SANDIA REPORT

SAND96–1620 • UC–706
Unlimited Release
Printed July 1996

Discovering System Requirements

A. Terry Bahill, Bo Bentz, Frank F. Dean

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550
for the United States Department of Energy
under Contract DE-AC04-94AL85000

Approved for public release; distribution is unlimited.

SF2900Q(8-81)

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

Available to DOE and DOE contractors from
Office of Scientific and Technical Information
PO Box 62
Oak Ridge, TN 37831

Prices available from (615) 576-8401, FTS 626-8401

Available to the public from
National Technical Information Service
US Department of Commerce
5285 Port Royal Rd
Springfield, VA 22161

NTIS price codes
Printed copy: A04
Microfiche copy: A01

SAND96-1620
Unlimited Release
Printed July 1996

Distribution
Category UC-706

Discovering System Requirements

A. Terry Bahill
Bo Bentz
Systems and Industrial Engineering
University of Arizona
Tucson, AZ 85721

Frank F. Dean
New Mexico Weapons Systems Engineering Center
Sandia National Laboratories
Albuquerque, NM 87185

Abstract

Cost and schedule overruns are often caused by poor requirements that are produced by people who do not understand the requirements process. This report provides a high-level overview of the system requirements process, explaining types, sources, and characteristics of good requirements. System requirements, however, are seldom stated by the customer. Therefore, this report shows ways to help you work with your customer to discover the system requirements. It also explains terminology commonly used in the requirements development field, such as verification, validation, technical performance measures, and the various design reviews.

Acknowledgments

We thank Patty Guyer for technical editing and Ron Andreé for technical illustrations.

Contents

VOLUME I

FIGURES

VOLUME II

Intentionally Left Blank

Discovering System Requirements

1. Introduction

No two systems are ever exactly alike in their requirements. However, there is a uniform and identifiable process for logically discovering the system requirements regardless of system purpose, size, or complexity (Grady, 1993). The purpose of this report is to expose this process.

This report represents the philosophy and terminology used by the New Mexico Weapons Systems Engineering Center (Directorate 2100) for discovering system requirements. Other organizations may use other procedures and terminology. However, we think a consensus is developing in the Systems Engineering community. It is hoped that this document is consistent with that consensus. Like Systems Engineering in general, the statements in this document are not dogmatic. Each statement has been rightfully violated many times (see for example Martin, 1995). However, these statements are generalizations of good engineering practices.

This report only explains a part of the systems requirements process. Large projects should use a computer system to help write, decompose and maintain system requirements. Many such computer systems are commercially available (NCOSE, 1995). Each project design team will select a specific tool and provide training for it. Therefore, this report will not discuss such tools. An important task in writing system requirements is modeling the proposed system. Dozens of tools are available; two recently popular ones are object-oriented design and functional decomposition (Bharathan, Poe, and Bahill, 1995). This report does not discuss tools for modeling systems.

Intentionally Left Blank

2. Stating the Problem

Stating the problem is one of the System Engineer's most important tasks in developing requirements. The problem must be stated in a clear, unambiguous manner.

State the problem in terms of what the world would be like if the problem did not exist, and not in terms of preconceived solutions. Do not believe the first thing your customer says. Verify the problem statement with the customer, and expect to iterate this procedure several times. For an excellent (and enjoyable) reference on stating the problem, see Gause and Weinberg (1990).

It is good engineering practice to state the problem in terms of the top-level function that the system must perform. However, it is better to state the problem in terms of the deficiency that must be ameliorated. This stimulates consideration of more alternative designs.

Example 1

Top-level function:	Design a system that will hold together 2 to 20 pieces of 8½ by 11-inch, 20 pound paper.
Alternatives:	stapler, paper clip, fold the corner, put them in a folder.

Example 2

The deficiency:	My reports are typically composed of 2 to 20 pieces of 8½ by 11-inch, 20 pound paper. The pages get out of order and get mixed up with pages of other reports.
Alternatives:	stapler, paper clip, fold the corner, put them in a folder, number the pages, put them in an envelope, throw away the report, convert it to electronic form, have it bound as a book, put it on audio tape, distribute it electronically, put it on a floppy disk, put it on microfiche, transform the written report into a videotape.

2.1 Do Not Use the Word Optimal

The word optimal should not appear in the statement of the problem, because there is no single optimal solution to complex systems problems. Most system designs have several performance and cost criteria. Systems Engineering creates a set of alternative designs that satisfies these performance and cost criteria to varying degrees. Moving from one alternative to another will usually improve at least one criterion and worsen at least one criterion (i.e., there will be trade-offs). None of the feasible alternatives is likely to optimize all the criteria (Szidarovszky, Gershon, and Duckstein, 1986). Therefore, we must settle for less than optimality. Some subsystems may be considered optimal, but when they are interconnected, the overall system will not be optimal; the best possible system is not that made up of optimal subsystems. Furthermore, if the system requirements demanded an optimal system, data could

not be provided to prove that any resulting system was indeed optimal. In general, it can be proven that a system is at a local optimum, but it cannot be proven that it is at a global optimum.

If it is required that optimization techniques be used, then they should be applied to subsystems. However, total system performance must be analyzed to decide if the cost of optimizing a subsystem is worthwhile. Furthermore, total system performance should be analyzed over the whole range of operating environments, because what is optimal in one environment will not necessarily be optimal in another.

2.2 Definition of Customer

The term *customer* includes anyone who has a right to impose requirements on the system. This includes end users, operators, bill payers, owners, regulatory agencies, victims, sponsors, etc. Because Systems Engineering delivers both a *product* and a *process* for manufacturing it, we must also consider the customer of the process.

Let us now illustrate some of these roles for a commercial airliner, such as the Boeing 777. The users are the passengers that fly on the airplane. The operators are the crew that fly the plane and the mechanics that maintain it. The bill payers are the airline companies, such as United, TWA, etc. The owners are the stockholders of these companies. The Federal Aviation Administration (FAA) writes the regulations and certifies the airplane. Among others, people who live near the airport are victims of noise and air pollution. If the plane is tremendously successful, McDonnell Douglas (the manufacturer of a competing airplane) would also be a victim. The sponsor would be the corporate headquarters of, for example, Boeing.

The users and operators of the process would be the employees in the manufacturing plant. The bill payer would be Boeing. The owner would be the stockholders of Boeing. Occupational Safety and Health Administration (OSHA) would be among the regulators. Victims would include physically injured workers and, according to Deming, workers stuck doing mindless, repetitive tasks who have little control of the output but are reviewed for performance (Latzko and Saunders, 1995).

2.3 Who Is the Audience?

Before writing a document you should consider who the audience is going to be. For a requirements document, the audience is the client and the designers.

System requirements communicate the customer's needs to the technical community that will design and build the system, and therefore they must be understandable by both. One of the most difficult tasks in creating a system is communicating with all subgroups within both groups (IEEE 1233).

The client and the designers have different backgrounds and needs. Wymore (1993) suggests two different documents for these two different groups: The Operational Need Document for the client and the System Requirements Document for the design engineers.

> The Operational Need Document is a detailed description of the problem in plain language. It is intended for management, the customer and systems engineering....The Systems Requirement Document is a succinct mathematical description or model of the...requirements as described in the Operational Need Document. Its audience is systems engineering.
>
> (Chapman, Bahill, and Wymore, 1992)

Sometimes these are referred to as customer requirements and technical requirements, respectively.

Intentionally Left Blank

3. What Are Requirements?

Requirements are the necessary attributes defined for a system before design development. The customer's need is the ultimate system requirement from which all other requirements and designs flow (Grady, 1993). In addition, requirements are statements that identify the essential needs of a system in order for it to have value and utility. Requirements may be derived or based upon interpretation of other stated requirements to assist in providing a common understanding of the desired characteristics of a system. Finally, requirements should state what the system is to do, but they should not specify how the system is to do it. Section 3.1 is an example of a requirement.

3.1. A Graphic-Editor Grid Facility (Sommerville, 1989)

To assist in positioning items on a diagram, the user may turn on a grid in either centimeters or inches, via an option on a control panel. Initially the grid is off. The grid may be turned on and off at any time during an editing session and can be toggled between inches and centimeters at any time. The grid option will also be provided on the reduce-to-fit view, but the number of grid lines shown will be reduced to avoid filling the diagram with grid lines.

Good points about this requirement: It provides rationale for the items. It explains why there should be a grid. It explains why the number of grid lines should be reduced for the reduce-to-fit view. It provides initialization information: initially the grid is off.

Bad points: The first sentence has three different components: (1) it states that the system should provide a grid, (2) it gives detailed information about grid units (centimeters and inches), and (3) it tells how the user will activate the grid. It provides initialization information for some but not all similar items: it specifies that initially the grid is off, but it does not specify the units when it is turned on. Section 3.2 shows how the requirement might be improved.

3.2 An Improved Requirement for the Graphic-Editor Grid Facility (Sommerville, 1989)

3.2.1 The Grid

3.2.1.1 The graphic-editor grid facility shall produce a pattern of horizontal and vertical lines forming squares of uniform size as a background to the editor window. The grid shall be passive rather than active. This means that alignment is the responsibility of the user and the system shall not automatically align items with grid lines.

Rationale: A grid helps the user to create a neat diagram with well spaced entries. Although an active grid might be useful, it is best to let the user decide where the items should be positioned.

3.2.1.2 When used in the "reduce-to-fit" mode the logical grid line spacing should be increased.

Rationale: If the logical grid line spacing were not increased the background would become cluttered with grid lines.

Specification: ECLIPSE/WORKSTATION/DEFS. Section x.y

This requirement definition references the requirement specification, which provides details such as units of centimeters and inches and the initialization preferences.

4. The Requirements Discovery Process

Requirements discovery is one subprocess of the Systems Design Process shown in Figure 1. Systems Engineering is a fractal process. It is applied at levels of greater and greater detail: It is applied to the system, then to the subsystems, then to the components, etc. It is applied to the system being designed and also to the enterprise in which the system will operate. This concept is shown in a poster that is available at

http://www.sie.arizona.edu/sysengr or

http://www.sie.arizona.edu (click on systems engineering) and at

http://dpopenet.sandia.gov/syseng/index.html

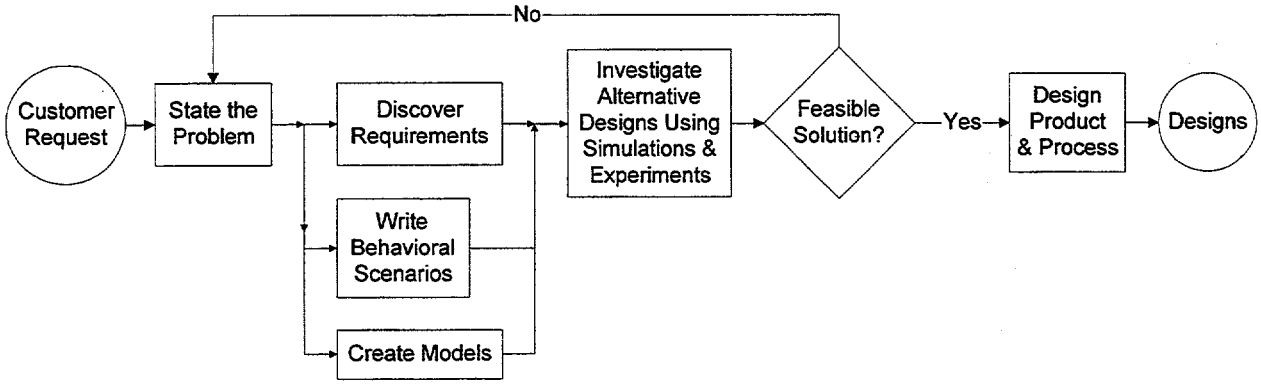

Figure 1. The system design process.

4.1 Identify Customers and Stakeholders

The first step in developing requirements is to identify the customer. The term *customer* includes anyone who has a right to impose requirements on the system. This includes end users, operators, bill payers, owners, regulatory agencies, victims, sponsors, etc. All facets of the customer must be kept in mind during system design. For example, in evaluating the cost of a system, the total life cycle cost and the cost to society should be considered. Frequently, the end user does not fund the cost of development. This often leads to products that are expensive to own, operate, and maintain over the entire life of the product, because the organization funding development saves a few dollars in the development process. It is imperative that the Systems Engineer understand this conflict and expose it. The sponsor and user can then help trade off the development costs against the cost to use and maintain. Total life cycle costs are significantly larger than initial costs. For example, in one of their advertisements, Compaq proclaimed "80% of the lifetime cost of your company's desktops comes after you purchase them." In terms of the

personal computer, if total life cycle costs were $10,000, purchase cost would have been $2,000 and maintenance and operation $8,000.

4.2 Understand the Customer's Needs

The system design must begin with a complete understanding of the customer's needs. The information necessary to begin a design usually comes from preliminary studies and specific customer requests. Frequently the customer is not aware of the details of what is needed. Systems Engineers must enter the customer's environment, discover the details, and explain them. Flexible designs and rapid prototyping facilitate identification of details that might have been overlooked. Talking to the customer's customer and the supplier's supplier can also be useful. This activity is frequently referred to as mission analysis.

It is the Systems Engineer's responsibility to ensure that all information concerning the customer's needs is collected. The Systems Engineer must also ensure that the definitions and terms used have the same meaning for everyone involved. Several direct interviews with the customer are necessary to ensure that all of the customer's needs are stated and that they are clear and understandable. The customer might not understand the needs; he may be responding to someone else's requirements. Often, a customer will misstate his needs; for example, a person might walk into a hardware store and say he needs a half-inch drill bit. But what he actually needs is a half-inch *hole* in a metal plate, and a chassis-punch might be more suitable.

If the organization does not have a Vision or Mission statement, then you should write one.

4.3 Define and State the Problem

What is the problem we are trying to solve? Answering this question is one of the Systems Engineer's most important and often overlooked task. An elegant solution to the wrong problem is less than worthless.

Early in the process, the customer frequently fails to recognize the scope or magnitude of the problem that is to be solved. The problem should not be described in terms of a perceived solution. It is imperative that the Systems Engineer help the customer develop a problem statement that is completely independent of solutions and specific technologies. Solutions and technologies are, of course, important; however, there is a proper place for them later in the Systems Engineering process. It is the Systems Engineer's responsibility to work with the customer, asking the questions necessary to develop a complete "picture" of the problem and its scope. The Air Force customer did not know that they wanted a stealth airplane until after the engineers showed that they could do it.

Figure 2, based on Grady (1995), shows the requirements discovery process. This whole diagram is the "Discover Requirements" box of the System Design Process shown in Figure 1.

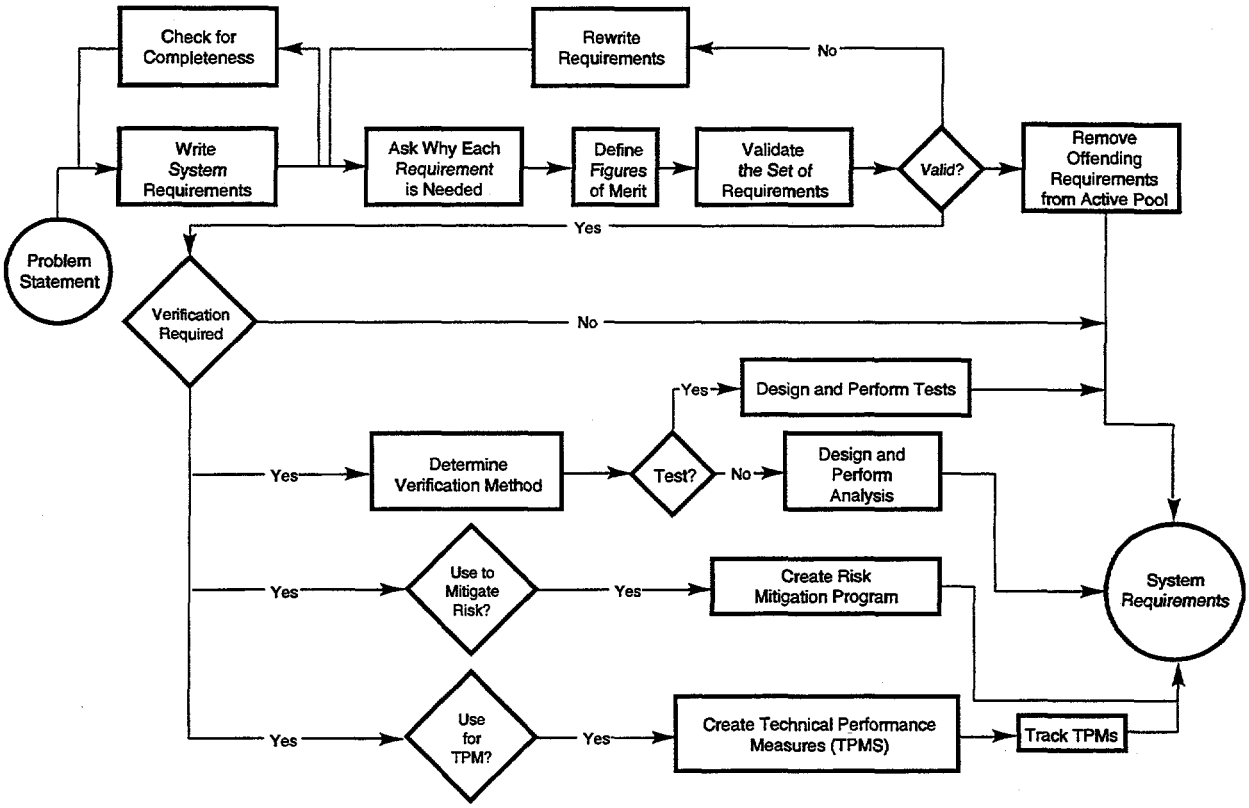

Figure 2. The requirements discovery process.

4.4 Write System Requirements

The process of developing and specifying requirements is often referred to as Requirements Analysis. The Systems Engineer must interact with the customer to develop the requirements. The Systems Engineer must involve the customer in the process of defining, clarifying, and prioritizing the requirements. It is prudent to involve users, bill payers, regulators, manufacturers, maintainers, and other key players in the process.

Next Systems Engineering must discover the functions that the system must perform in order to satisfy its purpose. The system functions form the basis for dividing the system into subsystems. QFD is useful for identifying system functions (Bahill & Chapman, 1993; Bicknell & Bicknell, 1994).

Although this makes it sound as if requirements are transformed into functions in a serial manner, that is not the case. It is actually a parallel and iterative process. First we look at system requirements, then at system functions. Then we re-examine the requirements and then re-examine the functions. Then we re-assess the requirements and again the functions, etc.

4.5 Consult with the Customer

The Systems Engineer must consult with the customer to ensure that the requirements are correct and complete. The Systems Engineer and the customer should identify which requirements can be used as trade-off requirements. The customer should be satisfied that if these requirements are met, then the system will do what it really needs to do. This should be done in formal reviews with the results documented and distributed to appropriate parties. All parties must agree to a way of measuring system performance to ensure that the system does what the customer wants it to do.

At these reviews it is important to ask why each requirement is needed. This can help to reduce the number of requirements.

Sometimes the customer is not available for consultation. In such unfortunate situations, a surrogate customer will have to be used.

4.6 Define Performance and Cost Figures of Merit

Figures of merit are the criteria on which the different designs will be "judged." Each figure of merit must have a fully described unit of measurement. Units of power could be horsepower, for example, and units of cost could be dollars (or inverse dollars if it is desirable to consistently have "more is better" situations). Suppose a figure of merit were acceleration, then the unit of measurement could be seconds taken to get from 0 to 60 mph. The units of measurement can be anything, as long as they measure the appropriate criterion, are fully described, and are used consistently for all designs. The value of a figure of merit describes how effectively a preference requirement has been met. For example, a car went from 0 to 60 in 6.5 seconds. It is these values that are often put into the scoring functions, (Section 6.1, Figure 5), to give the requirements scores, which are in turn used to perform trade-off studies. Such measurements are made throughout the development of the system.

4.7 Validate System Requirements

Validating requirements means ensuring that the requirements are consistent and that a real-world solution can be built and tested to prove that it satisfies the requirements. Each requirement should be technically feasible and fit within budget, schedule, and other constraints. Requirements are often validated by reference to an existing system that meets most of the requirements. The requirements that are not satisfied by the existing system are validated by argument, modeling, or simulation.

4.8 Describe the Verification Process

A critical element of the requirements development process is describing the tests, analysis or data that will be used to prove compliance of the final system with its requirements. Each test must explicitly link to a specific requirement; this will help expose untestable requirements. The specification of the testing process informs the producers of the systems how the system will be

tested. In other words, they know how they will be "graded." This process frequently uncovers overlooked requirements.

At this time it may be useful to examine the following definitions.

Validating a System: Building the right system; making sure that the system does what it is supposed to do. It determines the correctness of an end product, compliance of the system with the customer's needs, and completeness of the system.

Validating Requirements: Ensuring that the set of requirements is consistent, that a real-world solution can be built that satisfies the requirements, and that it can be proven that such a system satisfies its requirements. If Systems Engineering discovers that the customer has requested a perpetual-motion machine, the project should be stopped.

Verifying a System: Building the system right; ensuring that the system complies with its requirements. It determines the conformance of the system to its design requirements. It also guarantees the consistency of the product at the end of each phase, with itself and with the previous prototypes. In other words, it guarantees the honest and smooth transition from model to prototype to preproduction unit to production unit.

Verifying Requirements: Examination, analysis, test, or demonstration that proves whether a requirement has been satisfied. This process is iterative. The requirements should be verified with respect to the model, the prototype, the preproduction unit, and the production unit.

Verification and Validation: MIL-STD-1521B (and most Systems Engineers) and DoD-STD-2167A (and most software engineers) use the words verification and validation in almost the exact opposite fashion. For Systems Engineers, to validate a set of requirements is to prove that it is possible to satisfy them. System verification, on the other hand, is a process of proving that a system meets its requirements (Grady, 1994). To add further confusion, ISO-9000 tells you to verify that a design meets the requirements and validate that the product meets requirements. NASA has a different spin. It says that verification consists of proving that a system (or a subsystem) complies with its requirements, whereas validation consists of proving that the total system accomplishes its purpose. (Shishko, 1995). Thus it is necessary to agree on the definitions of verification and validation as these terms pertain to your system.

4.9 Define Technical Performance Measures

Technical performance measures (TPMs), or metrics, are used to track the progress of the design and manufacturing process. They are measurements that are made during the design and manufacturing process to evaluate the likelihood of satisfying the system requirements. Not all requirements have TPMs, just the most important ones. In the beginning of the design and manufacturing process, the prototypes will not meet the TPM goals. Therefore the TPM values are only required to be within a tolerance band. It is hoped that as the design and manufacturing process progresses the TPM values of the prototypes and preproduction units will come closer and closer to the goals.

As an example, let us consider the design and manufacture of solar ovens (Funk & Larson, 1994). In many societies, particularly in Africa, many women spend as much as 50% of their time acquiring wood for their cooking fires. To ameliorate this sink of human resources, people have been designing and building solar ovens. Let us now examine the solar oven design and manufacturing process that we followed in a Freshman Engineering class at the University of Arizona.

First we defined a TPM for our design and manufacturing process. When a loaf of bread is finished baking, its internal temperature should be 95°C (203°F). To reach this internal temperature, commercial bakeries bake the loaf at 230°C (446°F). As initial values for our oven temperature TPM, we chose a lower limit of 100°C, a goal of 230°C, and an upper limit of 270°C. The tolerance band shrinks with time as shown in Figure 3.

In the beginning of the design and manufacturing process, our day-by-day measurements of this metric increased because of finding better insulators, finding better glazing materials (e.g., glass and mylar), sealing the box better, aiming at the sun better, etc.

At the time labeled "Design Change-1," there was a jump in performance caused by adding a second layer of glazing to the window in the top of the oven. This was followed by another period of gradual improvement as we learned to stabilize the two pieces of glazing material.

At the time labeled "Design Change-2," there was another jump in performance caused by a design change that incorporated reflectors to reflect more sunlight onto the window in the oven top. This was followed by another period of gradual improvement as we found better shapes and positions for the reflectors.

But, in this case, it seemed that we might not attain our goal. Therefore we re-evaluated the process and the requirements. Bread baking is a complex bio-chemical process that has been studied extensively: Millions of loaves have been baked each day for the last four thousand years. These experiments have revealed the following consequences of insufficient oven temperature:

(1) Enzymes are not deactivated soon enough and excessive gas expansion causes coarse grain and harsh texture.

(2) The crust is too thick, because of drying caused by the longer duration of baking.

(3) The bread becomes dry, because prolonged baking causes evaporation of moisture and volatile substances.

(4) Low temperatures cannot produce carmelization and crust color lacks an appealing bloom.

After consulting some bakers, our managers decided that 190°C (374°F) would be sufficient to avoid the above problems. Therefore, the requirements were changed at the indicated spot and our TPM was then able to meet our goal. Of course this change in requirements forced a review

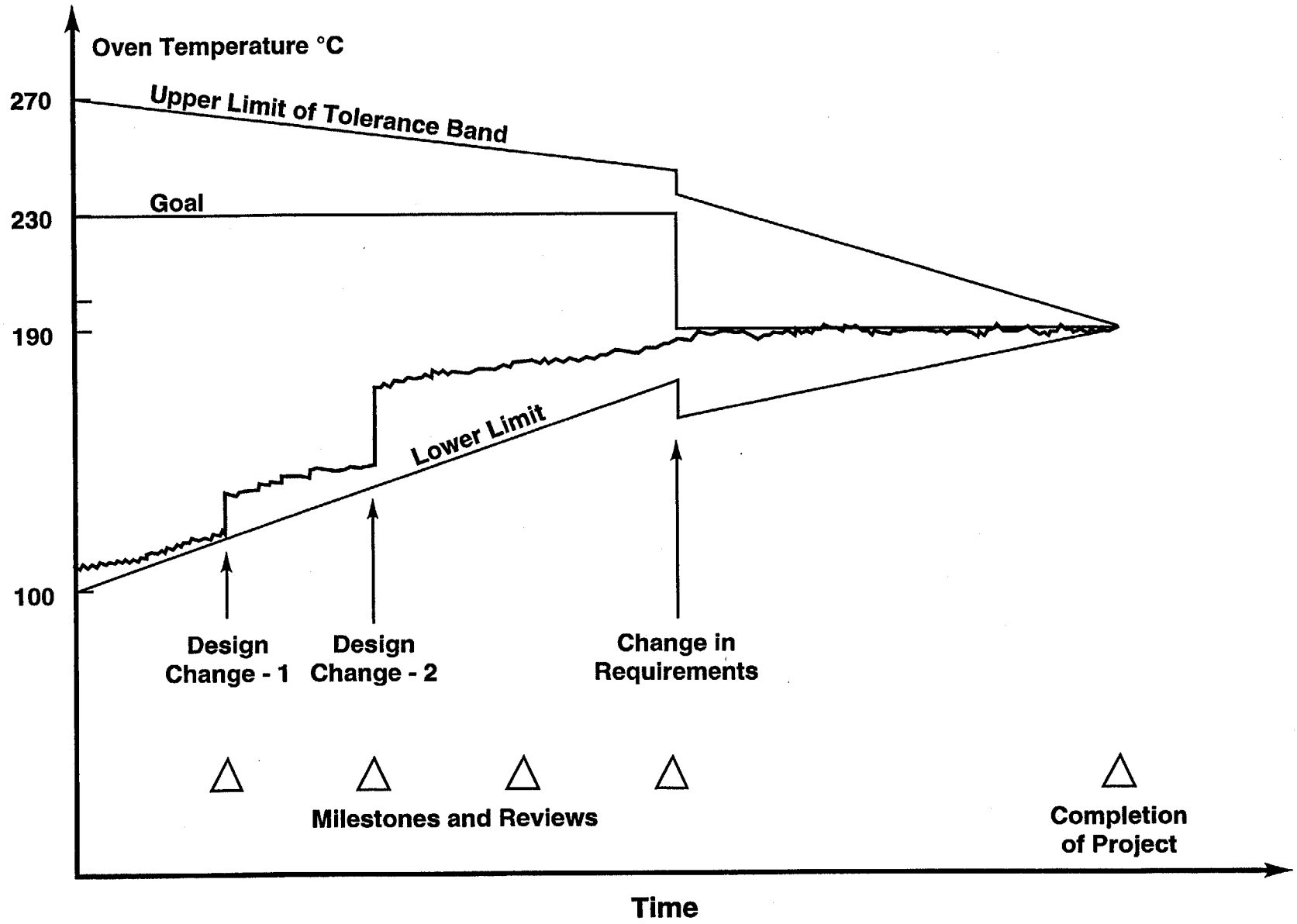

Figure 3. A technical performance measure.

of *all* other requirements and a change in many other facets of the design. For example the duration weight tables had to be re-computed.

If sugar, eggs, butter and milk were added to the dough, we could get away with temperatures as low as 175°C (347°F). But we decided to design our ovens to match the needs of our customers, rather than try to change our customers to match our ovens.

4.10 Risk Mitigation

Identifying and mitigating program risk is the responsibility of management at all levels in the company. Each item that poses a threat to the cost, schedule or performance of the project must be identified and tracked. The following information should be recorded for each identified risk: name, description, type, origin, probability, severity, identification number, identification date, identification on the work breakdown structure, risk mitigation plan, responsible team, needed resolution date, principal engineer, current status, date, signature of team leader. Forms useful in identifying and mitigating risk are given in chapter 17 of Kerzner (1995), and Section 4.10 of Grady (1995) and chapter 3 of this handbook. For the solar oven project we identified the following risks.

(1) Insufficient internal oven temperature was a performance risk. Its origin was the Design project area. It had high probability and high severity. We mitigated it by making it a technical performance measure, as shown in Figure 3.

(2) High cost of the oven was a cost risk. Its origin was the Design process. Its probability was low, and its severity was medium. We mitigated it by computing the cost for every design.

(3) Manufacturing the oven to be tested posed a schedule risk. Its origin was Design and Manufacturing. Its probability was low, but its severity was very high. We mitigated this risk by requiring final designs seven days before the scheduled test date and a preproduction unit three days in advance.

4.11 Review System Requirements

The system requirements must be reviewed with the customer many times. At a minimum they should be reviewed at the end of the modeling phase, after testing the prototypes, before commencement of production, and after testing production units.

The objective of these reviews is to find missing requirements, ensure that the requirements have been met, and verify that the system satisfies customer needs. Additional objectives include assessing the maturity of the development effort, recommending whether to proceed to the next phase of the project, and committing additional resources. These reviews should be formal. The results and conclusions of the reviews should be documented. The Systems Engineer is responsible for initiating and conducting these reviews.

The following definitions based on Sage (1992) and Shishko (1995) might be useful. They are arranged in chronological order. Although these definitions are written with a singular noun, they are often implemented with a collection of reviews. Each system, subsystem, subsubsystem, etc. will be reviewed and the totality of these constitutes the indicated review.

Mission Concept Review: The Mission Concept Review and the Mission Definition Review are the first formal reviews. They examine the mission objectives and the functional and performance requirements.

System Requirements Review (SRR): Demonstrates that the product development team understands the mission and the system requirements. It confirms that the system requirements are sufficient to meet mission objectives. It ensures that the performance and cost figures of merit are realistic, and that the verification plan is adequate.

System Definition Review: Examines the proposed system architecture, the proposed system design, and the flow down of functions to the major subsystems. It also ensures that the verification plan is complete. This is sometimes called the Conceptual Design Review

Preliminary Design Review (PDR): Demonstrates that the preliminary design meets all the system requirements with acceptable risk. System development and verification tools are identified, and the Work Breakdown Structure is examined. Full-scale engineering design begins after this review.

Critical Design Review (CDR): Verifies that the design meets the requirements. The CDR examines the system design in full detail, ensures that technical problems and design anomalies have been resolved, checks the technical performance measures, and ensures that the design maturity justifies the decision to commence manufacturing. Few requirements should be changed after this review.

Production Readiness Review (PRR): For some systems there is a long phase when prototypes are built and tested. At the end of this phase, and before production begins, there is a production readiness review.

System Test: At the end of manufacturing and integration, the system is tested to verify that it satisfies its requirements. Technical performance measures are compared to their goals. The results of these tests are presented at the System Acceptance and Operational Readiness Reviews.

Figure 4 shows the timing of these major reviews.

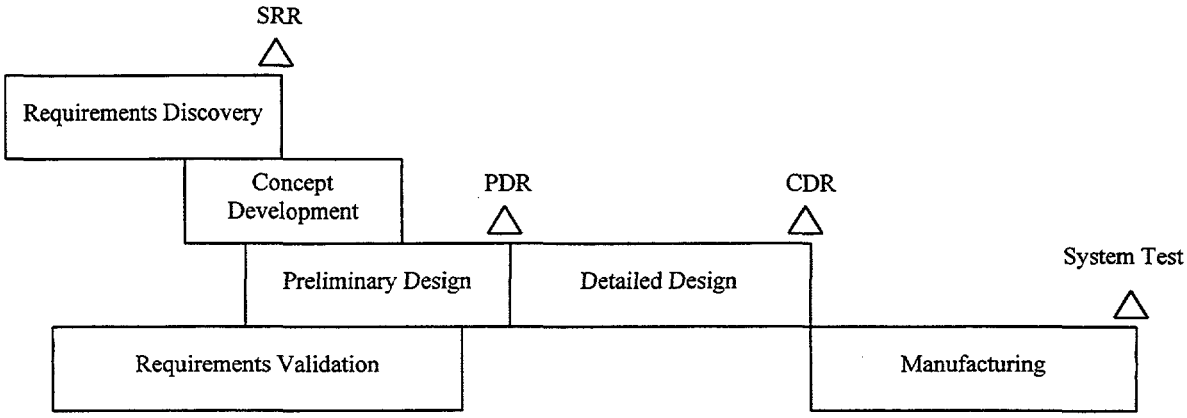

Figure 4. Timing of the major reviews.

5. Characteristics of a Good Requirement

5.1 Describes What, Not How

There are many characteristics of a good requirement. First and foremost, a good requirement defines what a system is to do and to what extent, but does not specify how the system is to do it. A statement of a requirement should not be a preconceived solution to the problem that is to be solved. It would be a mistake to require a relational database for the requirements. To avoid this trap, ask why the requirement is needed, then derive the real requirements. For example, the following requirements state what is needed, not how to accomplish it: provide the ability to store, provide the ability to sort, provide the ability to add attributes.

It should be noted that because QFD is often used iteratively to define requirements, the *hows* in one QFD chart become the *whats* in the next, making the above statements confusing.

5.2 Atomic

A requirement should be "atomic," not compound. That is, it should have a single purpose (one idea per requirement). Furthermore, each requirement should be allocated to a single entity.

5.3 Unique

A requirement should have a unique label, a unique name, and unique contents. Avoid repeating requirements.

5.4 Documented and Accessible

A requirement must be documented (writing, pictures, images, databases, etc.) and the documentation must be accessible. In situations where confidentiality is important, each requirement should clearly indicate classification status. Only those with appropriate clearance and the need to know should have access to that requirement.

5.5 Identifies Its Owner

A good requirement will identify its owner and custodian, which could be one and the same person. The requirement's owner must approve of any change in the requirement.

5.6 Approved

After a requirement has been revised, reviewed, and rewritten, it must be approved by its owner. Furthermore, each top-level requirement must be approved by the customer.

5.7 Traceable

A good requirement is traceable; it should be possible to trace each requirement back to its source. A requirement should also identify related requirements (i.e., parents, children, siblings, cousins).

5.8 Necessary

All requirements should be necessary. System Engineers should ask, "Is this requirement really necessary? Will the system necessarily be better because of this requirement?" Avoid over-specifying the system, writing pages and pages that no one will probably ever read. There are two common types of over specification: gold plating and specifying unnecessary things. For example, requiring that the outside of a CPU box be gold-plated is not a good requirement because something far less expensive would probably be just as effective. Also, requiring that the inside of the CPU box be painted pink is probably an unnecessary request. Over-specification (of both types) is how $700 toilet seat covers and $25,000 coffee pots get created (Hooks, 1994). The documentation should include a complete statement of the rationale behind each requirement.

5.9 Complete

The documentation must be as clear, concise, and complete as possible.

5.10 One Semantic Definition

Avoid the use of synonyms (e.g., The software requires 8 Mbytes of RAM but 12 Mbytes of memory are recommended) and homonyms (e.g., Summaries of disk X-rays should be stored on disk). There should only be one semantic definition of each requirement.

5.11 Is Not Always Written

It must be noted that all systems will undoubtedly have many "common sense" requirements that will not be written. This is acceptable as long as the requirements really are common sense. An exhaustive list of requirements would take years upon years and use reams of paper, and even then you would probably never finish.

5.12 Quantitative and Testable

Quantitative values must be given in requirements. A requirement states a necessary attribute of a system to be designed. The designer cannot design the system if a magnitude is not given for each attribute. Without quantification, system failure could occur because of: (1) exceeding the minimum necessary cost due to over design, or (2) failing to account for a needed capability. Quantitative values for attributes are also necessary in order to test the product to verify that it satisfies its requirements (Grady, 1993).

A requirement must be verifiable by examination, analysis, test, or documentation and therefore it must have well-defined technical performance measurements. Qualitative words like

low and *high* shall be (at least roughly) defined. What is low cost to a national laboratory and what is low cost to a small company may be very different. Only requirements that are clear and concise will be easily testable. Requirements with ambiguous qualifiers will probably have to be refined before testing will be possible. Furthermore, the value given should be fully described as, for example, an expected value, a median, a minimum, a maximum, etc. A requirement such as "reliability shall be at least 0.999" is a good requirement because it is testable, quantified, and the value is fully described as a minimum. Also the requirement "the car's gas mileage should be about 30 miles per gallon" is a good requirement as it establishes a performance measure and an expected value.

Note that often the customer will state a requirement that is not quantified. For example: "The system should be aesthetically pleasing." It is then the engineer's task to define a requirement that is quantified, i.e., "The test for aesthetics will involve polling two hundred potential users; at least 70% should find the system aesthetically pleasing."

5.13 Identifies Applicable States

Some requirements only apply when the system is in certain states or modes. If the requirement is only to be met sometimes, the requirement statement should reflect when. There may be two requirements that are not intended to be satisfied simultaneously, but they could be at great expense.

For example: The vehicle shall

(1) be able to tow a 2000-pound cargo trailer at highway speed (65 mph),

(2) accelerate from 0 to 60 mph in less than 9.5 seconds.

It would be expensive to build a car that satisfied both requirements simultaneously.

R Your Lights On?

However, as with everything, you can take this principle too far, as illustrated by the following, which is probably a true story. We first saw it in Gause and Weinberg (1990).

Recently the highway department tested a new safety proposal. They asked motorists to turn on their headlights as they drove through a tunnel. However, shortly after exiting the tunnel the motorists encountered a scenic-view overlook. Many of them pulled off the road to look at the beautiful hills and valleys that stretched as far as the eye could see. When they returned to their cars, they found that their car batteries were dead because they had left their headlights on. So the highway department decided to erect signs to get the drivers to turn off their headlights.

First they tried "Turn your lights off." But someone said that not everyone would heed the request to turn their headlights on. And it would be impossible for these drivers to turn their headlights off.

So they tried "If your headlights are on, then turn them off." But someone objected that would be inappropriate if it were night time.

So they tried "If it is daytime and your headlights are on, then turn them off." But someone objected that would be inappropriate if it were overcast and visibility was greatly reduced.

So they tried "If your headlights are on and they are not required for visibility, then turn them off." But someone objected that many new cars are built so that their headlights are on whenever the motor is running.

So they tried "If your headlights are on, and they are not required for visibility, and you can turn them off, then turn them off." But someone objected.

So they decided to stop trying to identify applicable states. They would just alert the drivers and let them make the appropriate actions. Their final sign said "Are your lights on?"

5.14 States Assumptions

All assumptions should be stated. Unstated bad assumptions are one cause of bad requirements. Frequently one assumes that weapons are submarine based, but this will not necessarily always be true, even for the Navy.

5.15 Use of Shall, Should, and Will

A mandatory requirement should be expressed using the word *shall* (e.g., The system shall conform to all state laws.). A preference requirement can be expressed using *should* or *may* (e.g., The total cost for the car's accessories should be about 10% of the total cost of the car.). The term *will* can be used to express a declaration of purpose on the part of a contracting agency, to express simple future tense, and for statement of fact (e.g., The resistors will be supplied by an outside manufacturer.) (Grady, 1993).

5.16 Does Not Use Certain Words

The words *optimize, maximize*, and *minimize* should not be used in stating requirements, because we could never prove that we were there. Consider the following criteria: (1) we should minimize human suffering, and (2) we should maximize the quality and quantity of human life. A starving child should be fed, even if the child continues to live in misery. However, the criteria of minimal suffering could lead to the conclusion that the child should die.

Requirements should not use the word *simultaneous* because it means different things to different people. Simultaneous might well mean anything from within a few fempto seconds to a millennium.

5.17 Might Vary in Level of Detail

The amount of detail in the requirements depends upon the intended supplier. For in-house work or work to be done by a supplier with well-established systems engineering procedures, the

requirements can be written at a high level. However, for outside contractors with unknown systems engineering capabilities, the requirements might be broken down to a very fine level of detail.

5.18 Respects the Media

Newspaper journalists quote out of context, and headlines do not reflect the content of their stories. It is important to write each requirement so that it cannot spark undue public criticism of your project.

Intentionally Left Blank

6. Requirements Characterizations

There are many orthogonal characterizations of system requirements. Four of those are: types, sources, expressions or modalities, and input-output trajectories. A thumbnail synopsis of the characterizations follows.

6.1 Types of Systems Requirements

There are two types of system requirements: mandatory and preference.

Mandatory requirements:

(1) specify the necessary and sufficient conditions that a minimal system must have in order to be acceptable and are usually expressed with shall and must,

(2) are passed or failed (must not use scoring functions), and

(3) must not be susceptible to trade-offs between requirements.

When mandatory requirements have been identified, Systems Engineers propose alternative candidate designs, all of which satisfy the mandatory requirements. Preference requirements are then evaluated to determine the "best" designs.

Preference requirements:

(1) state conditions that would make the customer happier and are often expressed with *should* and *want*,

(2) should use scoring functions (Chapman, Bahill, and Wymore, 1992) to produce figures of merit, and

(3) should be evaluated with a multicriteria decision technique (Szidarovszky, Gershon, and Duckstein, 1986) because none of the feasible alternatives is likely to optimize all the criteria, and there will be trade-offs between these requirements.

> *Example*: 1.3 Competing Characteristics Criteria. As a rule, priority of consideration shall be given to nuclear safety, reliability and other operational characteristics and restrictions, in that order. It is understood that technical feasibility, schedule, and cost are to provide the basis for making trade-offs among the desired competing characteristics.

Sometimes there is a relationship between mandatory and preference requirements in which a mandatory requirement might be a lower threshold of a preference requirement. For example, computer software where 8 Mbytes of RAM are required, but 12 Mbytes are preferred.

A scoring function is used to give a system a normalized score that reflects how the requirement has been met for each criterion. The value of the figure of merit, using the example of Mbytes of RAM (Figure 5), is put into the scoring function and a normalized score is returned. The use of scoring functions allows different criteria to be compared and traded off against each other. In other words, scoring functions allow apples to be compared to oranges and nanoseconds to be compared to billions of dollars.

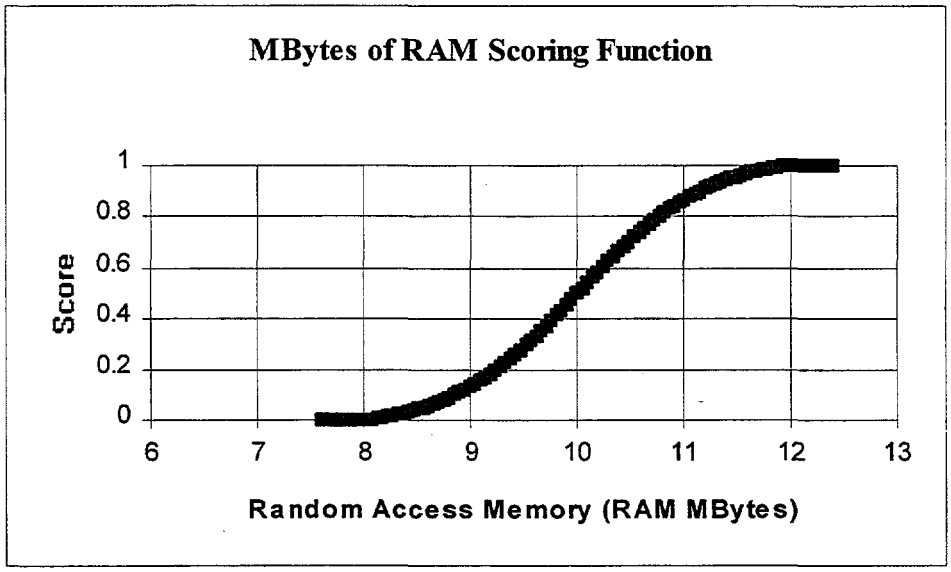

Figure 5. A scoring function for the amount of RAM.

6.2 *Sources of Systems Requirements*

There are many sources of requirements. The following is by no means an exhaustive list; however, Wymore (1993) says that only the first six categories are necessary: Input-Output, Technology, Performance, Cost, Trade-off, and System Test. He says all of the other sources can be put into one of these six. Grady (1993) says we should have only five categories: Functional, Performance, Constraints, Verification, and Programmatic. He thinks that most of our sources are constraints. The EIA/ANSI 632 Standard on Systems Engineering says there are only three: Functional, Performance, and Constraints (*Martin, personal communication*). We leave it to the reader to decide whether or not our list of sources can be condensed.

6.2.1 Input-Output

Wymore (1993) maintains that functional requirements are a subset of input-output requirements. If an input-output requirement is very tight, then it describes a function. For example, an input-output requirement for an electronic amplifier could be stated as "the ratio of the output to the input at 10 kHz shall be 20 dB." This input-output requirement describes the function "Amplify the input signal."

The functional requirement "The system shall fasten pieces of paper" is covered by the input-output requirement "The system shall accept 2 to 20 pieces of 8½ by 11 inch, 20 pound paper and fasten them together."

6.2.2 Technology

The technology requirement specifies the set of components — hardware, software, and bioware — that are available to build the system. The technology requirement is usually defined in terms of types of components that *cannot be used,* that *must be used,* or both. For example, a Sandia weapon component was required to use CMOS and MNOS integrated circuits with three - four micron lines. Further, the Intel 80×86 microprocessor family was required and the Motorola 68xxx family was forbidden. Admiral Rickover required that submarine nuclear instrumentation be done with magnetic amplifiers. Also, the Purchasing Department will often be a source of technology constraints.

6.2.3 Performance

Performance attributes include quantity (how many, how much), quality (how well), coverage (how much area, how far), timeliness (how responsive, how frequent), and readiness (availability, MTBF). An example of performance could be that a car shall accelerate from 0 to 60 mph in 7 seconds or less. Performance is an attribute of products and processes. Its requirements are initially defined through requirements analyses and trade studies using customer need, objective, and/or requirements statements (*MIL-STD-499B*).

> *Example*: 2.3 Performance. The system shall carry xx reentry systems to a range of xyz nautical miles.

6.2.4 Cost

An example cost requirement would be that the purchase price cannot be more than $10,000 and the total life cycle cost cannot exceed $18,000.

6.2.5 Trade-off

Trade-off between performance and cost is defined as the different relative value assigned to each factor. For example, the performance figures of merit may have a weight of 0.6 and the cost figures of merit may be given a weight of 0.4.

6.2.6 System Test

The purpose of system test is to verify that the design and the system satisfy the requirements. For example, in an electronics amplifier, a 3-mV, 10-kHz sinusoid will be applied to the input, and the ratio of output to input will be calculated.

6.2.7 Company Policy

Company policy is another way of stating requirements. For example, Learjet, Inc., has stated "We will make the airframe, but we will buy the jet engines and the electronic control systems."

6.2.8 Business Practices

Corporate business policies might require Work Breakdown Structures, PERT Charts, Quality Manuals, ES&H Plans, or a certain return on investment.

6.2.9 Systems Engineering

Systems or Software Engineering might require that every transportable disk (e.g., floppy or Bernoulli) have a Readme file that describes the author, date, contents, software program, and version (e.g., Word 6.0 or Excel 4.0).

6.2.10 Project Management

Access to source code for all software might be a project management requirement. It takes time and money to install new software. This investment would be squandered if the supplier went bankrupt and the customer could no longer update and maintain the system. Therefore, most customers would like to have the source code. However, few software houses are willing to provide source code, because it might decrease their profits and complicate customer support. When there is any possibility that the supplier might stop supporting a product, the source code should be provided and placed in escrow. This source code remains untouched as long as the supplier supports the product. But if the supplier ceases to support the product, the customer can get the source code and maintain the product in house. Therefore, placing the source code in escrow can be a requirement. Cost, schedule and performance requirements will also be suggested by project management.

> *Example*: 1.4 Development Schedule. The warhead development schedule shall support the Missile System Initial Operational Capability (IOC) of January 1, 2001.

6.2.11 Marketing

The marketing department wants features that will delight the customer. Kano calls them exciters. They are features that the customer did not know they wanted. In the 1970's, IBM queried customers to discover their needs. No one mentioned portability, so IBM did not make it a requirement. Compaq made a portable PC and then a laptop, dominating those segments of the market. In the 1950's IBM could have bought the patents for Xerox's photocopy machine. But they did a market research study and concluded that no one would pay thousands of dollars for a machine that would replace carbon paper. They did not realize that there was a need for a machine that could provide dozens of copies in just minutes.

6.2.12 Manufacturing Processes

Sometimes we might require a certain manufacturing process or environment. We might require our semiconductor manufacturer to have a Class 10 clean room. Someone might specify that Quality Function Deployment (QFD) be used to help elicit customer desires (although this would be in bad form because it states a how not a what).

6.2.13 Design Engineers

Design engineers impose requirements on the system. These are the "build to," "code to," and "buy to" requirements for products and "how to execute" requirements for processes.

> *Example*: 1.5 Design Philosophy. Attention must be directed toward a design that offers high reliability in a device that will remain in the strategic arsenal until 2050. Warhead design should be conservative and should not attempt to extend performance beyond well-established regimes. More specifically, the warhead design that is developed for this system should:
>
> - Minimize the likelihood of deleterious changes during stockpile life.
> - Enhance insensitivity to any changes that may occur.
> - Optimize the capability to replicate the design should a warhead rebuild program be required in the future.
> - Allow for unforeseeable excursion beyond those nominal conditions described in the Stockpile-to-Target Sequence.

6.2.14 Reliability

Reliability could be a performance requirement, or it could be broken out separately.

> *Example*: 3.3 Reliability Considerations. Warhead reliability including the AF&F shall be consistent with an overall reentry system reliability goal of 0.xxxxxx.

6.2.15 Safety

Some requirements may come from safety considerations. These may state how the item should behave under both normal and abnormal conditions.

> *Example*: 3.4.6 In the absence of arming and firing signals, the probability of nuclear detonation for normal and hostile environments specified in the Stockpile to Target Sequence (STS) shall not exceed:
>
> 3.4.6.1 One in 10^9 per warhead lifetime in the absence of warhead enabling stimuli.

3.4.6.2 One in 10^6 per occurrence after application of initial enabling stimuli and in the absence of final enabling stimuli.

3.4.8 HE Safety. The following are design objectives:

3.4.8.1 The warhead HE shall not ignite or detonate as a result of a free fall of the assembled reentry system from a distance specified in the STS.

3.4.8.2 The warhead HE shall not detonate when the reentry system is subjected to the fire environments described in the STS.

3.4.8.3 The warhead HE shall not ignite or detonate as a result of stress caused by other credible abnormal environments.

6.2.16 The Environment

Concern for the environment will produce requirements, such as forbidding the use of chlorofluorocarbons (CFCs) or tetraethylchloride (TEC).

6.2.17 Ethics

Ethics could require physicians to obtain informed consent before experimenting on human subjects.

6.2.18 Intangibles

Sometimes the desires of the customer will be hard to quantify, such as for intangible items such as beauty, aesthetics, national or company prestige (e.g., putting a man on the moon in the Apollo project), or ulterior motives such as trying to get a foot in the door using a new technology (e.g., the stealth airplanes) or starting business in a new country (e.g., China).

6.2.19 Common Sense

Many requirements will not be stated because they are believed to be common sense. For example, characteristics of the end user are seldom stated. If we are designing a computer terminal, it would not be stated that the end user would be a human with two hands and ten fingers. Common sense also dictates that computers not be damaged if they are stored at temperatures as high as 140°F. Furthermore, we do not write that there can be no exposed high voltage conductors on a personal computer, but it certainly is a requirement. Many of these requirements can be found in de facto standards.

6.2.20 Laws or Standards

Requirements could specify compliance with certain laws or standards, such as the National Electrical Code, City/County Building codes, or the IEEE 1220 Standard for Systems Engineering.

6.2.21 The Customer

Some requirements are said to have come from the customer, such as statements of fact and assumptions that define the expectations of the system in terms of mission or objectives, environment, constraints, and measures of effectiveness. These requirements are defined from a validated needs statement (Customer's Mission Statement), from acquisition and program decision documentation, and from mission analyses.

6.2.22 Legacy Requirements

Sometimes the customer has definite specific requirements that are not stated, for example, "Your last system was robust enough to survive a long trip on a dirt road, so we expect your new system to do the same."

6.2.23 Data Collection Activities

If an existing system is similar to the proposed new system, then existing data collection activities can be used to help discover system requirements because each piece of data that is collected should be traceable to a specific system requirement. Often it is difficult to make a measurement to verify a requirement. It might be impossible to meet the stated accuracy. Trying to make a measurement to verify a requirement might reveal more system requirements.

6.2.24 Government Agencies

Government Agencies are often the source of additional and sometimes obscure requirements. Sandia projects typically encounter requirements from the Department of Energy (DOE), the Department of Defense (DoD), the Environmental Protection Agency (EPA), and the Nuclear Regulatory Commission (NRC). DOE requirements are frequently derived from DOE Orders and Policies, Engineering Procedures (EPs), Field Office Directives such as the "Development and Production" Manual, and requirements such as QC-1. Additional DoD requirements are frequency derived from the DoD Orders and Policies as well as the Military Standards.

6.2.25 Industry Standards

Projects may be required to comply with certain Industrial or Commercial Standards. Typical standards that may be the source of requirements include those issues by ANSI, IEEE, EIA, and SAE.

6.2.26 Other Sources

There are many other sources of system requirements: human factors, the environment (e.g. temperature, humidity, vibration, etc.), the end user, the operator, victims, management, company vision, future expansion, logistics, the US Congress, and compatibility.

6.3 *Expressions or Modalities*

For some purposes, the best expression of the requirements will be a narrative where words are organized into sentences and paragraphs. Such documents are often called operation concepts or operational needs. But all descriptions in English will have ambiguities, both because of the language itself and the context in which the reader interprets the words. Therefore, for some purposes the best description of a system will be a list or string of *shall* and *should* statements. Such a list would be useful for acquisition or acceptance testing. However, it is still very difficult to write with perfect clarity so that all readers have the same understanding of what is written.

Other modalities that can be used instead of written descriptions include:

- Wymorian Notation (Wymore, 1993)
- Finite State Machines (Katz, 1994)
- Algorithmic State Machine Notation (Katz, 1994)
- Hardware
- Object-Oriented Models (Booch, 1994; Rumbaugh et al., 1991; Jacobson et al., 1995)
- Special purpose, requirements management, computer programs

The big advantage of these modalities over the English language is that they can be rigorous and executable by computer. This greatly helps to point out contradictions and omissions. It also allows you to perform a sensitivity analysis of the set of requirements to learn which requirements are the real cost drivers (Karnavas et al., 1993).

6.3.1 Prototype

A publicly assessable prototype can express the system requirements as they are currently understood. Of course many functions will not be implemented; instead there will only be a statement of what the functions are intended to do. Such a system is easy to update and it helps everyone understand what the requirements are.

6.3.2 Users Manual

The Users Manual should be written by future users early in the system design process (Shand, 1994). This helps get the system requirements stated correctly and increases user "buy in."

6.4 *Input and Output Trajectories*

Input and output trajectories are descriptions of input and output values as functions of time.

6.4.1 Behavioral Scenarios

A powerful technique for describing the functional behavior of a system and for discovering requirements is describing typical sequences of events that the proposed system will go through. Such descriptions of behavior as a function of time are called trajectories, behavioral scenarios, use cases, threads, operational scenarios, logistics, or interaction diagrams.

The basis of these diagrams is to list the system's objects (or components) along the top of the diagram. Then, with time running from top to bottom, list the messages that are exchanged between the objects. Alternatively, the arrows could be labeled with data that is exchanged between the components. Examples of behavioral scenarios for an AF&F are given in Figures 6 through 8. These examples were derived using object-oriented modeling. This technique relies on collecting a large number of behavioral scenarios. This collection then describes the desired system behavior. Additional scenarios can be incrementally added to the collection. Behavioral scenarios are easy for people to describe and discuss, and it is easy to transform them into a system design. (See also Appendix E of Volume II for an example of an automated teller machine (ATM) behavioral scenario.)

6.4.2 Input-Output Relationships

Wymore (1993) shows the following six techniques for writing input-output relationships. These techniques have different degrees of precision, comprehensibility, and compactness.

(1) For each input value, produce an output value. For example, multiply the input by 3:

$$output(t+1) = 3 * input(t)$$

(2) For each input string, produce an output value. For example, compute the average of the last three inputs:

$$output(t+1) = (input(t-2) + input(t-1) + input(t))/3$$

(3) For each input string, produce an output string. For example, collect inputs and label them with their time of arrival:

For an input string of 1, 1, 2, 3, 5, 8, 13, 21, the output string shall be (1,1), (2,1), (3,2), (4,3), (5,5), (6,8), (7,13), (8,21). All strings are finite in length.

(4) For each input trajectory, produce an output trajectory. For example, collect inputs and label them with their time of arrival.

For an input trajectory of 1, 1, 2, 3, 5, 8, 13, 21, . . . the output trajectory would be (1,1), (2,1), (3,2), (4,3), (5,5), (6,8), (7,13), (8, 21) . . . A trajectory may be infinite in length.

(5) For each state and input, produce a next state and next output. For example, design a Boolean system where the output is asserted whenever the input bit stream has an odd number of 1s. This Odd Parity Detector can be described as:

Z1 = (SZ1, IZ1, OZ1, NZ1, RZ1), where

SZ1 = {Even, Odd}, /* The 2 states are named Even and Odd. */

IZ1 = {0, 1}, /* A 0 or a 1 can be received on this input port. */

OZ1 = {0, 1}, /* The output will be 0 or 1. */

NZ1 = {((Even, 0), Even), /* If the present state is Even and the input is 0, then the next state will be Even. */ ((Even, 1), Odd), ((Odd, 0), Odd), ((Odd, 1), Even)},

RZ1 = {(Even, 0), (Odd, 1)} /* If the state is Even the output is 0, if the state is Odd the output is 1. */

(6) Qualitative descriptions, which includes words, sentences, paragraphs, blueprints, pictures, and schematics. Most of this document has focused on this technique.

The following is a behavioral scenario for an AF&F device in range detonation mode.

Note: Time runs from the top to the bottom of the page.

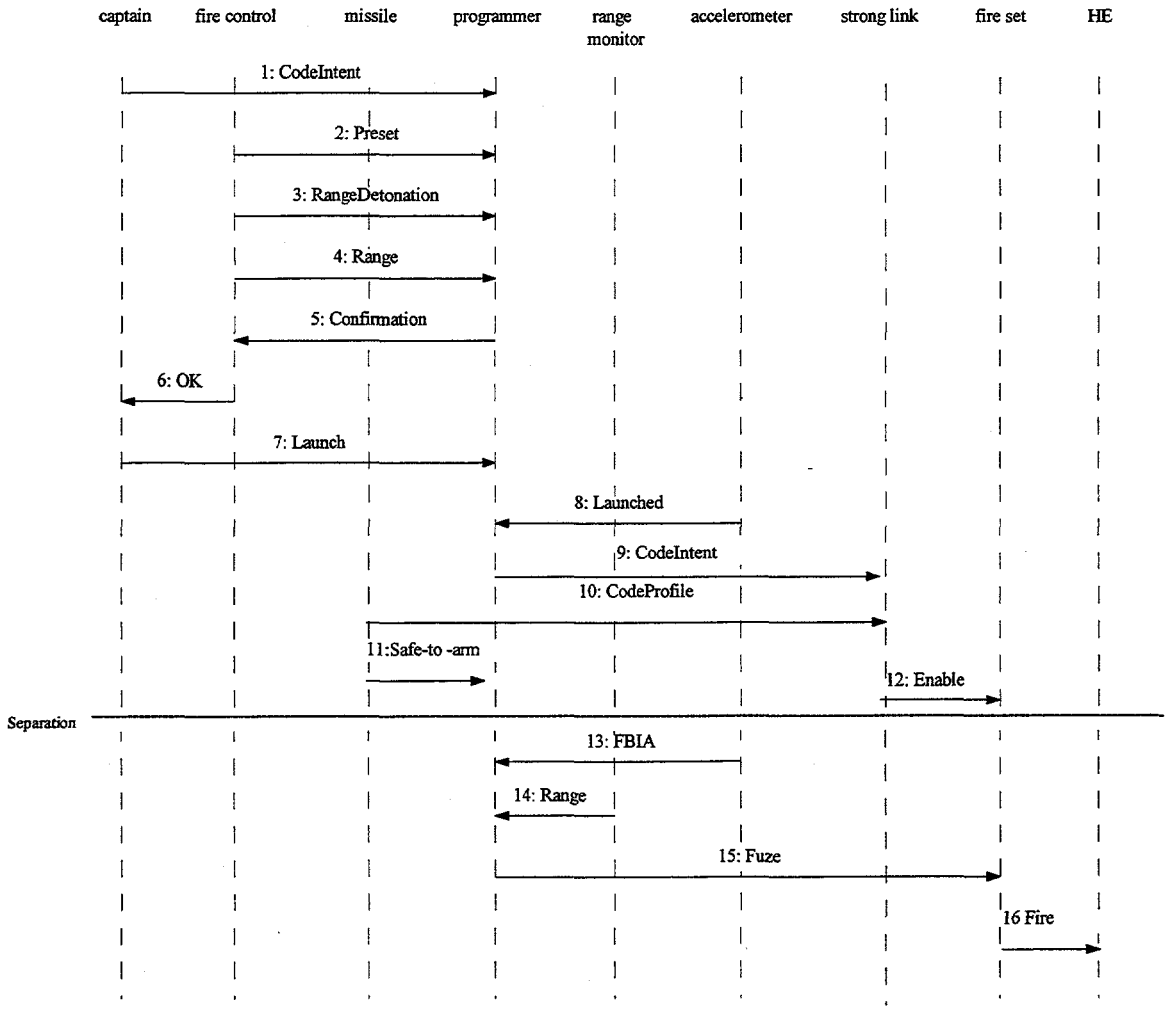

Figure 6. AF&F Range Detonation Mode

The following is a behavior scenario for an AF&F device in contact detonation mode.

Note: Time runs from the top to the bottom of the page.

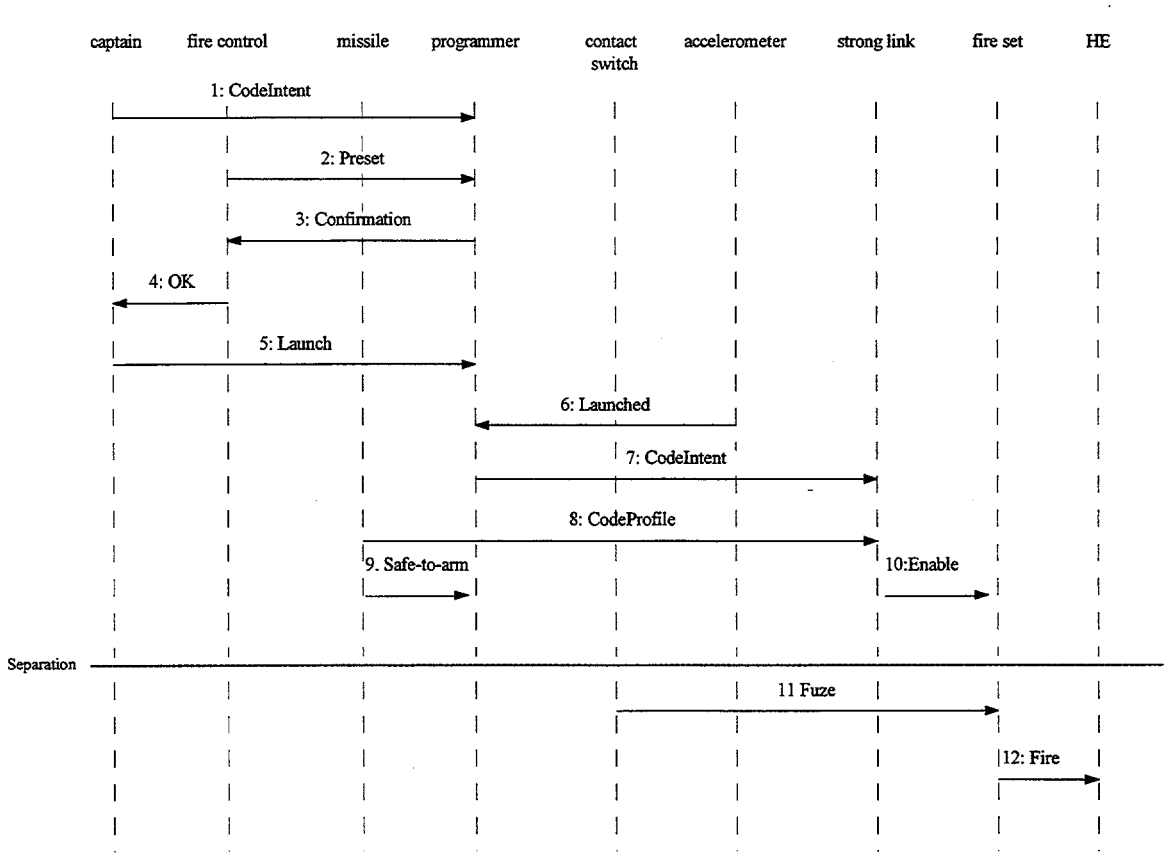

Figure 7. AF&F Contact Detonation Mode (Backup Only)

The following is a behavioral scenario for an AF&F device in time detonation mode.

Note: Time runs from the top to the bottom of the page.

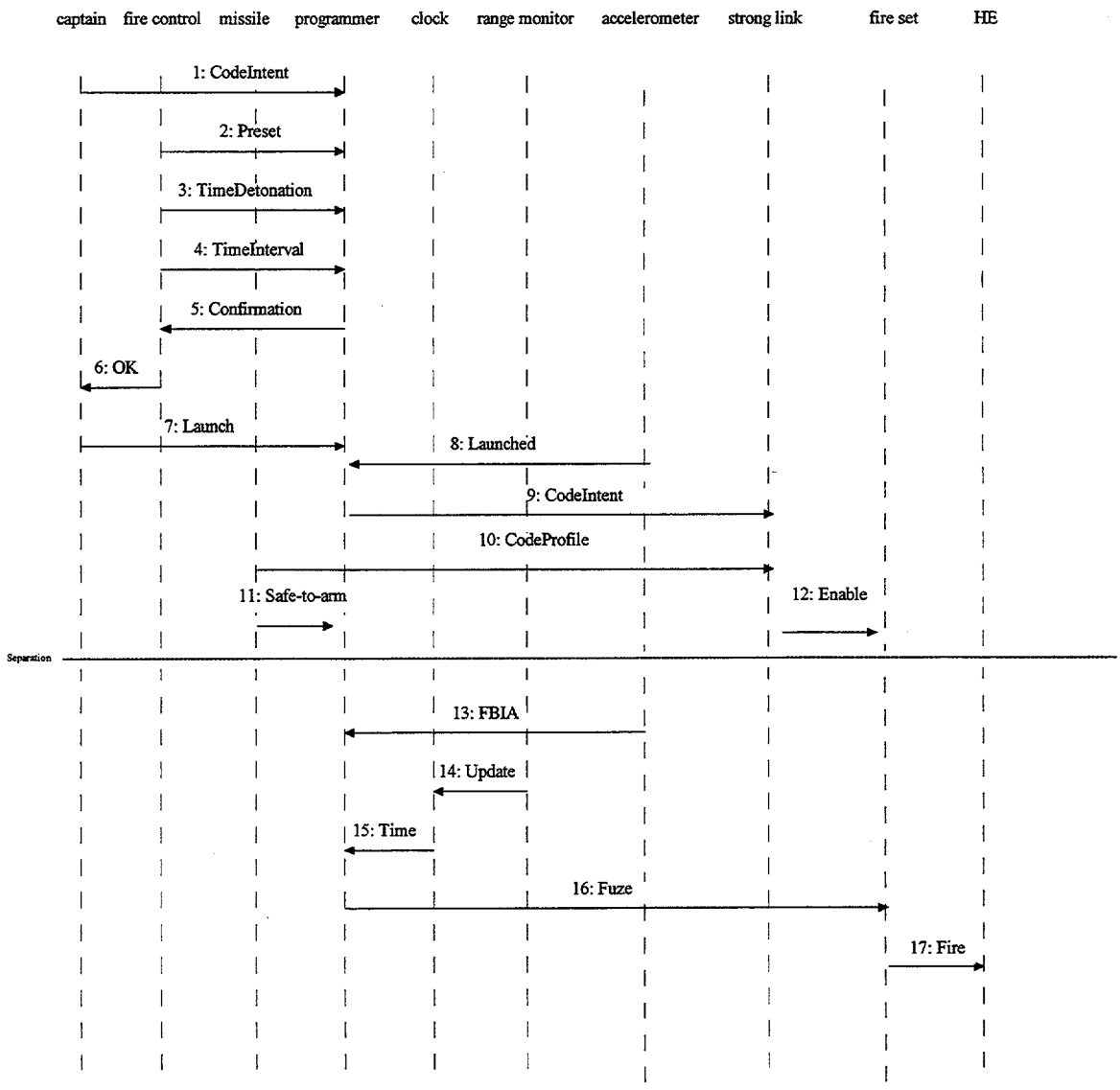

Figure 8. AF&F Time Detonation Mode

Another way to show the interaction between components is with an N-squared (N^2) diagram as shown in this figure. An x indicates an interaction between the components in that column and row.

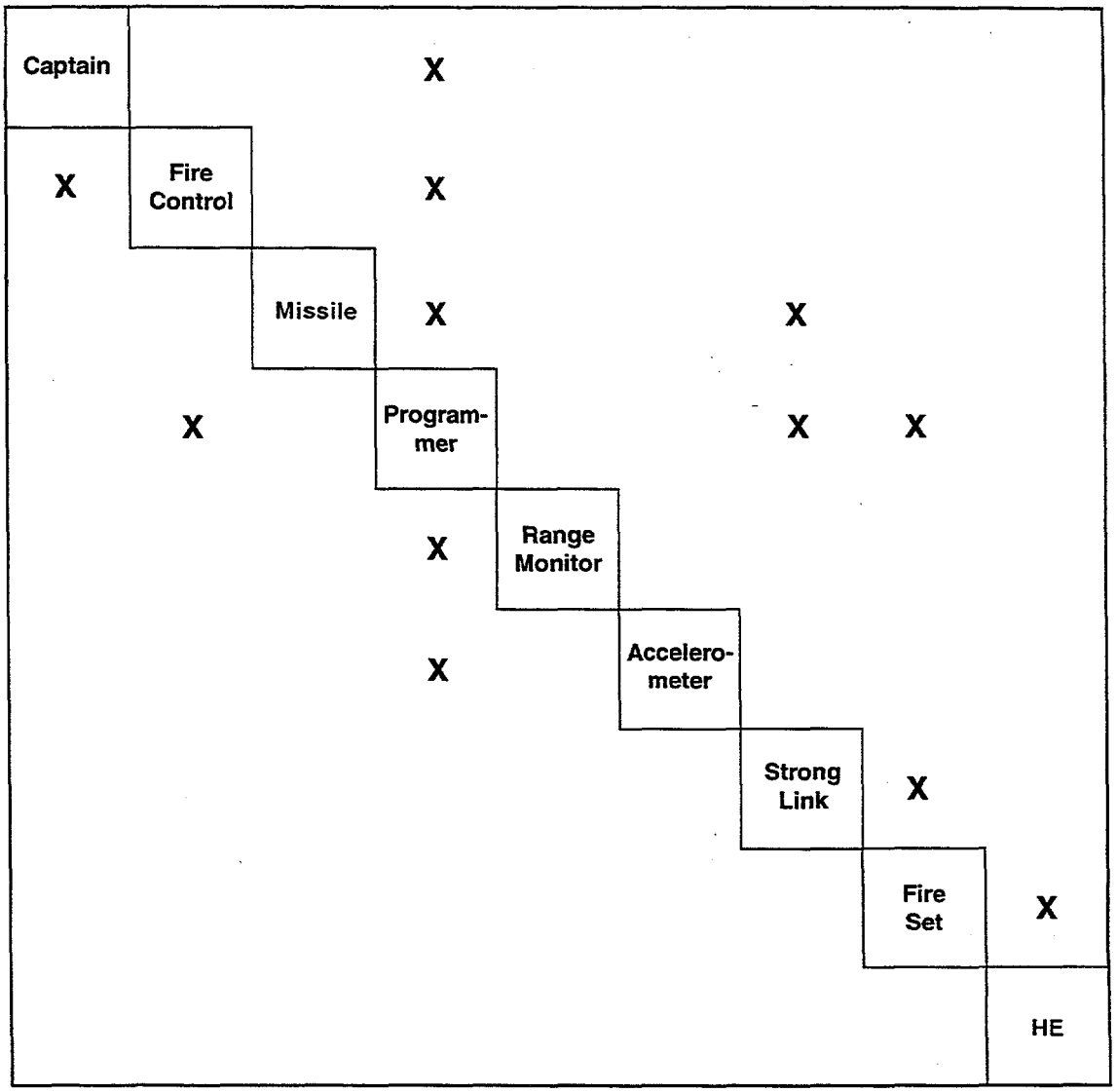

Figure 9. N^2 diagram for W88 AF&F.

However, we prefer to implement the N-squared (N^2) diagram with an arrow instead of an x, as shown in this diagram. The arrows show the direction of data or information flow.

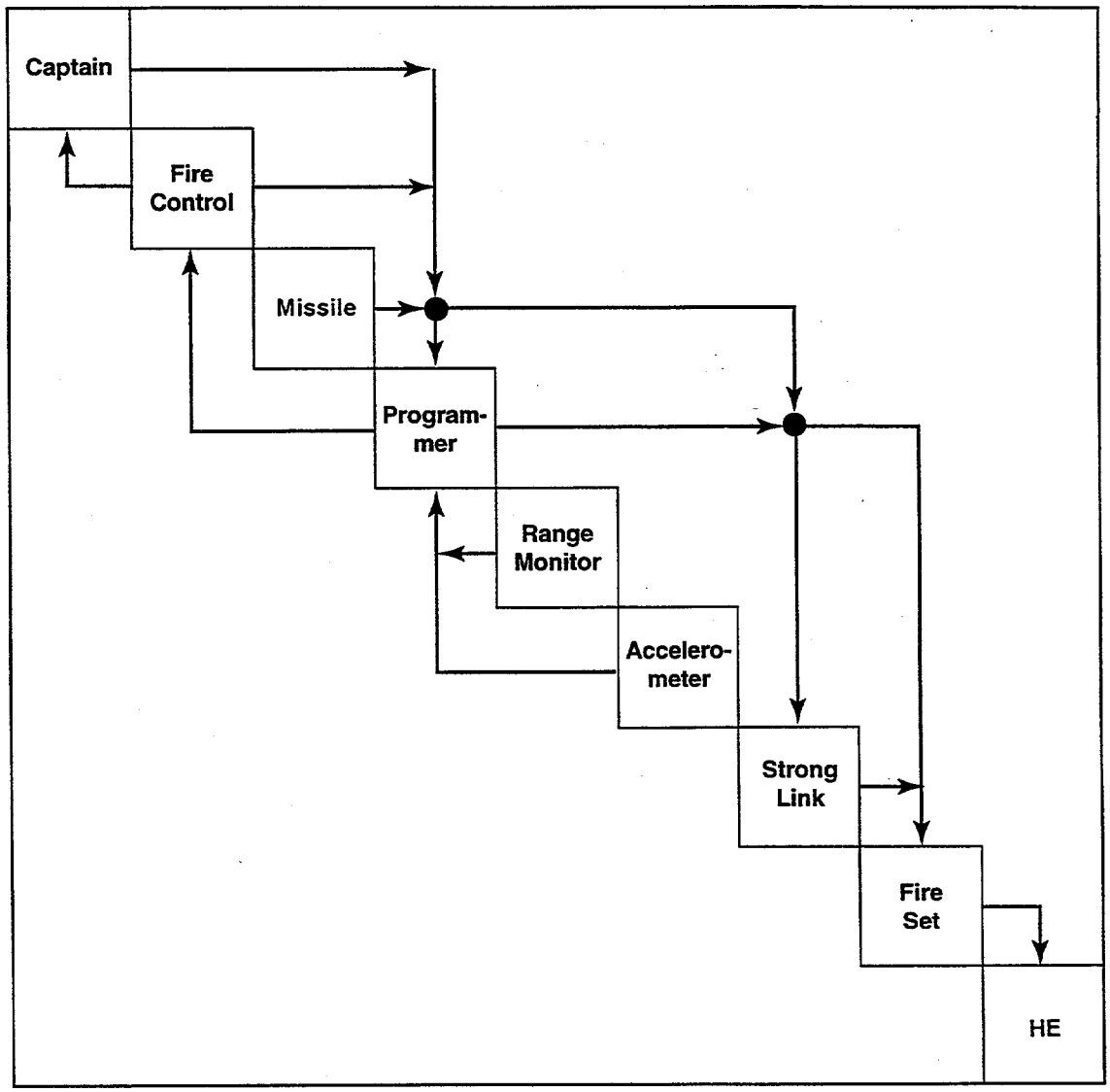

Figure 10. N^2 diagram for W88 AF&F.

Intentionally Left Blank

7. Tools for Gathering Requirements

The following tools are used to help discover and write requirements. See Appendix D for a comparison of these tools.

- Affinity diagrams
- Force-field analysis
- Ishikawa fishbone (cause-and-effect) diagrams
- Pugh charts
- Quality Function Deployment (QFD)
- Wymorian T3SD
- RDD-100
- CORE
- Slate

Grady (1995A and B) discusses many more tools that Systems Engineers can use to gain insight into the system and to derive appropriate requirements.

Intentionally Left Blank

8. Other Related Terms

8.1 Requirements Versus Constraints

The terms *requirements* and *constraints* are sometimes used interchangeably. However, a design constraint can be defined as a boundary condition within which the designer must remain while satisfying the performance requirements (Grady, 1993). With this definition, almost all of the requirements mentioned in this document (except for performance and system test) could alternatively be called constraints.

8.2 Requirements Versus Goals

The term goal is often used for a requirement that cannot be tested. Grady (personal communication) calls them requirements and desirements. For example, a requirement may be that "The hole shall be 5 mm in diameter, plus or minus 0.5 mm." According to Taguchi, a goal would say, "The hole shall be 5 mm in diameter and the standard deviation should be as small as feasible." Some DoD customers use "goal" as a specific value for a preference requirement.

8.3 External Versus Internal

Some engineers characterize requirements as external and internal. External requirements are driven by customer need. Internal requirements are driven by company practices and resources. For example, a company might use certain processes or technologies.

8.4 Outcomes, Environments, and Constraints

Some engineers also characterize requirements as outcomes, environments, and constraints. Outcomes are customer related and are often given in Military Characteristics Documents as shown in Appendix B. Environmental requirements are given, for example, in the Stockpile-to-Target Sequence. These requirements change as the system design progresses. Finally, constraints, such as laws that have to be obeyed or standards that have to be followed, are often left unstated for the sake of brevity.

8.5 Requirement Definition Versus Specification

A requirements definition set, which we usually call the requirements, describes the functions the systems should provide, the constraints under which it must operate, and the rationale for the requirements. It should be written in plain language. It is intended to describe the proposed system to both the customer and the designers. It should be broad so that many alternative designs fit into its feasibility space.

The requirements specification, which we usually call the specification, provides a precise description of the system that is to be built. It should have a formal format and might be written in a specialized language. It is intended to serve as the basis of a contract between Purchasing and Manufacturing. It should narrow the feasibility space to a single point that is the system to be manufactured.

The set of requirements determines the boundaries of the solution space. The specifications define one and only one solution within that space. The requirements say what, the specifications say how.

These definitions came out of the software engineering literature (Sommerville, 1989). The systems engineering literature is seldom as clear. Often the best we get is "A specification is a big document that contains a lot of requirements." (Jim Martin and Ivy Hooks, personal communications, 1995.) Because of the variable usage in the literature, if customers use the term specification, you should ask them what *they* mean by the term.

Why do so many people write the requirements after the system has been built? Perhaps they (1) write the requirements up front, (2) develop the requirements into specifications, and (3) build the system, continually updating the specifications but not the requirements. Consequently when they deliver the system and the customer asks for the requirements, they must go back and write them.

9. Heuristic Examples of Requirements

9.1 A Nuclear Warhead

Assumption: It is assumed that this warhead will be on a submarine based missile.

2.1 Destructive Capability

The warhead shall have at least a 90% kill probability against a certain target.

2.1.1 Yield

The warhead shall provide a nominal yield of xxx kilotons of TNT equivalent. (Note: It is probably inappropriate to specify yield in a requirement, because this restricts consideration of alternative designs.)

This requirement cross references 1.1 (below), The System Mission.

1.1 System Mission

The system shall carry yz reentry systems to a range of xxxx nautical miles with a circular error probable of yy meters.

Note: This mission statement violates our singleness of purpose rule of thumb.

9.2 An Automatic Teller Machine (ATM) Example

LaPlue, Garcia, and Rhodes (1995) say that a requirement should contain (1) specification of the system output, (2) conditions under which the requirement must be met, (3) external inputs associated with the requirement, and (4) all characteristics that determine if the system output is correct. They have organized this into a standard template:

The system name shall produce <output>

for use by <nodes>,

if <conditions>,

using <inputs>,

where <quality factor>.

They offer the following example.

Requirements for an Automated Teller Machine

3.0 Transaction Requirements

3.1 The ATM User

3.1.1 Produce Receipt

3.1.2 Give Cash

The ATM shall produce cash

- for use by the ATM user
- if the ATM user requested a withdrawal
- and if the Central Bank verifies the account and password
- and if the Central Bank validates the withdrawal amount
- and if the ATM cash on hand exceeds the cash requested
- using the Withdrawal Validation Message from the Central Bank
  and the Account Verification Message from the Central Bank
  and the withdrawal request from the user
- where the amount of cash produced equals the amount requested
- and where the cash is dispensed within 20 seconds of the receipt of the Withdrawal Validation Message from the Central Bank.

3.1.3 Eject Card

3.1.3.1 Eject bank card at end of session

The ATM shall produce the bank card

- for use by the ATM user
- if the ATM user has inserted a bank card
- and if the ATM user has requested termination of session
- using the Bank Card and the Terminate Request
- where the Bank Card is ejected within 2 seconds of the receipt of the Terminate Request

3.1.3.2 Eject unreadable cards

The ATM shall produce the bank card

for use by the ATM user

- if the ATM user has inserted a bank card
- and if the bank card does not contain a valid code
- using the bank card
- where the code reading and validation is as specified in Bank Card Specifications, section 3.1

3.1.4 Produce Error Messages

3.2 The Central Bank

3.2.1 Verify Account Message

The ATM shall produce the Verify Account Message

- for use by the Central Bank
- if the ATM user has entered a password
- and if the bank card contains a readable code
- using the bank card and user-entered password
- where the content and format is as specified in the Central Bank Interface Specification, section 3.2.18
- and where the message is issued within 2 seconds of the final digit of the password

This example shows many of the features of good requirements that were mentioned in this chapter. The numbering scheme manifests the tree structure of this set of requirements: parent, child and sibling relationships are clear. References are made to the specifications. In each requirement the customer is identified: e.g., the ATM user, the central bank. Many behavioral scenarios were used to elicit these requirements. Performance figures of merit are given, they are specified as maximum values, units are given, and they are testable: e.g., cash must be dispensed within 10 seconds. The requirements state what, not how: e.g., The ATM shall produce cash. The requirements identify applicable states with the conjunctive if clauses. The word choice is correct.

Intentionally Left Blank

10. Glossary

Behavioral Scenarios: A common technique for describing the functional behavior of a system and discovering requirements is to describe typical event sequences expected of the proposed system. Such descriptions of behavior as a function of time are called trajectories, behavioral scenarios, use cases, threads, operational scenarios, or interaction diagrams.

Critical Design Review (CDR): This review verifies that the system design meets its requirements. It examines the system design in full detail, ensures that technical problems and design anomalies have been resolved, and ensures that the design maturity justifies the decision to commence manufacturing. Few requirements should be changed after this review.

Customer: Anyone who has a right to impose requirements on the system. This includes end users, operators, bill payers, owners, regulatory agencies, victims, etc.

Customer Needs: The customer may not be aware of the details of what is needed. Systems Engineers must enter the customer's environment, discover the details, and explain them. Flexible designs and rapid prototyping facilitate identifying details that might have been overlooked. Talking to the customer's customer and the supplier's supplier is also useful.

Mission Concept Review: The Mission Concept Review and the Mission Definition Review are the first formal reviews. They examine the mission objectives and the functional and performance requirements.

Preliminary Design Review (PDR): This review demonstrates that the preliminary design meets all the system requirements with acceptable risk. System development and verification tools are identified, and the Work Breakdown Structure is examined. Full-scale engineering design begins after this review.

Production Readiness Review (PRR): For some systems there is a long phase when prototypes are built and tested. At the end of this phase, and before production begins, there is a production readiness review.

Requirement Analysis: Requirements analysis establishes what the system must be capable of accomplishing: how well system products must perform in quantitative, measurable terms; the environments in which system products must operate; and constraints that will affect design solutions. The requirements are derived from customer expectations, project constraints, external constraints, and higher level system requirements. These are documented in a requirements baseline. (IEEE P1220)

Requirement Analysis: Determining system characteristics based on analysis of customer needs, requirements, and objectives: missions, projected utilization environments for people, products and processes; and measures of effectiveness. Requirements analysis helps the customers refine their functional and performance requirements. It is a key link in establishing achievable requirements that satisfy needs. (Martin, 1996)

Requirements: Requirements are statements that identify the essential needs for a system in order for it to have value and utility. Requirements may be derived or based upon interpretation of stated requirements to assist in providing a common understanding of the desired operational characteristics of a system.

Specification: A document that contains the mission statement, technical requirements, verification criteria, functional decomposition, and interface definitions. When invoked by a contract, it is legally enforceable and contractually binding.

System Definition Review: This review examines the technical requirements, the proposed system architecture, the proposed system design, and the flowdown of functions to the major subsystems. It also ensures that the verification program is described.

System Life Cycle: The system life cycle has seven phases: (1) requirements development, (2) concept exploration, (3) full-scale engineering design and development, (4) manufacturing, (5) system integration and test, (6) operation, maintenance and modification, and (7) retirement, disposal, and replacement. However, the system life cycle is different for different industries, products, and customer. (Chapman, Bahill, and Wymore, 1992; Wymore, 1993; Kerzner, 1995)

System Requirements: System Requirements provide a description of desired capabilities, constraints, and other details that pertain to the product's functional, performance, and physical characteristics. These descriptions provide the stimulus for investigating product alternatives, and for making trade-offs among requirement sets. These requirements should establish the capabilities, physical characteristics, and customer quality attributes that define a quality product offering within the marketplace.

System Requirements Review (SRR): This review demonstrates that the product development team understands the mission and the system requirements. It confirms that the system requirements are sufficient to meet mission objectives. It ensures that the performance and cost figures of merit are realistic. It ensures that the verification plan is adequate.

System Test: At the end of manufacturing and integration, the system is tested to verify that the system satisfies its requirements. The results of these tests are presented at the System Acceptance and Operational Readiness Reviews.

Validating a System: Building the right system: making sure that the system does what it is supposed to do. It determines the correctness of an end product, compliance of the system with the customer's needs, and completeness of the system.

Validating Requirements: Ensuring that the set of requirements is consistent, that a real-world solution can be built that satisfies the requirements, and that it can be proven that such a system satisfies its requirements. If Systems Engineering discovers that the customer has requested a perpetual-motion machine, the project should be stopped.

Verifying a System: Building the system right; ensuring that the system complies with its requirements. It determines the conformance of the system to its design requirements. It also

guarantees the consistency of the product at the end of each phase, with itself and with the previous prototypes. In other words, it guarantees the honest and smooth transition from model to prototype to preproduction unit to production unit.

Verifying Requirements: Examination, analysis, test, or demonstration that proves whether a requirement has been satisfied. This process is iterative. The requirements should be verified with respect to the model, the prototype, the preproduction unit, and the production unit.

Verification and Validation: MIL-STD-1521B (and most Systems Engineers) and DoD-STD-2167A (and most software engineers) use the words verification and validation in almost the exact opposite fashion. For Systems Engineers, to validate a set of requirements is to prove that it is possible to satisfy them. System verification, on the other hand, is a process of proving that a system meets its requirements (Grady, 1994). To add further confusion, ISO-9000 tells you to verify that a design meets the requirements and validate that the product meets requirements. NASA has a different spin. They say verification consists of proving that a system (or a subsystem) complies with its requirements whereas validation consists of proving that the total system accomplishes its purpose. (Shishko, 1995)

Work Breakdown Structure: A product-oriented tree of hardware, software, data, facilities, and services. It displays and defines the products to be developed and relates the elements of work to be accomplished to each other and to the end product. It provides structure for guiding team assignments and cost and tracking control. (Martin, 1995)

Intentionally Left Blank

11. References

Bahill, A.T. and Chapman, W.L., "A tutorial on quality function deployment," *Engr Management J,* 5(3):24-35, 1993.

Bharathan, K., Poe, G.L. and Bahill, A.T., "Object-Oriented Systems Engineering," *Systems Engineering in the Global Market Place,* proceedings of the Fifth Annual Symposium of the National Council on Systems Engineering, July 22-26, 1995, St. Louis.

Bicknell, K.D. and Bicknell, B.A., *The Road Map to Repeatable Success: Using QFD to Implement Changes,* CRC Press, Boca Raton, 1994.

Booch, G., *Object-Oriented Analysis and Design*, Benjamin Cummings, 1994.

Chapman, W.L., Bahill, A.T. and Wymore, W., *Engineering Modeling and Design,* 1992.

Funk, P.A. and Larson, D.L., "Design features influencing thermal performance of solar box cookers," presented at the 1994 International Winter Meeting, paper No. 94-6546, American Society of Agricultural Engineers.

Gause, D.C. and Weinberg, G.M., *Are Your Lights On? How to Figure Out What the Problem Really Is*, Dorset House Publishing, NY, 1990.

Grady, J.O., *System Requirements Analysis,* McGraw Hill Inc., 1993.

Grady, J.O., *System Integration,* CRC Press, Boca Raton, 1994.

Grady, J.O., *System Engineering Planning and Enterprise Identity,* CRC Press, Boca Raton, 1995A.

Grady, J.O., *System Requirements Analysis Student Manual*, JOG System Engineering, San Diego, 1995B.

Hooks, I., *Writing Good Requirements*, Proceedings NCOSE, pp. 197-203, 1994.

IEEE P1220 Standard for Systems Engineering, IEEE Standards Dept., NY, 1994.

IEEE P1233 Guide For Developing System Requirements Specifications, IEEE Standards Dept., NY, 1993.

Jacobson, I., Ericsson, M. and Jacobson, A., *The Object Advantage: Business Process Reengineering with Object Technology,* Addison-Wesley, New York, 1995.

Karnavas, W.J., Sanchez, P. and Bahill, A.T., "Sensitivity analyses of continuous and discrete systems in the time and frequency domains," *IEEE Transactions on Systems, Man and Cybernetics*, SMC-23: 488-501, 1993.

Katz, R., *Contemporary Logic Design*, Benjamin Cummings, 1994.

Kerzner, H., *Project Management: a Systems Approach to Planning, Scheduling, and Controlling,* Van Nostrand Reinhold, New York, 1995.

LaPlue, L., Garcia, R.A., and Rhodes, R., "A rigorous method for formal requirements definition," *Systems Engineering in the Global Market Place*, Proceeding of the Fifth Annual Symposium of the National Council on Systems Engineering, July 22-26, 1995, St. Louis, pp. 401-406.

Latzko, W.J. and Saunders, D.M., *Four Days with Dr. Deming,* Addison-Wesley, Reading, Mass., 1995.

Lawton, R., *Creating a Customer-Centered Culture*, ASQC, Milwaukee, WI, 1993.

Martin, J. "Requirements methodology: Shattering myths about requirements and the management thereof," *Systems Engineering in the Global Market Place*, Proceeding of the Fifth Annual Symposium of the National Council on Systems Engineering, July 22-26, 1995, St. Louis, pp. 473-480.

Martin, J., *Systems Engineering Guidelines,* CRC Press, Boca Raton, 1996.

MIL-STD-499B, Draft Military Standard for Systems Engineering, AFSC/EN, 1993. (Note: This standard was not signed by the Department of Defense. They said that government should not be in the business of writing standards and that they will adopt standards written by professional societies.)

MIL-STD-1521B (referenced on pages 19 and 34), "Technical Reviews and Audits for Systems,"

NCOSE, *Systems Engineering in the Global Market Place,* Proceedings of the Fifth Annual Symposium of the National Council on Systems Engineering, July 22-26, 1995, St. Louis.

Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F. and Lorenson, W., *Object Oriented Modeling and Design*, Prentice Hall, 1991.

Sage, A.P., *Systems Engineering*, John Wiley, 1992.

Shand, R.M., "User Manuals as Project Management Tools," *IEEE Transactions on Professional Communications*, 37, 123-142, 1994.

Shishko, R., *NASA Systems Engineering Handbook*, SP-6105, 1995.

Sommerville, I., *Software Engineering,* Addison-Wesley, Reading, Mass., 1989.

Szidarovszky, F., Gershon, M. and Duckstein, L., *Techniques for Multiobjective Decision Making in Systems Management,* Elsevier Science Publishers, Amsterdam, 1986.

Wymore, W., *Model-Based Systems Engineering,* CRC Press, Boca Raton, 1993.

Intentionally Left Blank

Distribution

L. Coffman
1000 Independence Avenue SW
Room 1J054
Washington, DC 20585

	MS 0769	Dennis S. Miyoshi, 5800
	0461	L. Paul Page, 14702
	0461	Susan L. Harris, 14700
	0863	Carol A. Murry, 14004
	0458	Stephen J. Rottler, 09003
	0429	Ronald D. Andreas, 2100
	9005	J.B. Wright, 2200
	0509	W.D. Williams, 2300
	0301	D.J. Rigali, 2400
	0842	C.M. Hart, 2500
	0985	J.H. Stichman, 2600
	0481	T.M. Skaggs, 2167
	0481	E.D. Ayers, 2167
	0483	J.G. Lewis, 2165
	0479	E.J. Barkocy, 2151
	0487	G.J. Bloom, 2121
	0487	T.D. Geiwitz, 2121
	0447	A.L. Hillhouse, 2111
	0435	K.R. Eklund, 2102
	0435	J.C. Dalton, 2102
	0965	T.G. Taylor, 5711
(50)	0435	F.F. Dean, 2102
(42)	0435	A.T. Bahill, 2102
	0435	A.L. Bentz, 2102
	0435	A.R. Busse, 2102
	0435	M.H. Chapel, 2102
	0435	D.A. Geene, 2102
	0445	T.Hendrickson, 2166
	0435	G.A. Hultine, 2102
	0435	I.M. Jensen, 2102
	0435	V. Koonce, 2102
	0435	B.A. Lagree, 2102
	0435	T. Martinez, 2102
	0435	C.T. Naranjo, 2102
	0435	S.E. Ohrt, 2102
	0435	K. Ortiz, 2102
	0435	D.L. Poole, 2102
	0435	G.M. Pullen, 2102
	0435	R.C. Rentzsch, 2102
	0435	I.O. Rivera, 2102
	0435	M.E. Senglaub, 2102
	0435	F. Vigil, 2102

	0631	W.C. Nickell, 12300
	0435	M.M. Witkowski, 2102
	0435	J.K.H. Yip, 2102
	0435	E.M. Young, 2102
	0435	P.G. Guyer, 2102
	0435	D.R. Apodaca, 2102
	0427	T.D. Hernandez, 2101
	0427	W.R. Reynolds, 2103
	0453	D.L. McCoy, 2104
	0475	R.C. Hartwig, 2105
	0427	P.A. Longmire, 2106
	0447	J.D. Harrison, 2111
	0487	J.D. Mangum, 2121
	0469	D. Chadwick, 15102
	0486	M.E. Bleck, 2123
	0436	G.L. Maxam, 2147
	0479	P.A. Sena, 2151
	0482	A.B. Cox, 2161
	0483	R.L. Alvis, 2165
	0445	D.D. Tipton, 2166
	0445	M.A. Rosenthal, 2167
	9035	S.M. Ehle, 2201
	9006	D.J. Bohrer, 2203
	9005	W.G. Wilson, 2204
	9013	R.B. Nevin, 2266
	9032	M.A. Dremalas, 2211
	9015	C.A. Pura, 2221
	9036	D.R. Hensen, 2254
	9032	C.T. Oien, 2261
	9003	C.L. Knapp, 2262
	9034	D.J. Beyer, 2263
	9035	R.D. Monson, 2265
	9013	R.G. Miller, 2266
	9014	T.R. Harrison, 2271
	9038	R.W. Finn, 2281
	9039	V.E. ByField, 2282
	0479	M.G. Orrell, 2151
	0461	C.A. Yarnall, 14700
	0458	J.R. Asay, 5132
	0472	D.J. Allen, 5131
	0632	J.C. Hogan, 14707
	0458	L.R. Gilliom, 5133
	0985	M.W. Callahan, 5202
(1)	9018	Central Technical Files, 8523-2
(5)	0899	Technical Library, 4414
(2)	0619	Review & Approval Desk, 12630 For DOE/OSTI