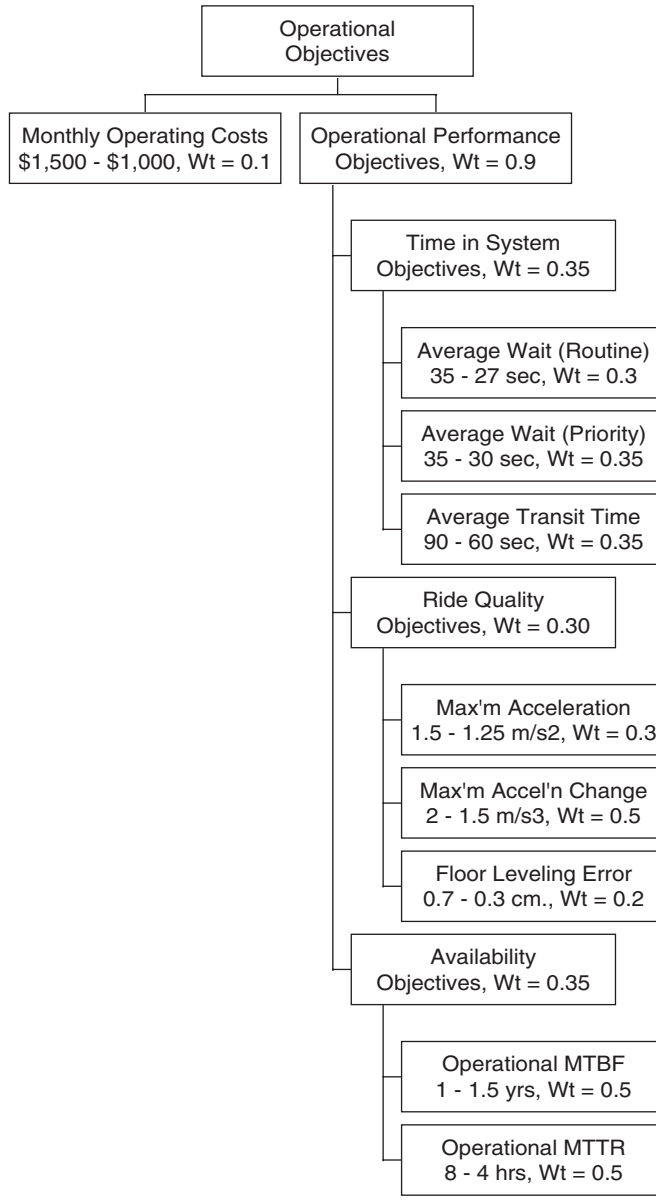

FIGURE 6.9 Fundamental objectives hierarchy for operational phase of elevator.

5. *Whenever possible, use simulation and experimental design to perform trade offs as methods that rely heavily on "engineering judgment" rating scales are more subject to bias and error.*
6. *Have the customer make all value judgments in trade offs.*

7. *Allow the customer to modify requirements and participate in developing the solution based on the trade offs.*

The objectives hierarchy (a directed tree) usually has two to five levels. The objectives in the hierarchy may include stakeholders explicitly and often include context (environmental) variables (e.g., weather conditions, peak versus non-peak loading) from the scenarios in the operational concept. If present, these scenarios are usually at the top of the hierarchy, shown as varying conditions for defining the objectives.

To make use of the objectives hierarchy for trade studies, additional information must be added; value curves must be added for each objective at the bottom of the objectives hierarchy and value weights for comparing the relative value of swinging from the bottom of each value scale to top. Figure 6.9 shows the thresholds and design goals for each objective; each threshold and design goal defines a "swing" in performance that is used to establish the "swing" weights in the value model (see Chapter 13). Figure 6.10 illustrates the value curves for a simplified objectives hierarchy for an elevator system. See Sailor [1990] for another example.

As mentioned above, decision analysis uses value curves and weights to support trade-off decisions. These value curves and weights need to be obtained from the stakeholders for two important reasons. First, the objectives typically span several groups of stakeholders, necessitating an agreement among these groups of stakeholders about the relative importance of one objective with others. Second, this objectives hierarchy and its associated value curves and weights represent the value structure needed by the systems engineering team to make many trade-off decisions during the design process. The values are those of the stakeholders, not the systems engineers. Far too often the systems engineers must guess at the stakeholders' values during design decisions, or even worse, are not even aware that design decisions have impacts on the ultimate satisfaction the stakeholders will experience.

The *objectives hierarchy* is typically used throughout the systems engineering design process as the *cornerstone of all of the trade studies* that compare one design alternative with another. In doing trade studies the evaluation should reveal which of several design alternatives is preferred; each design alternative will commonly have one advantage over the others, such as operational cost, reliability, accuracy of outputs, and the like. Since there is a system and associated qualification system for each phase of the life cycle, there should also be an objectives hierarchy for each of these systems.

This decision analysis approach has been used for many military acquisitions, two of which are covered in Buede and Bresnick [2007], in which the objectives hierarchy, value curves, and weights were developed with government users and included in the request for proposal (RFP) to industry; Chapter 13 provides a discussion of one of these two acquisitions. This explicit, quantitative approach received very positive responses from the industry design teams. Watson and Buede [1987] describe the analytic methodology that was

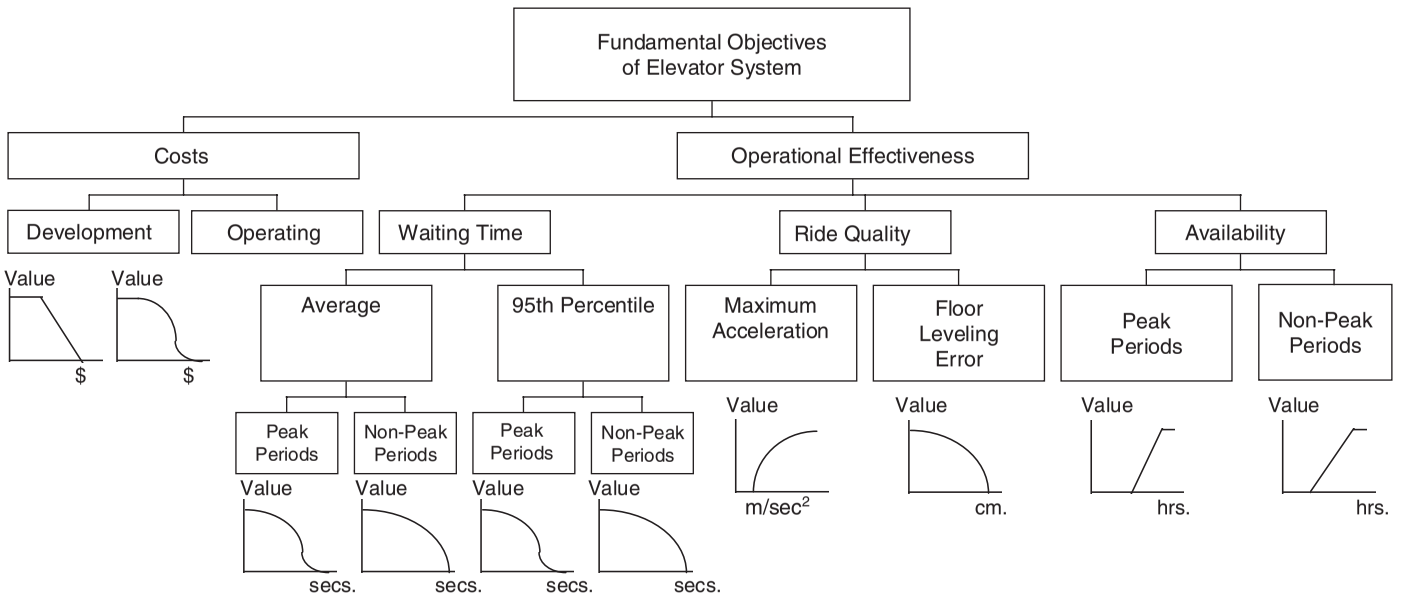

FIGURE 6.10 Objectives hierarchy with value curves.

used for these efforts. Other applications include Sailor [1990], Thurston and Carnahan [1993], and Walters [1994].

6.13 PROTOTYPING, ANALYSES AND USABILITY TESTING

Prototyping can apply to any aspect of the system and is synonymous with modeling. A *prototype* is a physical model of the system that ignores certain aspects of the system, glosses over other aspects, and is fairly representative of a third segment of aspects of the system. The prototype can range from a subscale model of the system to a paper display (storyboard) of the user interface of the system. Prototyping became strongly associated with software development in the 1980s, and it is this context that will be the focus of this section. Most discussions of prototyping focus on the development of the prototype and assume that the answers for requirements and design alternatives magically appear. However, in the real world the prototype has to undergo usability testing in order for this information to be gathered reliably.

The development of a prototype for a user interface ranges from a throwaway prototype to an evolutionary prototype [Connell and Shafer, 1989]. Throwaway prototypes are just what the name implies, prototypes that are developed for the main purpose of educating the users about the possibilities and extracting requirements from the users based upon their needs. Evolutionary prototypes are built for these educational and requirements development purposes as well, but with the idea that the prototype will eventually be turned into a working version of the system. The evolutionary prototype initially will only address a portion of the total functionality of the system, and that new functionality will be added on as the development and operational phases evolve together. Both of these concepts of prototyping have proven effective and continue today. In fact, software products for the rapid development of prototypes are now a business area in their own right.

In Chapter 9 we will introduce many types of analyses that should be conducted as part of the process for engineering systems. These analyses range from performance analyses to predict how far or well the system might be able to travel or see; timing analyses to determine how fast the system can respond or how many outputs the system can deliver per unit time; and "-ility" analyses to determine how available or safe the system is. There are also many cost and schedule analyses conducted. During the requirements phase these analyses should be conducted on the meta-system to determine what difference is made in the performance, cost, and schedule parameters of the meta-system as the performance, cost, and schedule of the system being engineered are varied. The results of these analyses provide very important information for the setting of minimum acceptable and desired marks in the system's requirements' statements.

Coupled with these analyses are many forms of elicitation of the viewpoints of the stakeholders. These elicitation sessions can be interviews with one or a

few stakeholders, facilitated group sessions, observations of stakeholder performance on the current system or with prototypes of the new system, and questionnaires. Questionnaires are the last resort when no other approach is available since questionnaires produce lots of random responses from stakeholders that were too busy or too confused to do better. Valuable information is usually only achieved through human interactions. Individual interviews are best at soliciting information from quiet people who might be silent during group sessions. Facilitated meetings are best used to surface disagreements and try to find common ground or reasons for the differences of opinion that trace back to context and external system interactions. Observations are best for stressful periods during which people do things that they may not consciously recall during discussions.

Usability testing is the process of obtaining samples of users and eliciting the reactions of these users about their needs and desires as they interact with prototypes. The prototypes can be as crude as written samples of screen interfaces or as sophisticated as working modules of the system. Usability [Bias and Mayhew, 1994; Nielsen, 1993; Wiklund, 1994] is a discipline associated with human-computer interaction that became very sophisticated in the 1980s and 1990s.

The performance elements of usability are ease of learning (learnability), ease of use (efficiency), ease of remembering (memorability), error rate, and subjectively pleasing (satisfaction). Table 6.5 provides a sample of common metrics for each of these elements. Each of these metrics has to be measured in the context of specific types of users and specific tasks. The tasks come from the scenarios in the operational concept. For the error rate element, categorizing errors into categories such as minor, major, and catastrophic is important. Care must be taken to separate random errors from those caused by the system. If necessary, baseline capabilities of the users must be measured in order to define a baseline error rate for categories of users. Satisfaction typically has to be measured by subjective, categorical questions; see Nielsen [1993].

TABLE 6.5 Metrics for Measuring Usability Elements

Usability Element	Metrics
Learnability	Time to master a defined efficiency level, e.g., 50 words per minute
	Time to master a defined skill, e.g., cut and paste
Efficiency	Time for a frequent user to complete a defined task
	Rate of producing a defined set of products for a frequent user
Memorability	Time for a casual user to complete a defined task
	Time for a casual user to achieve previously achieved rate of production
Error Rate	Number of errors of a specific type in a given period for a given task
Satisfaction	Stress level associated with use
	Fun level associated with use

Users can be categorized along three dimensions: domain knowledge, computer experience, and system use experience. Segments of users along these three dimensions should be developed for testing purposes. When a sample of users is developed for the usability testing, the population of actual system users must be considered, not the population of people in society.

Many guidelines have been developed for user interfaces. There is insufficient room to even summarize these guidelines here, but they should be consulted while developing requirements for user interfaces [see Brown 1988; Chapanis 1996; Marshall et al., 1987; Mayhew, 1992; Reason, 1990; Shneiderman, 1992].

6.14 DEFINING THE STAKEHOLDERS' REQUIREMENTS

The framework for defining requirements on the basis of the operational concept, the external system diagram, and the objectives hierarchy is presented here in detail. Recall that there are four requirements categories: input/output, system-wide and technology, trade-off, and qualification. The addendum, which can be downloaded from the author's web site, (http://www.theengineeringdesignofsystems.com) provides a detailed example of these requirements for the life-cycle phases of an elevator.

6.14.1 Input/Output Requirements

Input/output requirements are defined on the basis of the inputs, controls, and outputs of the system identified while bounding the system with the external systems diagram. This external systems diagram is the primary tool used to support the development of input/output requirements.

The systems engineering team must examine each input, control, and output in detail to discover every requirement associated with each of these items. One or more input requirements are written for each input and control; similarly, one or more output requirements are written for each output. For example, the potential passengers of the elevator have certain characteristics that impact the provision of information about the floor location of the elevator. The requirements should state that audible feedback is needed, but this would be wrong. Rather the requirements should dictate that feedback be provided to all relevant passengers, letting the engineers design a system to do this.

See Table 6.2 for examples of requirements that may be associated with inputs or outputs. Note there will be some controls such as policies and procedures that were included because each function requires at least one control. These controls are not really data elements that the system receives, and therefore there need not be any input requirements established for them.

The environment (e.g., weather and elements that are outside the control of the system) or "context" is typically defined as part of the scenarios of the

operational concept. This context should be addressed in the requirements. The questions typically addressed are:

1. What elements of the environment matter?
2. How much variation in the environmental elements must be planned for? At what priority?
3. How well can these variations be forecasted (predicted)? Can these forecasts be part of the system?
4. Can the environment be controlled by the system or external system? Must the system protect itself from the environment?

In addition to input and output requirements there are external interface constraints and the functions that should be used to decompose the system's function. Interface constraints address the physical aspects of the interface to which the system has to connect to obtain the inputs and disseminate the outputs. Examples include the standard connector type for electrical and mechanical connections. The characteristics of the power or data that come across the interface should be part of the input or output requirement.

Finally, the functional requirements are not meant to be a long list of specific, detailed functions the system has to perform to produce outputs needed by the system. Rather, the functional requirements should be the two to six functions that partition the system function in such a way that all of the inputs to the system can be transformed into all of the outputs that have been identified as part of the external systems diagram.

Several examples of input/output requirements are:

The elevator shall receive "calls" from all floors of the building. (Input requirement)

The elevator shall indicate to a prospective passenger that he/she has successfully called the elevator. (Output requirement)

The elevator shall use a standard phone line from the building for emergency calls. (External interface requirement)

6.14.2 System-Wide and Technology Requirements

The system-wide and technology requirements relate to the system as a whole and not to specific inputs or outputs. These system-wide and technology requirements are not represented in the external systems diagram and are not addressed in a substantial way in the operational concept. Yet every system should have several system-wide and technology requirements that are key to the system's success. Recall that the four major categories are technology, suitability, cost, and schedule.

A typical category of requirements relates to regulations or laws that pertain to the system. Consider the following requirement:

The elevator system shall comply with the Americans with Disabilities Act.

This requirement is considered a system-wide requirement because the requirement, like all system-wide requirements, requires knowledge of the whole system to determine whether the requirement has been met. This is a deceptive requirement though because the requirement relates directly to an external system of the elevator, the passengers, and the ability of a special class of passengers to use the system. This requirement defines input and output restrictions with which the elevator must comply. For this reason this requirement could be placed in both the input and output sections of the input/output requirements category. However, there are major disadvantages, as discussed before, in having one requirement in multiple places of the requirements document. For this reason placing such a regulation in the system-wide requirements category of suitability is wise.

Technology requirements are the ones that engineers would prefer not to have because they really do constrain the engineering creativity and should result from the other requirements if they are justifiable. These requirements are usually justified on the basis of interoperability or compatibility with an existing product line, which ultimately should be reflected in cost savings. Examples are:

The elevator system's software shall be written in C++.

The elevator system's CPU shall be Pentium 4.

Table 6.2 provides a list of common *suitability* issues, topics that address quality concerns of a system and are system-wide in scope. There are technical engineering definitions that are expressed mathematically behind each of these suitability issues. In fact, many systems engineers make a career by specializing in one or several of these suitability areas. The detailed discussion of these suitability issues is critical for understanding the engineering of systems but is beyond the scope of this book (which is to provide a set of methods and models for getting to the definition of requirements for these issues and developing a design that meets such requirements). Conducting analyses of system concepts or designs related to suitability issues is discussed in more detail in Blanchard and Fabrycky [1998] and Pohl [2007].

Besides the technology and suitability requirements, cost and schedule requirements are also part of this segment of the requirements' partition. A *cost requirement* deals with payment of money during the appropriate life-cycle phase for the system in question to be useful. A *schedule requirement* deals with a timing issue for the relevant system for the phase of life cycle in question. There is nearly always a cost and a schedule requirement for every phase of the system's life cycle. Table 6.2 provides examples of some of these.

The objectives hierarchy should address every system-wide requirement that is critical enough to be considered a performance requirement. These typically include the cost and schedule requirements as well as several suitability requirements.

6.14.3 Trade-Off Requirements

Trade-off requirements in the form of value curves and value weights were described above during the discussion of the objectives hierarchy. Chapter 13 provides much more detail into the theory and elicitation techniques that can be used to obtain this requirements information. This set of requirements relies solely on value judgments of each segment of the stakeholders. These value judgments must be obtained in a reliable manner from a reasonable sample of representatives of each segment of the stakeholders. For some segments, such as the bill payer, determining who should provide the value judgments is easy. For other systems that will be used by thousands or millions of people, talking to everyone is not feasible. Care must be taken to define a sufficiently large and representative sample of these users.

6.14.4 Qualification Requirements

The four elements of the qualification requirements for a system in any life-cycle phase are: (1) *observance:* how the estimates (qualification data) for each input/output and system-wide requirement will be obtained, that is, test, analysis and simulation, inspection, or demonstration; (2) *verification plan:* how the qualification data will be used to determine that the real system conforms to the design that was developed; (3) *validation plan:* how the qualification data will be used to determine that the real system complies with the stakeholders' requirements; and (4) *acceptance plan:* how the qualification data will be used to determine that the real system is acceptable to the stakeholders.

The observance qualification requirements deal with data collection activities, devices, and facilities. For example, on a consulting project the author learned that an aircraft manufacturer was developing a detailed qualification plan for a fire suppression system installed in the cockpit of the aircraft. Specific derived requirements for the pressure and concentration of a chosen fire suppression agent existed for the three-dimensional space of the cockpit based upon the distribution of people and critical equipment. These requirements were developed based upon calculations and simulations that had been developed to ensure that the release pressure of the fire suppression system would be great enough to distribute the agent in the correct spatial concentration to suppress the fire but not too great to damage the structural elements of the cockpit. Note all of this analytical work had been done to address a fire suppression agent that had never been used in a cockpit before, so there was a great deal of uncertainty about the validity of the calculations. Observance requirements were developed to identify places in the cockpit to measure the

concentration of the fire suppression agent at specific times during tests of the fire suppression system.

The verification plan was to activate the fire suppression system several times and take measurements of pressure and concentration at the spatial locations for which requirements had been defined. Note for verification, there was no test of the fire suppression system's ability to extinguish a real fire. This verification plan also addressed the examination of the structural elements of the cockpit to verify the requirement that there be no structural damage. The final part of the verification plan defined the criteria for determining that this verification test was passed or failed. (Note this level of detail would not be in the stakeholders' requirements for the aircraft system but would be in the specification for the fire suppression system, a component of the aircraft. Nonetheless, analogous system-level qualification information would be in the stakeholders' and system requirements for the aircraft system.) The data collection activity here was part of the observance qualification requirement.

Next, validation tests for the fire suppression system were defined based upon three safety scenarios that could be traced to the operational concept for the specification of the fire suppression system if not the aircraft system. The safety scenarios were defined for three different potential causes of a fire. The observance qualification requirement stated that a fire be started *in* the cockpit based upon each of three causes, and the test would determine whether the fire suppression was activated and effectively suppressed the fire. The validation test requirement defined what was meant by effectively suppressing the fire. A fourth cause of a fire is from a ballistic hit from a weapon fired at the aircraft (this was a military aircraft). As a result, the test requirement called for several test cockpits to be hit by a weapon, a fire started either spontaneously or through whatever means were necessary (a fire is not guaranteed with a ballistic hit), and the fire suppression system's ability to suppress this fourth type of fire tested. Again, the observance qualification requirement defines that these ballistic tests will be conducted, and the validation requirement defines what successful performance is.

The acceptance test requirement provides the stakeholders' definitions of what acceptable performance is for the system as a whole. Sometimes this is based upon the validation tests and is synonymous with the validation test plan. At other times the acceptance test requirements call for additional tests, simulations, or inspections with acceptance criteria that are different than those of the validation criteria. *These qualification requirements, for each phase of the life cycle, are used to design the qualification system to be used during integration for each phase of the life cycle.*

As a final note, the aircraft manufacturer had designed the fire suppression system so that detailed design changes could be made as part of this integration phase activity of testing. Since the fire suppression system agent was new, the manufacturer needed the flexibility to adjust the design of the fire suppression system if the fire suppression was either less or more effective than expected.

In fact, two locations had been designed for additional agent distribution tanks in case the design did not meet the requirements. In addition the tank pressure in the planned tanks could be increased or decreased as needed. In an aircraft the total system weight is so important that the manufacturer was planning additional verification tests to remove as much concentrated agent from the tanks as possible while meeting the pressure and concentration output requirements.

6.15 REQUIREMENTS MANAGEMENT

"Requirements Management is the identification, derivation, allocation, and control in a consistent, traceable, correlatable, verifiable manner of all the system functions, attributes, interfaces, and verification methods that a system must meet including customer, derived (internal), and specialty engineering needs." [Stevens and Martin, 1995, p. 11] This definition of requirements management is inclusive of everything discussed in this chapter. For example, requirements management addresses which requirements have been changed, when and by whom; to what documents does each requirement trace; to which components has each requirement been allocated. Requirements management is considered a key element of systems engineering as shown by INCOSE Pragmatic Principle 3.

A more limited, and perhaps more common, definition is the "care and feeding" of the requirements, sometimes called requirements traceability. More formally, requirements traceability "refers to the ability to describe and follow the life of a requirement, in both a forwards and backwards direction." [Gotel and Finkelstein, 1994, p. 95] Numerous techniques for tracing requirements and their sources and destinations are semantic networks, assumption-based truth maintenance networks, constraint networks, cross-referencing schemes, hypertext, integration documents, key phrase dependencies, matrices, and templates. Relational and object-oriented databases are used to implement requirements traceability tools.

Pragmatic Principle 3 [DeFoe, 1993] Establish and Manage Requirements

1. *Identify and distinguish between specified (fundamental or essential), allocated, implied, and derived requirements.*
2. *Carry analysis and synthesis to at least one level broader and deeper than seems necessary before settling on requirements and solutions at any given level. (Top down is a better recording technique than it is an analysis or synthesis technique.)*
3. *Write a rationale for each requirement. The attempt to write a rationale for a "requirement" often uncovers the real requirement.*
4. *Ensure the customer and consumer understand and accept all the requirements.*

5. *Explicitly identify and control all the external interfaces the system will have —signal, data, power, mechanical, parasitic, and the like. Do the same for all the internal interfaces created by the solution.*
6. *Negotiate interfaces with affected engineering staff on both sides of each interface and get written agreement by the two parties before the customer approves the interface documentation.*
7. *Document all requirements interpretations in writing. Don't count on verbal agreements to stand the test of time.*
8. *Plan for the inevitable need to correct and change requirements as insight into the need and the "best" solution grows during development.*
9. *Be careful of new fundamental requirements coming in after the program is underway. They invariably have a larger impact than is obvious.*
10. *Maintain requirements traceability.*

6.16 SUMMARY

Requirements are generally considered the cornerstone of the systems engineering process because requirements define the design problem. Stakeholders' requirements are those requirements initially established by the system's stakeholders with the help of the systems engineering team. The systems engineering design process is a mixture of establishing requirements to define the design problem and partitioning the physical resources of the system into components that perform functions that meet the requirements (the solution to the design problem). This partitioning process is decision rich in that many important decisions are made by the systems engineering team that will ultimately affect the performance of the system and the satisfaction of the stakeholders.

This chapter defines requirements and the characteristics that these requirements should satisfy. In addition, this chapter provides a method or process for developing these requirements. This process includes the concepts and associated models of an operational concept, external systems diagram, and objectives hierarchy, all of which are extremely valuable aids in the definition of requirements.

The key points made in this chapter concerning the systems engineering design process are that (1) all stakeholders have stakeholders' requirements that, taken together, address every phase of the system's life cycle. Capturing the complete set of stakeholders' requirements ensures a concurrent engineering process. (2) The set of stakeholders' requirements should ensure a decision rich design process by not over constraining the design. The following attributes of requirements are meant to ensure the process is not overconstrained: traced, correct, unambiguous, understandable, design independent, attainable, comparable, and consistent. (3) At the same time the stakeholders' requirements

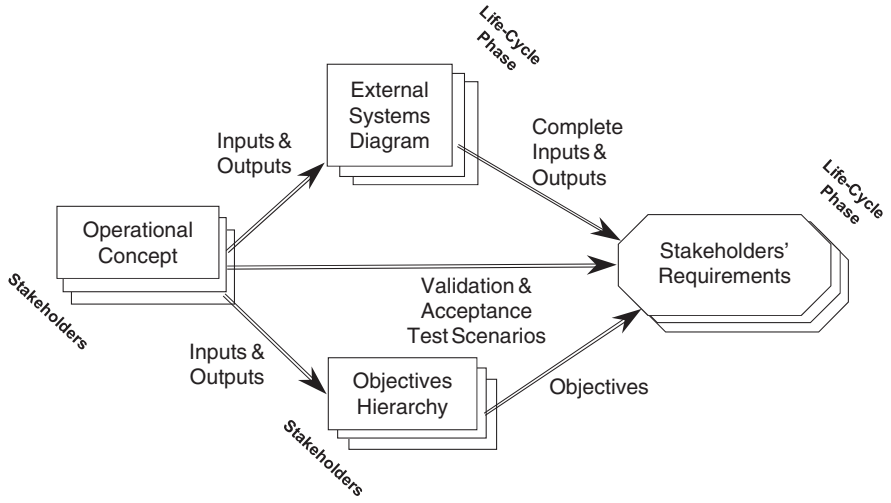

FIGURE 6.11 Summary of stakeholders' requirements development.

should not underconstrain the design because the stakeholders should be happy with the system that is created. Complete, verifiable, and traceable requirements should guarantee this.

The systems engineering design process defined in this chapter includes the development of an operational concept for each stakeholder group, external systems diagram for each life cycle phase, and an objectives hierarchy for each stakeholder group. These three concepts are then used to develop the stakeholders' requirements, organized by life-cycle phase. See Figure 6.11. Wymore's [1993] partition of requirements was adopted and modified: input/output requirements, technology and system-wide requirements, trade-off requirements, and system qualification requirements. In particular the trade-off information defining stakeholder values that is needed to support design decisions includes performance trade offs, cost trade offs, and cost–performance trade off information. This initial systems engineering phase is complete when the existence of at least one feasible solution is verified, the acceptance requirements for the qualification system are defined, and the stakeholders have approved the StkhldrsRD.

CASE STUDY: AIR BAG RESTRAINT SYSTEM

Air bags, a safety device appearing in automobiles in the early 1990s, became the cause of death for a noticeable number of individuals. This severe, undesirable impact can be traced to the requirements for the air bag system. The following requirements issues are paraphrased from

those published in 1984 by the National Highway Traffic Safety Administration (NHTSA) as part of Federal Motor Vehicle Safety Standard 208, Occupant Crash Protection [see Buede, 1998]:

1. The requirements defined a single safety scenario on which to base the design. This single scenario could only be justified if there was a single worst-case situation. Note this was not the approach with seat belts for which requirements were defined for the 50th percentile 6-year-old, 5th percentile adult female, and 95th percentile adult male.
2. The single, worst-case scenario for safety protection was the 50th percentile male not wearing a seat belt in a 30 mile per hour frontal collision. No specific attention was directed toward children and women, and small or large adults. As the results show, this is the root of the problem.
3. While there was a requirement that the air bag not deploy on a very rough or bumpy road or when the car hits a small pole, there was no requirement that the air bag remain undeployed during accidents at sufficiently slow speeds that no lives are in danger. A number of people have lost their lives in accidents in which the car was only moving at 5 or 10 miles per hour, speeds at which there was almost no chance of a fatality.
4. The test condition was defined such that the test dummy is only in an upright position with its hands at the 3 and 9 o'clock positions on the steering wheel, and a frontal accident with the crash force parallel to the length of the car occurs into a fixed barrier at 30 miles per hour. In fact, frontal accidents are likely to occur when the driver is not in this nominal driving position. Also there arc many accidents requiring an air bag safety restraint in which the crash force is close to being parallel to the length of the car but is not exactly parallel.
5. There was no requirement that addressed accidents involving pre-impact braking. For frontal accidents, pre-impact braking is common. In the case of the current air bag design, pre-impact braking clearly causes problems because the people being protected are beginning to move toward the air bag before the sensors for activating the air bag can be triggered. This leads to a need for even more rapid inflation of the air bag.
6. The issue of injuries inflicted on drivers and passengers when the person collides with the deployed air bag was not addressed in the safety standard. Such a requirement would lead to an evaluation of the elasticity of alternate fabrics for the air bag, as well as the final pressure in the inflated air bag. The first generation, fully inflated air bag is very inelastic.

7. There was no requirement that the disposal of unused or partially expanded air bags be safe and free of toxic waste. Sodium azide is considered a hazardous chemical by some. Also, uninflated air bag systems can explode when the car is crushed in a junkyard.

The requirements for air bags were placed in a federal regulation. It takes 16 months on average to change these regulations. "From 1970 until 1991, federal statutes requiring air bags were debated, imposed, revoked, and reinstated as consumer and safety groups battled it out with reluctant automobile manufacturers and mostly Republican administrations. It took a Supreme Court decision in 1983, overturning a Reagan administration revocation of the standard, before the campaign took on real momentum." [Ottaway, 1996, p. 48] Unfortunately, while so much attention was being paid to the concept of air bags, the requirements for the air bags were overlooked and remained unchanged.

CASE STUDY: *APOLLO 13* DISASTER

This case study is excerpted from Lovell and Kluger [1994], the book associated with the movie titled *Apollo 13*.

> Every major component in an Apollo spacecraft, from gyros to radios to computers to cryogenic tanks, was routinely tracked by quality control inspectors from the moment its first blueprints were drawn to the moment it left the pad on launch day; any anomaly in manufacturing or testing was noted and filed away. Generally, the thicker the file any part amassed by the time it was ready to fly, the more headaches it had caused. Oxygen tank two, it turned out, had quite a dossier.

The problems with the tank began in 1965, around the time Jim Lovell and Frank Borman were deep in training for the flight of Gemini 7, and North American Aviation was building the Apollo command-service module that would ultimately replace the two-man ship. … One of the most delicate of the delegated tasks was the construction of the spacecraft's cryogenic tanks, a job assigned to Beech Aircraft in Boulder, Colorado.

The Apollo spacecraft's electrical system was designed to operate on 28 volts of current [*derived requirement*] — the amount of juice provided by the service module's three fuel cells. Of all the systems inside the cryogenic tanks that would be driven by this relatively modest power system, none required more rigorous monitoring than the heaters. Ordinarily, cryogenic hydrogen and oxygen were maintained at a constant temperature of minus 340 degrees [*derived requirement*]. This was cold enough to keep the frigid gases in a slushy, non-gaseous state, but warm enough to allow some of the slush to vaporize and flow through the

lines that fed both the fuel cells and the atmospheric system of the cockpit. Occasionally, however, the pressure in the tanks dropped too low, preventing the gas from moving into the feed lines and endangering both the fuel cells and the crew. To prevent this, the heaters would occasionally be switched on, boiling off some of the liquid and raising the internal pressure to a safer level.

Beech and North American knew that the tanks the new ship needed would have to be more than just insulated bottles. To handle contents as temperamental as liquid oxygen, the spherical vessels would require all manner of safeguards, including fans, thermometers, pressure sensors, and heaters, all of which would have to be immersed directly in the supercold slush that the tanks were designed to hold, and all of which would have to be powered by electricity.

Of course, immersing a heating element in a pressurized tank of oxygen was, on its face, a risky business, and in order to minimize the danger of fire or explosions, the heaters were supplied with thermostat switches that would cut the power to the coils if the temperature in the tank climbed too far. By most standards, that upper temperature limit was not very high; 80 degrees was about as hot as the engineers ever wanted their supercold tanks to get [*derived requirement*]. But in insulated vessels in which the prevailing temperature was usually 420 degrees lower, that was a considerable warm-up. When the heaters were switched on and functioning normally, the thermostat switches remained closed — or engaged — completing the heating system's electrical circuit and allowing it to continue operating. If the temperature in the tank rose above the 80-degree mark, two tiny contacts on the thermostat would separate, breaking the circuit and shutting the system down.

When North American first awarded the tank contract to Beech Aircraft, the contractor told the subcontractor that the thermostat switches — like most of the switches and systems aboard the ship — should be made compatible with the spacecraft's 28-volt power grid, and Beech complied. This voltage, however, was not the only current the spacecraft would ever be required to accept. During the weeks and months preceding a launch, the ship spent much of its time connected to launch-pad generators at Cape Canaveral, so that preflight equipment test could be run [*missed operational concept scenario*]. The Cape's generators were dynamos compared to the service module's puny fuel cells, regularly churning out current at a full 65 volts.

North American eventually became concerned that such a relative lightning bolt would cook the delicate heating system in the cryogenic tanks before the ship ever left the pad, and decided to change its specs, alerting Beech that it should scrap the original heater plans and replace them with ones that could handle the higher launch pad voltage. Beech noted the change and modified the entire heating system — or almost the entire heating system. Inexplicably, the engineers neglected to change the

specifications on the thermostat switches, leaving the old 28-volt switches in the new 65-volt heaters. Beech technicians, North American technicians, and NASA technicians all reviewed Beech's work, but nobody discovered the discrepancy.

Although the 28-volt switches in a 65-volt tank would not necessarily be enough to cause damage to a tank — any more than, say, bad wiring in a house would necessarily cause a fire the very first time a light switch was thrown — the mistake was still considerable. What was necessary to turn it into a catastrophe were other, equally mundane oversights. The Cortright Committee soon found them.

The tanks that eventually flew aboard Apollo 13 were… installed in service module 106. Module 106 was scheduled to fly during 1969s Apollo 10 mission, … and the engineers decided to remove the existing tanks from the Apollo 10 service module and replace them with newer ones. …

Removing cryogenic tanks from an Apollo spacecraft was a delicate job. … Rockwell engineers unbolted the tank itself in spacecraft 106 and began to lift it carefully from the ship.

Unknown to the crane operators, one of the four bolts had been left in place. When the winch motor was activated, the shelf rose only two inches before the bolt caught, and the crane slipped, and the shelf dropped back into place. … The tanks on the dropped shelf were examined and found to be unharmed. Shortly afterward, they were removed, upgraded, and reinstalled in service module 109, which was to become part of the spacecraft more commonly known as Apollo 13. …

One of the most important milestones in the weeks leading up to an Apollo launch was the exercise known as the countdown demonstration test. … To make the dress rehearsal as complete as possible, the cryogenic tanks would be fully pressurized, the astronauts would be fully suited, and the cabin would be filled with circulating air at the same pressure used at liftoff.

During Apollo 13's countdown demonstration test with Jim Lovell, Ken Mattingly, and Fred Haise strapped into their seats, no significant problem occurred. At the end of the long dress rehearsal, however, the ground crew did report a small anomaly. The cryogenic system, which had to be emptied of its supercold liquids before the spacecraft was shut down, was behaving balkily. … Oxygen tank two seemed jammed, venting only about 8 percent of its 320 pounds of supercold slush and then releasing no more.

… When the tank was dropped eighteen months earlier, they now suspected, the tank had suffered more damage than the factory technicians at first realized, knocking one of the drain tubes in the neck of the vessel out of alignment. …

At its present supercold temperature and relatively low pressure, the liquid in the tank wasn't going anywhere. But what would happen, one of technicians wondered, if the heaters were used? Why not just flip the

warming coils on now, cook the slush up, and force the entire load of 0_2 out of the vent line? …

But the wrong thermostat switch — the 28-volt switch — was in the tank, and as it turned out, the heaters stayed on for a long, long time…. Given the huge load of 07 trapped in the tank, the engineers figured it would take up to eight hours before the last few wisps of gas would vent away. Eight hours was more than enough time for the temperature in the tank to climb above the 80-degree mark, but the technicians knew they could rely on the thermostat to take care of any problem. When this thermostat reached the critical temperature, however, and tried to open up, the 65 volts surging through it fused it instantly shut.

The technicians on the Cape launch pad had no way of knowing that the tiny component that was supposed to protect the oxygen tank had welded closed. …

Unfortunately, the readout on the instrument panel wasn't able to climb above 80 degrees. … The men who designed the instrument panel saw no reason to peg the gauge any higher, designating 80 as its upper limit. What the engineer on duty that night didn't know — couldn't know — was that with the thermostat fused shut, the temperature inside this particular tank was climbing indeed, up to a kiln-like 1000 degrees.

… At the end of eight hours, the last of the troublesome liquid oxygen had cooked away as the engineers had hoped it would — but so too had most of the Teflon insulation that protected the tank's internal wiring. Coursing through the now empty tank was a web of raw, spark prone copper, soon to be reimmersed in the one liquid likelier than any other to propagate a tire: pure oxygen.

[Lovell and Kluger, 1994, pp. 372–378] The words in italics inside the braces were inserted by the author of this text.

PROBLEMS

6.1 Use IDEF0 to develop an external system diagram for an information system to advise undergraduate systems engineering students on the development of their plans of study. The information system is the software and hardware system that the undergraduate systems engineering students will use. Assume the systems engineering faculty will maintain the accuracy of the courses and prerequisites. Assume the information system can obtain schedule information over a network from the registrar's office. Assume that the information system produces a written plan of study for each student.

6.2 Use the following operational concept for the operational phase of the ATM to:

i. Create one additional scenario for the operational concept.
ii. Develop an external system diagram using IDEF0.

iii. Create an objectives hierarchy for the ATM system.
iv. Develop a set of stakeholders' requirements. Use the format of the Stakeholders' Requirements Document and the taxonomy of four types of requirements from this chapter. Make every effort to develop as complete and unambiguous a set of stakeholders' requirements for the operational phase as possible using only the information provided in the following scenarios. Then add three system-wide requirements and four qualification requirements.

Automatic Teller Machine (ATM) for Money Mart Corporation. The ATM system is to provide a cost-effective service to bank customers that is convenient, safe, and secure 24-hour access to a common set of banking transactions and reduce the cost of providing these basic transaction. The ATM system shall provide a number of the most common banking transactions (deposit, withdraw, transfer of funds, balance query) without involvement of bank personnel.

The operational concept is comprised of a group of scenarios that are based upon the stakeholders' requirements and relates to both the bank's customers and employees.

Customer Scenarios

1. Customer makes deposits.
   a. Customer provides valid general identification information.
   b. ATM requests unique identification information.
   c. Customer enters unique identification information.
   d. ATM requests activity selection.
   e. Customer selects deposit.
   f. ATM requests account type.
   g. Customer identifies account type (i.e., savings, checking, bank credit card).
   h. ATM requests type of deposit (cash vs. check).
   i. Customer identifies type of deposit — cash/check.
   j. ATM provides a means to physically insert cash/check into ATM.
   k. Customer enters deposit.
      ATM transmits the transaction to the main bank computer, gives customer receipt, returns to main menu.
2. Customer requests cash to be withdrawn from an account.
   a. Customer provides valid general identification information.
   b. ATM requests unique identification information.
   c. Customer enters unique identification information.
   d. ATM requests activity selection.
   e. Customer selects withdrawal.
   f. ATM requests account type.

g. Customer identifies account type (i.e., savings, checking, bank credit card).
h. ATM requests amount of withdrawal.
i. Customer identifies amount of withdrawal (C_{req}).
j. ATM contacts the main bank computer and requests the amount of available funds from the selected account (F_{max}).
k. If $C_{req} > F_{max}$, ATM denies request.
l. If $C_{req} > C_{lim}$, ATM denies request. (C_{lim} is the maximum cash withdrawal allowed.)
m. $C_{req} > C_{left}$ ATM apologizes for inability to satisfy request and sends message to bank for more funds. (C_{left} is amount cash ATM has left).
n. Else, ATM transmits the transaction to the main bank computer, gives customer receipt, gives the customer money, and returns to the main menu.

3. Customer requests transfer of funds from one account to another.
a. Customer provides valid general identification information.
b. ATM requests unique identification information.
c. Customer enters unique identification information.
d. ATM requests activity selection.
e. Customer selects transfer of funds.
f. ATM requests account type for source of funds transfer.
g. Customer identifies source account type.
h. ATM requests account type for destination of funds transfer.
i. Customer identifies destination account type.
j. ATM queries the main bank computer to determine the availability of funds from the source account (F_{max}).
k. ATM requests the amount of the funds transfer.
l. Customer identifies the amount of funds to be transferred (F_{trns}).
m. If Ftrns > F max, the ATM denies the request.
n. Else the funds are transferred, ATM transmits the transaction to the main bank computer, gives the receipt, and returns to the main menu.

4. Customer requests the status of balance of an account.
a. Customer provides valid general identification information.
b. ATM requests unique identification information.
c. Customer enters unique identification information.
d. ATM requests activity selection.
e. Customer selects balance status of an account.
f. ATM requests account type for balance query.
g. Customer identifies account type.
h. ATM queries the main bank computer to obtain the needed information, gives customer receipt, and returns to the main menu.

5. Customer cancels request
   a. Customer provides valid general identification information.
   b. ATM requests unique identification information.
   c. Customer enters unique identification information.
   d. ATM requests activity selection.
   e. Customer selects withdrawal.
   f. ATM requests account type.
   g. Customer identifies account type (i.e., savings, checking, bank credit card).
   h. During the course of a transaction, the customer indicates the desire to cancel the current transaction.
   i. ATM returns to the main menu and gives the customer the choice to begin another transaction.
   j. Customer chooses to end the session.
   k. ATM resets for the next customer.
6. Customer input device is not working.
   a. Customer attempts to provide valid general identification information.
   b. ATM informs customer that the input device is not working.
   c. If this is the third straight customer for which the input device is not working, then the ATM sends a message to the bank about this problem.
7. ATM cannot verify the customer identification scheme.
   a. Customer provides valid general identification information.
   b. ATM requests unique identification information.
   c. Customer enters unique identification information.
   d. ATM checks unique identification, finds the identification incorrect, and requests customer to re-input identification.
   e. Customer enters unique identification information.
   f. ATM checks unique identification, finds the identification incorrect, and requests customer to re-input identification.
   g. Customer enters unique identification information.
   h. ATM checks unique identification, finds the identification incorrect, and alerts the customer that any attempts to re-input identification will result in an alarm to the bank.
   i. Customer leaves.
   j. ATM resets for the next customer.
8. ATM does not have receipts.
   a. When only 25 receipts remain, ATM sends message to bank to resupply receipts.
9. Hostile situations
   a. Robber attempts to break into ATM.
   b. ATM sends message to bank and sounds alarm.
   c. ATM shuts down operation.

Bank Employee Scenarios

1. Routine resupply operation
   a. Employee enters code into ATM.
   b. ATM provides access to valid employee.
   c. Employee opens ATM.
   d. Employee loads ATM with cash.
   e. Employee loads ATM with blank receipts.
   f. Employee removes deposits from ATM.
   g. Employee shuts ATM and initializes for operation.
2. Malfunction operations
   a. Employee enters code into ATM.
   b. ATM provides access to employee.
   c. Employee opens ATM.
   d. Employee runs built-in diagnostic tests to determine problem.
   e. ATM responds to diagnostic tests.
   f. Employee fixes ATM.
   g. Employee runs built-in diagnostic tests to determine if problem is solved.
   h. ATM responds to diagnostic tests.
   i. Employee shuts ATM and initializes for operation.

6.3 Use the following operational concept for the operational phase of an automobile system called OnStar to:

i. Create one additional scenario for the operational concept.
ii. Develop an external system diagram using IDEF0.
iii. Create an objectives hierarchy for the OnStar system.
iv. Develop a set of stakeholders' requirements. Use the format of the Stakeholders' Requirements Document and the taxonomy of four types of requirements from this chapter. Make every effort to develop as complete and unambiguous a set of stakeholders' requirements as possible for the operational phase using only the information provided in the following scenarios. Then add three system-wide requirements and four qualification requirements.

OnStar System for Cadillac. The OnStar system is an information system for Cadillac owners to provide emergency help and a wide range of support. Generally, the operational concept involves a satellite communications link between the car and a control center run by Cadillac.

The operational concept is comprised of a group of scenarios that are based upon the stakeholders' requirements and relates to both the OnStar's users and maintenance personnel.

User Scenarios

1. Driver uses cellular phone to contact control center to find directions.
   a. Driver pushes a single button on the OnStar cellular phone.
   b. OnStar calls control center.
   c. Control center person responds and inquires where driver wants to go via the OnStar cellular phone.
   d. Driver responds with location (tourist landmark, restaurant, hotel, ATM, Cadillac dealer, and gas station) via the OnStar cellular phone.
   e. Control center person responds with address and block by block directions via the OnStar cellular phone.
   f. Driver uses OnStar to record these directions and plays them back as needed.
2. Driver loses car in parking lot.
   a. Driver calls control center using a toll-free number from a pay phone.
   b. Control center person sends signal to OnStar.
   c. OnStar activates flashing lights and honking horn on driver's car.
   d. Driver goes to car and deactivates lights and horn.
3. Driver locks keys in car.
   a. Driver calls control center using a toll-free number from a pay phone.
   b. Control center person requests identification information.
   c. Driver provides identification information.
   d. Control center person sends signal to OnStar.
   e. OnStar unlocks your car.
4. Emergency support when an accident occurs.
   a. Car is involved in an accident in which the air bags are activated.
   b. OnStar sends a priority signal to the control center, with the exact location.
   c. Control center person calls driver on the OnStar cellular phone.
   d. If contact is not made, control center person contacts appropriate 911 number.
   e. Control center person provides information on driver's location, car, and license number.
   f. Police respond to driver.
5. Vandals/thieves break into driver's car and steal the car.
   a. Vandals/thieves break into driver's car and drive the car away.
   b. The security system of car is activated and sends a signal to OnStar.
   c. OnStar sends a signal to the control center.
   d. The control center person calls 911 and reports the break-in and provides information on driver's car to the police.
   e. OnStar sends signals to the control center allowing the car to be tracked.

   f. The control center person provides this tracking information to the police.
6. Carjackers steal car and kidnap driver and passengers.
   a. Thieves carjack the car with the driver (and possibly passengers).
   b. Driver pushes a red button on the cellular phone.
   c. OnStar sends a carjacking signal to the control center with an open phone line so that any conversations can be monitored.
   d. OnStar sends signals to the control center allowing the car to be tracked.
   e. The control center provides information about the situation and the location of the car to the police.
7. OnStar is deactivated.
   a. OnStar receives its power from the car's battery.
   b. The car's battery is dead or disconnected causing the deactivation of OnStar.

Maintainer Scenarios

1. Maintainer checks emergency carjacking capability.
   a. Maintainer tests emergency button on the cellular phone to determine that contact with control center is made. If tests show a problem, adjustments are made or cellular phone is replaced to correct any deficiencies.
   b. Maintainer tests link to control center to make sure that conversation can be heard and that car's location is transmitted. Adjustments or replacements are made as necessary to correct any deficiencies.
2. Maintainer tests ability of OnStar to unlock car.
   a. Maintainer checks that unlock signal is received by OnStar.
   b. Maintainer checks that OnStar unlocking signal is activated when control center unlock signal is received.
   c. Maintainer checks that car locks are unlocked when OnStar unlocking signal is sent.
   d. Maintainer makes repairs as needed.

6.4 Use the following operational concept for the *development* phase of an air bag system:

i. Create one additional scenario for the operational concept.
ii. Develop an external system diagram using IDEF0.
iii. Create an objectives hierarchy for the air bag development system.
iv. Develop a set of stakeholders' requirements. Use the format of the Stakeholders' Requirements Document and the taxonomy of four types of requirements from this chapter. Make every effort to develop

as complete and unambiguous a set of stakeholders' requirements as possible for the operational phase using only the information provided in the following scenarios. Then add three system-wide requirements and four qualification requirements.

Vision and Mission Requirement: The systems engineering team for an upgraded air bag safety restraint system shall design an air bag system that saves as many lives as possible while not subjecting any drivers or passengers to unneeded injuries or deaths. Cost of the air bag system will be kept within bounds and designs will be tailored to various automakers' needs.

Scenarios

1. The systems engineering team (SET) will review all safety regulations published by the National Highway Traffic Safety Administration (NHTSA), send questions and comments to NHTSA on a timely basis, receive responses, and incorporate these regulations into the air bag design.
2. The SET will seek out and review all research findings available on air bag systems, formulate questions and comments to the research teams on a timely basis, receive and review responses, and ensure that the air bag design is consistent with the best research available.
3. The SET will send their requirements documents on the air bag system and the manufacturing system for the air bag system to the appropriate corporations for comments and respond to any comments received from these corporations. Comments related to the cost of the systems and the fit of the designs will be of special interest.
4. The SET will send the entire set of required test results on its designs to the NHTSA for review and comment; any questions from NHTSA will be answered and further tests conducted as needed.
5. The SET will send all safety findings and liability issues and analyses of their designs to corporate headquarters and respond to corporate guidance concerning safety and liability issues.
6. The SET will receive "built to" configuration items (CIs) from the air bag manufacturer, will integrate these items into a test automobile, and will test the integrated air bag against the test requirements. Design changes will he identified and incorporated into the requirements documents as needed based upon the tests. The revised requirements documents will be sent to the automakers and manufacturers for comment.
7. The SET will use additional "built to" CIs to build and forward operational test items to the automakers for integration testing into the automobiles of the automakers. Based upon these operational tests

the automakers will forward additional comments on the air bag design. These comments will be incorporated into the requirements documents.

8. The air bag manufacturers will submit engineering change proposals (ECPs) to the SET as problems are encountered during production. The SET will adopt those ECPs that are warranted, reject those that are not warranted, and comment on the remaining so that an acceptable solution can be found to manufacturing problems.

6.5 Use the following operational concept for the *manufacturing* phase of an air bag system:

i. Create one additional scenario for the operational concept.
ii. Develop an external system diagram using IDEF0.
iii. Create an objectives hierarchy for the air bag development system.
iv. Develop a set of stakeholders' requirements. Use the format of the Stakeholders' Requirements Document and the taxonomy of four types of requirements from this chapter. Make every effort to develop as complete and unambiguous a set of stakeholders' requirements as possible for the operational phase using only the information provided in the following scenarios. Then add three system-wide requirements and four qualification requirements.

Vision and Mission Requirement: The Manufacturing Division for an upgraded air bag safety restraint system shall design the air bag manufacturing system to produce the air bag system with as low a long-term cost as possible. Long-term cost includes the discounted cost of producing acceptable air bags as well as providing free parts due to manufacturing flaws. The manufacturing system shall be capable of producing the tailored designs for various automakers.

Scenarios

1. The Manufacturing Division will review all safety regulations published by the National Highway Traffic Safety Administration (NHTSA), send questions and comments to NHTSA on a timely basis, receive responses, and incorporate these regulations into the manufacturing design for air bags.
2. The Manufacturing Division will receive requirements documents on the air bag system from the development team on a periodic basis. The Manufacturing Division will provide comments on these documents as regards any difficulties being forced on the manufacturing of air bags. These comments will be provided on a timely basis.
3. The Manufacturing Division will produce the appropriate number of "built to" configuration items (CIs) based upon the design

documentation and schedule requirements of the development team. In order to produce these "built to" CIs the Manufacturing Division will procure the necessary tools, parts, and supplies.

4. The Manufacturing Division will submit engineering change proposals (ECPs) to the development team as problems are encountered during production. The development team will adopt those ECPs that are warranted, reject those that are not warranted, and comment on the remaining so that an acceptable solution can be found to manufacturing problems. The Manufacturing Division will modify its production process and equipment in accordance with the accepted ECPs.
5. The automakers will send orders for air bags to Corporate Headquarters; Corporate Headquarters will send sales orders to the Manufacturing Division with delivery instructions; the Manufacturing Division will produce the needed air bags and send them to the appropriate automaker; and the Manufacturing Division will send documentation of delivered air bags to Corporate Headquarters.
6. Corporate Headquarters will send periodic projections of air bag production requirements to the Manufacturing Division along with additional corporate guidance regarding cost and quality issues. The Manufacturing Division will send periodic reports on cost and performance data regarding the production of air bags.
7. The Manufacturing Division will send request for quotations (RFQs) to other corporations for the needed tools and parts (CIs) that comprise the air bag system; the Manufacturing Division will receive and review quotes from various corporations and select those quotes providing best value to the Manufacturing Division; and the Manufacturing Division will then send orders for the delivery of the tools and parts on a timely basis and receive these tools and parts.
8. The Manufacturing Division will send request for quotations (RFQs) to other corporations for the needed consumables and supplies; the Manufacturing Division will receive and review quotes from various corporations and select those quotes providing best value to the Manufacturing Division; and the Manufacturing Division will then send orders for the delivery of the consumables and supplies on a timely basis and receive these consumables and supplies.
9. The Manufacturing Division will send that material (unused consumables and supplies, used tools and parts) that needs to be disposed of to Corporate Headquarters.