Chapter 1

Getting Design Right

What Do We Mean by "Design"?

To some people, the word *design* means graphical or artistic design and is therefore in the domain of the artist. To other people, it means engineering design and is therefore in the domain of the engineer. If you are neither an artist nor an engineer, it would seem that you have little to do with design. But, if you broaden your definition of design to include planning and general problem solving, then you will see that you are engaging in design on a daily basis. Design is the creative part of your day, as opposed to the routine part such as brushing your teeth. When you enter a kitchen and decide what you are going to make for dinner, you are engaged in design. When you are composing a speech in your head that you will be delivering to a community group, you are engaged in design. Rearranging furniture, applying for jobs, and planning vacations are all design activities in the broad sense of the word. *Design* is the activity of imagining the future, setting goals, and finding ways to achieve those goals.

This is a practical book about design. To be practical, we focus on product design problems. That is, we imagine that you want to make something or hire someone to make it. You want to be involved deeply in the design process to make sure you get what you want. The "something" you want can be practically anything. It could be a consumer product such as a household appliance; it could be a software application or a business Web site. As a result, this book will have the most immediate interest to you if you are engaged in a product development process. But it has broader applicability. Many of the principles and techniques in this book are generally applicable to all design problems. They can be used to design human organizations: the mission of the organization, the roles of different individuals in the organization, the rules governing their interaction, the flow of work, and the command and control. You could thus use these techniques to design both the product and the business that will make money from the product. Though we focus on product design, you should recognize that by studying this book you are learning a general approach to problem solving.

In this book, we also lay the foundation for the design of complex systems: integrated networks of hardware, software, and human organizations. As an example of a complex system, think of an air traffic control system in which airplane pilots and ground controllers work together using radar, computer, and communication systems to avoid aircraft collisions and to manage the sequence of takeoffs and landings at a busy airport. Perhaps that seems out of your grasp. We do

not address the technological details of such complex systems, but we do show how the complexity of such systems is managed. It is handled with a straightforward design process applied to layer after layer of detail. The design process we describe in this book works whether you are designing a child's toy or an air traffic control system, a new line of clothing or a spacecraft.

Why "Getting Design Right"?

Every job you do involves design. Perhaps the design is well established, and the job has become routine. You no longer think about objectives, materials, sequence, and technique; you simply perform the job in a mechanical fashion. But, once the job was new and you were either taught how to do it or you figured it out on your own. If you figured it out on your own, you were the designer. Even if the job was as simple as finding out how to get to work, we are, all of us, designers in this sense. Design is the creative part of any job: identifying the objective, exploring the possibilities, picking a course of action, and testing and improving until we get it right.

Getting design right—fitting the result to the need—is greatly satisfying. In particular, if you design a product that pleases everyone who comes into contact with it, you can experience great joy as a designer. If you do it with economy, using the least effort and purest form, then your work has grace and elegance. Getting design right can also be commercially rewarding. The invention of the MP3 player, a portable digital audio player, was a breakthrough. The design and marketing of the iPod, Apple's version of an MP3 player, was a triumph. Personal satisfaction is therefore one motivation, and commercial success, another. You can have both.

Getting design wrong is a waste. It is a waste of time, of money, and of creative energy. We have all struggled with products that seem to work against us: devices that are too heavy, too complicated, or too flimsy. We have all experienced processes that are inefficient or unreliable: waiting in line for one agent, who then tells us we are in the wrong line; multistep processes to perform an operation so common we think it should be only a single mouse-click; and car repairs that seem to last just longer than the repair warranty. If you can think of an example of consumer frustration—of something that has wasted your time or money—then you have likely identified some form of design flaw. The extent of consumer frustration you have experienced is an indication that getting design right is not as easy as you might imagine. Getting design right is a process of avoiding design flaws to the best of your ability.

What Can Go Wrong?

We have collected many stories about design, both failures and successes, to illustrate the importance of getting design right. Each chapter in this book features at least one example of a design failure. Each failure can be traced, in part, to a failure to follow the process of getting design right properly. There are also stories of design successes. All of these stories can be read independently of the main text. Discussion questions at the end of each chapter are meant to stimulate your thinking about the stories.

The Sinking of the *Vasa*

In 1625, King Gustavus Adolphus of Sweden commissioned construction of a new flagship, the *Vasa* (Figure 1.1). Sweden needed a strong navy to dominate the Baltic Sea.

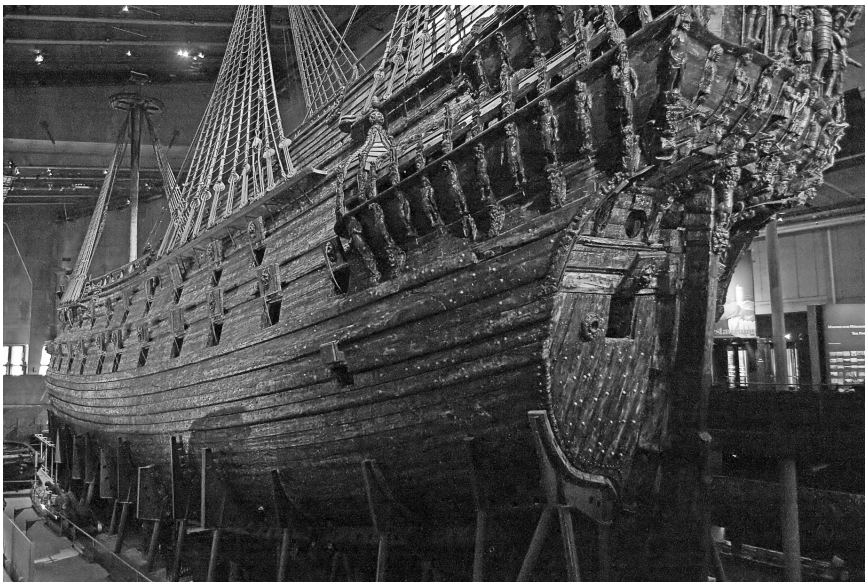

Figure 1.1 Restored seventeenth-century warship, the *Vasa,* Stockholm, Sweden. (Courtesy of Staale Sannerud.)

The *Vasa* was intended to be the world's largest warship, with two gun decks and 64 guns. Hendrick Hybertszoon, a master shipwright from Holland, was selected to build it. There were no written requirements from the king. Hybertszoon built a model of the ship and showed it to the king. The master shipwright assumed the ship would be 108 feet in length, but, after the first review, King Gustav requested a 135-ft ship. In response, the shipwright added timber to make the ship 120 ft.

While on vacation, King Gustav learned that the Danish king was building a ship with three gun decks. He then requested that a third gun deck be added to the two already on the *Vasa*. This meant adding 50 brass 24-lb cannons (at 1 ton each) to an already altered design. The master shipwright estimated that another 130 tons of rock ballast would be required under the deck to counterbalance the new cannons. However, there was no room for the additional ballast so it was left out.

The king was eager to show off the new flagship to the world. He ordered the shipwrights to complete the ship several months ahead of schedule. Hybertszoon died a year before the *Vasa* was finished. His less experienced brother, Arendt de Groot, completed the project.

Stability tests were conducted. These involved having 30 sailors run from one side of the ship to the other. The ship appeared to be unstable: It almost capsized on the third run. The navy admiral present at the test shrugged off the problem.

On a Sunday in August 1628, the *Vasa* set sail before a large audience that included King Gustav and foreign diplomats whom he hoped to impress. After a short distance, and in full view of the shocked audience, a gust of wind caught the mainsail, the ship heeled over, and water began to pour in through the gun ports. The gun ports had been left open even though this was the maiden voyage of a ship with known stability

problems. The ship capsized and immediately sank. At least 30 of the 150 people on board drowned.

The story of the *Vasa* is popular in design circles (Love 1993) because it illustrates the problems that continue to plague many projects, both large and small, even today: incomplete and shifting requirements, lack of meaningful trade-off analysis, accelerated construction timetables, leadership failures, and inadequate test and review cycles, to name a few.

What is There to Learn?

Getting design right is a skill. There are principles involved. Perhaps you already use some of these principles, though you use them unconsciously. Product successes can be traced to the consistent, intelligent use of these principles, and failures can be traced to their misapplication. There are also techniques developed by good designers that help them to implement these principles. Furthermore, there is a process for design: a place to start and steps to follow. This process of getting design right is applicable, with adaptation, in all design contexts. You may skip some steps and simplify others, but the basic process is sound. Getting design right, therefore, is something that can be taught and learned.

Why a Systems Approach?

We have qualified the title of this book with the phrase "a systems approach." Perhaps that is superfluous; any book that seeks to explain how to get designs right would necessarily have to adopt something like a systems approach. We use this phrase to emphasize that context is important. A successful race car design does not focus simply on speed: Strapping a rocket to the vehicle would maximize speed. Bad designs often do something extremely well. The failure comes in neglecting other aspects of the design that are important in the context of the product's use, such as the ability of a race car to maneuver around sharp corners at high speed. The phrase "a systems approach" suggests that you need to look at all aspects of the product in the context of its use within a larger system. There is a formal approach to studying an object in context—a systems approach.

A *system* is simply a collection of objects in relation to each other. A *systems approach* is therefore a procedural focus on relationships. Does your product vibrate in use? Taking a systems approach would lead you to explore the impact that vibration might have on the fasteners that hold your product together. Understanding relationships is critical to getting design right, and there are formal techniques to ensure that all relevant relationships in a design context are discovered and considered.

We also emphasize a systems approach because the language of systems is useful in all design contexts. Every discipline has evolved a language to describe the artifacts and methods within its design domain. Electrical engineers speak of circuits, mechanical engineers discuss forces, software engineers describe algorithms, and artists discuss medium and technique. The language of systems attempts to cut across disciplines and establish a more generic vocabulary and grammar with which to discuss design. We use abstractions such as entities, relationships, goals, functions, behavior, interfaces, and requirements to describe systems, and we become comfortable translating these concepts into specific artifacts and methods in different domains. The language of systems permits experts in different domains to talk with one another. It provides you with a universal language of design.

Finally, we use the phrase "a systems approach" because one of the techniques that is central to this approach is a technique known as "diving and surfacing." Diving refers to developing a detailed view of a system. Surfacing refers to abstracting from a detailed view to a more general view. It is the ability to surface, to abstract, and to "see the forest from the trees" that distinguishes the systems thinker from an individual who is trapped in discipline-specific details.

Design or Engineering?

This book is focused on product design and project management. It is not a text for detailed engineering. For the most part, we stop short of formulating and solving engineering problems. Instead, we focus on a general approach to design using generic techniques and expressing design in a systems language. It is a pre-engineering approach in the sense that it takes the design problem to the point where specific engineering problems can be posed and handed over to domain experts such as mechanical, electrical, or software engineers. But it is also a post-engineering approach in the sense that it identifies tests that the engineering solutions must satisfy. In short, it takes the perspectives of the product design owner and the project manager with the understanding that detailed design, build, and test can be subcontracted to other organizations.

Figure 1.2 illustrates a "Vee" diagram (Fossberg and Mooz 1992). A design project timeline moves from left to right in the figure, but the movement is downward into engineering detail on the left and then upward through testing on the right. The basic point is that each level of test should map back to an earlier design phase. In this text, our focus is not on the "engineering design–build–test" sequence below the line. Instead, in Getting Design Right we focus on the "pre-engineering design" and "post-engineering test" that occur before and after this "inner V." Ours is more of a systems view of design.

The examples and assignments in this book are incomplete because they stop short of creating true executable designs—that is, blueprints or drawings from which actual products can be constructed. The designs of this book are conceptual designs: product concepts thought out with as much detail as possible in a pre-engineering phase.

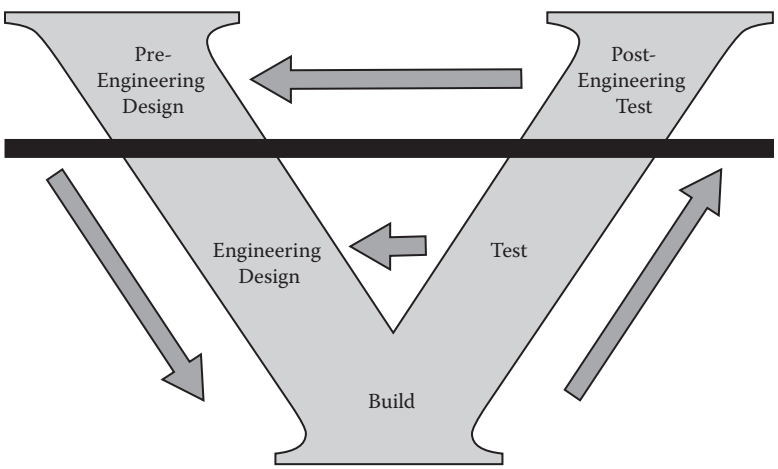

Figure 1.2 The "Vee" diagram: linking test to design.

In spite of the lack of traditional engineering content, this book is intended to be useful to engineers because the curricula of most engineering schools focus on analysis, not synthesis. As an engineering student, you are, or were, expected to discover the principles of design either on your own or on the job. The emphasis on design and synthesis is missing from many curricula. You may have been practicing engineering for many years without having been exposed to the formal approach presented here. If this is the case, you will readily see how to integrate these ideas into your daily engineering work. It will also help you to understand better your role in the overall product development process within your organization.

For Whom Is This Text Designed?

This book is written for a broad audience: students interested in the design–engineering interface, project managers and staff, and practicing engineers without formal design or project management training. No matter which category you fall into, you will find the material in this book accessible, provided that you have a reasonably good background in secondary school science and mathematics and you have used spreadsheet software. More advanced techniques of systems engineering and analysis can be found in graduate-level textbooks. We have excluded them from this book.

What is the Design Process?

As mentioned earlier, there is a process for design: a place to start and steps to follow. This process of getting design right is applicable, with adaptation, in all design contexts. In this book, we use the name "The Eight Steps to Getting Design Right" to describe this process. The eight steps are as follow:

1. *Define:* Define the problem. In this step, you will identify the product concept, define a project to design the product, establish the system context of the product, capture the "voice of the customer," and identify functional requirements for the product.
2. *Measure:* Measure the need and set targets. In this step, you will take ordinary English statements of product and functional requirements and, from them, develop measurable performance targets.
3. *Explore:* Explore the design space. The product concept you started with may not be the best design idea to meet all of the requirements that you have identified. In this step, you will use brainstorming and other techniques to explore the range of design possibilities systematically.
4. *Optimize:* Optimize design choices. After exploring the design space, you will have a number of workable integrated concepts from which to choose. In this step, you will compare these integrated concepts with each other from many perspectives and then select the best one. You will also set design parameters to optimize trade-offs.
5. *Develop:* Develop the architecture. In this step, you will define the functional behaviors of the product in greater detail, identify the components or subsystems, define how these components will work together to achieve the product behaviors, and identify the physical constraints that each component must satisfy in order meet the product level targets.

6. *Validate:* Validate the design. In this step, you will develop a master test plan to ensure that the product will meet the needs for which it is being designed. You will also identify the risks of the project and develop a strategy for mitigating those risks.
7. *Execute:* Execute the design. In this step, you will develop a plan for bringing the project to a successful and timely conclusion.
8. *Iterate:* Iterate the design process. For simple design problems, the preceding steps should be sufficient. However, complex systems must be designed in layers. In this step, you will learn how this design process is adapted and repeated for the next layer of detailed design. This time, however, you will understand the need for information tools to manage the explosion of detail and to ensure that the complex product will meet the needs for which it is being designed. Iteration is also required whenever you hit a roadblock in the design; further, you can learn to close out every design cycle with a systems view of your accomplishments.

The Getting Design Right process is depicted as a cycle of activities in Figure 1.3. The Eight Steps to Getting Design Right also provide the structure for this book. We devote a chapter to each step in the process: a total of eight chapters beyond this introduction. Refer to the table of contents to see how each of these major steps is broken down into further steps.

Learn by Example

The language and concepts of this book can seem quite abstract unless you can see clearly how to apply them. To make the book practical, we apply the steps of the design process to a specific example. It is a simple example: the design of a child's toy. Our initial design criteria are that the toy should be fun, safe, attractive, and affordable. From this simple starting point, we apply the Eight Steps to Getting Design Right process and we end with a product development plan and a collection of product definition documents that are ready for detailed engineering, design, and test. Throughout the book, we define system concepts abstractly and then illustrate them using this central example.

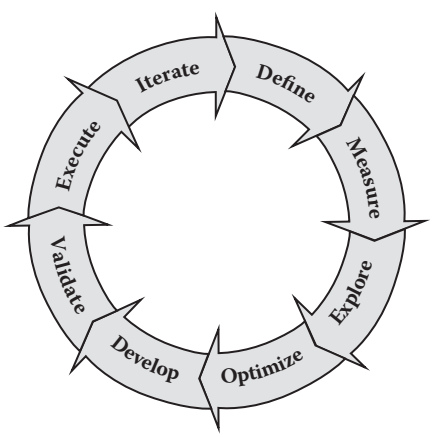

Figure 1.3 Getting design right cycle: Eight Steps to Getting Design Right.

Learn by Doing

This is not a book that you should attempt to read cover to cover without interruption. It is unlikely to have much impact upon the way you approach design unless you actually practice the steps. The techniques tend to build on each other, so you should master each step before advancing to the next. This book is intended to be part of a course of study in getting design right. In this course of study, you will apply the Eight Steps to Getting Design Right process to some project of your choosing. The project could be based on an idea of your own, or you may choose from some ideas we have created. In an appendix to this book you will find product design challenges for the following concepts:

- a bathroom-cleaning robot;
- a home health-care monitoring station;
- a night-vision system for automobiles; and
- an Internet-based meal delivery system.

No matter what project you work on, the goal of the exercise should be to create an initial product design and project plan. You should stop short of engineering-level detail. Even so, it is a substantial effort to carry out all the steps of the process. You should be prepared to devote many hours to the exercise.

Is it Worth the Effort?

This is not an easy book to read and these are not trivial concepts to master. Applying these techniques to a design problem requires effort and thought. Are this book and this approach worth all the effort to read, master, and apply? Absolutely! We have seen the benefits to students and ourselves from applying these ideas. Using a formal process to approach a design problem can seem awkward at first, but somewhere along the way you catch something that you would not have thought of otherwise, and, suddenly, you become a believer. On the second or third design problem, you find yourself using the techniques instinctively: They become the natural way to approach design.

What can you hope to see as a result of working your way through this book? If, presently, you do not consider yourself to be a designer, you may see your inhibitions to design disappear. You may discover a talent for design that you did not know you possessed. You may see your creativity unleashed. You may also find that your project leadership skills improve. Other members of your project team will start to look to you for leadership because you always seem to know what to do next, what pitfalls to avoid, and how to overcome obstacles. Even if you are an experienced designer, you may discover steps and techniques in this book of which you were unaware.

Why Use a Tabular Approach?

For the most part, the design approach we describe uses simple text-based tools and spreadsheet software (Microsoft Excel™). Recall that a system is a set of objects in relationship with another. A table is a standard way to represent relationships. Every cell of a table (a cell is the intersection of a row and a column) can be used to form a sentence relating the row heading to the column heading by means of the cell entry. For example, Table 1.1 is a simple description of a solar system showing the relation of moons to planets ("moons orbit planets") and planets to the sun ("planets

Table 1.1 Solar System in Tabular Form

	Sun	*Planets*	*Moons*
Sun		is larger than	
Planets	orbit		are larger than their
Moons		orbit	

orbit the sun"). It also captures relative sizes: "The sun is larger than any planet" and "the planets are larger than their moons."

Tables are our fundamental way to represent systems. The row and column headings will correspond to some subset of the objects of a system. The table entries will express the relationships. Because there are many different types of relationships, there will be many different tables.

Spreadsheet programs such as Microsoft Excel make it easy to create and manipulate textual lists, tables, and matrices (a *matrix* is a table of numbers or symbols). We make extensive use of Microsoft Excel in describing techniques for getting design right.

Excellent graphical techniques to describe systems are available. For example, Figure 1.4 displays a way to convey exactly the same information as that in Table 1.1 but in graphical form. This style of diagram is called an *entity–relationship diagram.* Objects ("entities") are drawn as nodes, and relationships are drawn as lines connecting the nodes. Lines are labeled with verb phrases, with the phrase positioned nearest the node that will form the subject of the sentence. Thus, "orbit" is placed near the node "planets" on the line connecting "planets" with the "sun" to suggest the relationship "planets orbit the sun."

Graphical representations have great visual power. It is likely that you prefer the representation of Figure 1.4 to that of Table 1.1. Why have we chosen the tabular approach over the graphical approach? The reason is that graphical presentations tend to be more time consuming to create and to make attractive than tabular views. You will find it requires a great deal of effort simply to apply the design steps of this book. It would be an extra burden to require you to complete the design steps and to present the results in attractive graphical form. We are content with having you present your work in a tabular format. We are therefore sacrificing visual power for ease of use.

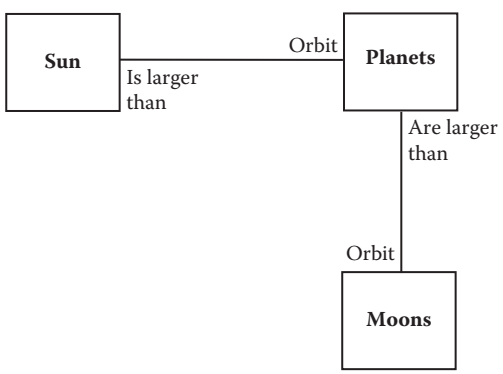

Figure 1.4 Solar system relationships in graphical form.

Graphical techniques are used extensively in advanced systems design, but there is a bewildering array of graphical styles to master (node shapes and meanings, line styles and meanings). We emphasize mastering the tabular technique first. As you discover different graphical presentations, you will be able to relate them to the underlying tabular views. Furthermore, advanced software packages for systems design can merge graphics, tables, and databases. Learning to use these sophisticated packages should be relatively easy after you have learned the simple tabular approach of this book.

The Getting Design Right Web Site

There are numerous spreadsheet files associated with this text. They can be downloaded by accessing the "Getting Design Right" Web site. The starting point is http://systemseng.cornell.edu/gettingdesignright/index.html. Follow the links there for the textbook, edition, and chapter in which you find the file referred.

Required Spreadsheet Skills

To do the exercises in this text, you will need to be familiar with Microsoft Excel (MS Excel™) or a spreadsheet program that can read MS Excel files. The following is a list of basic spreadsheet skills that will come in handy:

- copy transpose;
- drag and drop cells;
- merge cells;
- reorder rows and columns of a matrix;
- sum rows or columns;
- sort rows;
- create charts;
- use lookup formulas;
- compute sample mean and standard deviation; and
- display a Gantt chart.

If you are not familiar with these techniques, there is a tutorial, together with a fully worked example, available for download from the "Getting Design Right" Web site. The text will make reference to these skills prior to requiring them, so you can acquire the skills as you need them. Look for references in the text to a "spreadsheet skill."

To the Instructor: Where This Text Fits

This text was written to support a new type of course and fulfill a new need in engineering education. Traditional engineering curricula place a heavy emphasis on analysis and require a high level of mathematical maturity. Reclaiming the role of design and synthesis in engineering is an ongoing challenge. Design projects and design courses are enjoying a resurgence in engineering schools. But,

in spite of the many project experiences that the modern engineering student now acquires, he or she usually is still expected to pick up the principles of design and project work on his or her own.

Having developed a graduate-level curriculum in systems engineering and having taught many industrial short courses on these topics, we recognized that the basic principles do not require years of preparatory mathematics and analysis. They can be taught in much the same way the arts are taught: by example. Why not make these principles available to engineering students at the beginning of their undergraduate studies? By extension, why limit the course to engineering students? Design is ubiquitous and there is nothing in this text that is inaccessible to the arts student, the hotel school student, or the agriculture major. On the contrary, we have seen students from other backgrounds excel in this course. We have adopted a style of instruction that works with mature audiences in professional settings as well as with freshmen with limited design and project experience.

Essentially, this text is an introduction to systems engineering, though we consciously resist using the term until the final chapter. It is a radical departure from systems engineering texts in that it delays the discussion of the complexity of systems until late in the book. The fundamental concepts of systems engineering find their way into the text, but the order in which they are introduced is new. We lay primacy on the basic design cycle and introduce the level-by-level decomposition of systems only in the final step of the cycle, "iterate the process." For the instructor who feels strongly that iteration pervades all steps of the cycle, it would be a simple matter to cover the last chapter first, at the beginning of the course, and thus permit a discussion of levels and iteration throughout the course.

What is missing from the text, compared with other texts on design, are domain-specific topics, such as design for manufacturability (mechanical engineering) or design for security (software engineering). Instead, we introduce the concept of design for secondary uses. This topic becomes a placeholder for the instructor to tailor the course to his or her particular expertise and interest.

Two chapters may come as a surprise to design instructors. The first "Develop the Architecture," represents a blending of different threads in the design and systems engineering literature. It does not figure strongly in the Design for Six Sigma (DFSS) literature. It comes instead from the systems and software engineering literature. Even here, however, our approach is different from others. We support the object-oriented methodology that has proven so successful in software engineering, but we approach the topic from a behavioral perspective. The object view emerges from the behavioral view.

The second chapter that may come as a surprise is "Execute the Design." Project management and project scheduling are typically removed from design and, instead, covered in courses on project management. Our intent is to present a more holistic view of design. This chapter pushes the limit on mathematical complexity in a general purpose text, but by treating resource-constrained scheduling as a form of computer puzzle (a downloadable spreadsheet), we have maintained the accessibility of the concepts.

The product and service design challenges included in the appendices were developed for our graduate engineering curriculum in systems engineering. We have also used some of these cases in our industrial short courses. There is nothing, however, that prevents their use with an undergraduate audience. They are useful design challenges no matter what the student's level of experience and preparation is.

The text can stand on its own as a basic course on design. It can serve as an introductory course on systems engineering, and it can serve as a supplementary text in a graduate-level systems engineering program.

Discussion

1. Debate the adage "the customer is always right" in the context of the *Vasa* example. At least one person should defend the king's position.

References

Fossberg, K., and H. Mooz. 1992. The relationship of systems engineering to the project cycle. *Engineering Management Journal* 4 (3): 36–43.

Love, T. 1993. *Object lessons: Lessons learned in object-oriented development projects.* New York: SIGS Books, Inc.

The *Vasa* Museum (brochure). The National Maritime Museum. Vasa Museet: Box 27131, 102 52 Stockholm.