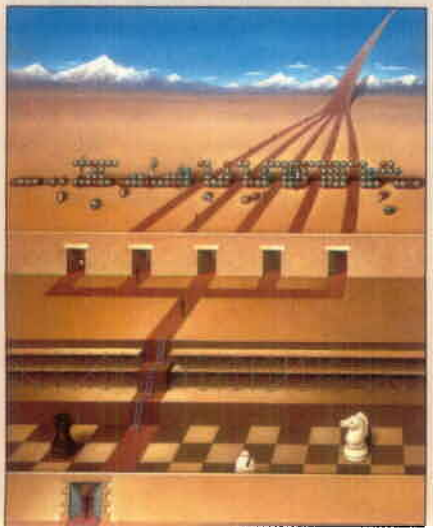
3

INFORMATION SYSTEM DEVELOPMENT

CHAPTER PREVIEW AND OBJECTIVES

This chapter introduces a system development life cycle-based methodology as the process used to develop information systems. System development is not a hit-or-miss process! As with any product, information systems must be carefully developed. Successful systems development is governed by some fundamental, underlying principles that we will introduce in this chapter. We also introduce a representative systems development methodology as a disciplined approach to developing information systems. Although such an approach will not guarantee success, it will improve the chances of success. You will know that you understand information systems development when you can:

- Differentiate between the system development life cycle and a methodology.
- Describe eight basic principles of systems development.
- Define problems, opportunities, and directives—the triggers for systems development projects.
- Describe a framework that can be used to categorize problems, opportunities, and directives.
- Describe a phased approach to systems development. For each phase or activity, describe its purpose, participants, prerequisites, deliverables, activities, postrequisites, and impact.
- Describe the cross life cycle activities that overlap the entire life cycle.
- Define computer-aided systems engineering (CASE) and describe the role of CASE tools in system development.



SCENE

A conference room where Sandra Shepherd has called a project planning meeting. Sitting around the table are Bob Martinez (a teammate), Terri Hitchcock (the business analyst assigned to the Member Services system project), Gary Goldstein (the Development Center manager), and Lonnie Bentley (a professor at State University).

SANDRA

Good afternoon. I thought the project launch meeting went very well. I called this meeting to start planning the Member Services project. I guess some introductions are in order. I am Sandra Shepherd, senior systems analyst and project manager for this project. The permanent members of our team will be Bob Martinez, a new systems analyst here at SoundStage, and Terri Hitchcock, who is on a two-year loan to Information Services from Member Services. She will be the business analyst for the project. I'd like to introduce you both to Gary Goldstein. He manages our Development Center. And Gary, I see you've brought a guest.

GARY

Yes. Bob, it is nice to meet you. Terri, it will be a pleasure working with you again. I'd like to introduce everybody to Professor Lonnie Bentley from State University. Lonnie has been hired by Nancy [Picard, vice president of Information Services] to perform a special role in this project. As you may be aware, I recently chaired a task force to study emerging object-oriented methodologies with the goal of determining how to best integrate object-oriented methods and technologies into future projects.

TERRI

I hate to interrupt, but I don't come from an information systems background. What is object-oriented?

LONNIE

Good question. For the last two decades, most of our methodologies have been based one way or the other on something called *structured* techniques—structured programming, structured design, structured analysis, information engineering, and so forth. These methodologies attempt to capture the essence of a business problem and its solution requirements in some sort of graphical picture called a model. The models become the basis for design. I believe that your methodology—What do you call it? *FAST*?—uses structured methods.

GARY

Yes, but in combination with prototyping methods to accelerate system development.

LONNIE

Yes. But all of these structured methods dealt with various aspects of the system differently and separately. You used one modeling approach to document data requirements and another to document process requirements. In some cases, you use different kinds of structured models to describe the same aspect of the system. It has all been very confusing, especially to users.

TERRI

Tell me about it!

LONNIE

Object methods are a dramatically different alternative to structured methods. They use the same models to describe all aspects of the system, and the chosen models don't change as you progress through the project?

TERRI

Sounds intriguing!

SANDRA

So, what's the angle on this project?

GARY

Well, Nancy has hired Lonnie and one of his colleagues to develop our system in parallel with our use of *FAST*'s structured methods. Lonnie will use object methods and technology. After the project is completed, Lonnie and his team will help me integrate object methods and technology into *FAST*. Initially, object methods will be an alternative strategy for us, but all indications are that object methods will eventually obsolete structured methods.

BOB

This sounds cool! Will we get to see this stuff?

SANDRA

Ditto on Bob's question. I'm fascinated!

GARY

Of course. Lonnie and his team will serve as observers in all of our facilitated sessions. They will develop the object models in parallel with our structured models and we will invite all interested staff to attend special comparison presentations. Of course, Lonnie's team will implement the system with object technology.

SANDRA

I thought our *Powerbuilder* language is object-oriented.

GARY

It is. But Lonnie has taught me the difference between object-oriented and object-based technology. *Powerbuilder* is object-based.

LONNIE

A detailed differentiation is not important now, but suffice it to say that any tool must meet three criteria to be object-oriented. First, it must support integration of data and methods that act upon that data into objects. *Powerbuilder* and languages like *Visual BASIC* meet that criterion. Second, it must support a concept called inheritance. Both *Powerbuilder* and *Visual BASIC* only partially meet that criterion in their current versions. Third, the tool must support something called polymorphism. Only true object languages do that.

SANDRA

I've heard the terms, and I assume you teach them to us at the appropriate time, but which language are you going to use?

BOB

I'd bet either *C++* or *Smalltalk*. I've read that they are pure object technologies.

LONNIE

No. Your current methodology is based on an important strategy called *rapid*

SOUNDSTAGE

SOUNDSTAGE ENTERTAINMENT CLUB

application development. My colleagues are not convinced that either *C++* or *Smalltalk* are especially conducive to rapid application development. Instead, we are going to apply Borland's *Delphi* or Microsoft's *Visual BASIC* to the project. It has the rapid application development capabilities of *Powerbuilder*, and it fulfills most of the three object-oriented criteria.

SANDRA
Too bad we can't use it for our structured methods. That would give us an even stronger basis for the final comparison. But neither *Delphi* nor *Visual BASIC* are in our approved application architecture.

GARY
Actually, you can. And Nancy is encouraging it. She has requested a waiver of the *Powerbuilder* standard for this project—if you're willing.

SANDRA
[noting Bob's obvious interest] I think I can support that!

[pause]

Welcome to the team, Lonnie. Let's get to the purpose of this meeting. Gary, Bob, and Terri are new to the *FAST* methodology.

GARY
I know. I've scheduled a two-day workshop for all new hires and business analysts starting Monday of next week. Since Lonnie is also new to the team, maybe I should provide a quick overview.

[noting no objections, Gary continues]

FAST is a rapid application development methodology. For SoundStage, it is our second-generation methodology.

BOB
Second generation?

GARY
Yes, our first methodology was based solely on structured analysis and design. It was shipped in the form of seven three-ring binders, which had to be purchased for every developer. It was so detailed, complex, and inflexible that many developers resisted its use and deployment. So we replaced it with *FAST*.

TERRI
Why is *FAST* better?

GARY
First, it is simpler and more flexible. We found that to be highly desirable in our development culture. It is customizable to our standards and extensible to emerging methods, like object-oriented. And it is on-line. The only documentation is a "Getting Started" booklet and a reference manual for the on-line service. There are also some training materials, but you receive those when you complete the various in-house courses I teach.

LONNIE
On-line?

GARY
Yes, the methodology database is stored on the local area network, and it's available in its most current version from any developer's workstation. Simple diagrams walk the developers through the various phases, activities, roles, deliverables, and so forth. It can even automatically invoke the correct CASE tools we use to design and build our systems at the correct time during the project.

BOB
This sounds like fun.

GARY
Let's introduce the phases briefly today. I'll project the on-line services from my laptop to the screen . . .

DISCUSSION QUESTIONS

1. Some people believe that an enforced methodology for building systems stifles creativity. Why would they think that is true? Why might SoundStage still implement a common methodology?
2. Why must a methodology be adaptable to emerging methods and technologies?
3. How do you perceive Gary's role in this meeting and project?

SYSTEM DEVELOPMENT LIFE CYCLES AND METHODOLOGIES

This chapter introduces the business process used to develop information systems. In practice, the process is called a methodology. In theory, all methodologies are derived from a logical problem-solving process that is sometimes called a system development life cycle.

> A **system development life cycle (SDLC)** is a logical process by which systems analysts, software engineers, programmers, and end-users build information systems and computer applications to solve business problems and needs. It is sometimes called an **application development life cycle.**

The life cycle is essentially a project management tool used to plan, execute, and control systems development projects.

System development methodologies are frequently confused with the system development life cycle. Is there a difference? Some experts claim that methodologies have replaced the life cycle. In reality, they are one and the same! The intent

of the life cycle is to plan, execute, and control a systems development project. It defines the phases and tasks that are *essential* to systems development, no matter what type or size system you may try to build. Note the emphasis on the word *essential*. For instance, we should always study and analyze the current system (at some level of detail!) before defining and prioritizing user requirements. It's common sense!

So, what *is* a methodology? At the risk of oversimplification, we offer the following definition.

> A **methodology** is the physical implementation of the logical life cycle that incorporates (1) step-by-step activities for each phase, (2) individual and group roles to be played in each activity, (3) deliverables and quality standards for each activity, and (4) tools and techniques to be used for each activity.

Two points are important. First, a true methodology should encompass the entire systems development life cycle. Second, most modern methodologies incorporate the use of several development tools and techniques.

Why do companies use methodologies? Methodologies ensure that a consistent, reproducible approach is applied to all projects. Methodologies reduce the risk associated with shortcuts and mistakes. Finally, methodologies produce complete and consistent documentation from one project to the next. All of these advantages provide one overriding benefit—as development teams and staff constantly change, the results of prior work can be easily retrieved and understood by those who follow!

Methodologies can be home grown; however, many businesses purchase their development methodology. For example, SoundStage has purchased a methodology called *FAST*. Why purchase a methodology? For one thing, most information system organizations can't afford to dedicate staff to the continuous improvement of a homegrown methodology. Also, methodology vendors have a vested interest in keeping their methodologies current with the latest business and technology trends.

As you will discover in this book, *FAST,* like most modern, commercial methodologies, is based on some combination of techniques called best practices. In this chapter, we will explore *FAST* to learn about both the life cycle and a representative methodology.

UNDERLYING PRINCIPLES OF SYSTEMS DEVELOPMENT

Before we study the life cycle, let's introduce some general principles that should underlie all systems development methodologies.[1]

Principle 1: Get the Owners and Users Involved

Analysts, programmers, and other information technology specialists frequently refer to "my system." This attitude has, in part, created an "us-versus-them" attitude between technical staff and their users. Although analysts and programmers work hard to create technologically impressive solutions, those solutions often backfire because they don't address the real organization problems or they introduce new organization or technical problems. For this reason, owner and user involvement is an absolute necessity for successful systems development.[2] The individuals responsible for systems development must make time for owners and users, insist on their participation, and seek agreement from them on all decisions that may affect them.

[1] Adapted from R. I. Benjamin, *Control of the Information System Development Cycle* (New York: Wiley-Interscience, 1971).

[2] We are using the term *owner,* introduced in Chapters 1 and 2, to mean *management.*

Miscommunication and misunderstandings continue to be a significant problem in systems development. However, owner and user involvement and education minimize such problems and help to win acceptance of new ideas and technological change. Because people tend to resist change, the computer is often viewed as a threat. Through education, information systems and computers can be properly viewed by users as tools that will make their jobs less mundane and more enjoyable.

Principle 2: Use a Problem-Solving Approach

A methodology is, first and foremost, a problem-solving approach to building systems. The term *problem* is used throughout this book to include real problems, opportunities for improvement, and directives from management. The classical problem-solving approach is as follows:

1. Study and understand the problem (opportunity and/or directive) and its system context.
2. Define the requirements of a suitable solution.
3. Identify candidate solutions and select the "best" solution.
4. Design and/or implement the solution.
5. Observe and evaluate the solution's impact, and refine the solution accordingly.

The notion here is that systems analysts should approach all projects using some sort of problem-solving approach.

There is a tendency among inexperienced problem solvers to eliminate or abbreviate one or more of the above steps. The result can range from (1) solving the wrong problem, to (2) incorrectly solving the problem, to (3) picking the wrong solution. The problem-solving orientation of a methodology, when correctly applied, can reduce or eliminate the above risks.

Principle 3: Establish Phases and Activities

Most life cycles and methodologies consist of phases. In its simplest, classical form, the life cycle consists of four phases: systems planning, analysis, systems design, and systems implementation. (A fifth activity, systems support, refines the resulting system by iterating through the previous four phases on a smaller scale to refine and improve the system.)

Figure 3.1 illustrates the four key phases in the context of your information systems framework (from Chapter 2). In each phase, the analysts and participants define and/or develop the building blocks in that corresponding row of the framework. (Recall that the building blocks are DATA, PROCESSES, INTERFACES, and GEOGRAPHY. Also notice that as you progress top to bottom through the phases, you are moving closer to the technology base of the pyramid—suggesting that your early concerns are with the *business* and your later concerns become more *technology*-driven.

Because projects may be quite large and each phase usually represents considerable work and time, the phases are usually broken down into activities and tasks that can be more easily managed and accomplished. This chapter will focus only on phases, albeit somewhat more refined than shown in Figure 3.1. The underlying activities and tasks will be covered in later chapters.

Generally, the phases of a project should be completed top to bottom, in sequence. This may leave you with the impression that once you finish a phase, you are done with that phase. That is not true. At any given time, you may be performing tasks in more than one phase simultaneously. You may have to backtrack to previous phases and activities to make corrections or to respond to new requirements. Obviously, you can't get carried away with this type of backtracking or you might never implement the new system.

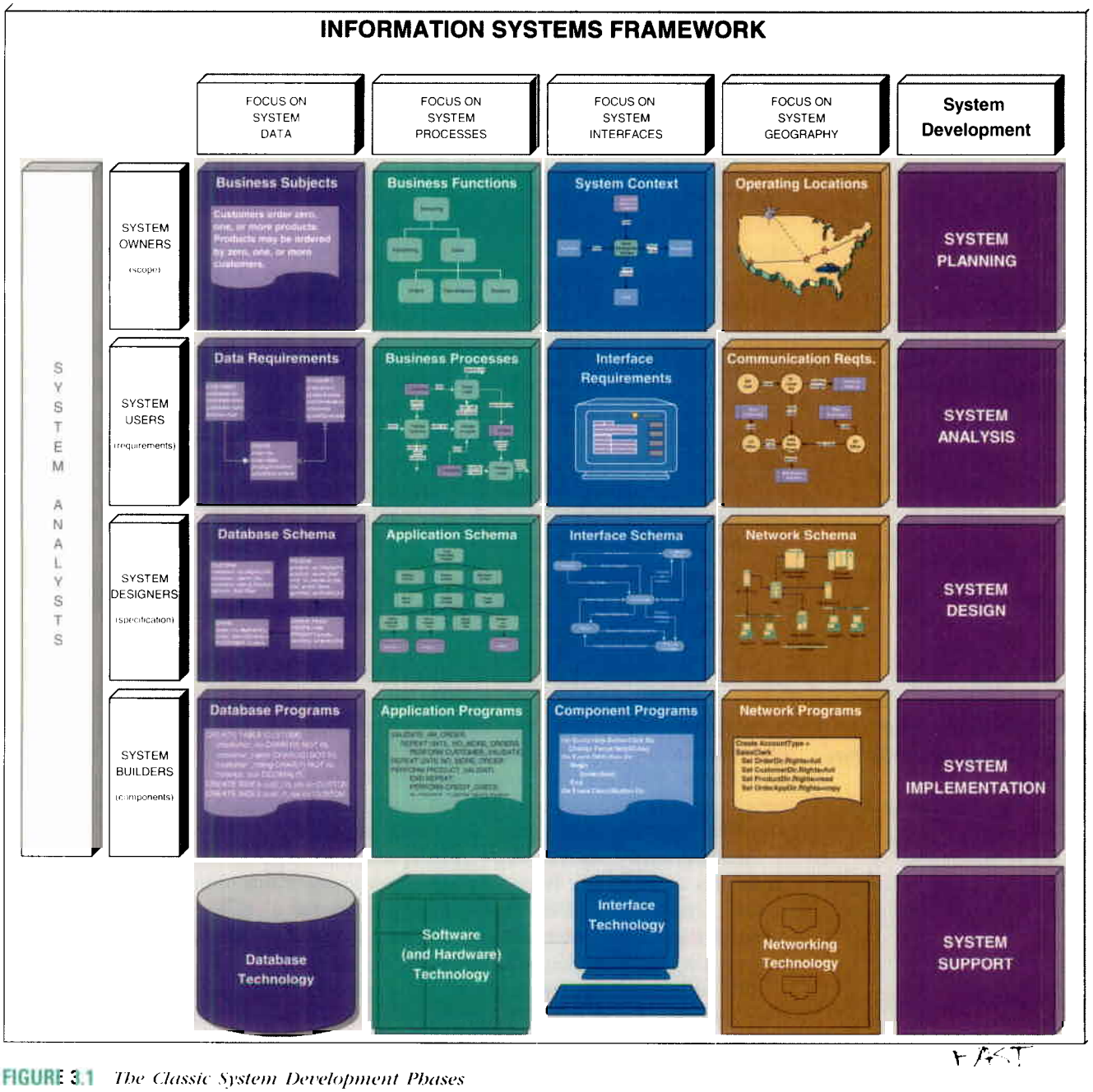


FIGURE 3.1 *The Classic System Development Phases*

An organization has many information systems that may include thousands of programs and software packages. If all analysts and programmers were to adopt their own preferred methodology and use their own tools and techniques to develop and document systems, chaos would quickly result. In medium-to-large information systems shops, systems analysts, programmers, and technical specialists come and go (as do system owners and users!). Some will be promoted; some will quit the organization; and still others will be reassigned. To promote good communication between this constantly changing base of users and information systems professionals, you must develop standards to ensure consistent systems development.

Principle 4: Establish Standards for Consistent Development and Documentation

Systems development standards usually describe (1) activities, (2) responsibilities, (3) documentation guidelines or requirements, and (4) quality checks. These four standards should be established for every phase in the methodology.

The need for documentation standards underscores a common failure of many analysts—the failure to document as an ongoing activity during the life cycle. Most students and practitioners talk about the importance of documentation, but talk is cheap! When do you really place comments in your computer programs? After you finish, of course! Therein lies the problem: Most of us tend to post-document software. Unfortunately, we often carry this bad habit over to systems development.

Documentation should be a working by-product of the entire systems development effort. Documentation reveals strengths and weaknesses of the system to others—before the system is built. It stimulates user involvement and reassures management about progress. This book teaches technique, but it also teaches documentation. Learn to use the tools and techniques to communicate with users during the life cycle, not after!

Principle 5: Justify Systems as Capital Investments

Information systems are capital investments, just as are a fleet of trucks or a new building. Even if management fails to recognize the system as an investment, you should not. When considering a capital investment, two issues must be addressed.

First, for any problem, there are likely to be several possible solutions. The analyst should not accept the first solution that comes to mind. The analyst who fails to look at several alternatives is an amateur. Second, after identifying alternative solutions, the systems analyst should evaluate each possible solution for feasibility, especially for cost-effectiveness.

Cost-effectiveness is defined as the result obtained by striking a balance between the cost of developing and operating a system, and the benefits derived from that system.

Cost-benefit analysis is an important skill to be mastered.

Principle 6: Don't Be Afraid to Cancel or Revise Scope

A significant advantage of the phased approach to systems development is that it provides several opportunities to reevaluate feasibility. There is often a temptation to continue with a project only because of the investment already made. In the long run, canceled projects are less costly than implemented disasters! This is extremely important for young analysts to remember.

Similarly, many analysts allow project scope to increase during a project. Sometimes this is inevitable because the analyst learns more about the system as the project progresses. Unfortunately, most analysts fail to adjust estimated costs and schedules as scope increases. As a result, the analyst frequently and needlessly accepts responsibility for cost and schedule overruns.

The authors of this text advocate a creeping commitment approach to systems development.[3] Using the creeping commitment approach, multiple feasibility checkpoints are built into the systems development methodology. At any feasibility checkpoint, all costs are considered sunk (meaning irrecoverable). They are, therefore, irrelevant to the decision. Thus, the project should be reevaluated at each checkpoint to determine if it is still feasible.

At each checkpoint, the analyst should consider (1) cancellation of the project if it is no longer feasible, (2) reevaluation of costs and schedule if project scope is to be increased, or (3) reduction of scope if the project budget and schedule are frozen, but not sufficient to cover all project objectives.

The concept of sunk costs is more or less familiar to some financial analysts and managers, but it is frequently forgotten or not used by the majority of practicing analysts and users.

Principle 7: Divide and Conquer

All systems are part of larger systems (called supersystems). Similarly, virtually all systems contain smaller systems (called subsystems). These facts are significant for

[3] Thomas Gildersleeve, *Successful Data Processing Systems Analysis,* 2nd ed. (Englewood Cliffs, NJ: Prentice Hall, 1985), pp. 5–7.

two reasons. First, systems analysts must be mindful that any system they are working on interacts with its supersystem. If the supersystem is constantly changing, so might the scope of any given project. Most systems analysts can still be faulted for underestimating the size of projects. Most of the fault lies with not properly studying the implications of a given system relative to its larger whole—its supersystem.

Consider the old saying, "If you want to learn anything, you must not try to learn everything—at least not all at once." For this reason, we divide a system into its subsystems in order to more easily conquer the problem and build the larger system. By dividing a larger problem (system) into more easily managed pieces (subsystems), the analyst can simplify the problem-solving process. We'll be applying this principle throughout this book.

Principle 8: Design Systems for Growth and Change

There is a critical shortage of information systems professionals needed to develop systems. Combined with the ever-increasing demand for systems development, many systems analysts have fallen into the trap of developing systems to meet only today's user requirements. Although this may seem to be a necessary approach at first glance, it actually backfires in almost all cases.

Entropy is the term system scientists use to describe the natural and inevitable decay of all systems. Entropy is illustrated in the context of the system development life cycle in Figure 3.2. After a system is implemented, it enters the support phase of the life cycle. During the support phase, the analyst encounters the need for changes that range from correcting simple mistakes, to redesigning the system to accommodate changing technology, to making modifications to support changing user requirements. As indicated by the blue arrows, many of these changes direct the analyst and programmer to rework former phases of the life cycle. Eventually, the cost of maintenance exceeds the costs of starting over—the system has become obsolete. This is indicated by the red arrow in the figure.

Frequently, systems that are designed to meet only current requirements are difficult to modify in response to new requirements. The systems analyst is frequently forced to duplicate files and "patch" programs in ways that make the system very costly to support over the long run. As a result, many systems analysts become frustrated with how much time must be dedicated to supporting existing systems (often called **legacy systems**) and how little time is left to work on important, *new* systems development.

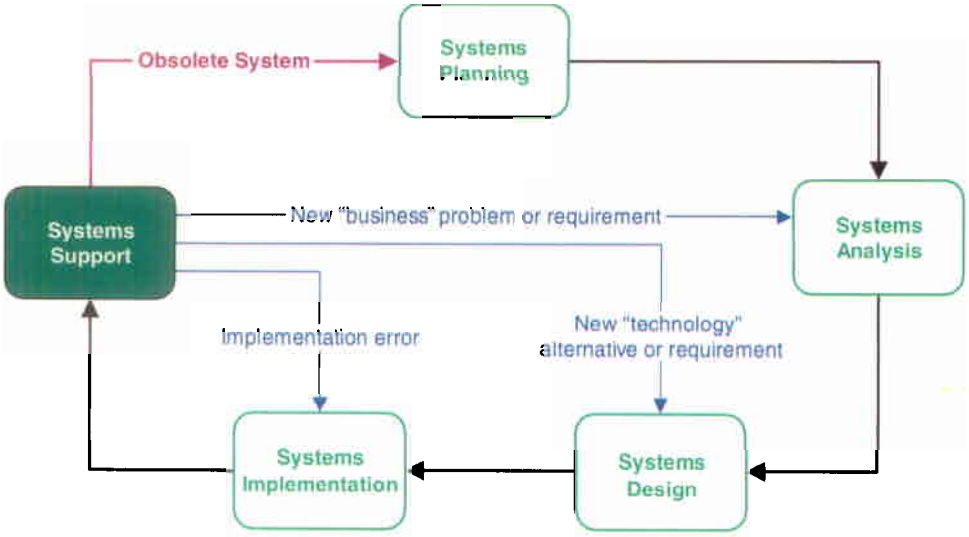


FIGURE 3.2
Systems Support and Entropy

FIGURE 3.3 *Principles of Systems Development*

- Get the owners and users involved.
- Use a problem-solving approach.
- Establish phases and activities.
- Establish standards for consistent development and documentation.
- Justify systems as capital investments.
- Don't be afraid to cancel.
- Divide and conquer.
- Design systems for growth and change.

Even if you design the system to easily adapt to change (our last principle), at some point, it will become too costly to simply support the existing system. Why? Perhaps the organization itself has changed too dramatically to be supported by the system. Or perhaps the requirements have become too complex to be integrated into the current system. In either case it is time to start over! This situation puts the term *cycle* into the term *systems development life cycle*. No system lasts forever (although many do last for a decade or longer).

But system entropy can be managed. Today's tools and techniques make it possible to design systems that can grow and change as requirements grow and change. This book will teach you many of those tools and techniques. For now, it's more important to simply recognize that flexibility and adaptability do not happen by accident—they must be built into a system.

We have presented eight principles that should underlie any methodology. These principles, summarized in Figure 3.3, can be used to evaluate any methodology, including *FAST*.

FAST—A SYSTEM DEVELOPMENT METHODOLOGY

In this section we'll examine a modern system development life cycle using a representative methodology we call *FAST* (used in the SoundStage case study). We'll begin by studying how a *FAST* project gets started. Then we'll examine *FAST*'s phases with a high-level picture that omits many details. Finally, we'll study a more complete picture of the *FAST* phases.

How a *FAST* Project Gets Started

Traditionally, system owners and system users have initiated most projects because they are closer to the organization's activities that need improvement. Alternatively, systems analysts are expected to survey the organization for possible improvements. When system owners, system users, or systems analysts initiate a project, *FAST* calls this an **unplanned system request**. These unplanned system requests can easily overwhelm the largest information services organization; therefore, they are frequently screened and prioritized by a **steering committee** of system owners to determine which requests get approved. Those requests that are not approved are often said to be **backlogged** until resources become available (which sometimes never happens).

The opposite of an unplanned system request is a **planned system initiative**. A planned system initiative is the result of one of the following earlier projects:

- An **information strategy plan** that has examined the business as a whole to identify those systems and application development projects that will return the greatest strategic (long-term) value to the business. An example of a methodology that incorporates an information strategy plan is *information engineering*.
- A **business process redesign** that has thoroughly analyzed a series of fundamental business processes to eliminate redundancy and bureaucracy and to improve efficiency and value added—now it is time to redesign the

supporting information systems for those business processes. Business process redesign (BPR) is a source of many current systems development projects.

Planned or unplanned, the impetus for most projects is some combination of problems, opportunities, or directives.

Problems are undesirable situations that prevent the organization from fully achieving its purpose, goals, and objectives.

For example, an unacceptable increase in the time required to fill an order can trigger a project to reduce that delay. Problems may either be current, suspected, or anticipated.

An **opportunity** is a chance to improve the organization even in the absence of specific problems. (Note: You could argue that any unexploited opportunity is, in fact, a problem.)

For instance, management is always receptive to cost-cutting ideas, even when costs are not currently considered a problem. Opportunistic improvement is expected to be the source of today's most important systems development projects.

A **directive** is a new requirement that's imposed by management, government, or some external influence. (Note: You could argue that until a directive is fully complied with, it is, in fact, a problem.)

For example, the Equal Employment Opportunity Commission, a government agency, may mandate that a new set of reports be produced each quarter. Similarly, company management may dictate support for a new product line or policy. Some directives may be technical. For instance, systems may be strategically directed to convert from batch to on-line processing or from conventional files to database. Such measures are appropriate if the current technology is obsolete, difficult to maintain, slow, or cumbersome to use.

There are far too many potential problems, opportunities, and directives to list them all in this book. However, James Wetherbe has developed a useful framework for classifying problems, opportunities, and directives.[4] He calls it **PIECES** because the letters of each of the six categories, when put together, spell the word *pieces*. The categories are:

- **P** the need to improve *performance*.
- **I** the need to improve *information* (and data).
- **E** the need to improve *economics*, control costs, or increase profits.
- **C** the need to improve *control* or security.
- **E** the need to improve *efficiency* of people and processes.
- **S** the need to improve *service* to customers, suppliers, partners, employees, etc.

Figure 3.4 expands on each of these categories.

The categories of the PIECES framework overlap. Any given project can be characterized by one or more categories, and any given problem or opportunity may have implications with respect to more than one category. PIECES is a practical framework (used in *FAST IV*), not just an academic exercise! We'll revisit PIECES later in the book.

An Overview of the *FAST* Life Cycle and Methodology

Figure 3.5 shows the context of a *FAST* system development project. You can clearly see the two aforementioned project triggers: **unplanned system request**

[4] James Wetherbe and Nicholas P. Vitalari, *Systems Analysis and Design: Traditional, Best Practices*, 4th ed. (St. Paul, MN: West Publishing, 1994), pp. 196–99.

FIGURE 3.4 *The PIECES Problem-Solving Framework*

The following checklist for problem, opportunity, and directive identification uses Wetherbe's PIECES framework. Note that the categories of PIECES are not mutually exclusive; some possible problems show up in multiple lists. Also, the list of possible problems is not exhaustive. The PIECES framework is equally suited to analyzing both manual and computerized systems and applications.

PERFORMANCE Problems, Opportunities, and Directives

A. Throughput—the amount of work performed over some period of time.
B. Response time—the average delay between a transaction or request and a response to that transaction or request.

INFORMATION (and Data) Problems, Opportunities, and Directives

A. Outputs
   1. Lack of any information
   2. Lack of necessary information
   3. Lack of relevant information
   4. Too much information—information overload
   5. Information that is not in a useful format
   6. Information that is not accurate
   7. Information that is difficult to produce
   8. Information is not timely to its subsequent use

B. Inputs
   1. Data are not captured
   2. Data are not captured in time to be useful
   3. Data are not accurately captured
   4. Data are difficult to capture
   5. Data are captured redundantly—same data captured more than once
   6. Too much data are captured
   7. Illegal data are captured

C. Stored data
   1. Data are stored redundantly in multiple files and/or databases
   2. Stored data are not accurate (may be related to 1)
   3. Data are not secure to accident or vandalism
   4. Data are not well organized
   5. Data are not flexible—not easy to meet new information needs from stored data
   6. Data are not accessible

ECONOMICS Problems, Opportunities, and Directives

A. Costs
   1. Costs are unknown
   2. Costs are untraceable to source
   3. Costs are too high

B. Profits
   1. New markets can be explored
   2. Current marketing can be improved
   3. Orders can be increased

CONTROL (and Security) Problems, Opportunities, and Directives

A. Too little security or control
   1. Input data are not adequately edited
   2. Crimes are (or can be) committed against data
      a. Fraud
      b. Embezzlement
   3. Ethics are breached on data or information—refers to data or information getting to unauthorized people
   4. Redundantly stored data are inconsistent in different files or databases
   5. Data privacy regulations or guidelines are being (or can be) violated
   6. Processing errors are occurring (either by people, machines, or software)
   7. Decision-making errors are occurring

B. Too much control or security
   1. Bureaucratic red tape slows the system
   2. Controls inconvenience customers or employees
   3. Excessive controls cause processing delays

EFFICIENCY Problems, Opportunities, and Directives

A. People, machines, or computers waste time
   1. Data are redundantly input or copied
   2. Data are redundantly processed

FIGURE 3.4 *Concluded*

3. Information is redundantly generated

B. People, machines, or computers waste materials and supplies
C. Effort required for tasks is excessive
D. Materials required for tasks are excessive

SERVICE Problems, Opportunities, and Directives

A. The system produces inaccurate results
B. The system produces inconsistent results
C. The system produces unreliable results
D. The system is not easy to learn
E. The system is not easy to use
F. The system is awkward to use
G. The system is inflexible to new or exceptional situations
H. The system is inflexible to change
I. The system is incompatible with other systems
J. The system is not coordinated with other systems

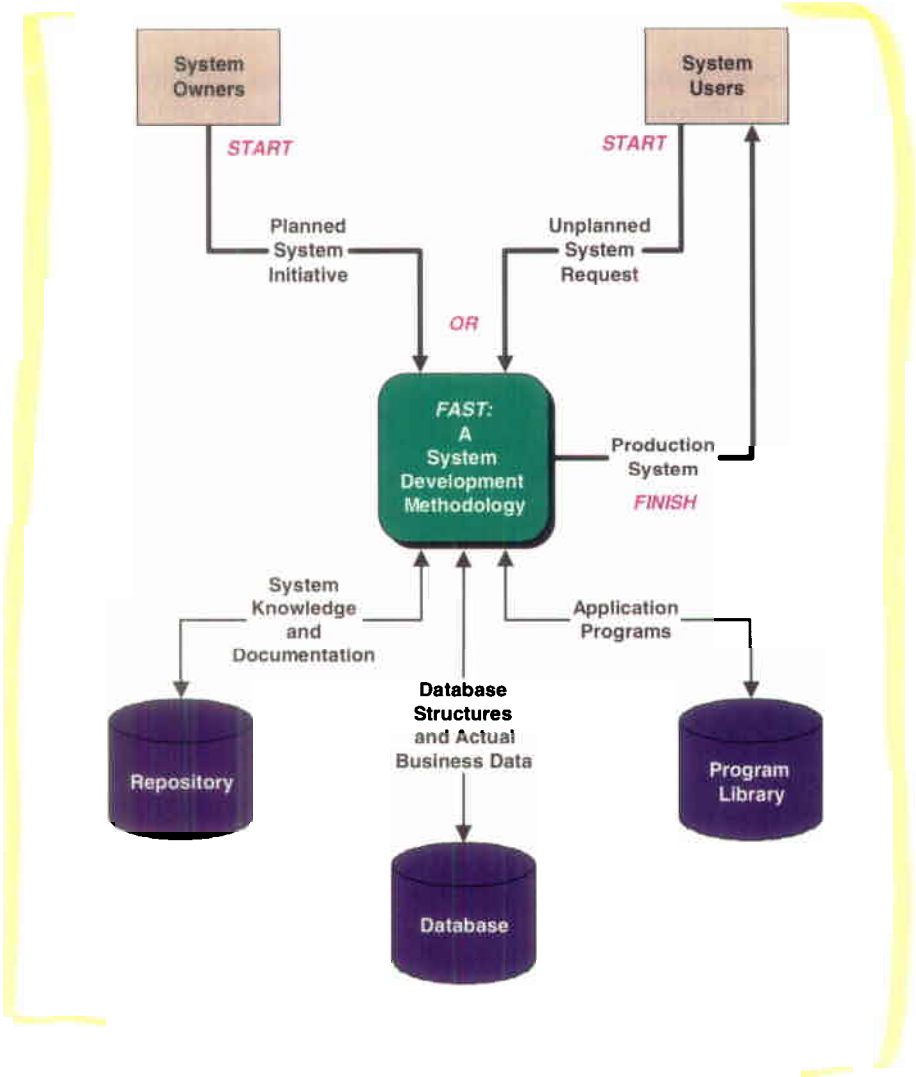


FIGURE 3.5
The Context of System Development

and **planned system initiative.** The final output of the methodology is the **production system** (so named because the system produces results). Also notice that, as you develop a system, you need a place to store various artifacts such as documentation, production data, and software. The three data stores are described as follows:

- The **repository** is a place where systems analysts and other developers store documentation about the system. Examples of such documentation might include *written memos, user requirements,* and *program flowcharts.*
- The **database** is built during the project to store actual business data about such things as CUSTOMERS, PRODUCTS, and ORDERS. This database will be maintained by the application programs written (or purchased) for the information system.
- The **program library** is where any application software and programs will be stored once they are written (or purchased).

We'll discuss these three data stores in more detail later in the chapter and book. Now we're ready to study the phases of a project in somewhat greater detail.

Let's look inside the *FAST* methodology process symbol. Figure 3.6 depicts a simplified *phase diagram* of the *FAST* methodology. The symbology, used throughout this chapter, is described as follows:

- The rounded rectangles represent *phases* in a *FAST* system development project.
- The thick black arrows represent the major deliverables (or outputs) of the phases. Each deliverable contains important documentation and/or specifications. Notice that the deliverable of one phase may serve as input to another phase.
- Although not used in Figure 3.6, we will eventually add thin black, double-ended arrows to represent other secondary information and communication flows. These flows can take the form of conversations, meetings, letters, memos, reports, and the like.
- The rectangles indicate people or organizations with whom the analyst may interact. Notice that the roles introduced in Chapter 2 have been carried over to this figure. For instance, we see system owners and system users participating in a project. Once again, we did not show all people interactions on this overview diagram of the methodology.
- Finally, consistent with our creeping commitment principle, the black circles indicate checkpoints at which time the project participants should reevaluate feasibility and/or project scope.

As you can see, the *FAST* methodology consists of eight phases. We'll get into details later, but each of these phases deserves a brief explanation:

1. The *survey phase* establishes the project context, scope, budget, staffing, and schedule.
2. The *study phase* identifies and analyzes both the business and technical problem domains for specific problems, causes, and effects.
3. The *definition phase* identifies and analyzes *business* requirements that should apply to any possible technical solution to the problems.
4. The *configuration phase* identifies and analyzes candidate technical solutions that might solve the problem and fulfill the business requirements. The result is a feasible application architecture.

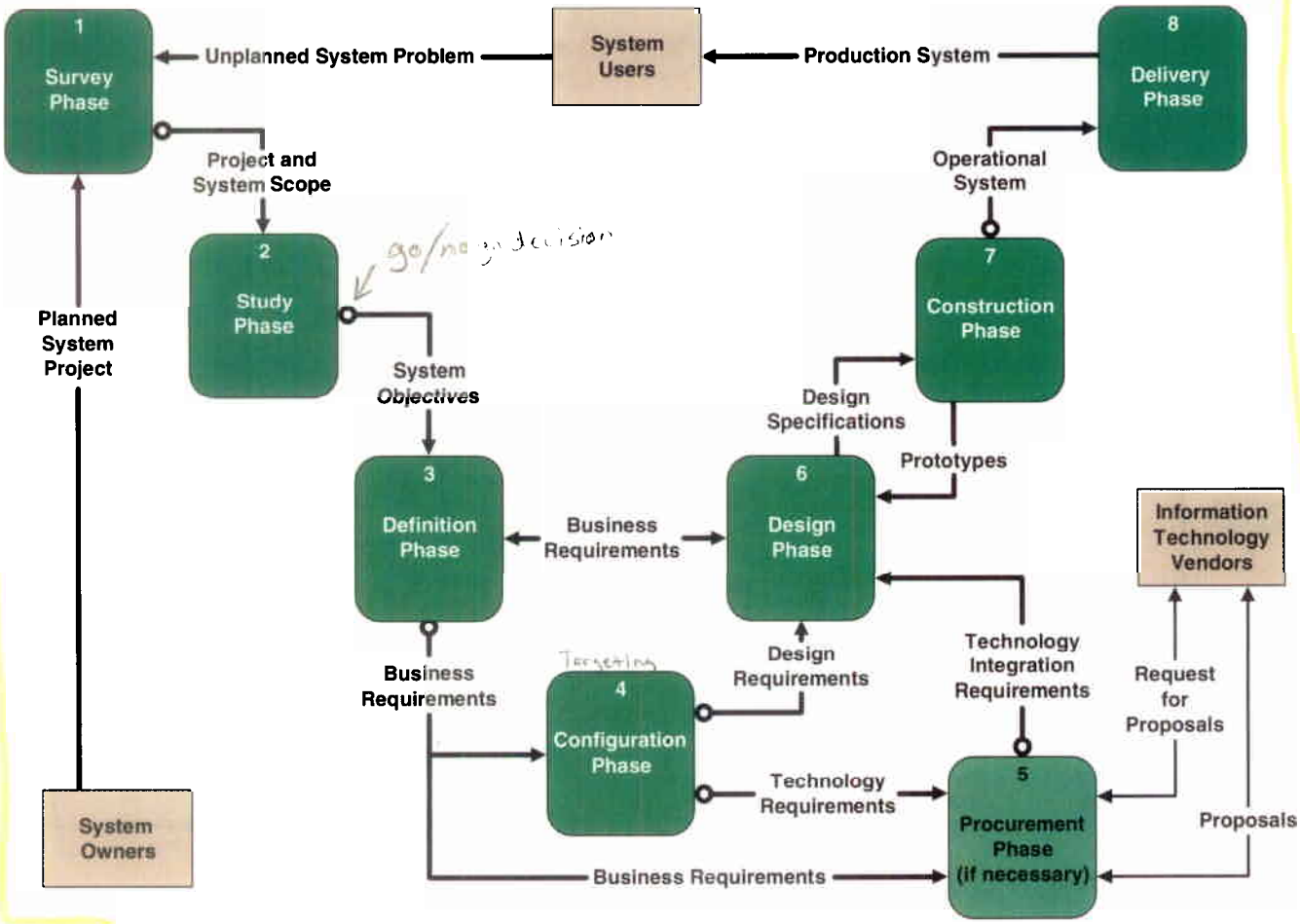


FIGURE 3.6 *System Development Phases*

5. The *procurement phase* (optional) identifies and analyzes hardware and software products that will be purchased as part of the target solution.
6. The *design phase* specifies the technical requirements for the target solution. Today, the design phase typically has significant overlap with the construction phase.
7. The *construction phase* builds and tests the actual solution (or interim prototypes of the solution).
8. The *delivery phase* puts the solution into daily production.

Some of these phases take little time to complete. The time required to complete other phases depends on strategy and risk. The phases are not necessarily sequential; they typically overlap.

Given this high-level overview of *FAST*, let's examine each of the phases with respect to (1) purpose, (2) participants and roles, (3) prerequisites, (4) activities, (5) deliverables, (6) postrequisites (based on continuous feasibility reassessment), and (7) impact analysis.

Figure 3.7 provides a single reference point for all this discussion. You might want to mark that page for easy access. Figure 3.8 illustrates the phases in the context of your information systems framework (from Chapter 2). You might also want to mark that figure for easy access.

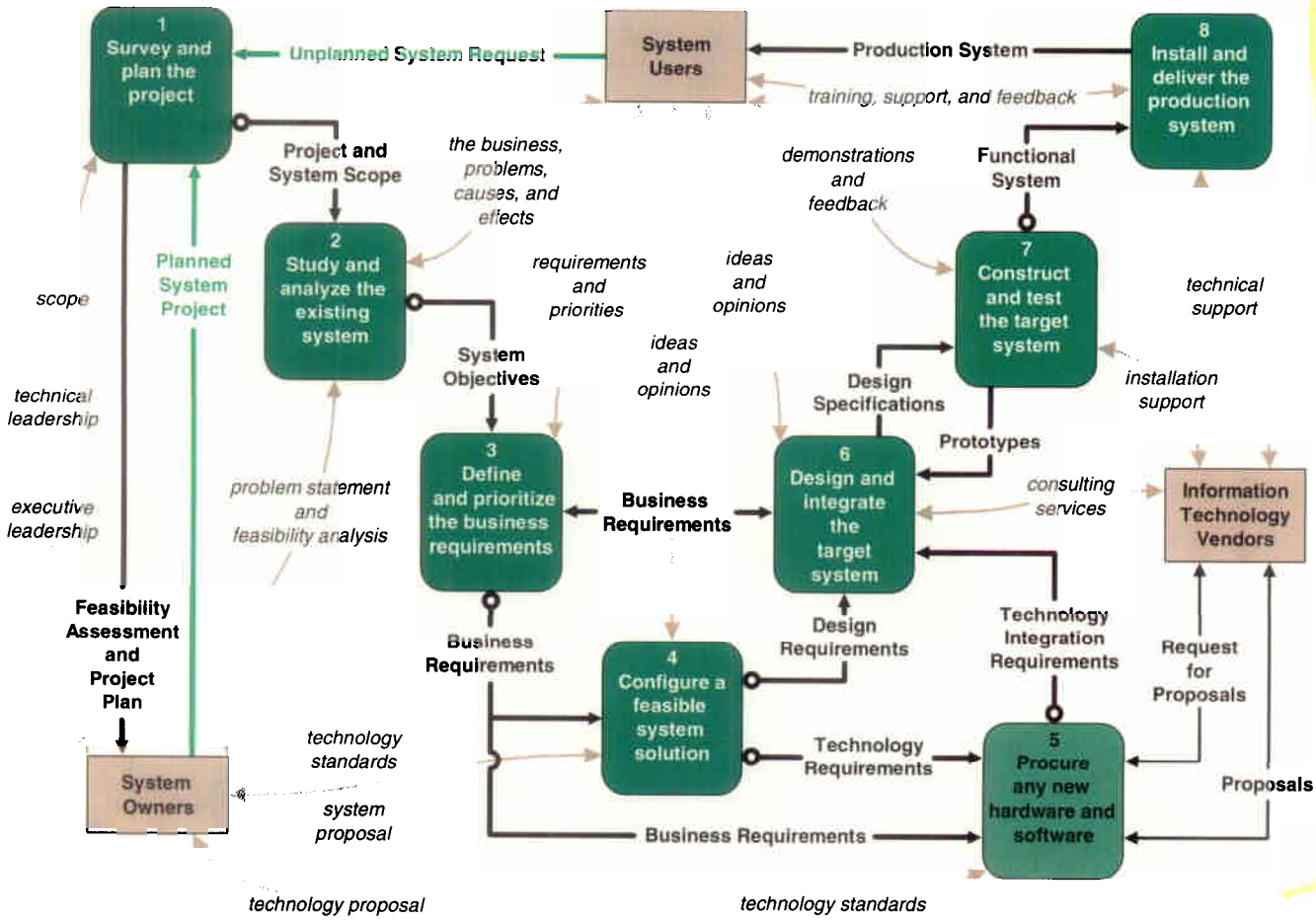


FIGURE 3.7 *The FAST Phase Diagram*

The Survey Phase

Systems development can be very expensive. Thus, it pays to answer the important question, "Is this project worth looking at?" Thus, the first phase of a project is to survey the project. The **survey phase** is also sometimes called a preliminary investigation or feasibility study. It amounts to a "quick-and-dirty" (usually two to three days) preliminary investigation of the problems, opportunities, and/or directives that triggered the project.

Purpose The purpose of the survey phase is threefold. First, the survey phase answers the question, "Is this project worth looking at?" To answer this question, the survey phase must define the scope of the project and the perceived problems, opportunities, and directives that triggered the project. Assuming the project *is* worth looking at, the survey phase must also establish the project team and participants, the project budget, and the project schedule.

Participants and Roles The facilitator of this phase is the systems analyst. Because this phase describes the system and project from the perspective of system owners, those owners are usually the only other participants. It is appropriate, however, to now expand our definition of system owner to include the following roles:

- *Executive sponsor*—the highest-level manager who will pay for the project. More importantly, this manager should provide visible support and leadership to the project participants. Cooperation and compromise among participants are often dependent on this leadership.

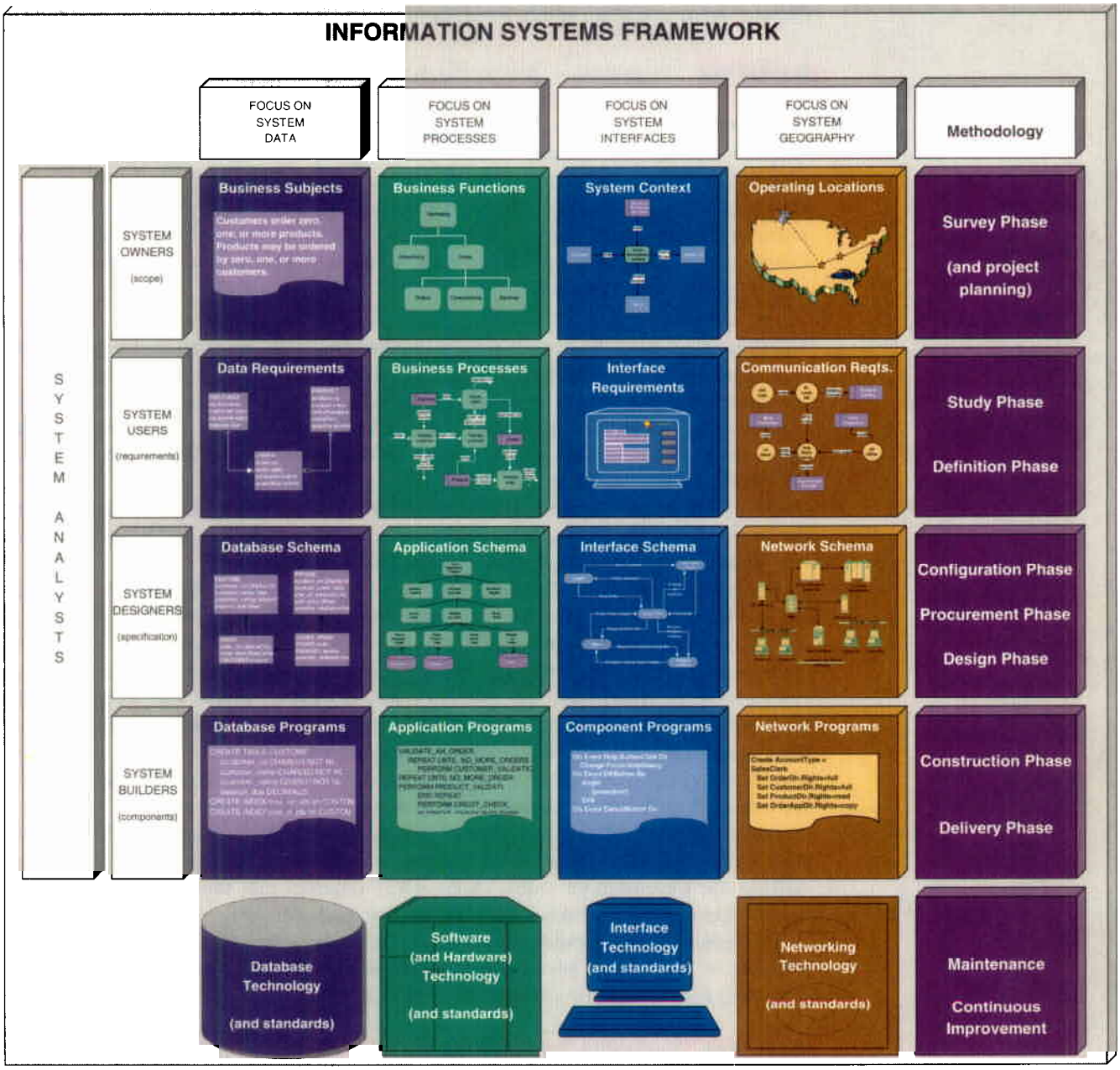


FIGURE 3.8 *FAST and Its Relationship to the Information System Framework*

- *Technical sponsor*—the highest-level manager from the information services organization who will pay for the project. This may be the chief information officer, the director of all systems development, or the manager of all information systems for a specific business function.
- *Project managers*—the managers of the project team. This person is responsible for the staffing, budget, and schedule. The project manager may be a senior systems analyst, a full-time project manager, or a significant staff member from the system user community. *FAST* encourages joint management of the project by an information services worker and a ranking manager from the user community.

Other system users are rarely involved until the next phase.

Prerequisites Prerequisites define the required and optional inputs for a phase. The key input to the phase is either the **unplanned system request** or the **planned system initiative.** These project triggers were described earlier in the chapter. *FAST* accommodates both formal triggers (such as a form) and informal triggers (such as a memo).

Figure 3.7 also illustrates the need for *executive leadership* and *technical leadership* from the system owners. (Note: This is our first example of secondary information and communication flows.) Also, *scope* is negotiated between the systems analyst and system owners.

Activities The most important activity in the survey phase is to define the scope or size of the project. Many projects fail because of poor scope management. If uncontrolled, scope tends to grow (along with costs and missed deadlines). You can't manage scope if you don't define it. Think of scope as a statement of the users' expectations of the project. Once defined, you can renegotiate for additional resources if users increase that scope. Using the information systems framework (Figure 3.8), project scope can be defined in terms of the system owner building blocks for DATA, PROCESSES, INTERFACES, and GEOGRAPHY. Project scope also includes the identification of other system owners and system users.

Once scope has been defined, we need to answer the question—"Is this project worth looking at?" To answer this question, the analyst should discuss the perceived problems, opportunities, and directives with the system owners. The PIECES framework provides an excellent outline for such a discussion. The goal here is not to solve the problems, just to catalog and categorize them. The analyst should also explore any constraints that may impact the project, such as budget, schedule, or technology standards. Finally, the analyst can discuss the question with the system owners. What impact would solving these problems have on the business? Generally, it is too early to assess the cost of developing the system.

Assuming the system is worth looking at, the project manager should formally plan the project. This includes establishing a preliminary budget and schedule and staffing the development team. (Note: *FAST* requires that the budget and schedule be reevaluated and refined at the end of each phase.)

The entire phase should not consume more than two to three days since the purpose of the survey phase is to decide whether or not more significant time and resources should be committed to the project.

Deliverables Deliverables are the outputs of a phase. As shown in Figure 3.7, a key deliverable for the survey phase is a *feasibility assessment and project plan*. This might be a report or verbal presentation, possibly both. The report version is sometimes called an *initial study report*. The analyst's recommendation may prescribe (1) a "quick fix," (2) an enhancement of the existing system and software, or (3) a completely new information system. For the latter possibility, a statement of **project and system scope** should be prepared as a deliverable to the next phase.

Postrequisites and Feasibility Checkpoints Recall that a circle at the beginning of any information flow in Figure 3.7 indicates the flow may or may not occur based on our creeping commitment principle. Thus, these circles define feasibility checkpoints in *FAST*. In other words, project and system scope will only occur if the project has been approved to continue to the next phase.

The **feasibility assessment and project plan** must usually be reviewed by the system owners (or a steering committee that includes system owners). This is especially common when many proposed projects are competing for the same

resources. One of four decisions is possible: (1) approve the project to continue to the study phase, (2) change the scope and continue on to the study phase, (3) reject the project, or (4) delay the project in favor of some other project.

Impact Analysis Can the survey phase ever be skipped? Scope definition is critical to all projects, planned and unplanned, but it could be deferred until the study phase for those projects that have already been determined to be worth looking at. For example, some organizations plan all or most projects as part of an ongoing *strategic information systems plan*. For another example, some projects are driven by extremely strong administrative directives. Thus, if a vice president *insists* you look at an application, it would not be wise to question worthiness at this point in the project.

The Study Phase

There's an old saying that suggests, "Don't try to fix it unless you understand it." With those words of wisdom, the next phase of a *FAST* project is to study and analyze the existing system. There is always an existing business system, regardless of whether it currently uses a computer. The **study phase** provides the project team with a more thorough understanding of the problems, opportunities, and/or directives that triggered the project. The analyst frequently uncovers new problems and opportunities. The study phase may answer the questions, "Are the problems worth solving?" and "Is a new system worth building?"

Purpose The purpose of the study phase is threefold. First and foremost, the project team must gain an appropriate understanding of the business problem domain. Second, we need to answer the question, "Are these problems (opportunities and directives) worth solving?" Finally, we need to determine if the system is worth developing. The study phase provides the systems analyst and project team with a more thorough understanding of the problems, opportunities, and/or directives that triggered the project. In the process, they frequently uncover new problems and opportunities.

Participants and Roles Once again, the systems analyst facilitates this phase. But system users are actively involved in the study. Although some system users (or business analysts) may be permanently assigned to the project team, all system users should be involved to some degree (such as interviews, group meetings, formal presentations, progress reports, reports of findings, etc.).

System owners must visibly support the study to ensure that all system users actively participate. Also, the findings of the study phase should be presented to or reviewed with the system owners.

Prerequisites As shown in Figure 3.7, the key input is the statement of project and system scope from the survey phase. The project team studies the existing system by collecting factual information from the system users concerning *the business* and the perceived *problems, causes, and effects*. It is also appropriate to discuss system limitations with information services specialists who support the existing system. From all of this information, the project team gains a better understanding of the existing system's problems and limitations.

Activities Every existing system has its own terminology, history, culture, and nuances. Learning those aspects of the system is the principle activity in this phase. Depending on the complexity of the problem domain and the project schedule, the team may or may not choose to formally document the system. Your information system framework (Figure 3.8) provides an effective outline for studying the existing system from the perspective of system users as they see the DATA, PROCESS,

INTERFACE, and GEOGRAPHY building blocks. In the next unit of the book, you will discover and learn how to use specific tools and techniques for this purpose.

Given an understanding of the current system, the project team analyzes the perceived problems, opportunities, and directives. Do the problems and opportunities really exist? If so, how serious are they? Many times, the initial problems are mere symptoms, frequently of more serious or subtle problems. During the study phase, we need to address the causes and effects of the problems, opportunities, and directives. The PIECES framework presented earlier in this chapter can be used as an effective tool for organizing the analyses.

Deliverables The findings of the study phase are reviewed with the system owners as a business problem statement and feasibility analysis (sometimes called a detailed study report). *FAST* encourages that the findings also be shared with system users. This problem statement may take the form of a formal written report, an updated feasibility assessment, or a formal presentation to management and users. SoundStage has standardized on a formal written report distributed during a formal presentation to owners and users.

If the project continues to the next phase, the project team should include system objectives in the aforementioned report or presentation. These objectives do not define inputs, outputs, or processes. Instead, they define the business criteria on which any new system will be evaluated. For instance, we might define an objective that states the new system must REDUCE THE TIME BETWEEN ORDER PROCESSING AND SHIPPING BY THREE DAYS, OR REDUCE BAD CREDIT LOSSES BY 45 PERCENT. Think of objectives as the "grading curve" for evaluating any new system that you might eventually design and implement.

Postrequisites and Feasibility Checkpoints After reviewing the findings, the system owners will either agree or disagree with the recommendations of the study phase. Thus, the system objectives information flow is marked as optional (with the circle) based on a feasibility reassessment. The project can be (1) canceled if the problems prove not worth solving, or a new system is not worth building, (2) approved to continue to the definition phase, or (3) reduced in scope or increased in budget and schedule, and then approved to continue to the definition phase.

Impact Analysis Can you ever skip the study phase? Rarely! You almost always need some understanding of the existing system. But there may be reasons to complete the phase as quickly as possible. If the project was triggered by a planned system initiative, the worthiness of both the project and system was determined earlier. Accordingly, the study phase requirements are reduced to understanding the current system.

As another example, if the project was triggered by a management directive (e.g., "We must have this system by February 1 in order to comply with new federal regulations.") then worthiness is definitely not in question. In fact, we may want to get through the study phase rather quickly to increase the likelihood of meeting the deadline.

FAST is flexible insofar as determining how much time should be spent on this phase. Like many other businesses, SoundStage used to spend much time documenting the existing system using *structured analysis* techniques. As you'll learn by studying the SoundStage project, it now spends much less time on documenting the existing system and more time on cause/effect analysis. The goal is to move every project into a "new system focus" (the definition phase) as quickly as possible.

The Definition Phase

Given approval of the business problem statement, now you can design a new system, right? No, not yet! What capabilities should the new system provide for its users? What data must be captured and stored? What performance level is expected?

Careful! This requires decisions about what the system must do, not how it should do those things. The next phase of a *FAST* project is to define and prioritize the business requirements. It is sometimes called a requirements analysis phase or simply, the **definition phase.** Simply stated, the analyst approaches the users to find out what they need or want out of the new system. This is perhaps the most important phase of the life cycle. Errors and omissions in the definition phase result in user dissatisfaction with the final system and costly modifications to that system.

Purpose Essentially, the purpose of requirements analysis is to identify the DATA, PROCESS, INTERFACE, and GEOGRAPHY requirements for the users of a new system. Most importantly, the purpose is to specify these requirements without expressing computer alternatives and technology details; at this point, keep analysis at the business level!

Participants and Roles The systems analyst or business analyst facilitates the definition and prioritization of business requirements. System users assigned to the team play an essential role in specifying, clarifying, and documenting the business requirements. It is, however, extremely important to involve system users not on the team. These users may provide perspectives and requirements not known to the users who are assigned to the project team. *Joint requirements planning* sessions (Chapter 5) are highly recommending as a technique to fully involve the user community in the requirements identification process.

Prerequisites As shown in Figure 3.7, the definition phase is triggered by an approved statement of system objectives. From the system users, the team collects and discusses requirements and priorities. This information is collected by way of interviews, questionnaires, and facilitated meetings. The challenge to the team is to validate those requirements.

Notice the two-way information flow, **business requirements,** between the definition phase and the design phase. Through this information flow, *FAST* recognizes that business requirements are sometimes not discovered until some level of design or prototyping activity occurs (described below).

Activities Intuitively, the identification of business requirements is the principle activity in this phase. Your information systems framework (Figure 3.8) indicates that the definition phase explores the requirements from the system users' perspectives of the DATA, PROCESS, INTERFACE, and GEOGRAPHY building blocks. The real challenge is to organize and synchronize these requirements in a way that permits system users to validate and prioritize the business requirements.

Clearly, the new system objectives from the study phase provide some degree of validation. But detailed validation requires the analyst to translate the users' words into a more precise representation of the requirements. The system users can then validate the requirements and offer corrected requirements and priorities. The most popular approach to documenting and validating users' requirements is modeling.

> **Modeling** is the act of drawing one or more graphical (meaning *picture-oriented*) representations of a system. The resulting picture represents the users' DATA, PROCESSING, INTERFACE, or GEOGRAPHY requirements from a business point of view.

This is the "picture is worth a thousand words" approach to validation. Different models will describe data requirements, process requirements, interface requirements, and geographic requirements. (You may be familiar with computer program

models such as structure charts, flowcharts, and pseudocode.) They model the modular structure and logic of a program. In this book you will learn several tools and techniques for modeling business requirements.

Another approach to documenting and validating requirements is prototyping.

> **Prototyping** is the act of building a small-scale, representative or working model of the users' requirements to discover or verify those requirements.

This is the "they'll know what they need when they see it" approach. The analyst uses powerful prototyping tools to quickly build computer-based prototypes. The users can then react to the prototype to help the analyst refine or add to the requirements. Prototypes can also be used to develop or refine the aforementioned system models.

Another activity in the definition phase is to prioritize requirements. This can be accomplished during or after the other activities. At the highest level, requirements can be classified as mandatory, desirable, or optional. Within these classifications, it may be necessary to rank the requirements to break the implementation into versions, the first of which could relatively quickly provide some value to the business.

Deliverables The final models and prototypes are usually organized into a business requirements statement. Some approaches call for great detail in this requirements statement, whereas others emphasize "the big picture." *FAST* supports both approaches based on the complexity and schedule of the project. We'll examine these approaches in Chapters 4 through 9. The requirements statement becomes the trigger for systems design.

Postrequisites and Feasibility Checkpoints Although it is rare, the project could still be canceled at the end of this phase—hence, the feasibility notation at the beginning of the business requirements information flow.

More realistically, the project scope (or schedule and budget) could be adjusted if it becomes apparent the new system's requirements are much more substantive than originally anticipated. Today, it is popular to time box a project based on the business requirements.

> **Time boxing** is a technique that divides the set of all business requirements for a system into subsets, each of which will be implemented as a version of the system. Essentially, the project team guarantees that new versions will be implemented on a regular and timely basis.

Time boxing is a modern response to historical system owner and user complaints that the Information Services unit of the business takes too long to develop systems. Usually, the time box is defined to be six to nine months in duration for each release. (Note: The packaged software industry has long used the time box approach to define and manage the release of new software versions.)

Regardless, if the project is not canceled, it proceeds to the targeting and design phases. Actually, the targeting phase can begin before we complete the definition phase because the targeting phase is not dependent on complete, detailed business requirements. Also, as previously noted, the design phase may have already started if we use prototyping techniques to define requirements.

Impact Analysis You can never skip the definition phase. One of the most common complaints about new systems and applications is that they don't really satisfy the users' needs. This is usually because information systems specialists found it difficult to separate "what" the user needed from "how" the new system would work. In other words, they became so preoccupied with the technical solution that they failed to consider the users' essential requirements. Requirements are a statement of what the system must do no matter how you design and implement it.

The definition phase formally separates "what" from "how" to properly define and prioritize those requirements.

FAST recognizes the need for more rapid system and application development, but the methodology also insists, "If you don't have time to do it right, how will you find (and justify) the time to do it over?"

The Configuration Phase

System design and construction are detailed, technical, and time-consuming phases. Given any statement of business requirements, there are usually numerous alternative ways to design the new system. Some of the pertinent questions include the following:

- How much of the system should be computerized?
- Should we purchase software or build it ourselves (called the make-versus-buy decision)?
- Should we design the system for centralized computing on a mainframe or minicomputer, or should we design a distributed computing solution utilizing PCs, server computers, and networks?
- What (emerging) information technologies might be useful for this application?

The questions are answered in the next phase of a *FAST* project, *configure a feasible system solution*.

Purpose There are almost always multiple candidate solutions to any set of business requirements. The purpose of the **configuration phase** is to identify candidate solutions, analyze those candidate solutions, and recommend a target system that will be designed and implemented.

Participants and Roles The systems analyst usually facilitates this phase. Alternatively, *FAST* allows information services to designate an *application architecture manager* to facilitate this phase for all projects.

All members of the project team including system owners, system users, and system designers must be involved in this key decision-making phase. Each class of participant brings its own ideas and perspectives to the table. Clearly, this is the first phase that involves system designers such as database administrators, software engineers or programmers, human interface specialists, and network administrators. Technology consultants may also become involved.

Prerequisites As shown in Figure 3.7, the configuration phase is triggered by a reasonably complete specification of business requirements. These business requirements are usually not extremely detailed; they identify inputs, outputs, databases, key functions or programs, and so forth. They do not typically specify precise details of those elements. In fact, the configuration phase begins before completion of the definition phase and may end after the design phase has already begun.

The project team also solicits ideas and opinions from a diverse audience. The project team also identifies or reviews any technology standards via the technology-oriented system owners.

Activities In your information system framework (Figure 3.8), the configuration phase is the first phase that begins to look at the perspectives of system designers. Also notice that the configuration phase is significantly influenced by the existing technology and standards that exist for the technology (shown at the bottom of the framework).

The first activity is to define the candidate solutions. Some candidate solutions will be proposed as design ideas and opinions by various sources (e.g., systems analysts and system designers, other information system managers and staff, technical consultants, system users, or information technology vendors). Some

technical choices may be limited by a predefined approved technology architecture provided by system managers.

After defining candidates, each candidate is evaluated by the following criteria:

- *Technical feasibility.* Is the solution technically practical? Does our staff have the technical expertise to design and build this solution?
- *Operational feasibility.* Will the solution fulfill the user's requirements? To what degree? How will the solution change the user's work environment? How do users feel about such a solution?
- *Economic feasibility.* Is the solution cost-effective (as defined earlier in the chapter)?
- *Schedule feasibility.* Can the solution be designed and implemented within an acceptable time period?

The final activity is to recommend a feasible candidate as the target system. Infeasible candidates are immediately eliminated from further consideration; however, several alternatives usually prove to be feasible. The project team is usually looking for the *most* feasible solution—the solution that offers the best combination of technical, operational, economic, and schedule feasibility.

Deliverables The key deliverable of the configuration phase is a formal *systems proposal* to system owners. The double-ended arrows indicate the system proposal must be presented, and usually negotiated, with the system owners who will usually make the final business and financial decisions. This proposal may be written or verbal.

If it is decided to purchase some or all of the target system (hardware or application software), the technology requirements must be forwarded to the purchasing phase (as shown on Figure 3.7).

Regardless, the solution design requirements must be provided to the design phase (which may already be in progress). These design requirements are not specifications. They are high-level architectural decisions that will constrain the detailed design of the system.

Postrequisites and Feasibility Checkpoints Clearly, several outcomes are possible from the configuration phase. Notice the deliverables are all marked as optional (circles) based on the feasibility analyses described earlier. Specifically, system owners might choose any one of the following options:

- Approve and fund the system proposal (possibly including an increased budget and timetable if scope has significantly expanded).
- Approve or fund one of the alternative system proposals.
- Reject all the proposals and either cancel the project or send it back for new recommendations.
- Approve a reduced-scope version of the proposed system.

Based on the decision, a procurement phase may be triggered. Also, based on the decision, the design phase (possibly already in progress) may be canceled or modified in scope or direction.

Impact Analysis The configuration phase is not always required. Like many modern methodologies, *FAST* encourages businesses to develop an application architecture.

> An **application architecture** defines an approved set of technologies to be used when building any new information system.

Using *FAST*'s application architecture project template, SoundStage has already defined its approved technologies. For example, it has standardized on Microsoft *Access* as its PC database technology and Microsoft *SQL Server* as its enterprise

database engine. It has also standardized its approved programming languages, network topologies, and user and system interface technologies. We'll see SoundStage apply these standards in its member services project.

Businesses that apply an application architecture must be willing to invest the time and effort to continuously improve that architecture based on business and technological change.

The Procurement Phase

This phase is missing from many methodologies. College graduates are often shocked to discover the high percentage of computer software that is purchased (or leased) rather than built. Also, any new system may present the need to acquire additional hardware, such as personal computers or printers. Recall that the make-versus-buy decision was made in the configuration phase. If the decision includes a "buy" component, then the next phase of systems design is to purchase any new hardware and software. It is sometimes called the acquisition or purchasing phase; *FAST* calls it the **procurement phase.** This phase is only required if new technology needs to be purchased.

Purpose The purpose of the procurement phase is to research the information technology marketplace, solicit vendor proposals, and recommend (to management) the proposal that best fulfills the business and technology requirements. Why include this phase in a methodology? The selection of hardware and software takes time. Much of that time can occur between order and delivery. This time lag must be figured into the methodology to schedule the subsequent life cycle phases!

Participants and Roles The key facilitator in the procurement phase is still the systems analyst; however, several other parties get involved. Clearly, the information technology vendors (who sell hardware and/or software) get involved. Also, users (both those on the project team and those in the user community) must be involved since they must ultimately live with the system. System owners must be involved because these purchases usually exceed the authorized spending limits of the average project team. In most businesses, purchasing agents and legal staff must be involved in negotiations for any contracts and service agreements.

Prerequisites As shown in Figure 3.7, the key inputs to the phase are business requirements from the definition phase and technology requirements from the configuration phase. The project team should also be aware of any technology standards imposed by systems management.

Activities In your information system framework (Figure 3.8), the procurement phase considers the perspectives of system designers. It should be noted, however, that the activities of the procurement phase ultimately result in the selection of one or more of the information technologies illustrated at the bottom of the information system framework. The most common purchased component is an application software package to implement the PROCESS column. But in some cases, the project team may only purchase the technologies required to design and construct the system (for example, a database management system, a new programming language, or a local area network operating system).

The project team's initial activity is to research the technology and marketplace. Although initial inquiries might be made to information technology vendors, the first source of information is usually gained by studying the many periodicals and services that survey the technology marketplace.

Subsequently, the project team organizes the business, technology, and relationship requirements and establishes the mechanisms that will be used to evaluate the technical alternatives. These requirements and mechanisms are communicated to the vendors as a **request for proposals.** The two-way arrow indicates that clarification of these requirements is normal.

The vendors usually respond with formal proposals that may also have to be clarified or negotiated. The project team must evaluate proposals and quotes to determine (1) which ones meet requirements and specifications and (2) which one is the *most* cost-effective. The analysts make a recommendation to the system owners (and usually the information system managers as well). This recommendation may also be negotiated. Finally, the authorized agents of the business execute the final orders, contracts, licenses, and service agreements.

Deliverables The key deliverable of the procurement phase (once again shown on Figure 3.7) is a technology proposal to system owners to acquire specific hardware and/or software. If that proposal is approved, then a technology integration requirements statement is passed on to the design phase. This statement describes, in detail, the approved hardware and/or software and requirements for integrating that technology into the overall system design. It is important to include any unfilled requirements in this statement since the design phase must then build in those missing features and capabilities.

Postrequisites and Feasibility Checkpoints The procurement phase is followed by the design phase unless the purchased software fully meets the business and technology requirements of the project, a very rare occurrence! The design phase will integrate the purchased system into the overall system, adding any necessary functionality and features that could not be purchased.

In the rare case where a purchased system fully meets requirements (sometimes called a *turn-key system* because you just turn the key to start the system), the project proceeds immediately to the *delivery phase*.

Sometimes the procurement phase results in a "no decision." Either the users were unsatisfied with the business solution of any package, or the technical staff was unsatisfied with the technology or support. In this case, the project proceeds directly to the design phase to be designed and constructed in-house as a custom solution. That explains the optional notation on the technology integration requirements information flow.

It is rare that the entire project would be canceled at this stage; however, if a purchased package was targeted as the *only* feasible solution, and no satisfactory package could be found, then the project could be canceled.

Impact Analysis This phase is entirely optional based on the make-versus-buy decision in the configuration phase. If the decision is to buy, shortcuts through *FAST*'s recommended procurement activities (covered in detail later in the book) are strongly discouraged. You tend to get what you pay for and only what you pay for in the software industry. Unfortunately, the software industry is somewhat notorious for unfulfilled promises; it pays to thoroughly research, evaluate, and negotiate all agreements.

The Design Phase

Given the approved, feasible solution from the configuration phase, you can finally design and integrate the target system. You understand what the business requirements are from the definition phase and how you plan to fulfill those requirements from the configuration phase. Thus, you can now justify the time and cost to fully design the new system. Ideally, any new system should work in harmony with other current information systems. Similarly, if we have purchased software packages, those packages must work in harmony with any components of the systems that are to be built in-house. For these reasons, we not only design the new system but we also integrate the new system as well.

Purpose The purpose of the **design phase** is to transform the business requirements from the definition phase into a set of technical design blueprints for con-

struction. *FAST* does not support the traditional design-then-construct strategy wherein you complete a detailed design before starting construction. Instead, *FAST* encourages an iterative design-and-construct strategy. Some specification must precede the first pass through the design-and-construct approach; however, *FAST* encourages that most of the design be tested by developing a series of prototypes that evolve into the final system. This contemporary strategy is called rapid application development or RAD.

Participants and Roles The key facilitator of the design and integration phase is still the systems analyst. But various other design specialists play important roles. For instance, database specialists might design or approve the design of any new or modified databases. Network specialists might design or modify the structure of any computer networks. Microcomputer specialists may assist in the design of workstation-based software components. Human interface specialists may assist in the design of the user interface. And as always, the system users must be involved. They evaluate the new system's ease of learning, ease of use, and compatibility with the stated business requirements.

Prerequisites As shown in Figure 3.7, the design phase has two triggers: the business requirements (from the definition phase) and the design requirements (from the configuration phase). In those projects that will purchase hardware and/or software, the design phase also receives technology integration requirements (from the purchasing phase). System users provide various ideas and opinions into or about the system's design.

When appropriate and cost-effective, the design phase can employ consulting services from information technology vendors. For example, if a specific vendor's bar-coding technology has been chosen for the target system, the vendor of that technology might be engaged to help integrate the technology into the target system.

Activities In your information system framework (Figure 3.8), and as expected, the design phase builds views of the system from the perspective of the system designers. Various design specialists design databases, programs, user interfaces, and networks to support the new system.

As noted earlier, *FAST* is like many contemporary methodologies in that it has effectively merged the design and construction phases to form a **rapid application development** (or **RAD**) approach based on iterative prototyping. Shown more clearly in Figure 3.7, this strategy designs and constructs the system as a series of **prototypes** to which the system users react. (Notice the loop between the design and construction phases, and the two-way arrow between the design and definition phases.)

Prototyping was introduced in the discussion of the definition phase. Design by prototyping allows the analyst to quickly create scaled-down but working versions of a system or subsystem. In the interest of speed, certain features and capabilities, such as input editing, may be left out of prototypes. Prototypes will typically go through a series of iterations and user reviews until they evolve into an acceptable design. The entire process works something like this:

1. Define the base-level scope of the first (or next) version of the system.
2. Define, design, and construct the database (only!). Load that database with some test data and review it with the system users. Make any corrections they request (to the test database only!).
3. Define, design, and construct the inputs (only!). Demonstrate this prototype to the system users. Repeat step 3 until system users are satisfied. If necessary, return to step 1 to add new requirements to the database design.

4. Define, design, and construct the outputs (only!). Demonstrate this prototype to system users. Repeat step 4 until system users are satisfied. If necessary, return to step 1 to add new database requirements or step 2 to add new input requirements.
5. Define, design, and construct the interface. The interface is the glue that integrates all the above system components. Demonstrate this prototype to system users. Repeat step 5 until system users are satisfied. If necessary, return to step 1, 2, or 3 to add new database, input, or output requirements, respectively.
6. Design and construct any missing system controls such as security, backup, recovery, etc.
7. Implement this version of the system.
8. Go to step 1 to begin the RAD cycle for the next version of the system.

The basic idea of RAD is (1) to actively involve system users in the design process, (2) to accelerate the definition/design/construction process by catching errors and omissions earlier in the process, and (3) to reduce the amount of time that passes before the users begin to see a working system. RAD has become the strategy of choice in modern design methodologies.

Deliverables As described above, the design and construction phases are effectively merged in the *FAST* methodology. This section will describe only the design deliverables of the phase. The final deliverable is a technical set of design specifications. These specifications can take several forms, but the most common approach is modeling. (Modeling was introduced in the definition phase description.) Normally, general design models will depict:

- The structure of the database.
- The structure of the overall application.
- The overall look and feel of the user interface.
- The structure of the computer network.
- The design structures for any complex software to be written.

Postrequisites and Feasibility Checkpoints Once again, in a *FAST* project the design and construction phases are effectively merged. Thus, a project is rarely canceled in the design phase, unless it is hopelessly over budget or behind schedule. As shown in Figure 3.7, each constructed prototype is refined or expanded by another pass through system design until the final system is constructed.

Impact Analysis Not surprisingly, the design phase cannot be skipped. In the true spirit of classical engineering, there is no such thing as a prototype that was not preceded by some level of design.

The Construction Phase

We just learned that in the *FAST* methodology, the **construction phase** is actually part of a design/construction loop that implements rapid application development. Given some aspect of the system design, we construct and test the system components in that design. After several iterations of the design/construction loop, we will have built the functional system to be implemented.

Purpose The purpose of the construction phase is twofold: (1) to build and test a functional system that fulfills business and design requirements, and (2) to implement the interfaces between the new system and existing production systems.

Participants and Roles The construction phase is still facilitated by the systems analyst; however, the analysts and users often play a less visible role. The analyst

serves as a general contractor for work done by technical specialists or subcontractors. Examples include database programmers, application programmers, and network administrators. (Keep in mind that *system builders* are roles, not job descriptions. In practice, it is common for a single person to play the role of both designer and builder.) System users' responsibilities are usually limited to reacting to the functional system's ease of learning and ease of use.

Prerequisites As shown in Figure 3.8, the design specifications (general or detailed) are the key input to the construction phase. Information technology vendors may provide installation support for any packaged software or software development tools.

Activities Your information system framework (Figure 3.8) identifies the relevant building blocks and activities for the construction phase. The project team must construct the database, application programs, user and system interfaces, and networks. Some of these elements may already exist (subject to enhancement).

Usually, the database and networks provide the system's infrastructure; therefore, they must be constructed first (unless they already exist). Next, any new software packages must be installed and tested. Finally, any new programs must be constructed and tested. You may already have some experience with the principal technique in this activity—application programming. Programs can be written in many different languages, but the current trend is toward the use of visual and object-oriented programming languages such as Powersoft's *Powerbuilder,* Microsoft's *Visual BASIC,* Borland's *Delphi,* Digitalk's *Visual Smalltalk,* Sun's *Java,* and Microsoft's *Visual C++*. (Note: Visual and object-oriented versions of our old standby, *COBOL,* are also beginning to emerge.)

One of the most important aspects of application programming is testing—both unit and system testing.

Unit tests ensure that the applications programs work properly when tested in isolation from other applications programs.

System tests ensure that applications programs written in isolation work properly when they are integrated into the total system.

It is common for programs that work perfectly by themselves to fail to work when combined with other related programs. If this happens, the programmer must often return to the build/test phase.

The install/test phase is also used to install purchased or leased software packages. Even those packages must be system tested to ensure they properly interact with other programs and packages. Furthermore, the programmer may have built special programs to connect the package with other programs, files, and databases. These integrated solutions must also be system tested.

Deliverables The final deliverable of the construction phase is the functional system; however, the rapid application development strategy of *FAST* results in several interim deliverables called prototypes. The prototypes are subject to demonstrations and feedback from the system users. Each prototype then cycles back through the design and definition phase until the final, functional system is acceptable.

Postrequisites, Feasibility Checkpoints, and Impact Analysis After the functional system has been completed, all that remains is the implementation phase. Although Figure 3.7 marks this phase as optional, the project is rarely canceled after the construction phase. It is possible that a prototype might be implemented as a first (next) version before the system has been fully constructed.

The Delivery Phase

What's left to do? New systems usually represent a departure from the way business is currently done; therefore, the analyst must provide for a smooth transition from the old system to the new system and help users cope with normal start-up problems. Analysts must also train users, write various manuals, and load files and databases. Thus, the last phase of implementation is to deliver the production system into operation.

Purpose The purpose of the **delivery phase** is to install, deploy, and place the new system into operation or production.

Participants and Roles The system analysts are still the facilitators but again become the most visible players as they communicate implementation problems and issues among system users, system designers, and system builders. The entire project team, inclusive of owners, users, designers, and builders, is active in this phase. Ideally, system owners and users step to the forefront as cheerleaders for the new system.

Prerequisites The functional information system is the key input to the delivery phase of the functional system (see Figure 3.7). System users provide continuous feedback as new problems and issues are common (note: no system has achieved the nirvana goal of perfection). For new information technology (hardware and software), the information technology vendors provide necessary technical support.

Activities In your information system framework (Figure 3.8) the delivery phase considers the system builders' perspective. During this phase, tests are conducted to ensure that the new system works properly. A conversion plan must then be prepared to provide a smooth transition to the new system. This plan may call for an abrupt cutover where the old system is terminated and replaced by the new system on a specific date. Alternatively, the plan may run the old and new systems in parallel until the new system has been monitored and deemed acceptable to replace the old system. Prior to converting to the new system, the system builders must install databases to be used by the new system. This involves loading production data to be used by the delivered system.

The delivery phase involves training individuals that will use the final system and developing documentation to aid the system users. The delivery phase normally concludes with a postaudit to gauge the success of the completed systems project. This activity represents a TQM effort that will contribute to the success of future systems projects.

Deliverables The final deliverable of the delivery phase (and the project) is the production system for the system users. Various follow-up reports or meetings (not depicted in Figure 3.7) are recommended to evaluate and improve both the final product (the information system), the process used to build it (the *FAST* methodology), and the participants (the project team).

Another output of the delivery phase is training and support. Training includes user documentation and formal classes. Support includes day-to-day assistance to system users when they encounter problems and undiscovered errors.

Postrequisites, Feasibility Checkpoints, and Impact Analysis The project is complete! There is no further feasibility analysis. But, as noted earlier, there may be a project postaudit to evaluate the system, methodology, and team. The system is continuously evaluated to log problems and new requirements that can be fixed in future versions. The system is also evaluated to determine whether it truly solved the intended problems and exploited the opportunities. The methodology is evaluated to reuse and improve the tools, techniques, and decision making of the systems development activities. The team is evaluated to assess education and training needs that can improve future performance.

Beyond Systems Development—Systems Support

Once the system is placed into production, the analyst's role changes to systems support. In fact, a significant portion of most systems analysts' time and effort is spent providing ongoing support for existing systems.

Systems support is the ongoing maintenance of a system after it has been placed into operation. This includes program maintenance and system improvements.

In your information system framework, systems support maintains all the building blocks (and their documentation) for a production information system. Systems analysts usually coordinate systems support, calling on the services of maintenance programmers and system designers as necessary. Systems support doesn't consist of phases so much as it does ongoing activities. These activities include:

- *Fixing software bugs.* Software bugs are errors that slipped through the testing phases during software construction.
- *Recovering the system.* From time to time, a system failure will result in an aborted program or loss of data. This may have been caused by human error or a hardware or software failure. The systems analyst may then be called on to recover the system—that is, to restore a system's files and databases, and to restart the system.
- *Assisting users.* Regardless of how well the users have been trained and how good the end-user documentation is, users will eventually require additional assistance—unanticipated problems arise, new users are added, and so forth.
- *Adapting the system to new requirements.* New requirements may include new business problems, new business requirements, new technical problems, or new technology requirements.

All these support activities continue through the lifetime of the production system. In *FAST*, the support activities are supported by a simplified version of the same phases used to build the system.

CROSS LIFE CYCLE ACTIVITIES

Systems development also involves a number of activities called cross life cycle activities.

Cross life cycle activities are activities that overlap many or all phases of the methodology. In fact, they are normally performed in conjunction with several phases of the methodology.

Cross life cycle activities include fact-finding, documentation and presentation, estimation and measurement, feasibility analysis, project management, and process management. Let's briefly examine each of these activities. The cross life cycle nature of these activities is illustrated in the time bar chart in Figure 3.9.

Fact-Finding

There are many occasions for fact-finding (or information gathering) during the project.

Fact-finding—also called information gathering or data collection—is the formal process of using research, interviews, meetings, questionnaires, sampling, and other techniques to collect information about systems, requirements, and preferences.

Fact-finding is most crucial to the survey, study, and definition phases of a *FAST* project. It is during these phases that the project team learns about a business's and system's vocabulary, problems, opportunities, constraints, requirements, and priorities.

Fact-finding is also used during the configuration, design, and construction phases but to a lesser extent. For instance, in systems design, fact-finding becomes technical as the project team attempts to learn more about the technology selected for the new system.

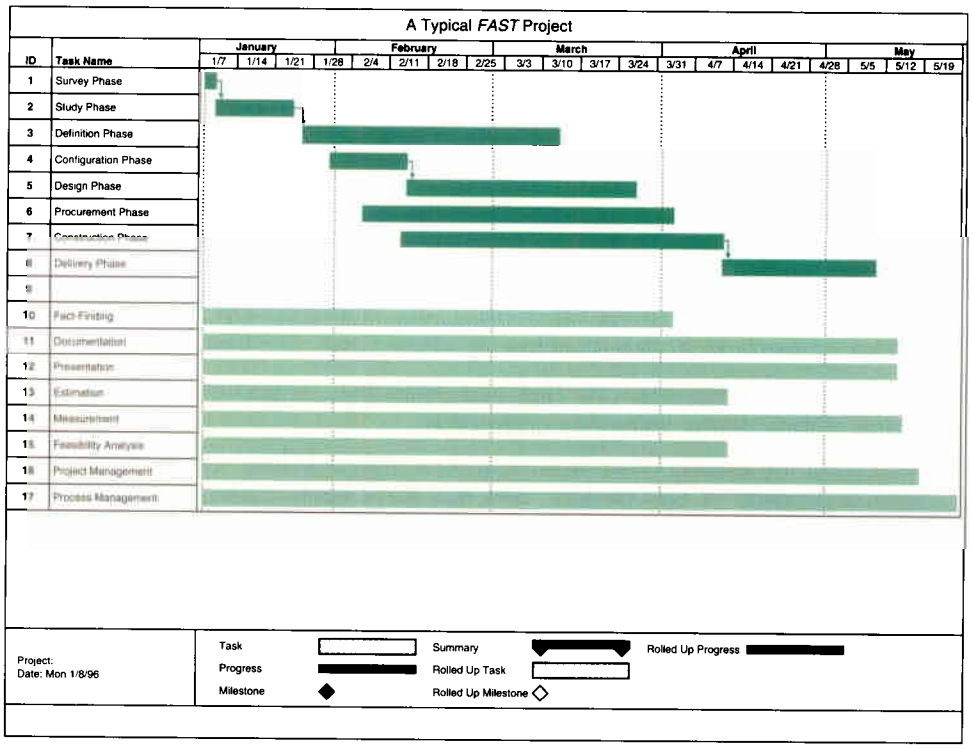


FIGURE 3.9
Cross Life Cycle Activities (and Overlap of the Other Development Phases)

Documentation and Presentations

Communication skills are essential to the successful completion of a project. Two forms of communication that are common to systems development projects are documentation and presentation.

Documentation is the activity of recording facts and specifications for a system.

Presentation is the related activity of formally packaging documentation for review by interested users and managers. Presentations may be either written or verbal.

Clearly, documentation and presentation opportunities span the entire *FAST* methodology and support all the phases.

As described throughout this chapter, documentation for a project is built during various phases and activities. Over time, a sizable base of documentation for various systems and applications is compiled. Many businesses try to get control over such documentation so it is kept and maintained for future use. Version control over documentation has become a critical success factor; it involves keeping and tracking multiple versions of a system's documentation. At a minimum and at any given time, most information systems shops want to keep documentation for all the following versions:

- One or more previous versions of the system.
- The current production version of the system.
- Any version of the system going through the build and test activity.
- Any version going through the life cycle to create a new version.

In Figure 3.7, the arrows on the flow-oriented models have been used to represent the typical use and updating of project documentation and the presentation of documentation to various audiences by systems analysts and programmers. In practice, these information flows don't go from phase to phase. Instead, interim results are stored in the repository, database, and program libraries introduced earlier in the chapter (refer back to Figure 3.6).

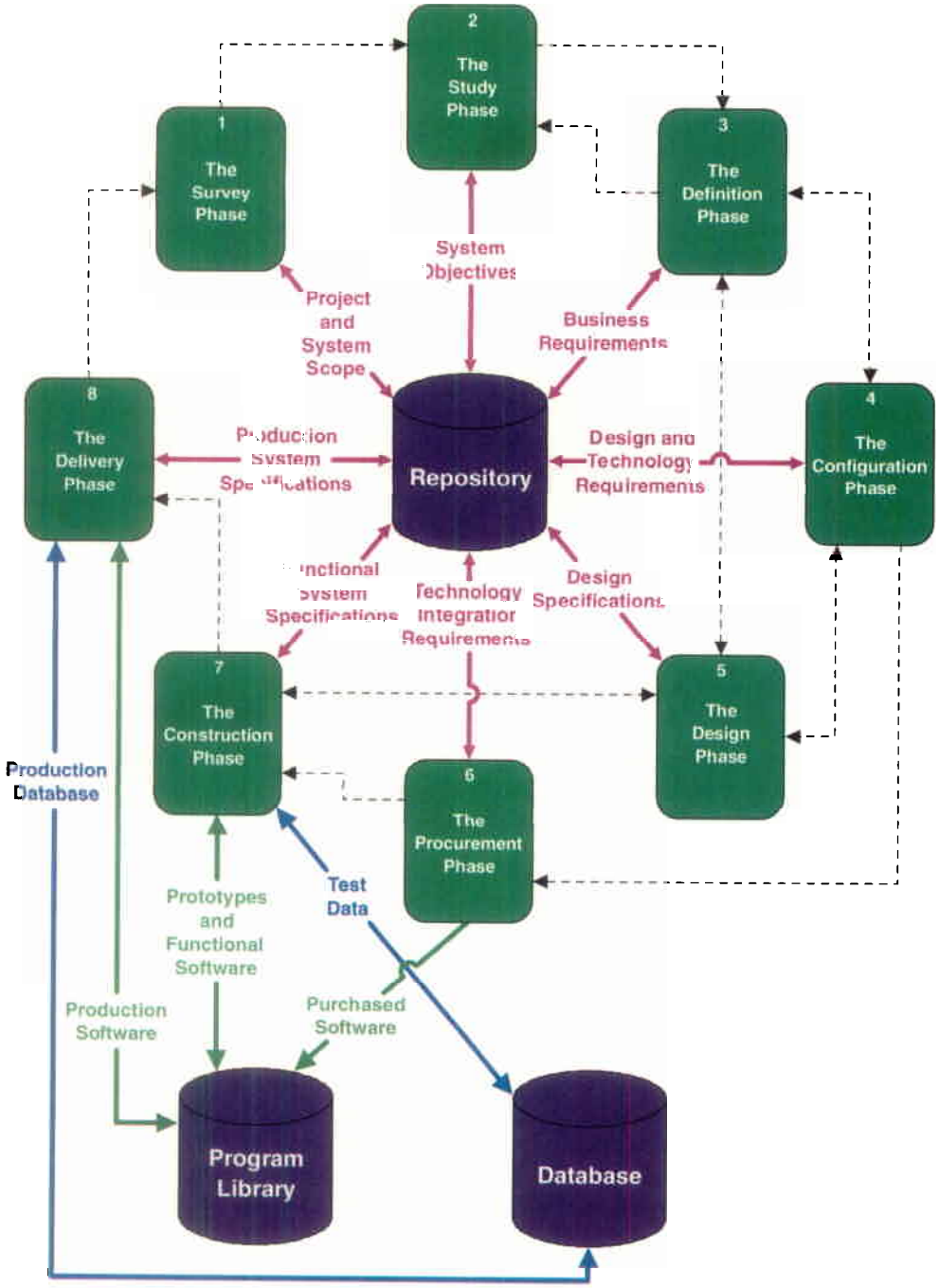


FIGURE 3.10
The FAST Phase Diagram—Repository View

To better illustrate the repository concept, the *FAST* methodology phases are redrawn in Figure 3.10 to illustrate the sharing of knowledge and system components. To simplify this diagram, we (1) used the phase nicknames and (2) eliminated the interpersonal communications information flows. The dashed lines show the flow of activities through the project. Because the phases communicate across the common data stores, they can overlap, thus accelerating the project.

Estimation and Measurement

Information systems are significant capital investments. For this reason, estimation and measurement activities are commonly performed to address the quality and productivity of systems.

Estimation is the activity of approximating the time, effort, costs, and benefits of developing systems. The term guesstimation (as in "make a guess") is used to describe the same activity in the absence of reliable data.

Measurement is the activity of measuring and analyzing developer productivity and quality (and sometimes costs).

Estimating is an extremely important activity because information systems, as noted earlier in the chapter, are capital investments. You don't want to spend $25,000 of time and effort to solve a problem that is costing your organization $2,000 per year. The payback would take more than a decade.

Estimating can be a difficult and frustrating activity. It is difficult to translate an abstract problem statement into a precise estimate of the time, effort, and costs needed to solve that problem. Multiple factors influence the estimate. There are two common approaches to estimation. First, some analysts avoid estimation out of fear, uncertainty, or lack of confidence. In this case, the analyst may resort to what are jokingly called "guesstimates." Alternatively, better analysts draw on experience and data (both their own and the collective experience of others) from previous projects to continually improve their estimates.

Measurement has become important because of the productivity and quality problems that plague systems development. In response to those problems, the industry has developed methods and tools to improve both quality and productivity. These methods and tools can be costly. Formal measurement of development productivity may be the only way to justify the cost of these expenses.

The field of software and systems metrics offers hope for the future.

Software and systems metrics provides an encyclopedia of techniques and tools that can both simplify the estimation process and provide a statistical database of estimates versus performance.

Feasibility Analysis

A system development life cycle that supports our creeping commitment approach to systems development recognizes feasibility analysis as a cross life cycle activity.

Feasibility is a measure of how beneficial the development of an information system would be to an organization.

Feasibility analysis is the activity by which feasibility is measured.

Too many projects call for premature solutions and estimates. This approach often results in an overcommitment to the project. If analysts are so accurate in feasibility estimates, why then are so many information system projects late and over budget? Systems analysts tend to be overly optimistic in the early stages of a project. They underestimate the size and scope of a project because they haven't yet completed a detailed study.

A project that is feasible at any given stage of system development may become less feasible or infeasible later. For this reason, we use the creeping commitment approach to reevaluate feasibility at appropriate checkpoints (indicated by small circles) in Figure 3.7.

Various measures of feasibility were introduced in the targeting phase. These measures included technical feasibility, operational feasibility, economic feasibility, and schedule feasibility.

Project and Process Management

Systems development projects may involve a team of analysts, programmers, users, and other IS professionals who work together.

Project management is the ongoing activity by which an analyst plans, delegates, directs, and controls progress to develop an acceptable system within the allotted time and budget.

Failures and limited successes of systems development projects far outnumber

very successful information systems. Why is that? One reason is that many systems analysts are unfamiliar with or undisciplined in the tools and techniques of systems development. But most failures are attributed to poor leadership and management. This mismanagement results in unfulfilled or unidentified requirements, cost overruns, and late delivery.

The systems development life cycle provides the basic framework for the management of systems projects. Because projects may be quite large and complex, the life cycle's phased approach to the project results in smaller, more measurable milestones that are more easily managed.

Process management is an ongoing activity that establishes standards for activities, methods, tools, and deliverables of the life cycle.

Process management is a relatively new concept in systems development. The intent is to standardize both the way we approach projects and the deliverables we produce during projects.

COMPUTER-AIDED SYSTEMS ENGINEERING (CASE)

You may be familiar with the old story of the cobbler (shoemaker) whose own children had no shoes. That situation is not unlike the one faced by systems developers. For years we've been applying information technology to solve our users' business problems; however, we've been slow to apply that same technology to our own problem—developing information systems. In the not-too-distant past, the principal tools of the systems analyst were paper, pencil, and flowchart template.

Today, an entire technology has been developed, marketed, and installed to assist systems developers. Chances are that your future employer is using or will be using this technology to develop systems. That technology is called computer-aided systems engineering.

Computer-aided systems engineering (CASE) is the application of information technology to systems development activities, techniques, and methodologies. *CASE tools* are programs (software) that automate or support one or more phases of a systems development life cycle. The technology is intended to accelerate the process of developing systems and to improve the quality of the resulting systems.

Some people refer to this as computer-aided *software* engineering, but software is only one component of information systems. (Recall your information system building blocks.) Thus, we prefer the broader context of the term *systems*.

CASE is not a methodology (although some vendors sell it as such). Nor is CASE an alternative to methodologies. Instead, it is an enabling technology that supports a methodology's preferred strategies, techniques, and deliverables.

The term *systems engineering* is based on a vision for CASE technology—that systems development can and should be performed with engineering-like precision and rigor. In its broadest context, the use of CASE technology automates your entire systems development methodology. *FAST* recommends CASE tools to implement all its phases and activities. Let's take a look at the CASE concept and survey some representative product categories and tools.

The History and Evolution of CASE Technology

The concept of using computers to automate systems development is not new. In a sense, common language compilers and interpreters can be thought of as CASE tools. Let's briefly examine the history and evolution of this technology, a technology that will almost certainly affect your future.

The true history of CASE dates to the early- to mid-1970s. The ISDOS project, under the direction of Dr. Daniel Teichrowe at the University of Michigan, developed a language called *Problem Statement Language (PSL)* for describing user problems and solution requirements for an information system into a computerized dictionary. A companion product called *Problem Statement Analyzer (PSA)*

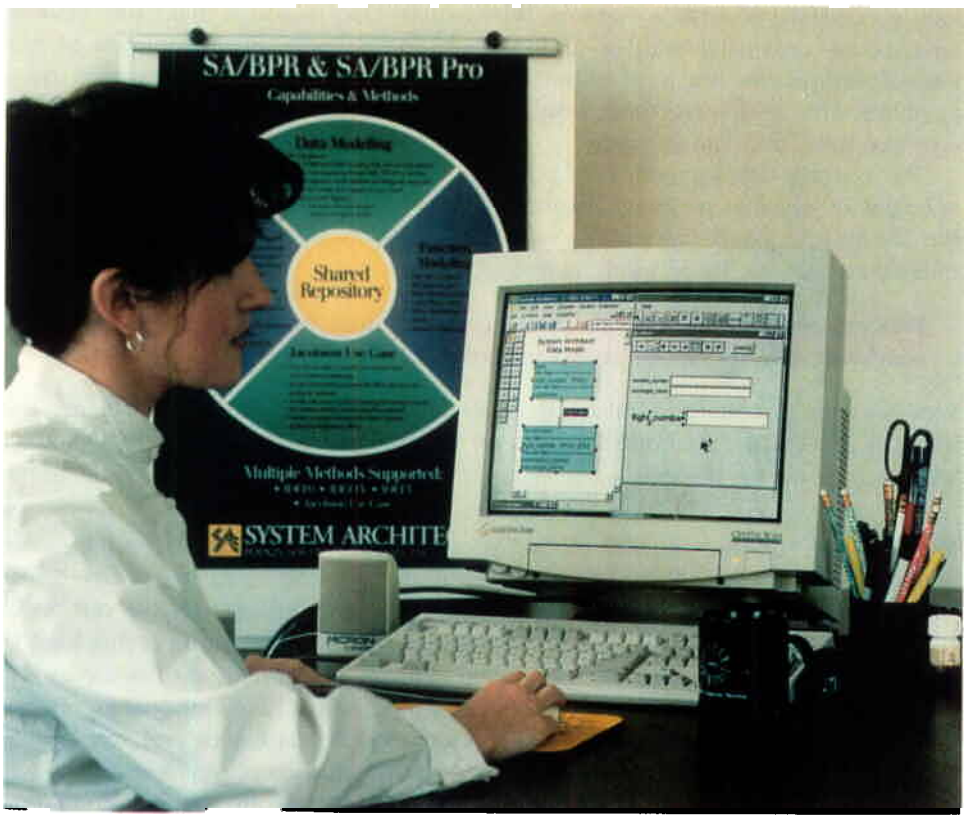


FIGURE 3.11
A Typical CASE Workbench

was created to analyze those problems and requirements for completeness and consistency. *PSL/PSA* ran on large mainframe computers and consumed precious and expensive machine resources. Few companies could afford to dedicate computer resources to *PSL/PSA*.

The real breakthrough came with the advent of the personal computer. Not long after, in 1984, an upstart company called Index Technology (now known as Intersolv) created a PC software tool called *Excelerator*. Its success established the CASE acronym and industry. Today, hundreds of CASE products are available to various systems developers.

Most CASE products run on personal computers or UNIX workstations. In some environments, these workstations are networked to shared CASE tools and a common repository. A typical CASE workbench is shown in Figure 3.11.

CASE technology bears a remarkable similarity to another engineering technology: computer-aided design/computer-aided manufacturing (CAD/CAM). Modern engineers use CAD tools to design and analyze new products. CAM tools then automatically generate the computer programs that will run the shop floor machinery needed to produce the design. CASE seeks to do for information system developers what CAD/CAM does for engineers: help them design better products (systems) and automatically generate the computer programs.

A CASE Tool Framework

Let's base our CASE tool framework on a concept that is already familiar to you, the systems development life cycle. In other words, we will classify tools according to which phases of the life cycle they support. Our CASE framework is based on the following popular terminology:

> The term **upper-CASE** describes tools that automate or support the upper or earliest phases of systems development—the survey, study, definition, and design phases.

The term **lower-CASE** describes tools that automate or support the lower or later phases of systems development—detailed design, construction, and delivery (and also support).

There is some overlap between upper- and lower-CASE tools. This is because our profession has never reached agreement on when systems analysis ends and when systems design begins.

A typical business's complete CASE tool kit should include one or more products from each category. Although some CASE vendors offer an integrated CASE product family that rather comprehensively covers all categories, it is highly unlikely that any firm will find a single source for every CASE tool they need, or might want to use.

What about SoundStage? The *FAST* methodology recommends that CASE tools be used, but does not specify which CASE tools should be used. SoundStage has used various CASE tools, but it is currently using Popkin Software's *System Architect* as its upper-CASE tool, and Powersoft's *Powerbuilder* as its lower-CASE tool. As described in the chapter-opening minicase, they have decided to pilot (meaning "try") Borland's *Delphi* rapid application development environment as an object-oriented alternative to *Powerbuilder* for the new Member Services system.

CASE Tool Architecture

How is a CASE tool structured? At the center of any true CASE tool's architecture is a database called a repository (or a link into such a repository). Around that repository is a collection of tools or facilities to create documentation or other system components. Let's briefly examine this architecture.

Repositories The repository concept was introduced earlier as a data store for all the knowledge, specifications, and documentation by-products created during a systems development project.

Our experience suggests that most people are first attracted to CASE tools by their graphics capabilities (see Figure 3.12). "Wow! Here is a neat way to produce high-quality entity relation diagrams or system flowcharts!" But the real power of

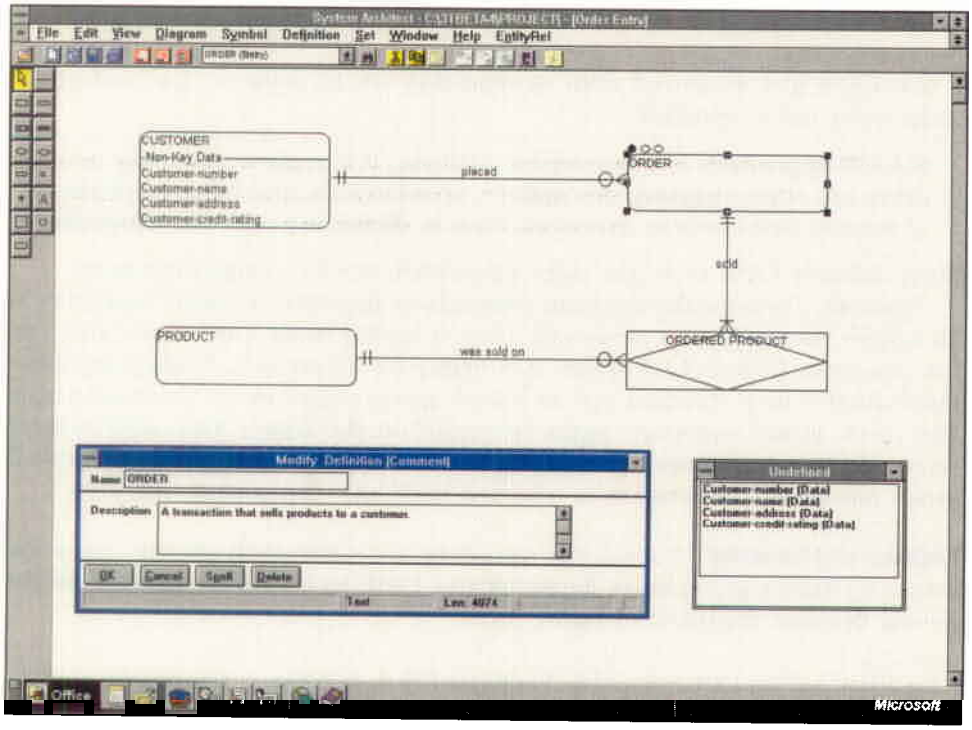


FIGURE 3.12
A CASE Tool Screen (Popkin System Architect*)*

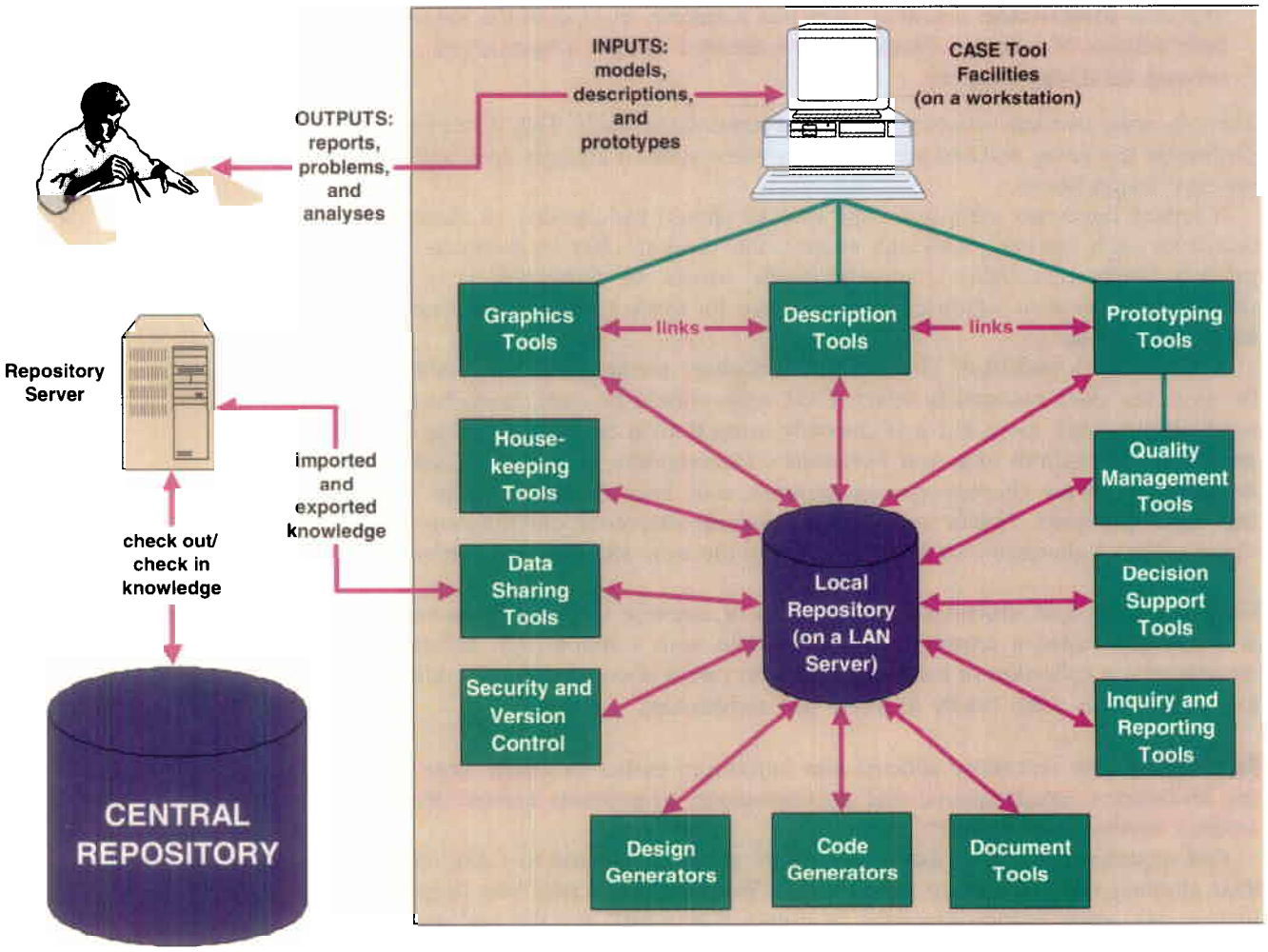


FIGURE 3.13 *CASE Architecture*

a true CASE tool is derived from its repository (or its ability to use and update some other tool's repository).

> A **CASE repository** is a developers' database. It is a place where the developers can store diagrams, descriptions, specifications, and other by-products of systems development. Synonyms include **dictionary** and **encyclopedia.**[5]

Many different CASE tools can share information across a single repository.

Typically, a mature development organization maintains a central repository of all system documentation developed since it started using CASE tools. This central repository (Figure 3.13) serves as a library for all projects. Existing repository information is then "checked out" to a work group repository for the project team. This work group repository might be stored on the team's local area network server. After various project milestones, or after the project is completed, the work group repository's information is "checked back into" the central repository.

Facilities and Functions To use the repository, the CASE tools provide input and output facilities and functions. Representative CASE tools provide some of the following facilities, illustrated in Figure 3.13:

[5] The *System Architect* CASE tool used by SoundStage calls its repository an encyclopedia.

- *Diagramming tools* are used to draw the system models required or recommended in most methodologies. Usually, the objects on one model can be linked to other graphical models and to descriptive details.
- *Description tools* are used to record, delete, edit, and output nongraphical documentation and specifications. As shown in Figure 3.13, the descriptions can be created and accessed directly by the description tools or directly from the objects on the aforementioned diagrams.
- *Prototyping tools* are used to construct system components including inputs, outputs, and programs. Today, most of these prototypes can evolve into the final, working system. As shown in the figure, the prototypes can directly use and maintain the models and descriptions in the repository.
- *Inquiry and reporting tools* are used to extract models, descriptions, and specifications from the repository.
- *Quality management tools* analyze models, descriptions, and prototypes for consistency, completeness, or conformance to accepted rules of the methodologies the CASE tools support. Some CASE tools can identify and report quality errors as they occur.
- *Decision support tools* provide information for various decisions that occur during systems development. For example, some CASE tools help systems analysts estimate and analyze feasibility.
- *Documentation organization tools* are used to assemble, organize, and report repository information that can be reviewed by system owners, users, designers, and builders. Based on customizable templates, they implement the deliverables of the chosen methodology.
- *Design generation tools* automatically generate first-draft designs for various system components based on the business requirements recorded in the repository and technology standards provided by the system designer. For example, many CASE tools can generate a database given from a business data model and the chosen database technology.
- *Code generator tools* automatically generate application programs or significant portions of those programs.
- *Testing tools* help the system designers and builders test databases and application programs.
- *Data sharing tools* provide for import and export of repository information to and from other software tools that cannot directly access the repository. (Increasingly, CASE tool vendors are providing direct access to their repositories, reducing the need for import and export capabilities.)
- *Version control tools* maintain the integrity of the repository by preventing unauthorized or inadvertent changes and saving prior versions of various information stored in the repository.
- *Housekeeping tools* establish user accounts, privileges, repository subsets, tool defaults, backup and recovery, and other essential facilities.

The Benefits of CASE

Most methodologies do not require CASE. But most methodologies do benefit from and recommend CASE technology be used. Some of the most commonly cited benefits include:

- Improved productivity (through automation of tasks and rapid application development).
- Improved quality (because CASE tools check for completeness, consistency, and contradictions).
- Better documentation (mostly because the tools make it easier to create and assemble consistent, high-quality documentation).

- Reduced lifetime maintenance (because of the aforementioned system quality improvements combined with better documentation).
- Methodologies that really work (through rule enforcement and built-in expertise).

These benefits do not come free. CASE can be very expensive, but some relatively inexpensive CASE tools now provide much of the functionality of CASE tools that used to cost $10,000 per workstation. Popkin's *System Architect* is one example of such a cost-effective CASE tool. But the real costs include more than the CASE software. Organizations must provide suitably configured workstations, training, and a support infrastructure to fully exploit CASE technology. Many information system organizations provide training and support through a development center.

A **development center** is a central group of information system professionals who plan, implement, and support a systems development environment for other developers. They provide training and support for both the methodology and CASE tools.

Thus, the development center might be thought of as consultants to the systems analysts, system designers, and system builders in the organization.

WHERE DO YOU GO FROM HERE?

This chapter, along with the first two, completes your context for systems analysis and design. We have described that context in terms of the three Ps—participants (with special emphasis on the systems analyst), product (the information system), and process (the system development methodology). Armed with this understanding, you are now ready to study systems analysis and/or design methods.

Depending on whether your course is an only course on the subject, a systems analysis course, or a systems design course, you can take different routes through this book. For a first systems development course or a systems analysis course, we recommend you continue your sequential path into the next unit, "Systems Analysis Methods." In Chapter 4, "Systems Analysis," you will look deeper into the survey, study, and definition phases of the *FAST* methodology. You will learn about alternative routes through those activities. Then, in Chapters 5 to 8, you will learn to apply specific tools and techniques to those systems analysis activities.

If you are in a system design course, you might want to quickly review Chapter 4 and then begin your detailed study in Chapter 9, "System Design." In Chapter 9, you will learn more about the configuration and design phases of the *FAST* methodology. Then in Chapters 10 to 16, you will learn to apply specific tools and techniques to those system design activities.

Regardless of the focus of your course, you and your instructor have a couple of additional options. You could begin by taking a detour to Part Five, "Cross Life Cycle Activities Skills." You already have the prerequisite background to study both modules in Part Five. Module A, "Project and Process Management," teaches you tools and techniques for managing a *FAST* project. Module D, "Joint Application Development," teaches you how to manage the methodology itself (and the CASE technology too).

SUMMARY

1. A **system development life cycle** is a logical process by which developers build information systems and computer applications. A **methodology** is the physical implementation of that logical life cycle including activities, roles, deliverables, and tools and techniques.
2. The following principles should underlie all system development methodologies:
   a. Get the owners and users involved.
   b. Use a problem-solving approach. The life cycle is such an approach.
   c. Establish phases and activities.
   d. Establish standards for consistent development and documentation.
   e. Justify systems as capital investments.
   f. Don't be afraid to cancel the project or revise scope.
   g. Divide and conquer.
   h. Design systems for growth and change.
3. There are two types of projects, unplanned and planned. Both types are responses to some combination of problems, opportunities, and directives.
   a. **Problems** are undesirable situations that prevent a business from achieving its purpose, goals, or objectives.
   b. **Opportunities** are chances to improve an organization even in the absence of problems.
   c. **Directives** are new requirements imposed by management or external influences.
4. Wetherbe's PIECES framework is useful for organizing problems, opportunities, and directives. The letters of PIECES correspond to *performance, information, economics, control, efficiency,* and *service.*
5. The book will teach systems analysis and design methods in the context of a typical, but simple methodology called *FAST. FAST* consists of eight phases:
   a. Survey and plan the project.
   b. Study and analyze the existing system.
   c. Define and prioritize the business requirements.
   d. Configure a feasible system solution.
   e. Procure any new hardware and software.
   f. Design and integrate the target system.
   g. Construct and test the target system.
   h. Deliver the production system.
6. During systems development, by-products are stored in the following data stores:
   a. A **repository** is a place where documentation about the system is stored.
   b. The **database** is where actual business data will be stored.
   c. The **program library** is where application software will be stored.
7. **Systems support** is the ongoing maintenance of a system after it has been placed into production.
8. **Cross life cycle activities** are activities that overlap many or all of the phases of system development. They include fact finding, documentation and presentation, estimation and measurement, feasibility analysis, project management, and process management.
9. **Computer-aided systems engineering (CASE)** is the application of information technology to systems development activities, techniques, and methodologies.
   a. **Upper-CASE** tools support the survey, study, definition, and design phases of the methodology.
   b. **Lower-CASE** tools support the design, construction, and delivery phases of the methodology.
   c. CASE tools are built around an automated **repository** (see above).
   d. A **development center** is a central group of information system professionals who plan, implement, and support a systems development environment for other developers.

KEY TERMS

application architecture, p. 92
application development life cycle, p. 72
backlog, p. 78
business process redesign, p. 78
CASE repository, p. 106
computer-aided systems engineering (CASE), p. 103
configuration phase, p. 91
construction phase, p. 96
cost-effectiveness, p. 76
cross life cycle activity, p. 99
database, p. 82
definition phase, p. 89
delivery phase, p. 98
design phase, p. 94
development center, p. 108
dictionary, p. 106
directive, p. 79
documentation, p. 100
encyclopedia, p. 106
entropy, p. 77
estimation, p. 102
fact-finding, p. 99
feasibility, p. 102
feasibility analysis, p. 102
feasibility, types of, p. 92
information strategy plan, p. 78
legacy system, p. 77
lower-CASE, p. 105
measurement, p. 102
methodology, p. 73
modeling, p. 89
opportunity, p. 79
PIECES, p. 79
planned system initiative, p. 78
presentation, p. 100
problem, p. 79
process management, p. 103
procurement phase, p. 93
production system, p. 82
program library, p. 82
project management, p. 102
prototyping, p. 90
rapid application development (RAD), p. 95

repository, p. 82
software and systems metrics, p. 102
steering committee, p. 78
study phase, p. 87
survey phase, p. 84
system development life cycle (SDLC), p. 72
systems support, p. 99
system test, p. 97
time boxing, p. 90
unit test, p. 97
unplanned system request, p. 78
upper-CASE, p. 104

REVIEW QUESTIONS

1. What is the difference between the *system development life cycle* and a *methodology*?
2. Why do companies use methodologies? Why do many choose to purchase a methodology?
3. What are the eight fundamental principles of systems development? Explain what you would do to incorporate those principles into a systems development process.
4. What is cost-effectiveness and how does its measurement impact system development?
5. What is the creeping commitment approach?
6. What is entropy and how does it impact system development?
7. What are the two triggers for a system development project?
8. Briefly describe the two types of planned system initiatives.
9. Differentiate among problems, opportunities, and directives.
10. Name the six problem classes of the PIECES framework.
11. What is a repository? How is it different from a database? How is it different from a program library?
12. Identify and briefly describe the eight high-level phases that are common to most modern system development life cycles.
13. Describe each phase of the *FAST* life cycle in terms of purpose, participants, inputs, outputs, and activities.
14. Differentiate between *modeling* and *prototyping* as requirements definition techniques.
15. What is time boxing? Why is it now popular?
16. Describe four types of feasibility.
17. What is an application architecture? How does it impact the targeting phase?
18. What's the difference between unit testing and system testing?
19. Name five cross life cycle activities.
20. What is CASE? Why is CASE not a methodology or an alternative to a methodology?
21. Differentiate between upper-CASE and lower-CASE.
22. What is the role of a repository in CASE?
23. List several common facilities that might be implemented by a CASE tool using a repository.
24. What is a development center?

PROBLEMS AND EXERCISES

1. Using the PIECES framework, evaluate your local course registration system. Do you see any problems or opportunities? (Alternative: Substitute any system with which you are familiar.)
2. Assume you are given a programming assignment that requires you to make some modifications to a computer program. Explain the problem-solving approach you would use. How is this approach similar to the phased approach of the methodology presented in this chapter?
3. Which phases of the *FAST* methodology presented in this chapter do the following tasks characterize?
   a. The analyst demonstrates a prototype of a new sequence of work order terminal screens.
   b. The analyst observes the order-entry clerks to determine how a work order is currently processed.
   c. The analyst develops the internal structure for a database to support work order processing.
   d. An analyst is teaching the plant supervisor how to inquire about work orders using the new microcomputer.
   e. A plant supervisor is describing the content of a new work order progress report that would simplify tracking.
   f. The analyst is reading an inquiry concerning whether or not a computer system might solve the current problems in work order tracking.
   g. The analyst is installing the microcomputer and database management system needed to run work order processing programs.
   h. The analyst is reviewing the company's organizational chart to identify who becomes involved in work order processing and fulfillment.
   i. The analyst is comparing the pros and cons of a software package versus writing the programs for a new work order system.
   j. An analyst is testing a computer program for entering work orders into the system.
   k. The analyst is correcting a program to more accurately summarize weekly progress.
   l. The analyst is buying a microcomputer and some available software.

m. Information systems management and top business executives are identifying and prioritizing business area applications that should be developed.

4. Management has approached you to develop a new system. The project will last seven months. Management wants a budget next week. You will not be allowed to deviate from that budget. Explain why you shouldn't overcommit to early estimates. Defend the creeping commitment approach as it applies to cost estimating. What would you do if management insisted on the up-front estimate with no adjustments?
5. You have a user who has a history of impatience—encouraging shortcuts through the systems development life cycle and then blaming the analyst for systems that fail to fulfill expectations. By now, you should understand the phased approach to systems development. For each activity, compile a list of possible consequences to use when the user suggests a shortcut through or around that activity.
6. Our company does not have a methodology, but realizes it needs one. The president has suggested that we start a project to build a methodology. Explain to her the advantages of purchasing a methodology instead.
7. You are in a meeting with two other systems analysts. One is arguing that the business should adopt a model-driven development strategy. The other is arguing that a prototyping strategy should be used instead. Explain to them how the two approaches might complement one another.
8. In a staff meeting, a middle-level manager says, "I suggest we dump our methodology and replace it with CASE technology." Respond to this manager.
9. Some people believe that CASE code generators will eventually replace programmers. What do you think? Why or why not? Now suppose such an evolution does occur. How will this impact programmers? How about systems analysts?
10. When describing the phased approach that you plan to follow in developing a new information system, your client asks why your approach is missing a feasibility analysis and project management phase. How would you respond?

PROJECTS AND RESEARCH

1. Make an appointment to visit a systems analyst at a local information system installation. Discuss the analyst's current project. What problems, opportunities, and directives triggered the project? How do they relate to the PIECES framework?
2. Visit a local information system installation. Compare the methodology with the one in this chapter. Evaluate the company's methodology with respect to the eight systems development principles. (Alternative: Substitute the system development life cycle or methodology used in another systems analysis and design book, possibly assigned by your instructor.)
3. Read the mini-book *The One Minute Methodology* (see Suggested Readings). Write a paper that describes what the author was trying to teach, and relate it to the subject presented in this chapter.
4. Visit a local information system installation. Pick a CASE tool that they are using and describe its capabilities using the CASE tool architecture described in this chapter. Did you discover any new capabilities?
5. Research the upper-CASE or lower-CASE marketplace. (Be careful. The industry's terminology is constantly changing.) What are some products and their vendors? On which operating systems do they run? How are they similar and different? What do they cost?
6. Research some of the current trade literature about CASE experiences. What do the writers like about their CASE tools? What are their major complaints? How would you rate their overall satisfaction? Why? What are some future directions of the CASE tool industry?

MINICASES

1. Jeannine Strothers, investments manager, has submitted numerous requests for a new investment tracking system. She needs to make quick decisions regarding possible investments and divestments. One hour can cost her thousands of dollars in profits for her company.

   She has finally given up on Information Systems for not giving her requests high enough priority to get service. Therefore, she goes to a computer store and buys a microcomputer along with spreadsheet, database, and word processing software. The computer store salesperson suggests she build a database of her investments and options, subscribe to a computer investment databank (accessed via a modem in the microcomputer), feed data from her database and the bulletin board into the spreadsheet, play "what if" investment games on the spreadsheet, and then update the database to reflect her final

decisions. The word processor will draw data from the database for form letters and mailing lists.

After discussing her plans with Jeff, a systems analyst at another company, he suggests she take a systems analysis and design course before beginning to use the spreadsheet and database. The local computer store, on the other hand, says she doesn't need any systems analysis and design training to be able to develop systems using the spreadsheet and database programs. Their reasoning is that spreadsheets and database tools are not programming languages; therefore, she shouldn't need analysis and design to build systems with them. Is the computer store correct? Why or why not? Can you convince Jeannine to take the systems analysis and design course? What would your arguments be?

2. Jeannine Strothers, the impatient manager in the earlier minicase, did not take Jeff's advice. She built the new system, but she can't get top management to allow her to use it. And she's run into a number of other problems.

First, the financial comptroller has been reevaluating company investment strategies and policies. Jeannine wasn't aware of that. The new system does not account for many of the policies being considered.

Her own staff has rejected the investment and divestment orders generated by the system. She used Information Systems' existing file structure to design those orders, only to find out that her clerks had abandoned those files two years ago because they didn't include the data necessary to execute order transactions. Her staff is also critical of the design, saying that minor mistakes send them off into the "twilight zone" with no easy way to recover.

The computer link to the investment databank has been useless. The data received and its format are not compatible with systems requirements. Although other databanks are available, the current databank has been prepaid for two years. Additionally, Jeannine is now skeptical of such services.

Some of her subordinate managers are insisting on graphic reports. Unfortunately, neither her database management nor spreadsheet package supports graphics. She's not sure how to convert the data of either package to a graphic format (assuming it is possible).

To top off her problems, she isn't sure that her existing database structure can be modified to meet new requirements without having to rewrite all the programs, even those that appear to be working. And her boss is not sure he wants to invest the money in a consultant to fix the problems.

Jeannine's analyst friend Jeff is not very sympathetic to her problems: "Jeannine, I don't have any quick answers for you. You've taken too many shortcuts through the project life cycle. When we do a system, we go through a carefully thought-out procedure. We thoroughly study the problem, define needs, evaluate options, design the system and its interfaces, and only then do we begin programming."

Jeannine replies, "Wait a minute. I only bought a microcomputer. It's not the same as your mainframe computer. I didn't see the need for going through the ritual you guys use for mainframe applications. Besides, I didn't have the time to do all those steps."

Jeff's parting words are philosophical: "You didn't have the time to do it right. Where will you find the time to do it over?"

What principles did Jeannine violate? Why do so many people today fall prey to the belief that the life cycle for an application is somehow different when using microcomputers? What conclusions can you draw from Chapter 2 (the information systems chapter) that might help Jeannine learn from her mistakes? What would you recommend to Jeannine if she were to decide to approach her manager with a plan to salvage the system?

3. Evaluate the following scenarios using the PIECES framework. Do not be concerned that you are unfamiliar with the application. That situation isn't unusual for a systems analyst. Use the PIECES framework to brainstorm potential problems or opportunities you would ask the user about.

a. The staff benefits and payroll counselor is having some problems. Her job is to counsel employees on their benefit options. The company has just negotiated a new medical insurance package that requires employees to choose from among several health maintenance organizations (HMOs). The HMOs vary according to employee classifications, contributions, deductibles, beneficiaries, services covered, and service providers permitted. The intent was to provide the most flexible benefits possible for employees, to minimize costs to the company, and to control costs to the insurance agency (which would affect subsequent premiums charged back to the company).

The counselor will be called on to help employees select the best plan for themselves. She currently responds manually to such requests. But the current options are more straightforward than those under the new plan. She can explain the options, what they do and do not cover, what they cost and may cost, and the pros and cons. However, current employee distrust of the new plan suggests she will need to provide more specific suggestions and responses to employees.

She may have to work up scenarios—possibly worst-case scenarios—for many employees. The scenarios will have to be personalized for each employee's income, marital and family status, current health risks, and so on. In working up a few sample scenarios, she discovered first that it takes one full day to get salary and personnel data from the Information Systems department. Second, employee data are stored in many files that are not always properly updated. When conflicting data become apparent, she can't continue her projections until that conflict has been resolved. Third, the computations are complex. It often takes one full day or more to create investment and/or retirement scenarios for a single employee. Fourth, there are some concerns that projections are being provided to unauthorized individuals, such as former spouses or nonimmediate relatives. Finally, the complexity of the variations in the calculations (there are a lot of "If this, do that" calculations) results in frequent errors, many of which probably go undetected.

b. The manager of a tool and die shop needs help with job processing and control. Jobs are currently

processed by hand. First, a job number is established. Next, the job supervisor estimates time and materials for the job. This is a time-consuming process, and delays are common. Then, the job is scheduled for a specific day and estimated time.

On the day the job is to be worked on, materials orders have to be issued. If materials aren't available, the order has to be rescheduled.

Time cards are completed in the shop when workers fulfill the work order. These time cards are used to charge back time to the customer. Time cards are processed by hand, and the final calculations are entered on the work order. The work orders are checked for accuracy and sent to CIS, where accounting records are updated and the customer is billed.

The problem is that the customer frequently calls to inquire about costs already incurred on a work order, but it's not possible to respond because CIS sends a report of all work orders only once a month. Also, management has no idea of how good initial estimates are or how much work is being done on any tool or machine, or by any worker.

c. State University's Development Office raises funds for improving instructional facilities and laboratories at the university. It has uncovered a sensitive problem: The data are out of control.

The Development Office keeps considerable redundant data on past gifts and givers, as well as prospective benefactors. This results in multiple contacts for the same donor—and people don't like to be asked to give to a single university over and over!

To further complicate matters, the faculty and administrators in most departments conduct their own fund-raising and development campaigns, again resulting in duplication of contact lists.

Contacts with possible benefactors are not well coordinated. Whereas some prospective givers are contacted too often, others are overlooked. It is currently impossible to generate lists of prospective givers based on specific criteria (e.g., prior history, socioeconomic level), despite the fact that data on hundreds of criteria have been collected and stored. Gift histories are nonexistent, which makes it impossible to establish contribution patterns that would help various fund-raising campaigns.

4. Century Tool and Die, Inc., is a major manufacturer of industrial tools and machines. It is located in Newark, New Jersey. Larry is the assistant A/R manager. Valerie is a systems analyst for the Information Systems department. Valerie and Larry have just sat down to discuss their current project—improving the recently implemented Accounts Receivable (A/R) information system. As they start to discuss the project, they are interrupted by Robert Washington, the executive vice president of finance, and Gene Burnett, the A/R manager. Larry suddenly looks very nervous. And for good reason—he had suggested the new system, and it has not turned out as promised. Gene's support had been lukewarm, at best. Robert initiates the conversation.

"We've got big problems," said Robert. "This new A/R system is a disaster. It has cost the company more than $625,000, not to mention lost customer goodwill and pending legal costs. I can't afford this when the board of directors is complaining about declining return on investment. I want some answers. What happened?"

Gene responds, "I was never really in favor of this project. Why did we need this new computer system?"

Larry gets defensive. "Look, we were experiencing cash flow problems on our accounts. The existing system was too slow to identify delinquencies and incapable of efficiently following up on those accounts. I was told to solve the problem. A manual system would be inefficient and error-prone. Therefore, I suggested an improved computer-based system."

Gene replies, "I'm not against the computer. I approved the original computer-based system. And I realized that a new system might be needed. It's just that you and Valerie decided to redesign the system without considering alternatives—just like that! In my opinion, you should always analyze options. And let's suppose that a new computerized system was our best option. Why did we have to build the system from scratch? There are good A/R software packages available for purchase. I . . ."

Sensing a confrontation, Robert interrupts, "Gene has a point, Larry. Still, the system you proposed was defended as feasible. And yet it failed! Valerie, as lead analyst, you proposed the new system, correct?"

Nervously, Valerie responds, "Yes, with Larry's help."

Robert continues, "And you wrote this feasibility report early in the project. Let's see. You proposed replacing the current batch A/R system with an on-line system using a database management package."

Valerie replies, "Strictly speaking, the database management package wasn't needed. We could have used the existing VSAM files."

With a troubling look, Robert says, "The report says you needed it. I paid $15,000 to get you that package!"

Valerie answers, "Bill, our database administrator, made that recommendation. The A/R system was to be the pilot database project."

Robert continues. "You also proposed using a network of microcomputers as a front end to the mainframe computer?"

"Yes," answers Valerie. "Larry felt a mainframe-based system would take too long to design and implement. With microcomputers, we could just start writing the necessary transaction programs and then transfer the data to the database on the mainframe computer."

Robert looked puzzled. "I'm a former engineer. It seems to me that some sort of design work should have been done no matter what size computer you used . . . *[brief pause]* In any case, the bottom line in this report is that your projected benefits outweighed the lifetime costs. You projected a 22 percent annual return on investment. Where did you get that number?"

Valerie answers, "I met with Larry four times—about six hours total, I'd say—and Larry explained the problems, described the requirements, made suggestions, and then projected the costs, benefits, and rate of return."

Robert replies, "But that return hasn't been realized, has it? Why not?" After a long, silent pause, Robert continues, "Valerie, what happened after this proposal was approved?"

"We spent the next nine months building the system."

Robert responds, "And how much did you have to do with that, Larry?"

"Not a lot, sir. Valerie occasionally popped into my office to clarify requirements. She showed me sample reports, files, and screens. Obviously, she was making progress, and I had no reason to believe that the project was off schedule."

Robert continues his investigation. "Were you on schedule, Valerie?"

"I don't believe so, Mr. Washington. My team and I were having some problems with certain business aspects of the system. Larry was unfamiliar with those aspects and he had to go to the account clerks and the accountants for answers."

Visibly irritated, Robert asks, "I don't get it! Why didn't you go to the clerks and accountants?"

Although the question was directed to Valerie, Gene replies, "I can answer that. I designated Larry as Valerie's contact. I didn't want her team wasting my people's time—they have jobs to do!"

Robert is silent for a moment, but then he responds, "Something about that bothers me, Gene. In any case, when this project got seriously behind, Larry, why didn't you consider canceling it, or at least reassessing the feasibility?"

"We did!" answers Larry. "Gene expressed concern about progress about seven months into the project. We called a meeting with Valerie. At that meeting, we learned that the new database system wasn't working properly. We also found that we needed more memory and storage on the microcomputers. And to top it off, Valerie and her staff seemed to have little understanding of the business nature of our problems and needs."

This time, Valerie gets defensive. "As I already pointed out, I wasn't permitted contact with the users during the first seven months. Besides, we were asked by Larry to start programming as quickly as possible so that we would be able to show evidence of progress."

Looking at Larry, Robert asks, "Larry, I'm no computer professional; however, my gut instinct suggests that some design or prototyping should have been done first."

Valerie responds for Larry, "Yes, but that would have required end-user participation, which Larry and Gene would not permit."

Larry interrupts. "As I was saying, we considered canceling the project. But I pointed out that $150,000 had already been spent. It would be stupid to cancel a project at that point. I did reassess the feasibility, and concluded that the project could be completed in four more months for another $50,000."

Robert responds, "Nobody asked me if I wanted to spend that extra money."

Larry answers, "We realize that the system hasn't worked out as well as we had hoped. We are trying to redesign . . ."

Robert interrupts, "As well as you hoped? That's an understatement! Let me read you some excerpts from Gene's last monthly report. Customer accounts have mysteriously disappeared, deleted without explanation. Later, we discovered that data-entry clerks didn't know that the F2 key deletes a record. Also, customers have been legally credited for payments that were never made! Customers have been double billed in some cases! Reports generated by the system are late, inaccurate, and inadequate. Cash flow has been decreased by 35 percent! My sales manager claims that some customers are taking their business elsewhere. And the Legal department says we may be sued by two customers and that it will be impossible to collect on those accounts where customers received credit for nonpayments. You tell me, what would you do if you were in my shoes?"

a. Evaluate this project against the eight principles of systems development.
b. What would you do if you were in Mr. Washington's shoes? How would you react to Larry's performance? Valerie's? Gene's?
c. What did Valerie do wrong? Was she in control of her own destiny on this project? Why or why not?
d. What did Larry or Gene do wrong? Can either be held responsible for the failure of a computer project when they have limited computer literacy or experience?
e. What was wrong with the feasibility report? Did Valerie and Larry meet often enough? Was the input to that report sufficient? Did the team commit to a solution too early? Did programming begin too soon? Why or why not?
f. Why were Valerie and her staff uncomfortable with the business problem and needs?
g. Should the project have been canceled? What about the $150,000 investment that had already been made?
h. If you were Valerie or Larry, what would you have done differently?

5. Assume that you are a newly hired CASE coordinator for a major company. You are to make a presentation to management proposing that large sums of money be spent to buy a methodology and CASE tools for your company. Research the market and prepare a spreadsheet that details projected costs and benefits. State all your assumptions. Don't forget training and support.

SUGGESTED READINGS

Application Development Strategies (monthly periodical). Arlington, MA: Cutter Information Corporation. This is our favorite theme-oriented periodical that follows system development strategies, methodologies, CASE, and other relevant trends. Each issue focuses on a single theme.

Application Development Trends (monthly periodical). Westborough, MA: Software Productivity Group, Inc. This is our favorite periodical for keeping up with the latest trends in methodology and CASE. Each month features several articles on different topics and products.

Benjamin, R. I. *Control of the Information System Development Cycle*. New York: Wiley-Interscience, 1971. Benjamin's 16 axioms for managing the systems development process inspired our adapted principles to guide successful systems development.

Bouldin, Barbara. *Agents of Change: Managing the Introduction of Automated Tools*. Englewood Cliffs, NJ: Prentice Hall, 1989. This book documents experiences and makes recommendations for introducing CASE tools into an organization.

Gane, Chris. *Rapid Systems Development*. Englewood Cliffs, NJ: Prentice Hall, 1989. This book presents a nice overview of RAD that combines model-driven development and prototyping in the correct balance.

Gildersleeve, Thomas. *Successful Data Processing Systems Analysis*, 2nd ed. Englewood Cliffs, NJ: Prentice Hall, 1985. We are indebted to Gildersleeve for the creeping commitment approach.

Hammer, Mike. "Reengineering Work: Don't Automate, Obliterate." *Harvard Business Review*, July–August 1990, pp. 104–11. Mike Hammer is a noted consultant and author (and entertaining speaker!) who talks of a new development paradigm that suggests systems analysts take a more "active" (as opposed to "passive") role in reshaping underlying business processes instead of merely automating existing and dated business approaches. This article documents several classic examples where the business process was successfully redefined (during systems analysis), resulting in a more competitive business and information system.

Hobus, James J. *Application Development Center: Implementation and Management*. New York: Van Nostrand Reinhold, 1991. This work describes the implementation of the development center concept described at the end of our chapter.

Martin, James. *Information Engineering: Volumes 1–3*. Englewood Cliffs, NJ: Prentice Hall, 1989 (Vol. 1), 1990 (Vols. 2, 3). These works present information engineering, the dominant methodology after which our *FAST* methodology is patterned.

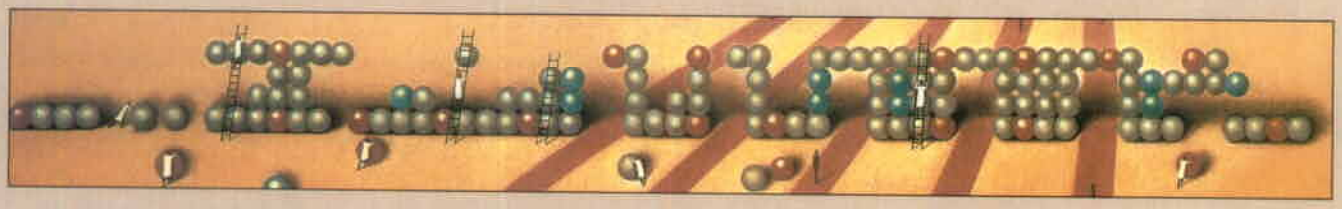
Orr, Ken. *The One Minute Methodology*. New York: Dorsett House Publishing, 1990. Must reading for those interested in exploring the need for methodology. This very short book can be read in one sitting. It follows the story of an analyst's quest for the development silver bullet, "the one minute methodology."

Wetherbe, James. *Systems Analysis and Design: Best Practices*, 4th ed. St. Paul, MN: West, 1994. We are indebted to Wetherbe for the PIECES framework.

Zachman, John A. "A Framework for Information System Architecture." *IBM Systems Journal* 26, no. 3 (1987). This article presents a popular conceptual framework for information systems survey and the development of an information architecture.

PART TWO

SYSTEMS ANALYSIS METHODS



The five chapters in Part Two introduce you to systems analysis activities and methods. Chapter 4, "Systems Analysis," provides the context for all the subsequent chapters by introducing the activities of *systems analysis*. Systems analysis is the most critical phase of a project. It is during systems analysis that we learn about the existing business system, come to understand its problems, define objectives for improvement, and define the detailed business requirements that must be fulfilled by *any* subsequent technical solution. Clearly, any subsequent design and implementation of a new system depends on the quality of the preceding systems analysis. Systems analysis is often shortchanged in a project because (1) many analysts are not skilled in the concepts and logical modeling techniques to be used, and (2) many analysts do not understand the significant impact of those shortcuts. Chapter 4 introduces you to systems analysis and its overall importance in a project. Subsequent chapters teach you specific systems analysis skills with an emphasis on logical system modeling.

In Chapter 5, "Data Modeling," we teach you *data modeling*, a technique for organizing and documenting the stored data requirements for a system. You will learn to draw entity relationship diagrams as a tool for structuring business data that will eventually be designed as a database. These models will capture the business associations and rules that must govern the data.

Chapter 6, "Process Modeling," introduces *process modeling*. It explains how data flow diagrams can be used to depict the essential business processes in a system, the flow of data through a system, and policies and procedures to be implemented by processes. If you've done any programming, you recognize the importance of understanding the business processes for which you are trying to write the programs.

Chapter 7, "Distribution Modeling," explains the concept of *distribution modeling*. Today's computer networking technology allows us to extend the reach of information systems to a larger geography. But to intelligently use that technology, we need a better understanding of the business geography to be supported. You'll learn how to draw location diagrams and map data and process requirements to those locations.

Finally, Chapter 8, "Object Modeling," introduces the latest systems analysis trend—*object modeling*. Object technology was introduced in prior chapters as an emerging technology to accelerate systems development. But new technologies require new techniques to properly apply the technologies. Eventually, object modeling will become the preferred modeling technique, replacing (or to be more accurate, *integrating*) separate data, process, and distribution modeling. Your early understanding of the concepts and techniques should prove to be a career advantage.

4

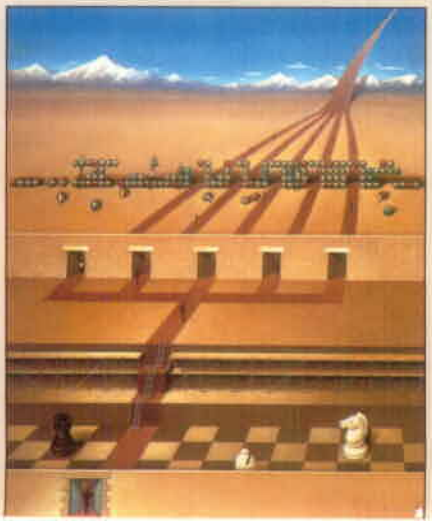
SYSTEMS ANALYSIS

CHAPTER PREVIEW AND OBJECTIVES

In this chapter you will learn more about the systems analysis phases in a systems development project: survey, study, and definition. These three phases are collectively referred to as systems analysis. You will know that you understand the process of systems analysis when you can:

- Define systems analysis and relate the term to the survey, study, and definition phases of the *FAST* methodology.
- Describe a number of systems analysis strategies for solving business system problems.
- Describe the survey, study, and definition phases in terms of your information system building blocks.
- Describe the survey, study, and definition phases in terms of objectives, roles, inputs, outputs, techniques, and steps.
- Identify those chapters and modules in this textbook that can help you perform the activities of systems analysis.

Although some techniques of systems analysis are introduced in this chapter, it is *not* the intent of this chapter to teach the *techniques*. This chapter teaches only the *process* of systems analysis. The techniques will be taught in the subsequent four chapters.



SCENE

When we last visited SoundStage, a survey of the existing information system had been completed. (This chapter will more closely examine the activities that were part of that survey.) This episode begins shortly after the project's executive sponsor (the person who pays for the system to be built) has approved the feasibility assessment and scope and authorized a more detailed study of the current system. We join Sandra, Bob, and Sarah Hartman (the business analyst assigned to this project) as they plot strategy in Sandra's office.

SANDRA

Good morning! I asked you here so Bob and I could review our strategy for the next phase of the project. This won't take long.

SARAH

As you know, I was just appointed to this business analyst position, and I'm not that familiar with your methods and terminology yet. I'm kind of looking forward to working for the next year or so in Information System Services. I have to admit, I avoided most of the computer courses when I was in school. To be honest, I thought you were all nerds—no offense!

BOB

None taken.

SARAH

Anyway, Bob's e-mail message said we are starting the study phase?

SANDRA

That's right. This phase will be accomplished in five steps. First, we will draw some simple models of the current system. We want to get a basic grasp of how the current system works.

SANDRA

Models?

BOB

Actually pictures. We'll draw four of them. One will show how this system connects to the business and other systems. One will show the things about which the current system collects data. A third picture will show all of the operating locations that are affected by the current system. And the fourth will show the basic functions performed by the system.

SARAH

Sounds like a lot of trouble.

BOB

Not really. Our *FAST* methodology encourages rapid model development. There will be no multipage models. Four models—four pages! We'll do them in one morning or afternoon.

SARAH

Really? It seems it should take a lot longer than that just to collect the facts.

SANDRA

Ah, but that's where you come in, Sarah. As the business analyst, you are going to plan, organize, and schedule a facilitated group meeting to collect the facts as a team and construct the models.

SARAH

I'm not sure I can lead such a meeting. I mean, I don't have the skill or experience.

BOB

Relax! Sandra will serve as facilitator. She's gone through special training to lead such discussions. I'll be drawing the models on the white board, and you'll be recording any miscellaneous facts . . . as well as encouraging all the managers and users to actively participate.

SARAH

Cool! I can do that. What happened to interviews?

SANDRA

We use them when necessary, but these facilitated sessions are far more productive.

[pause]

The second step will be to analyze all problems and directives.

BOB

Can I assume that *FAST* has a preferred technique for that? I'm new here too

[nodding to Sarah].

SANDRA

Of course! *FAST* recommends that we use something called the PIECES framework to brainstorm our problems and opportunities. I'll get you both some information on that framework later. Then we'll analyze each problem and opportunity for causes and/or effects. Sometimes a cause might actually be another problem, but if we keep applying the problem analyses, we'll get to the root causes and effects.

SARAH

Don't tell me, another facilitated meeting?

SANDRA

Actually, a continuation of the first meeting; that is, if you can talk our participants into giving up a whole day. The productivity and informational benefits are enormous.

SARAH

I think we can do it, if I get the right managers to buy in.

SANDRA

The third step occurs in that same meeting. For each problem, we'll establish some business-oriented objectives to improve the system.

SARAH

Is that like a *requirements statement*?

BOB

No, the requirements statement will be developed in the next project phase. Think of objectives as business criteria for measuring how good the new system will have to be.

SARAH

Got it. And the fourth step?

SANDRA

We're starting to wind the phase down now. The three of us will reassess our project scope, to see if it has changed. Then we'll adjust our project plan. The fifth and final step will be to present our findings to management and seek approval to continue to the next phase.

SARAH

Which is . . . ?

SOUNDSTAGE

SOUNDSTAGE ENTERTAINMENT CLUB

BOB
Requirements definition. We're going to identify and document the business requirements for the new system, a system that will achieve the aforementioned system improvement objectives. We call it the definition phase.

SARAH
I'm impressed. So far, all you've talked about are business issues—business this and business that. I know that we'll eventually get technical, but it is refreshing to hear your business focus. Maybe you guys aren't all nerds after all.

SANDRA
Thank you. Bob, why don't you tell Sarah how we're going to document the *business* requirements.

BOB
My pleasure. We will identify and document four categories of business requirements—data, process, interfaces, and geography. In each case, we'll draw models.

SARAH
Pictures?

BOB
You got it! These pictures will be considerably more detailed than those we previously discussed. They will not only show more detail, but will also be described by even greater detail in a dictionary-like database called a repository.

SARAH
And that's when we start getting technical?

BOB
No! We'll still leave our taped eyeglasses and pocket protectors in Information Services. These models and their supporting details will still focus entirely on the business requirements—those requirements that must be fulfilled by *any* technical solution we might come up with later. It is this requirements statement that will bridge the gap to technical design.

SARAH
I'm really going to enjoy this a lot more than I thought!

SANDRA
Well, at the risk of destroying our non-nerd image, we are going to get slightly technical during one aspect of requirements definition.

BOB
We are?

SANDRA
Right! *FAST* does recommend that the user interface requirements be prototyped.

BOB
Oh yes! I forgot. The prototypes of inputs and outputs can help us expand and refine our understanding of the data and process requirements.

SARAH
User interface?

BOB
Forgive our techno-terminology. The user interface consists of the inputs, outputs, and screens.

SARAH
Well that's the only part of the technical solution that I'm opinionated about, so you haven't scared me off yet. So, the definition phase will deliver system models and prototypes of inputs and outputs. Don't tell me, more facilitated sessions to capture and document these requirements, right?

DISCUSSION QUESTIONS

1. How would you characterize the focus of the early stages of this project? Why are technical concerns being avoided?
2. What is the advantage of asking Sarah to organize the facilitated meetings instead of Sandra or Bob? How much time do you think the facilitated group meeting might take as compared to a series of one-on-one interviews? Do you see advantages other than speed? Do you see any disadvantages?
3. Prepare an agenda for the full-day facilitated meeting suggested by Sandra and Bob.
4. What kind of pictures would you draw to illustrate the system aspects that Bob described?
5. What kinds of questions would facilitate discussion of problems and their underlying causes and effects?
6. Looking ahead to the requirements definition phase, do you think facilitated group meetings accelerate that process? Why or why not?

WHAT IS SYSTEMS ANALYSIS?

Let's begin with the formal definition of systems analysis.

Systems analysis is the dissection of a system into its component pieces to study how those component pieces interact and work.

We do a systems analysis to subsequently perform a systems synthesis.

Systems synthesis is the re-assembly of a system's component pieces back into a whole system—it is hoped an improved system.

Through systems analysis and synthesis, we may add, delete, and modify system components toward our goal of improving the overall system.

Moving from the theoretical definition to something a bit more contemporary, *systems analysis* is a term that collectively describes the early phases of systems

development. There has never been a universally accepted definition. And there has never been agreement on when analysis ends and design begins. To further confuse the issue, some methodologies refer to systems analysis as logical design. Typically, each organization's methodology of choice determines the defintion for that organization. In the *FAST* methodology, systems analysis is defined as those phases and activities that focus on the business problem, independent of technology (for the most part). Specifically, we refine our definition of systems analysis as follows.

> **Systems analysis** is (1) the survey and planning of the system and project, (2) the study and analysis of the existing business and information system, and (3) the definition of business requirements and priorities for a new or improved system. A popular synonym is **logical design.**

This definition corresponds to the first three phases of *FAST* (which were introduced in Chapter 3). The phase "configure a feasible solution" would be considered part of systems analysis by some experts. We prefer to think of it as an analysis-to-design transition phase.

Systems analysis is driven by business concerns, specifically, those of system users. Hence, it addresses the DATA, PROCESS, INTERFACE, and GEOGRAPHY building blocks from a system user perspective. Emphasis is placed on business issues, not technical or implementation concerns.

Figure 4.1 is a view of a *FAST* phase diagram that illustrates the three analysis phases only. Notice the repository.

> A **repository** is a collection of those places where we keep all documentation associated with the application and project.

Although we show only one project repository in the figure, the repository is normally implemented as some combination of the following:

- A disk or directory of word processing, spreadsheet, and other computer-generated files that contain project correspondence, reports, and data.

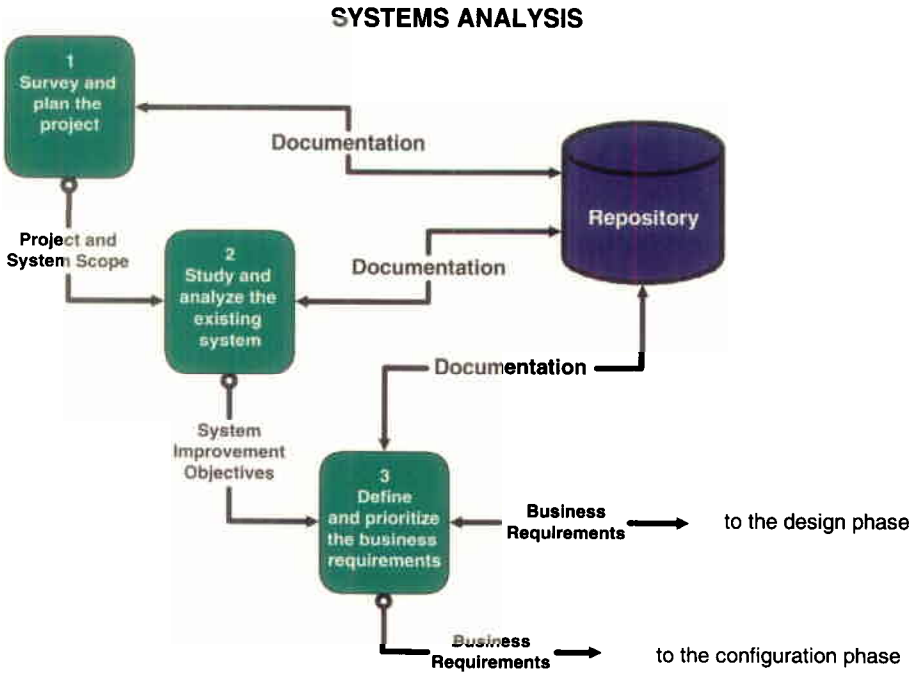


FIGURE 4.1
The Systems Analysis Phases of a Project

- One or more CASE local repositories (as discussed in Chapter 3).
- Hard-copy documentation (stored in notebooks, binders, and system libraries).

Hereafter, we will refer to these as making up a singular project repository.

FAST is a repository-based methodology. This means that phases (and activities included in phases) communicate across a shared repository. Thus, the phases and activities are not really sequential! Work in one phase can and should overlap work in another phase, so long as the necessary information is already in the repository. This accelerates development and allows *FAST* to live up to its name. Furthermore, this model permits the developer to backtrack when an error or omission is discovered.

This chapter examines each of the above phases in greater detail. But first, let's examine some overall strategies for systems analysis.

STRATEGIES FOR SYSTEMS ANALYSIS AND PROBLEM SOLVING

Traditionally, systems analysis is associated with application development projects, that is, projects that produce information systems and their associated computer applications. Your first experiences with systems analysis will likely fall into this category. But systems analysis methods can be applied to projects with different goals and scope. In addition to single information systems and computer applications, systems analysis techniques can be applied to strategic information systems planning and to the redesign of business processes.

There are also many strategies or techniques for performing systems analysis. They include modern structured analysis, information engineering, prototyping, and object-oriented analysis. These strategies are often viewed as competing alternatives. In reality, certain combinations complement one another. Let's briefly examine these strategies and the scope or goals of the projects to which they are suited. The intent is to develop a high-level understanding only. The subsequent chapters in this unit will actually teach you the techniques.

Modern Structured Analysis

Structured analysis was one of the first formal strategies developed for systems analysis of information systems and computer applications. Modern structured analysis[1] is still one of the most widely practiced techniques.

> **Modern structured analysis** is a process-centered technique that is used to model business requirements for a system. The models are structured pictures that illustrate the processes, inputs, outputs, and files required to respond to business events (such as ORDERS).

By process-centered, we mean the initial emphasis in this technique is on the PROCESS building blocks in our information system framework. The technique has evolved to also include the DATA building blocks as a secondary emphasis.

Structured analysis was not only the first popular systems analysis strategy; it also introduced an overall strategy that has been adopted by many of the other techniques—model-driven development.

> A **model** is a representation of reality. Just as "a picture is worth a thousand words," most models use pictures to represent reality.

> **Model-driven development** techniques emphasize the drawing of models to define business requirements and information system designs. The model becomes the design blueprint for constructing the final system.

Modern structured analysis is simple in concept. Systems and business analysts draw a series of process models called *data flow diagrams* (Figure 4.2) that depict the essential processes of a system along with inputs, outputs, and files. Because

[1] Edward Yourdon, *Modern Structured Analysis* (Englewood Cliffs, NJ: Yourdon Press, 1989.)

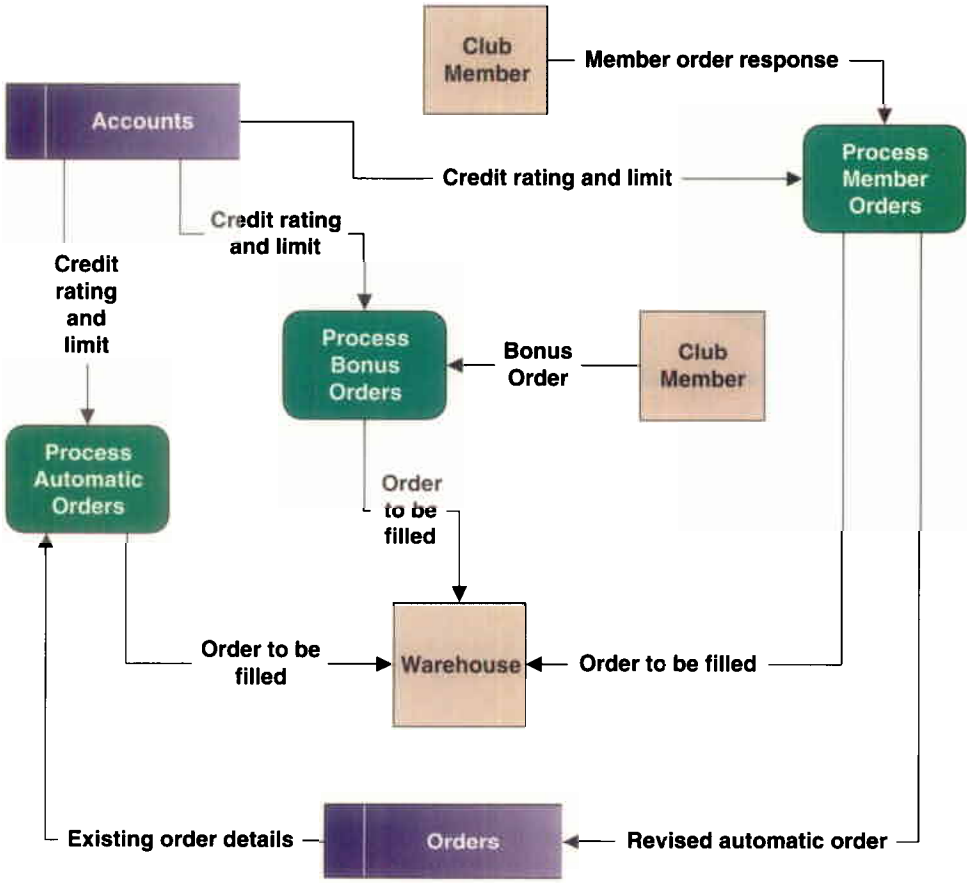


FIGURE 4.2
A Process Model (also Called a Data Flow Diagram)

these pictures represent the *logical* business requirements of the system independent of any *physical*, technical solution, the models are said to be a *logical design* for the system.

Information Engineering (IE)

Today, many organizations have evolved from a structured analysis approach to an information engineering[2] approach.

> **Information engineering** is a data-centered, but process-sensitive technique that is applied to the organization as a whole (or a significant part, such as a division), rather than on an ad-hoc, project-by-project basis (as in structured analysis).

The basic concept of information engineering is that information systems should be engineered like other products. Information engineering books typically use a pyramid framework to depict information systems building blocks and system development phases. As shown in Figure 4.3, the phases are:

1. **Information strategy planning (ISP)** applies systems analysis methods to examine the business as a whole to define an overall plan and architecture for subsequent information systems development. No actual information systems or computer applications are developed. Instead, the project team *studies* the business mission and goals and *defines* an information systems architecture and plan to optimally align information systems to help the organization achieve its business goals.

[2] James Martin, *Information Engineering: Volumes 1–3* (Englewood Cliffs, NJ: Prentice Hall, 1989).

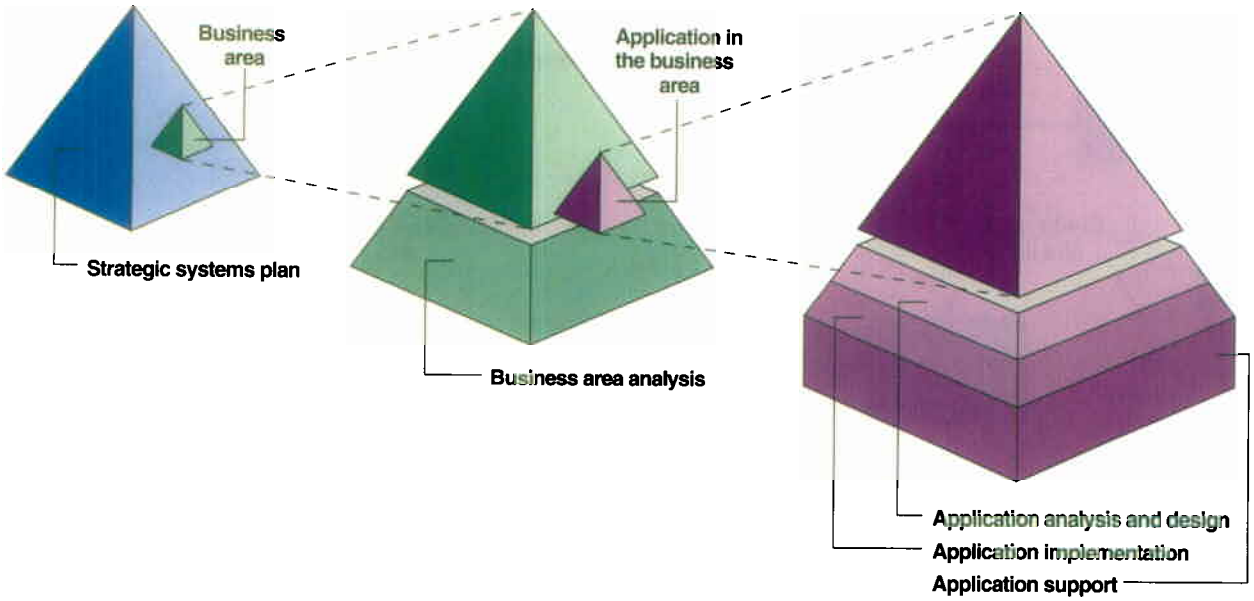


FIGURE 4.3 *Information Engineering Phases*

2. Based on the strategic plan, business areas are carved out and prioritized. A business area is a collection of cross-organizational business processes that should be highly integrated to achieve the information strategy plan (and business mission). A **business area analysis (BAA)** uses systems analysis methods to study the business area and define the business requirements for a highly streamlined and integrated set of information systems and computer applications to support that business area.
3. Based on the business area requirements analysis, information system applications are carved out and prioritized. These applications become projects to which other systems analysis *and* design methods are applied to develop production systems. These methods may include some combination of structured analysis and design, prototyping, and object-oriented analysis and design.

Information engineering is said to be a data-centered paradigm because it emphasizes the study and definition of DATA requirements before those of PROCESS, INTERFACE, or GEOGRAPHY requirements. This is consistent with the contemporary belief that information is a corporate resource that should be planned and managed. Since information is a product of data, data must be planned first! Data models, such as that shown in Figure 4.4, are drawn first. In addition to data models, information engineers also draw process models similar to those drawn in structured analysis.

Although information engineering has gradually replaced structured analysis and design as the most widely practiced strategy for systems analysis, information engineering actually integrates all the process models of structured analysis with its data models. That should make sense, since we know (from Chapter 2) that an information system must include both DATA and PROCESS building blocks. Information engineering was the first formal strategy for synchronizing those building blocks! Information engineering was also the first widely practiced strategy that considered GEOGRAPHY building blocks through application of tools that plan and document the distribution of data and processes to locations.

Prototyping

Another strategy for systems analysis is prototyping.

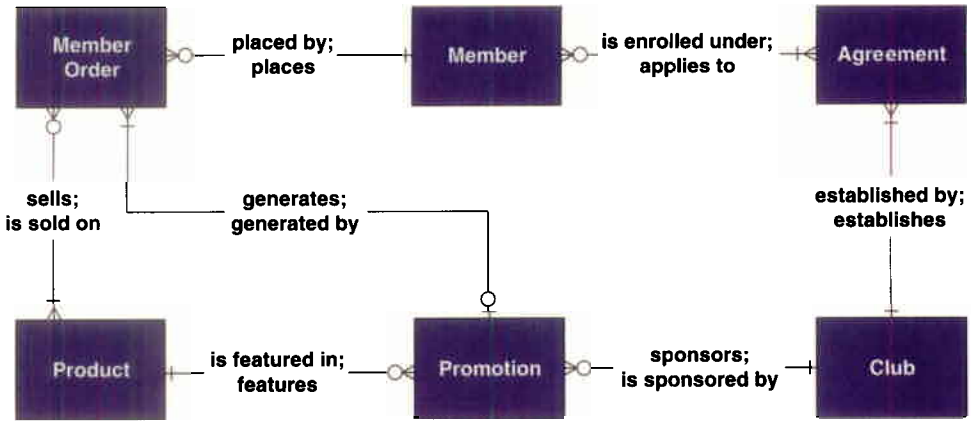


FIGURE 4.4 *A Data Model (also Called an Entity Relationship Diagram)*

Prototyping is an engineering technique used to develop partial but functional versions of a system or applications. When extended to system design and construction, a prototype can evolve into the final, implemented system.

Two flavors of prototyping are applicable to systems analysis:

- **Feasibility prototyping** is used to test the feasibility of a specific technology that might be applied to the business problem. For example, we might use Microsoft *Access* to build a quick-but-incomplete prototype of the feasibility of moving a mainframe application to a PC-based environment.
- **Discovery prototyping** (sometimes called *requirements prototyping*) is used to discover the users' business requirements by having them react to a quick-and-dirty implementation of those requirements. For example, we might again use Microsoft *Access* to create sample forms and reports to solicit user responses as to whether those forms and reports truly represent business requirements. (Note: In discovery prototyping, we try to discourage users from worrying about the style and format of the prototypes; that can be changed during system design!)

In response to the faster pace of the economy in general, prototyping has become a preferred technique for accelerating systems development. Many system developers extend the prototyping techniques to perform what they call **rapid application development.** Unfortunately, some developers are using prototyping to replace model-driven strategies, only to learn what true engineers have known for years—you cannot prototype without some degree of more formal design models.

Joint Application Development (JAD)

As previously described, modern structured analysis and information engineering both emphasize *model-driven development*. Prototyping places emphasis on construction of the working prototypes. Joint application development (JAD) complements both of these techniques by emphasizing *participative development* among system owners, users, designers, and builders.

Joint application development (JAD) uses highly organized and intensive workshops to bring together system owners, users, analysts, designers, and builders to jointly define and design systems. Synonyms include *joint application design* and *joint requirements planning*.

A JAD-trained systems analyst usually plays the role of facilitator for a workshop that will typically run from three to five full working days. This workshop may replace months of traditional interviews and follow-up meetings.

JAD provides a working environment in which to accelerate methodology activities and deliverables. It promotes enhanced system owner and user participation

in system development. But it also requires a facilitator with superior mediation and negotiation skills to ensure that all parties receive appropriate opportunities to contribute to the system's development.

Business Process Redesign (BPR)

One of the most interesting contemporary applications of systems analysis methods is business process redesign.

> **Business process redesign** (also called *business process reengineering*) is the application of systems analysis (and design) methods to the goal of dramatically changing and improving the fundamental business processes of an organization, independent of information technology.

The interest in BPR was driven by the discovery that most current information systems and applications have merely automated existing and inefficient business processes. Automated bureaucracy is still bureaucracy; it does not contribute value to the business and may actually subtract value from the business. Introduced in Chapter 1, BPR is one of many types of projects triggered by the trend we call *total quality management* (TQM).

BPR projects focus almost entirely on noncomputer processes. Each process is studied and analyzed for bottlenecks, value returned, and opportunities for elimination or streamlining. Once the business processes have been redesigned, most BPR projects conclude by examining how information technology might best be applied to the improved business processes. This creates new application development projects to which the other techniques described in this section might be applied. Business process redesign is a subject that deserves its own course and book.

Object-Oriented Analysis (OOA)

Object-oriented analysis is the new kid on the block. The concepts behind this exciting new strategy (and technology) are covered extensively in Chapter 8, but a simplified introduction is appropriate here.

For the past 30 years, most systems development strategies have deliberately separated concerns of DATA from those of PROCESS. The COBOL language, which dominated business application programming for years, was representative of this separation—the DATA DIVISION was separated from the PROCEDURE DIVISION. Most systems analysis (and design) techniques similarly separated these concerns to maintain consistency with the programming technology. Although most systems analysis and design methods have made significant attempts to synchronize data and process models, the results have been less than fully successful.

Object technologies and techniques are an attempt to eliminate the separation of concerns about DATA and PROCESS. Instead, data and the processes that act on that data are combined or encapsulated into things called *objects*. The only way to create, delete, change, or use the data in an object (called *properties*) is through one of its encapsulated processes (called *methods*). The system and software development strategy is changed to focus on the "assembly" of the system from a library of reusable objects. Of course, those objects must be defined, designed, and constructed. Thus, in the early part of the systems development process, we need to use object-oriented analysis techniques.

> **Object-oriented analysis (OOA)** techniques are used to (1) study existing objects to see if they can be reused or adapted for new uses, and to (2) define new or modified objects that will be combined with existing objects into a useful business computing application.

Object-oriented analysis techniques are best suited to projects that will implement systems using emerging object technologies to construct, manage, and assemble those objects into useful computer applications. Examples include *Smalltalk*, *C++*, *Delphi*, and *Visual BASIC*.

Today, most computer operating systems use graphical user interfaces (GUIs)

such as Microsoft *Windows* and IBM's *OS/2 Presentation Manager*. GUIs are built with object-oriented (or object-like) technologies. The development of GUI applications can be based on libraries of reusable objects (sometimes called *components*) that exhibit the same behaviors in all applications. For example, *Delphi* and *Visual BASIC* contain all the necessary objects (called components) to assemble the desired GUI screens for any new application (without programming!).

FAST Systems Analysis Strategies

The *FAST* methodology used by SoundStage Entertainment Club does not impose a single technique on system developers. Instead, it integrates all the popular techniques: structured analysis (via process modeling), information engineering (via data modeling), prototyping (via rapid application development), and joint application development (for all methods). Progressive *FAST* developers can use object-oriented analysis in conjunction with object technology for prototyping to fully exploit the object paradigm.[3]

Finally, the *FAST* methodology supports different types of projects including (1) application development, (2) information strategy planning, (3) business area analysis, (4) decision support system development, and (5) business process redesign. The SoundStage case study will demonstrate application development, a typical first assignment for a systems analyst.

FAST analysis techniques are applied within the framework of (1) your information system building blocks (which were introduced in Chapter 2), (2) the *FAST* phases (which were introduced in Chapter 3), and (3) *FAST* activities (described in this chapter). Given this overview of systems analysis scope and strategy, we can now explore the systems analysis activities. Each *FAST* phase will be described in terms of your information system building blocks and the activities that constitute that phase. For each activity, we will examine the following methodology elements:

- *Purpose* (self-explanatory).
- *Roles.* All *FAST* activities are completed by individuals who are assigned to **roles.** Roles are not the same as job titles. One person can play many roles in a project. Conversely, one role may require many people to adequately fulfill that role. For example, a system user role may require several users in order to adequately represent the interests of an entire system. *FAST* roles are assigned to the following role groups: system owner roles, system user roles, systems analyst roles, system designer roles, and system builder roles.

Notice that these groups correspond with the perspectives in your information system framework (from Chapters 2 and 3).

Every activity is described with respect to:

- *Prerequisites and inputs* (to the activity).
- *Deliverables and outputs* (produced by the activity).
- *Applicable techniques*—which techniques (from the previous section) are applicable to this phase.
- *Steps*—a brief description of the steps required to complete this activity. In the spirit of continuous improvement, all *FAST* steps are fully customizable for each organization.

Furthermore, all roles, inputs, outputs, techniques, and steps are presented with the following designations:

- (REQ) indicates that the role, input, output, technique, or step is required.
- (REC) indicates that the role, input, output, technique, or step is recommended but not required.

[3] Recall from the Chapter 3 opening case study that SoundStage has elected to test the use of object techniques and technology in the Member Services Information System project.

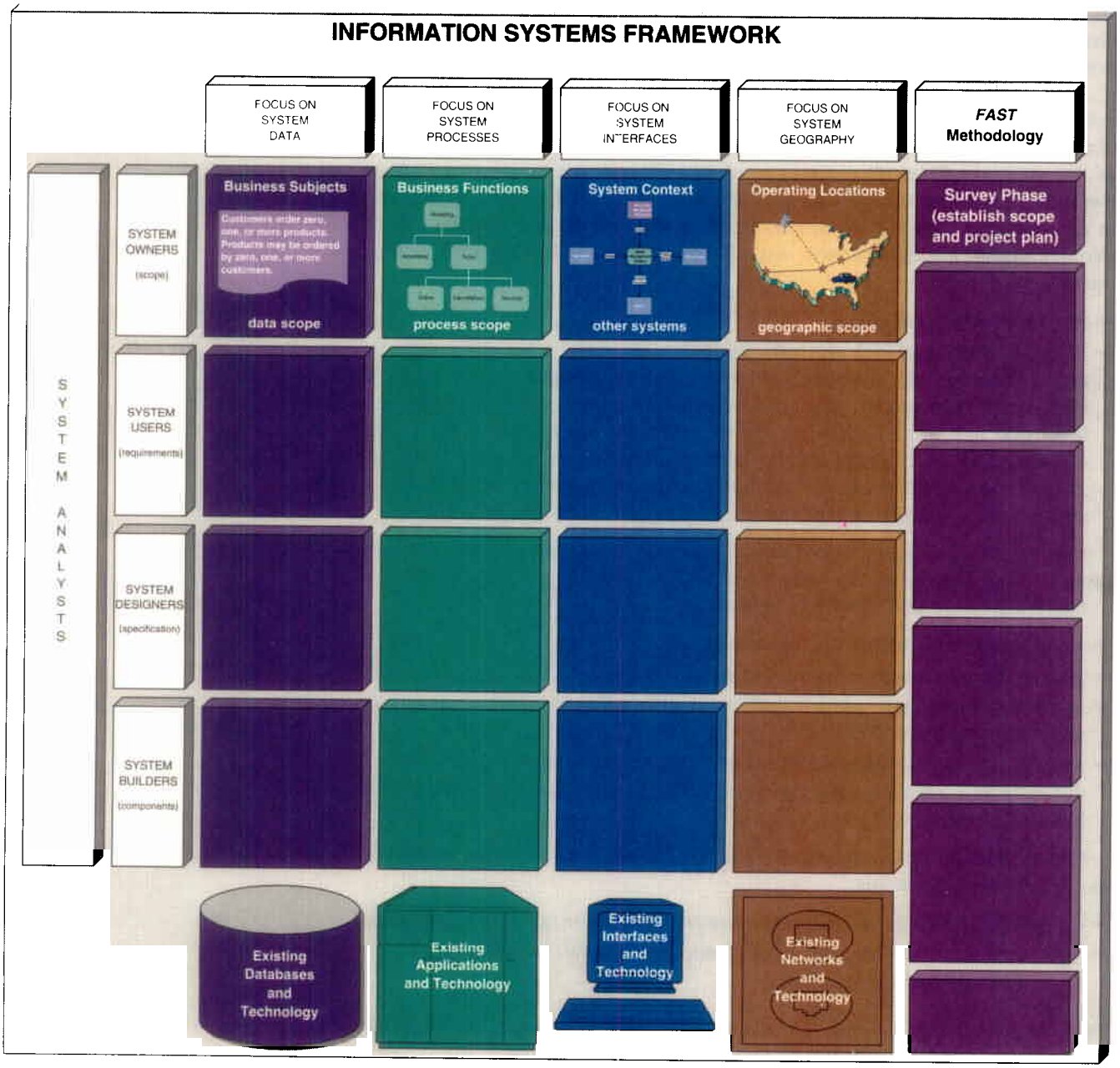
— (OPT) indicates that the role, input, output, technique, or step is optional but not required.

THE SURVEY PHASE OF SYSTEMS ANALYSIS

Recall from Chapter 3 that the first phase of a *FAST* project is to survey the project. The purpose of the survey phase is threefold. First, the survey phase answers the question, "Is this project worth looking at?" To answer this question, the survey phase must define the scope of the project and the perceived problems, opportunities, and directives that triggered the project. Assuming the project *is* worth looking at, the survey phase must also establish the project team and participants, the project budget, and the project schedule.

Your information system framework (Figure 4.5) provides the context for defining scope and understanding the basic problem domain. As shown in the model,

FIGURE 4.5 *Building Blocks for the Survey Phase*



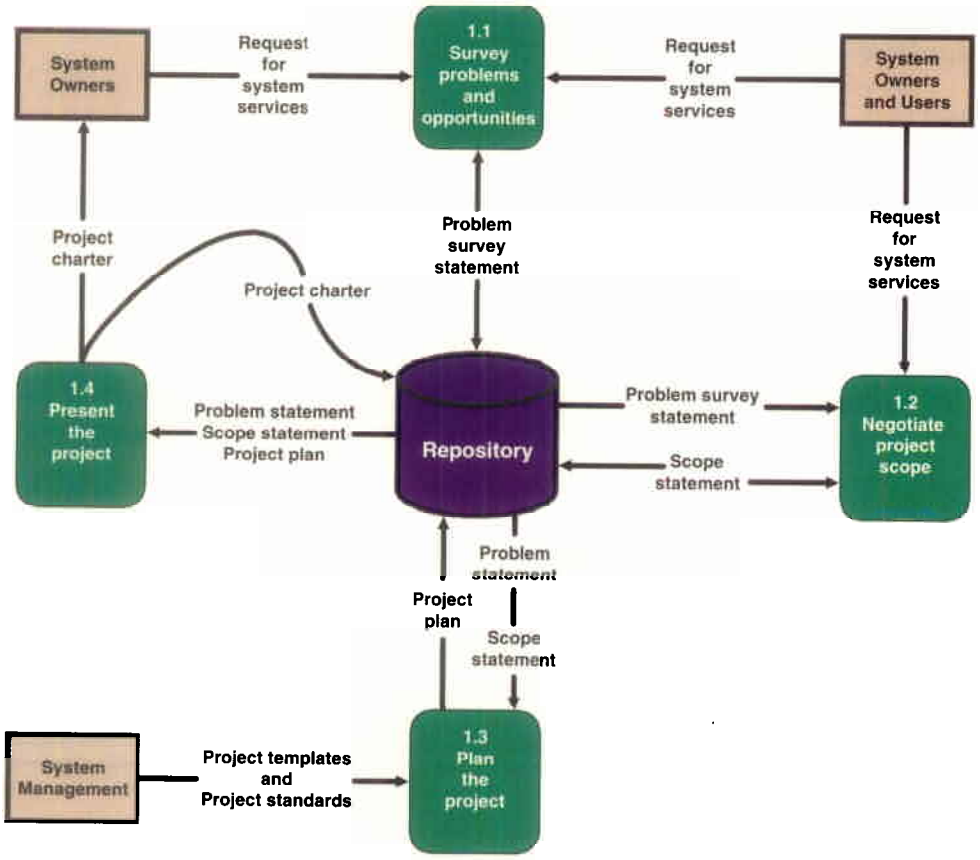


FIGURE 4.6
Activity Diagram for the Survey Phase

the survey phase is concerned with the system owner's view of the overall information system, which includes very few details. More detailed information will be collected if the project is approved to continue to the next phase.

Figure 4.6 is the first of three activity diagrams for systems analysis.

A *FAST* **activity diagram** shows the activities or work that must be completed to accomplish a *FAST* phase.

Solid lines indicate information and documentation flows. A small, shaded circle at the beginning of any input or output information flow indicates a feasibility checkpoint.[1]

Recall that the survey phase is intended to be quick. The entire phase should not exceed two or three days for most projects. Let's now examine each activity in greater depth.

Activity: Survey Problems, Opportunities, and Directives

One of the most important activities of the survey phase is to establish an initial reading of the problems, opportunities, and/or directives that triggered the project. Participants should keep in mind that a more thorough analysis of the same will occur during the study phase.

Purpose The purpose of this activity is to quickly survey and evaluate each identified problem, opportunity, and directive with respect to urgency, visibility, tangible benefits, and priority. Optionally, the participants can explore possible

[1] This is consistent with the *creeping commitment* principle of systems development that was introduced in Chapter 3.

solutions, although everyone should be informed that other solutions may and should be explored at later stages of the project.

Roles The activity is facilitated by the *project manager*. Other roles are defined as follows:

System owner roles:

- Executive sponsor (REQ)—the highest-level manager who will pay for and support the project.
- User managers (REQ)—the managers of the organizational units most likely to be supported by the system developed in this project.
- System managers (OPT)—the information systems unit managers to whom the project manager reports (e.g., the manager of systems development).
- Project manager (REQ)—the information systems unit manager who will

FIGURE 4.7
A Request for System Services

SoundStage Entertainment Club
Information System Services
Phone: 494-0686 Fax: 494-0999
Internet: http://www.soundstage.com
Intranet: http://www.soundstage.com/iss

REQUEST FOR INFORMATION SYSTEM SERVICES

DATE OF REQUEST	SERVICE REQUESTED FOR DEPARTMENT(S)
January 10, 1997	Member Services, Warehouse, Shipping

SUBMITTED BY (key user contact)
Name: Sarah Hartman
Title: Business Analyst, Member Services
Office: B035
Phone: 494-0867

EXECUTIVE SPONSOR (funding authority)
Name: Galen Kirkhoff
Title: Vice President, Member Services
Office: G242
Phone: 494-1242

TYPE OF SERVICE REQUESTED:
- ☐ Information Strategy Planning
- ☒ Business Process Analysis and Redesign
- ☒ New Application Development
- ☐ Other (please specify) _____
- ☐ Existing Application Enhancement
- ☐ Existing Application Maintenance (problem fix)
- ☐ Not Sure

BRIEF STATEMENT OF PROBLEM, OPPORTUNITY, OR DIRECTIVE (attach additional documentation as necessary)
The information strategy planning group has targeted member services, marketing, and order fulfillment (inclusive of shipping) for business process redesign and integrated application development. Currently serviced by separate information systems, these areas are not well integrated to maximize efficient order services to our members. The current systems are not adaptable to our rapidly changing products and services. In some cases, separate systems exist for similar products and services. Some of these systems were inherited through mergers that expanded our products and services. There also exist several marketing opportunities to increase our presence to our members. One example includes Internet commerce services. Finally, the automatic identification system being developed for the warehouse must fully interoperate with member services.

BRIEF STATEMENT OF EXPECTED SOLUTION
We envision completely new and streamlined business processes that minimize the response time to member orders for products and services. An order shall not be considered fulfilled until it has been received by the member. The new system should provide for expanded club and member flexibility and adaptability of basic business products and services.

We envision a system that extends to the desktop computers of both employees and members, with appropriate shared services provided across the network, consistent with the ISS distributed architecture. This is consistent with strategic plans to retire the AS/400 central computer and replace it with servers.

ACTION (ISS Office Use Only)
- ☐ Feasibility assessment approved
- ☒ Feasibility assessment waived
- ☐ Request delayed
- ☐ Request rejected

Assigned to: Sandra Shepherd
Approved Budget: $ 450,000
Start Date: ASAP Deadline: ASAP
Backlogged until date: _____
Reason: _____

Authorized Signatures:
Rebecca J. Todd — Chair, ISS Executive Steering Body
Galen Kirkhoff — Project Executive Sponsor

FORM ISS-100-RFSS (Last revised December, 1996)

directly manage the project team. This is usually a senior systems analyst (e.g., Sandra will play this role in the SoundStage project).

- System user roles.
  - Business analyst (OPT)—an analyst who is on loan from the user community to the information systems unit (applicable only to organizations that practice this concept).
  - Other users are typically not involved in this activity at this time.
- Systems analyst roles.
  - System modelers (REC)—systems analysts who are skilled with the system modeling techniques and CASE tools that will be used in the project.
  - System designer roles are not typically involved in this activity unless deemed appropriate by a system owner.
  - System builder roles are not typically involved in this activity unless deemed appropriate by a system owner.

Prerequisites (Inputs) This activity is triggered by a request for system services (REQ), shown in Figure 4.7. This input implements the two logical project triggers that were described in Chapter 3—a *planned system project directive* or an *unplanned system request*.

Deliverables (Outputs) The principle deliverable of this activity is a problem statement (REQ), which documents the problems, opportunities, and directives that were discussed. Figure 4.8 is a sample document that summarizes problems, opportunities, and directives in terms of:

- *Urgency*. In what time frame must/should the problem be solved or the opportunity or directive realized? A rating scale could be developed to consistently answer this question.
- *Visibility*. To what degree would a solution or new system be visible to customers and/or executive management? Again, a rating scale could be developed for the answers.
- *Benefits*. Approximately how much would a solution or new system increase annual revenues or reduce annual costs. This is often a guess, but if all participants are involved in that guess, it should prove sufficiently conservative.
- *Priority*. Based on the above answers, what are the consensus priorities for each problem, opportunity, or directive. If budget or schedule becomes a problem, these priorities will help to adjust project scope.
- *Possible solutions* (OPT). At this early stage of the project, possible solutions are best expressed in simple terms such as (1) leave well enough alone, (2) a quick fix, (3) a simple-to-moderate enhancement of the existing system, (4) redesign the existing system, or (5) design a new system. The participants listed for this activity are well suited to an appropriately high-level discussion of these options.

Applicable Techniques The following techniques are applicable to this activity. For each activity, we indicate which chapters or modules of the book teach that technique.

- *Fact-finding*. Fact-finding methods (REQ) are used to interact with people to identify problems, opportunities, and directives. Typically, scope is defined through interviews or a group meeting. Interviewing and meeting methods and frameworks are discussed in Part Five, Module B, "Fact-Finding and Information Gathering."

PROBLEM STATEMENTS

PROJECT:	Member Services Information System	**PROJECT MANAGER:**	Sandra Shepherd
CREATED BY:	Sandra Shepherd	**LAST UPDATED BY:**	Robert Martinez
DATE CREATED:	January 15, 1997	**DATE LAST UPDATED:**	January 17, 1997

Brief Statements of Problem, Opportunity, or Directive	Urgency	Visibility	Annual Benefits	Priority or Rank	Proposed Solution
1. Order response time as measured from time of order receipt to time of customer delivery has increased to an average of 15 days.	ASAP	High	$175,000	2	New development
2. The recent acquisitions of Private Screenings Video Club and GameScreen will further stress the throughput requirements for the current system.	6 months	Med	75,000	2	New development
3. Currently, three different order entry systems service the audio, video, and game divisions. Each system is designed to interface with a different warehousing system; therefore, the intent to merge inventory into a single warehouse has been delayed.	6 months	Med	515,000	2	New development
4. There is a general lack of access to management and decision-making information. This will become exasperated by the acquisition of two additional order processing systems (from Private Screenings and GameScreen)	12 months	Low	15,000	3	After new system is developed, provide users with easy-to-learn and use reporting tools.
5. There currently exists data inconsistencies in the member and order files.	3 months	High	35,000	1	Quick fix; then new development
6. The Private Screenings and GameScreen file systems are incompatible with the SoundStage equivalents. Business data problems include data inconsistencies and lack of input edit controls.	6 months	Med	unknown	2	New development. Additional quantification of benefit might increase urgency.
7. There is an opportunity to open order systems to the Internet, but security and control is an issue.	12 months	Low	unknown	4	Future version of newly developed system
8. The current order entry system is incompatible with the forthcoming automatic identification (bar coding) system being developed for the warehouse.	3 months	High	65,000	1	Quick fix; then new development

FIGURE 4.8 *A Representative Problem Statement*

— *Interpersonal skills*. Interpersonal skills (REQ) are related to fact-finding skills. They impact the way we communicate and negotiate with one another. Good interpersonal relations are essential to this activity. Interpersonal skills are taught in Part Five, Module E, "Interpersonal Skills and Communications."

Steps The following steps are suggested to complete this activity.

1. (REQ) Collect and review all documentation submitted to begin this project.
2. (REQ) Schedule and conduct a meeting of the people tentatively assigned to the aforementioned roles for this activity. (Alternative: Interview the people tentatively assigned to those roles.)
3. (REC) Document problems, opportunities, and constraints.

Activity: Negotiate Project Scope

Scope defines the boundary of the project—what aspects of the system will and will not be included in the project. Scope can change during the project; however, the initial project plan must be based on some agreement regarding scope. Then if the scope changes significantly, all parties involved will have a better appreciation for why the budget and schedule have also changed. This activity can occur in parallel with the prior activity.

Purpose The purpose of this activity is to define the boundary of the system and project. The boundary should be defined as precisely as possible to minimize the impact of creeping scope. Creeping scope is the subtle, but significant increase of scope that frequently occurs during system projects. Scope can increase for many legitimate reasons. By defining scope, we are not eliminating creeping scope. We

are merely providing a mechanism to document and track that scope so the impact on budget and schedule can be continuously reassessed.

Roles The activity is facilitated by the *project manager*. Other roles are defined as follows:

- System owner roles (defined in the previous activity).
  - Executive sponsor (REC).
  - User managers (REQ).
  - System managers (OPT).
  - Project manager (REQ).
- System user roles.
  - Business analysts (OPT).
  - Other users are typically not involved in this activity at this time.
- Systems analyst roles.
  - System modelers (REC).
- System designer roles are not typically involved in this activity unless deemed appropriate by a system owner.
- System builder roles are not typically involved in this activity unless deemed appropriate by a system owner.

Prerequisites (Inputs) This activity is triggered by the same *request for system services* (REQ) described as an input to the previous activity. The *problem survey statement* (REQ) produced by the previous activity can be a useful input for defining scope.

Deliverables (Outputs) The principal deliverable of this activity is a *scope statement* (REQ) that corresponds to the four building blocks for this phase:

- *Business subjects* define the scope for DATA. This might be a simple list of things about which the system needs to know information.
- *Business functions* define the scope for PROCESSES. This might be a simple list of ongoing business functions that would be included in or affected by this system.
- *System context* defines the scope for INTERFACES. This might be a list of external people, organization units, organizations, or other systems with which the system might have to interact.
- *Operating locations* define the scope for GEOGRAPHY. This might be a simple list of specific business operating locations that will be included within the scope of the project.

Notice that each statement of scope was described as a simple list. We don't necessarily define the items in the list. Nor are we very concerned with precise definitions. And we definitely are not concerned with any time-consuming steps such as modeling or prototyping.

Applicable Techniques The following techniques are applicable to this activity. For each activity, we indicate which chapters or modules of the book teach that technique.

- *Fact-finding.* Fact-finding methods (REQ) are used to interact with people to define scope. Typically, scope is defined by interviews or a group meeting. Interviewing and meeting methods and frameworks are discussed in Part Five, Module B, "Fact-Finding and Information Gathering."

- *Interpersonal skills.* Once again, good interpersonal skills (REQ) are essential to this activity. Applicable interpersonal skills are taught in Part Five, Module E, "Interpersonal Skills and Communications."

Steps The following steps are suggested to complete this activity.

1. (REC) Collect and review all documentation submitted to begin this project.
2. (REQ) Schedule and plan a meeting of the people tentatively assigned to the aforementioned roles for this activity. (Alternative: Interview the people tentatively assigned to those roles.) The meeting or interviews should focus on negotiating the scope in terms of the four building blocks of information systems: DATA, PROCESSES, INTERFACES, and GEOGRAPHY. Figure 4.8 provides sample questions for the meeting or interviews.
3. (REQ) Document scope.

Activity: Plan the Project

Based on the prior activities, you can now answer the question, Is the project worth looking at? If the answer is yes, the group needs to formally plan the project. At this stage of the project, we know too little to accurately predict costs or schedules. Thus, our initial project plan should consist of the following:

- A first-draft master plan and schedule for completing the entire project. This schedule will be modified at the end of each phase of the project. This is sometimes called a *baseline plan.*
- A detailed plan and schedule for completing the next phase of the project (the study phase). In most cases this schedule will be more accurate but still subject to a lack of detailed knowledge about the current system and user requirements.

The project plan is derived from experience plus the problem survey and scope definitions produced in the prior activities.

Purpose The purpose of this activity is to develop the initial project schedule and resource assignments. This includes selecting the appropriate *FAST* project template, assigning *FAST* roles to personnel, estimating time commitments for both people and activities, and generating the initial schedule and resource requirements.

Roles The activity is facilitated by the *project manager.* Other roles are defined as follows:

- System owner roles (defined in the previous activity).
  - Executive sponsor (REQ).
  - User managers (REC).
  - System managers (REC).
  - Project manager (REQ).
  - Steering body (OPT)—many organizations require that all project plans be formally presented to a steering body (sometimes called a steering committee) for final approval.
- System user roles.
  - Business analysts (OPT).
- Systems analyst, system designer, and system builder roles are not typically involved in this activity unless deemed necessary by the project manager.

Prerequisites (Inputs) This activity is triggered by completion of the problem survey and scope definition activities. The *problem statement* (REC) and the *scope*

statement (REC), if formally documented, are very helpful references for the project planning group.

Deliverables (Outputs) The principle deliverable of this activity is the *project plan* (REC). Many project managers use project management software to create and maintain their project plans. This initial project plan consists of two components:

- A *phase-level* plan that covers the entire project (REC).
- An *activity-level* plan that details the study phase of the project (REC).

Applicable Techniques The following techniques are applicable to this activity. For each activity, we indicate which chapters or modules of the book teach that technique.

- *Process management.* Process management (REC) (also called *methodology management*) defines the standards for applying the methodology to a project. It defines skill requirements and training for each role, CASE tool standards, documentation standards, quality management standards, and project management standards. Process management techniques are taught in Part Five, Module A, "Project and Process Management."
- *Project management.* Project management (REC) builds on process management by applying the methodology to specific projects in the form of schedule planning, staffing and supervision, progress reporting, management of expectations, budgeting, and schedule management. Project management techniques are taught in Part Five, Module A, "Project and Process Management."
- *Presentation skills.* The project charter and any verbal presentations of the project and plan obviously require presentation skills (REC). Presentation skills are taught in Part Five, Module E, "Interpersonal Skills and Communications."

Steps The following steps are suggested to complete this activity.

1. (REQ) Review system problems, opportunities, and directives, as well as project scope.
2. (REC) Select the appropriate *FAST* project template. *FAST* templates support different strategies and/or different system development goals (e.g., purchase a package versus object-oriented development).
3. (REQ) Assign specific people to each *FAST* role.
4. (REC) Estimate time required for each project activity, assign roles to activities, and construct a schedule.
5. (OPT) Negotiate expectations. (See Part Five, Module A, "Project and Process Management" for a simple expectations management instrument.)
6. (REC) Negotiate the schedule with system owners, adjusting resources, scope, and expectations as necessary.
7. (REC) Write the project charter.

Activity: Present the Project

In most organizations, there are more potential projects than resources to staff and fund those projects. If a project has not been predetermined to be of the highest priority (by some sort of prior tactical or strategic planning process), then it must be presented and defended to a steering body for approval.

A **steering body** is a committee of executive business and system managers that studies and prioritizes competing project proposals to determine which projects will return the most value to the organization and thus should be approved for continued systems development.

The majority of any steering body should consist of noninformation systems professionals or managers. Many organizations designate vice presidents to serve on a steering body. Other organizations assign the direct reports of vice presidents to the steering body. And some organizations utilize two steering bodies, one for vice presidents and one for their direct reports. Information systems managers serve on the steering body only to answer questions and to communicate priorities back to developers and project managers.

Purpose The purpose of this activity is (1) to secure any required approvals to continue the project and (2) to communicate the project and goals to all staff.

Roles Ideally, the activity should be facilitated by the *executive sponsor*. Other roles are defined as follows:

- System owner roles (defined in the previous activity).
  - Executive sponsor (REQ).
  - User managers (REC).
  - System managers (REC).
  - Project manager (REQ).
  - Steering body (OPT or REQ—depends on the organization).
- System user roles.
  - Business analysts (REC).
  - All direct and indirect users (REC).
- System designers.
  - Any systems analysts assigned to the project (REC).
  - Any system designers and specialists likely to be assigned to the project (REC).
- System builders.
  - Any system builders likely to be assigned to the project (REC).
  - Representatives of any technology vendors whose products are likely to be involved in the project (OPT).

Prerequisites (Inputs) This activity is triggered by the completion of the project planning activity. The inputs include the *problem statement* (REC), the *scope statement* (REC), and the *project plan* (REC) generated by the prior activities. Additionally, *project templates* (OPT) and *project standards* (REC) may be provided by systems management.

Deliverables (Outputs) The key deliverable of this activity is the *project charter* (REQ). This charter is usually a formal consolidation of all the inputs to the activity. It might be thought of as an internal contract for the project, should the project continue to the next phase.

The final deliverables of the activity (as discussed in Chapter 3) are the *problem statement* and *scope statement* that become the triggers for various study phase activities. They may take the form of a verbal presentation, a written document (possibly the project charter or a summary thereof), a letter of authority from the executive sponsor, or some combination of these formats. Figure 4.9 provides a representative outline of a written report.

Applicable Techniques The following techniques are applicable to this activity. For each activity, we indicate which chapters or modules of the book teach that technique.

- *Interpersonal skills*. Once again, good interpersonal skills (REQ) are essential to this activity. These include persuasion, sales (of ideas), writing, and

FIGURE 4.9 *Outline for a Survey Phase Final Report*

Project Feasibility Assessment Report

I. Executive summary (1 page)
   A. Summary of recommendation
   B. Brief statement of anticipated benefits
   C. Brief explanation of report contents
II. Background information (1–2 pages)
   A. Brief description of project request
   B. Brief explanation of the summary phase activities
III. Findings (2–3 pages)
   A. Problems and analysis (optional: reference *problem statement matrix*)
   B. Opportunities and analysis (optional: reference *problem statement matrix*)
   C. Directives and implications
IV. Detailed recommendation
   A. Narrative recommendation (1 page)
      1. Immediate fixes
      2. Quick fixes
      3. Enhancements
      4. New systems development
   B. Project plan
      1. Initial project objectives
      2. Initial master project plan (phase level)
      3. Detailed plan for the study or definition phase
V. Appendixes
   A. Request for system services
   B. Problem statements matrix
   C. (other documents as appropriate)

speaking. Applicable interpersonal skills are taught in Part Five, Module E, "Interpersonal Skills and Communications."

Steps The following steps are suggested to complete this activity.

1. (REQ) Review the deliverables of all prior activities.
2. (OPT) Reformat the project charter for presentation to the steering body.
3. (REQ in some organizations) Present the project proposal (charter) to the steering body. Be prepared to defend recommendations, address issues and controversies, and answer questions as posed by the steering body.
4. (REC) Plan an event to communicate the approved project to any and all affected staff, or (OPT) distribute the project charter or summary over a cover letter of authority from the executive sponsor. This launch event (e.g., a pizza luncheon) presents the project and plan to both participants and all interested parties. The executive sponsor's visible support of the project can prevent many political problems from surfacing.

This concludes the survey phase. The participants in the survey phase might decide the project is not worth proposing. It is also possible the steering body may decide that other projects are *more* important. Or the executive sponsor might not endorse the project. In each of these instances, the project is terminated. Little time and effort have been expended.

On the other hand, with the blessing of all system owners, the project can now proceed to the study and/or definition phases.

THE STUDY PHASE OF SYSTEMS ANALYSIS

You may recall the old saying, "Don't try to fix it unless you understand it." That statement aptly describes the next phase of a systems analysis, *study and analyze the current system*. There is always a current system, regardless of whether or not

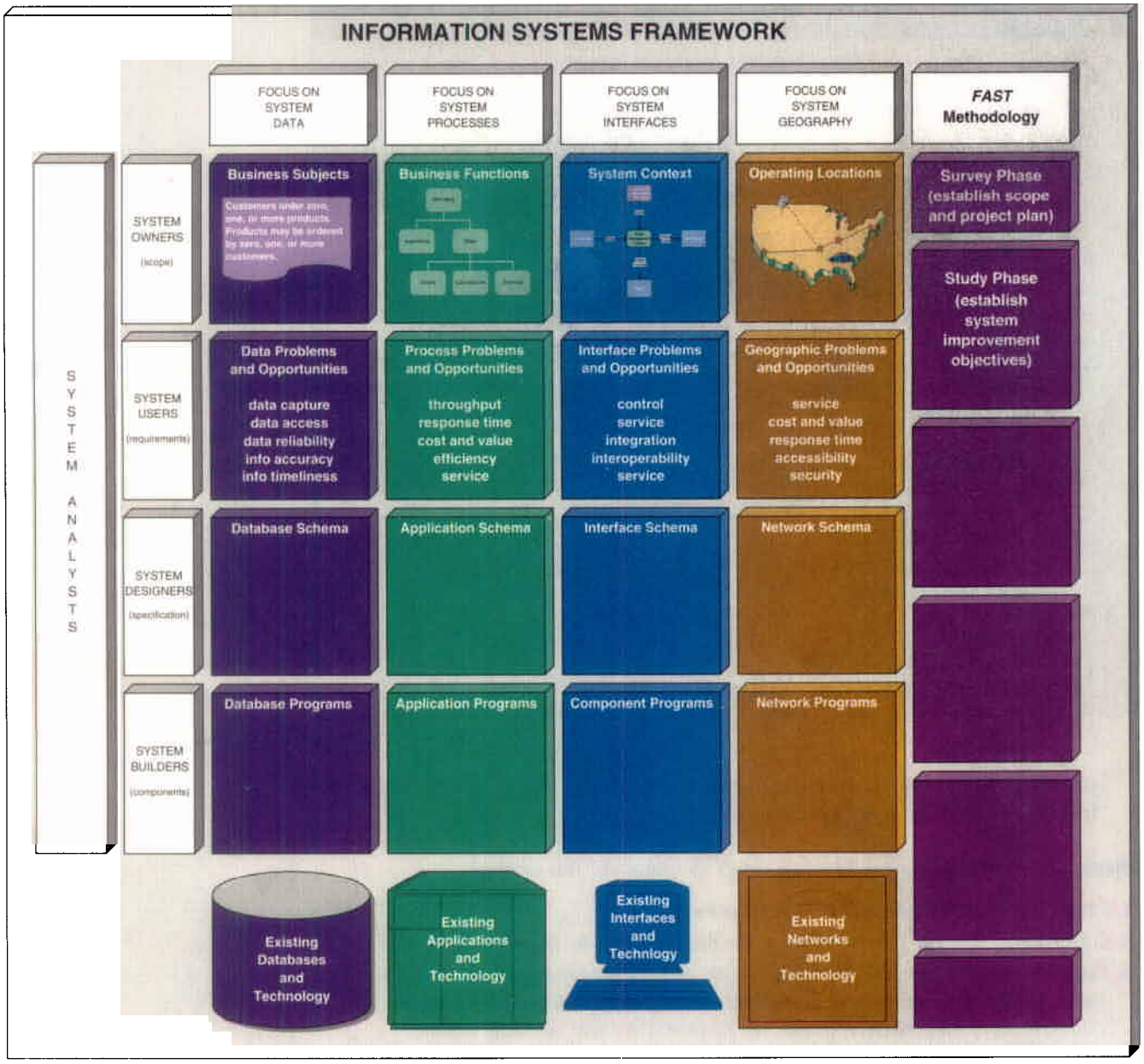
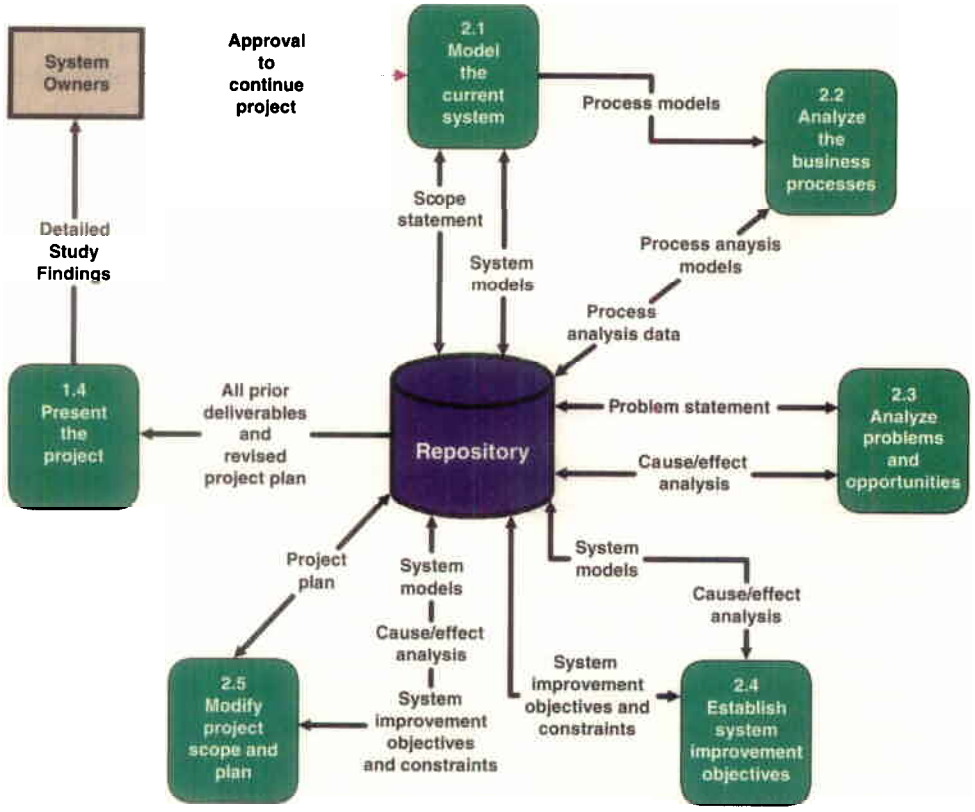


FIGURE 4.10 *Building Blocks for the Study Phase*

it currently uses computers. The study phase provides the analyst with a more thorough understanding of problems, opportunities, and/or directives. The study phase answers the questions, Are the problems really worth solving? and Is a new system really worth building? In different methodologies it may be called the detailed study or problem statement phase.

Can you ever skip the study phase? Rarely! You almost always need some understanding of the current system. But there may be some reasons to accelerate the study phase. First, if the project was triggered by systems planning, the worthiness of the project is not in doubt—the study phase is reduced to understanding the current system, not analyzing it. Second, if the project was initiated by a directive (such as, "Process financial aid applications according to new federal regulations that go into effect on July 1"), then worthiness is, once again, not in doubt.

The goal of the *FAST* study phase is to understand the problem domain well



Note: The dashed line represents a permission to continue.

FIGURE 4.11
Activity Diagram for the Study Phase

enough to thoroughly analyze its problems, opportunities, and constraints. Like many other methodologies, *FAST* once encouraged a detailed understanding of the current system that was documented with painstaking precision in various models such as data flow diagrams. Historically, the value added by this documentation proved questionable. Thus, the current version of *FAST* encourages some modeling, but only enough to refine our understanding of project scope and to define a common vocabulary for the system.

Your information systems framework (Figure 4.10) illustrates the context of the study phase as it relates to the information systems building blocks. Notice that the study phase is concerned jointly with the system owners' and users' views of the overall information system in somewhat greater detail than the survey phase.

Figure 4.11 is the activity diagram for the study phase. Depending on the size of the system, its complexity, and the degree to which project worthiness is already known, these activities may consume one to six weeks. Most of these activities can be accelerated by JAD sessions. (Joint application development was introduced earlier in this chapter and is taught in depth in Module D.)

Let's explore the study phase activities in greater depth.

Activity: Model the Current System

During the study phase, the team is attempting to learn about the current system. Each team member brings a different level of understanding to the process—different scope, different detail, different vocabulary, different perceptions, and different opinions. A well-conducted study can prove revealing to all parties, including management and users. How do you verify the team's consensus understanding of the current system? One way is to *model the current system*. Recall that a model is a representation of reality, usually pictorial.

The value of modeling in the study phase has come under some debate in recent years. Many early techniques, such as De Marco's structured analysis, called

for very detailed models of the current system. This consumed considerable time. Users and managers frequently became frustrated with endless reviews of a system that they wanted to replace anyway! This problem has sometimes been referred to as *analysis paralysis*. *FAST* suggests one of two modeling strategies for the study phase:

- A combination of high-level data, process, and geographic models.
- A combination of object and geographic models.

Let's examine the activity in greater detail.

Purpose The purpose of this activity is to learn enough about the current system's data, processes, interfaces, and geography to expand the understanding of scope and to establish a common working vocabulary for that scope.

Roles The activity may be facilitated by either the project manager or a systems analyst (sometimes one and the same person). Other roles are defined as follows:

- System owner roles.
  - User managers (REQ)—the managers of the organizational units most likely to be supported by the system developed in this project.
  - System managers (OPT)—the information systems unit managers to whom the project managers report (e.g., the manager of systems development).
  - Project manager (REQ)—the information systems unit manager who will directly manage the project team. This is usually a senior systems analyst (e.g., Sandra will play this role in the SoundStage project).
- System user roles.
  - Business analyst (REC)—an analyst who is on loan from the user community to the information systems unit (applicable only to organizations that practice this concept).
  - Other users (REC) as needed to fully represent the business scope of the project.
- System analyst roles.
  - System modeler (REQ)—a systems analyst who is skilled in the modeling techniques to be used in this activity.
- System designer roles are not typically involved in this activity unless deemed appropriate by a system owner.
- System builder roles are not typically involved in this activity unless deemed appropriate by a system owner.

Prerequisites (Inputs) This activity is triggered by completion of the survey phase activities and approval from the system owners to continue the project. The key informational input is the project and system *scope statement* (REC) that was completed as part of the survey phase. These information flows were covered in the previous section.

Deliverables (Outputs) The principal deliverable of this activity are *system models* (REQ) that serve two purposes: (1) to expand understanding of scope, and (2) to verify the team's consensus understanding of the business situation. The overriding modeling strategy is information hiding.

> The principle of **information hiding**, as applied to system models, suggests that models should hide inappropriate details in an effort to focus attention on what's really important. In other words, "If we want to learn *anything*, we must not try to learn *everything*—at least not all at once."

According to our information system framework, appropriate current system models may include:

- DATA—A one-page data model (REQ) is very useful for establishing business vocabulary, rules, and policies.
- PROCESSES—Today, it is widely accepted that a one- or two-page functional decomposition diagram (REC) should prove sufficient to get a feel for the current system processing.
- INTERFACES—A one-page context diagram (REQ) is useful for clarifying the system's inputs and outputs with other systems, organizations, and departments.
- GEOGRAPHY—A one-page geographic model (OPT) adequately identifies current operating locations that are relevant to the system.

Samples of some of these models were presented earlier in the chapter (as Figures 4.2 and 4.3).

Applicable Techniques The following techniques are applicable to this activity. For each activity, we indicate which chapters or modules of the book teach that technique.

- *Fact-finding.* By now, a common theme has emerged. Good fact-finding skills (REQ) are essential to most activities in the systems analysis phases. Fact-finding skills include interviewing, sampling, questionnaires, and research. These skills are covered in Part Five, Module B, "Fact-Finding and Information Gathering."
- *Joint application development.* The preferred technique for gathering information as rapidly as possible is joint application development (REC). The requisite system models can be developed in one or two facilitated group sessions with all the participants. JAD techniques are covered in Part Five, Module D, "Joint Application Development."
- *Data, process, and geographic modeling.* System modeling (REQ) is taught in the following chapters: Chapter 5, "Data Modeling"; Chapter 6, "Process Modeling"; and Chapter 7, "Distribution Modeling." These chapters cover modeling for both the study and definition phases.
- *Interpersonal skills.* And yet another common theme of systems analysis emerges—good interpersonal skills (REQ) are essential to most systems analysis activities. These skills are explored in Part Five, Module E, "Interpersonal Skills and Communications."

Steps The following steps are suggested to complete this activity.

1. (REC) Review the scope statement completed in the survey phase.
2. (REQ) Collect facts and gather information about the current system. The preferred technique is JAD, but JAD sessions may be preceded or followed by traditional fact-finding and information gathering activity.
3. (REQ) Draw system models. The recommended sequence of models is (1) INTERFACE, (2) DATA, (3) PROCESS, and (4) GEOGRAPHY. The interface model is first because it verifies and expands on the project scope. The data model is second because it helps establish basic business vocabulary and rules. The process model identifies high-level business functions. The geography model identifies the potential operating locations to which data, processes, and interfaces might eventually be distributed. Together, the models provide a solid foundation for problem and opportunity analysis.

4. (REQ) Verify the system models. The goal is to reach consensus on what the current system is all about. If JAD techniques are used, steps 2, 3, and 4 are consolidated into the group sessions.

(OPT) Activity: Analyze Business Processes

This activity is only applicable to *business process redesign* (BPR) projects. In such a process, the team is asked to examine business processes in much greater detail. The team models those business processes in greater detail and measures the value added or subtracted by each process as it relates to the total organization. Business process analysis can be politically charged. Managers and users alike can become very defensive about their existing business processes. The analysts involved must keep the focus on the processes, not the people, and constantly remind everyone that the goal is to identify opportunities for fundamental business change that will benefit the business and everyone in the business.

Entire books are written on the subject of business process analysis and redesign. This activity summary is intended only to skim the surface of the topic.

Purpose Applicable only to business process redesign projects, the purpose of this activity is to analyze each business process in a set of related business processes to determine if the process is necessary and what problems might exist in that business process.

Roles The activity is facilitated by a *business process analyst*. Other roles are defined as follows:

- System owner roles.
  - User managers (REQ) of the business processes that will be analyzed.
- System user roles.
  - Appropriate business users (REQ)—those who are involved or familiar with the intricacies of the business processes.
- Systems analyst roles.
  - Business process analyst (REQ)—an individual who is skilled in the BPR techniques being applied. This individual should also be skilled in keeping the team emphasis on the process, not the people who perform the process.
- System designers are never involved. The focus is on analyzing the business, not the technology.
- System builders are never involved in this activity.

Prerequisites (Inputs) Once again, this activity is applicable only to business process redesign projects. In those projects, this activity is triggered by completion of the *system models* (REQ) from the previous activity. This activity is only interested in the process models. These process models are much more detailed than in other types of projects. They show every possible work flow path through the system, including error processing.

Deliverables (Outputs) The deliverables of this activity are *process analysis models* (REQ) and *process analysis data* (REQ). The process analysis models look very much like data flow diagrams (Figure 4.2) except they are significantly annotated to show: (1) the volume of data flowing through the processes, (2) the response times of each process, and (3) any delays or bottlenecks that occur in the system. The process analysis data provides additional information such as: (1) the cost of each process, (2) the value added by each process, and (3) the consequences of eliminating or streamlining the process. The combination of the models and analysis will be used to redesign the business processes in the systems design phases.

Applicable Techniques The following techniques are applicable to this activity. For each activity, we indicate which chapters or modules of the book teach that technique.

- *Process modeling.* Basic process modeling (REQ) is taught in Chapter 6; however, this book defers any process model analysis material to books that can provide more extensive coverage.
- *Process analysis.* Process analysis techniques are not covered in this book. BPR books are a better source of such education. They also tend to include the use of CASE technology specifically geared to business process redesign.

Steps The following steps are suggested to complete this activity.

1. (REQ) If necessary, refine process models to include all possible work flows and data flows that can occur in the business area under examination.
2. (REQ) For each primitive business process, analyze throughput and response time, as well as any average delays that may occur.
3. (REQ) For each primitive business process, analyze cost and value added. Identify candidates for elimination, consolidation, and optimization.

Activity: Analyze Problems and Opportunities

In addition to learning about the current system, the project team must work with system owners and system users to *analyze problems and opportunities.* The team may use the current system model(s) to serve as a framework that identifies those aspects of the current system to be analyzed. (Note: Some problems and opportunities may be apparent on the models, but many will not.)

You might be asking, Weren't problems and opportunities identified earlier in the survey phase? Yes, they were. But initial problems may be only symptoms of other problems, perhaps not well known or understood by the users. Besides, we haven't yet really analyzed any problems in the classical sense.

True problem analysis is a difficult skill for inexperienced systems analysts. Experience suggests that most new analysts (and many users and managers) try to solve problems without truly analyzing them. They might state a problem like this: "We need to . . ." or "We want to . . ." They are stating the problem in terms of a solution. More effective problem solvers have learned to state and analyze the problem, not the solution. They analyze each perceived problem for causes and effects. That cause-effect analysis provides true understanding that can lead to not-so-obvious solutions. And that analysis can also uncover other, more significant problems to be analyzed. Let's study the activity in greater detail.

Purpose The purpose of this activity is to: (1) understand the underlying causes and effects of all perceived problems and opportunities; and (2) understand the effects and potential side effects of all perceived opportunities. The analysis should be kept at a business (nontechnical) level during the entire analysis.

Roles The activity is facilitated by either the *project manager* or *systems analyst.* Other roles are defined as follows:

- System owner roles.
  - User managers (REQ)—the managers of the organizational units most likely to be supported by the system developed in this project.
  - Project manager (REC)—the information systems unit manager who will directly manage the project team. This is usually a senior systems analyst (e.g., Sandra will play this role in the SoundStage project).
- System user roles.
  - Business analysts (OPT).

  - Other user experts (REC) as necessary to fully analyze the problems and opportunities.
- Systems analyst roles.
  - Systems analyst (REQ)—specifically, systems analysts who are skilled at cause-effect analysis.
- System designer roles are not typically involved in this activity unless deemed appropriate by a system owner.
- System builder roles are not typically involved in this activity unless deemed appropriate by a system owner.

Prerequisites (Inputs) Like the modeling activity, this activity is triggered by completion of the survey phase activities and approval from the system owners to continue the project. In fact, this activity begins in parallel with the modeling activity. One key informational input is the *problem statement* (REC) that was completed as part of the survey phase. Other key informational inputs are *problems and opportunities* and *causes and effects*, which are collected from the business analysts and other system users.

Deliverables (Outputs) The principal deliverable of this activity is the aforementioned *cause-effect analysis*. Figure 4.12 illustrates a representative way to document cause-and-effect analysis.

Applicable Techniques The techniques applicable to this activity are almost identical to those of the modeling activity. For each activity, we indicate which chapters or modules of the book teach that technique.

FIGURE 4.12 *Cause-Effect Analysis*

PROBLEMS, OPPORTUNITIES, OBJECTIVES, AND CONSTRAINTS MATRIX

Project:	Member Services Information System	**Project Manager:**	Sandra Shepherd
Created by:	Robert Martinez	**Last Updated by:**	Robert Martinez
Date Created:	January 21, 1997	**Date Last Updated:**	January 31, 1997

CAUSE-AND-EFFECT ANALYSIS		SYSTEM IMPROVEMENT OBJECTIVES	
Problem or Opportunity	**Causes and Effects**	**System Objective**	**System Constraint**
1. Order response time is unacceptable.	1. Throughput has increased while number of order clerks was downsized. Time to process a single order has remained relatively constant. 2. System is too keyboard dependent. Many of the same values are keyed for most orders. Net result is (with the current system) each order takes longer to process than is ideal. 3. Data editing is performed by the AS/400. As that computer has approached its capacity, order edit responses have slowed. Because order clerks are trying to work faster to keep up with the volume, the number of errors has increased. 4. Warehouse picking tickets for orders were never designed to maximize the efficiency of order fillers. As warehouse operations grew, order filling delays were inevitable.	1. Decrease the time required to process a single order by 30%. 2. Eliminate keyboard data entry for as much as 50% of all orders. 3. For remaining orders, reduce as many keystrokes as possible by replacing keystrokes with point-and-click objects on the computer display screen. 4. Move data editing from a shared computer to the desktop. 5. Replace existing picking tickets with a paperless communication system between member services and the warehouse.	1. There will be no increase in the order processing workforce. 2. Any system developed must be compatible with the existing *Windows 95* desktop standard. 3. New system must be compatible with the already approved automatic identification system (for bar coding).

- *Fact-finding.* Fact-finding skills (REQ) are necessary to both identify and analyze the problems and opportunities. These skills are covered in Part Five, Module B, "Fact-Finding and Information Gathering."
- *Joint application development.* The preferred technique for rapid problem analysis is joint application development (JAD) (REC). The requisite analysis can usually be completed in one full-day session or less. The JAD facilitator must be especially skilled at conflict resolution because people tend to view problem analysis as personal criticism. JAD techniques and conflict resolution are explored in Part Five, Module D, "Joint Application Development."
- *Interpersonal skills.* This activity can easily generate controversy and conflict. Good interpersonal skills are necessary to maintain a focus on the problems and not the personalities. These skills are explored in Part Five, Module E, "Interpersonal Skills and Communications."
- *Cause-effect analysis.* Cause-effect analysis, when applied with discipline, can help the team avoid a premature concern with solutions. Cause-effect analysis is explored in Part Five, Module B, "Fact-Finding and Information Gathering."

Steps The following steps are suggested to complete this activity.

1. (REC) Review the problem statement completed in the survey phase.
2. (REQ) Collect facts and gather information about the perceived problems and opportunities in the current system. The preferred technique is JAD, but JAD sessions may be preceded or followed by traditional fact-finding and information gathering activity.
3. (REQ) Analyze and document each problem and opportunity. The PIECES framework (introduced in Chapter 3) is most useful for cause-effect analysis. As you collect facts, note problems and limitations according to the PIECES categories. Remember, a single problem may be recorded into more than one category of PIECES. Also, don't restrict yourself to only those problems and limitations noted by end-users. As the analyst, you may also identify potential problems! Next, for each problem, limitation, or opportunity, ask yourself the following questions and record answers to them.
   a. What is causing the problem? What situation has led to this problem? Understanding why is not as important. Many current systems were never designed; they simply evolved. It is usually pointless to dwell on history. In fact, you should be careful not to insult system owners and users who may have played a role in how things evolved.
   b. What are the negative effects of the problem or failure to exploit the opportunity? Learn to be specific. Don't just say, excessive costs. How excessive? You don't want to spend $20,000 to solve a $1,000 problem.
   c. The effect sometimes identifies another problem. If so, repeat steps 1 and 2.

Opportunities do not have causes. But they do have effects (and side effects) that will eventually have to be weighed against the cost of implementing the opportunity.

Activity: Establish System Improvement Objectives and Constraints

Given our understanding of the current system's scope, problems, and opportunities, we can now *establish system improvement objectives.* How can we determine the success of a systems development project? Success should be measured in terms of the degree to which objectives are met for the new system.

An **objective** is a measure of success. It is something that you expect to achieve, if given sufficient resources.

Objectives represent the first attempt to establish expectations for any new system.

In addition to objectives, we must also identify any known constraints.

A **constraint** is something that will limit your flexibility in defining a solution to your objectives. Essentially, constraints cannot be changed.

A deadline is an example of a constraint.

Given this overview, let's further study this important activity.

Purpose The purpose of this activity is to establish the criteria against which any improvements to the system will be measured and to identify any constraints that may limit flexibility in achieving those improvements.

Roles The activity is facilitated by the *project manager* or a *systems analyst*. Other roles are defined as follows:

- System owner roles.
  - User managers (REQ)—the managers of the organizational units most likely to be supported by the system developed in this project.
  - Project manager (REQ)—the information systems unit manager who will directly manage the project team. This is usually a senior systems analyst (e.g., Sandra will play this role in the SoundStage project).
- System user roles.
  - Business analysts (OPT).
  - Other user experts (REC) as necessary to fully analyze the problems and opportunities.
- Systems analyst roles.
  - Systems analysts (REC)—systems analysts who are skilled in the formulation and documentation of objectives and constraints.
- System designer roles are not typically involved in this activity unless deemed appropriate by a system owner.
- System builder roles are not typically involved in this activity unless deemed appropriate by a system owner.

Prerequisites (Inputs) This activity is triggered by the completion of the two previous activities. The inputs are the *system models* (REQ) and the *cause-effect analysis* (REQ). Together, they define the context for establishing objectives and constraints.

Deliverables (Outputs) The deliverable of this activity is *system improvement objectives and constraints*. This deliverable also corresponds to the net deliverable of the study phase, *system objectives,* as described in Chapter 3. Objectives should be precise, measurable statements of business performance that define the expectations for the new system. Some examples might include the following:

- Reduce the number of uncollectible customer accounts by 50 percent within the next year.
- Increase by 25 percent the number of loan applications that can be processed during an eight-hour shift.
- Decrease by 50 percent the time required to reschedule a production lot when a workstation malfunctions.

The following is an example of a poor objective:

Create a delinquent accounts report.

This is a poor objective because it only states a requirement, not an actual objective. Now, let's reword that objective:

Reduce credit losses by 20 percent through earlier identification of delinquent accounts.

This gives us more flexibility. Yes, the delinquent accounts report would work. But a customer delinquency inquiry might provide an even better way to achieve the same objective.

Objectives must be tempered by identifiable constraints. Constraints fall into four categories as listed below (with examples):

- *Schedule.* The new system must be operational by April 15.
- *Cost.* The new system cannot cost more than $350,000.
- *Technology.* The new system must be on-line, or all new systems must use the DB2 database management system.
- *Policy.* The new system must use double-declining balance inventory techniques.

The last two columns of Figure 4.12 document typical system improvement objectives and constraints.

Applicable Techniques The following techniques are applicable to this activity. For each activity, we indicate which chapters or modules of the book teach that technique.

- *Joint application development.* The preferred technique for this activity is joint application development (REC). The requisite brainstorming can usually be completed in a half-day session or less. JAD techniques and conflict resolution are explored in Part Five, Module D, "Joint Application Development."
- *Benefit analysis.* Whenever possible, objectives should be stated in terms that can be measured. Tangible and intangible benefit estimation techniques are surveyed in Part Five, Module C, "Feasibility and Cost/Benefit Analysis."
- *Interpersonal skills.* Like the previous activity, this activity can easily generate controversy and conflict. Good interpersonal skills are necessary to maintain a focus on what's best for the organization. These skills are explored in Part Five, Module E, "Interpersonal Skills and Communications."

Steps The following steps are suggested to complete this activity.

1. (REC) Review scope and problem analyses from the prior activities.
2. (REQ) Negotiate business-oriented objectives to solve each problem and exploit each opportunity. Ideally, each objective should establish the way you will measure the improvement over the current situation. Measures should be as tangible (measurable) as you can possibly make them.
3. (REQ) Brainstorm any constraints that may limit your ability to fully achieve objectives. Use the four categories previously listed in this section (time, cost, technology, and policy) to organize your discussion.

Activity: Modify Project Scope and Plan

Recall that project scope is a moving target. Based on our initial understanding and estimates from the survey phase, it may have grown or diminished in size and complexity. (Growth is much more common!) Now that we're approaching the completion of the study phase, we should reevaluate project scope and revise the project plan accordingly. The systems analyst and system owner are the key individuals in this activity.

The analyst and system owner will consider the possibility that not all objectives may be met by the new system. Why? The new system may be larger than expected, and they may have to reduce the scope to meet a deadline. In this case the system owner will rank the objectives in order of importance. Then, if the

scope must be reduced, the higher-priority objectives will tell the analyst what's most important.

Purpose The purpose of this activity is to reevaluate project scope, schedule, and expectations. The overall project plan is then adjusted as necessary, and a detailed plan is prepared for the next phase.

Roles The activity is facilitated by the *project manager*. Other roles are defined as follows:

- System owner roles (defined in the previous activity).
  - Executive sponsor (OPT)—It may be necessary to renegotiate scope, expectations, schedule, and budget with the executive sponsor.
  - User managers (OPT)—User managers should also be involved in any renegotiation of scope, expectations, and schedule.
  - System managers (OPT)—System managers commit information services resources to projects; therefore, they need to be made aware of any scope, schedule, or budget changes.
  - Project manager (REQ)
- System users are not typically involved in this activity unless deemed appropriate by the project manager.
- Systems analyst, system designer, and system builder roles are not typically involved in this activity unless deemed necessary by the project manager.

Prerequisites (Inputs) This activity is triggered by completion of the system modeling, problem analysis, and objective definition activities. The *system models* (REC), *cause-effect analysis* (REC), and *system improvement objectives and constraints* are inputs for the activity. The original *project plan* from the survey phase (REC, if available) is also an input.

Deliverables (Outputs) The principal deliverable of this activity is a revised *project plan* (REC). Many project managers use project management software to create and maintain their project plans. Additionally, a detailed *definition phase plan* (REC) may be produced.

Applicable Techniques The following techniques are applicable to this activity. For each activity, we indicate which chapters or modules of the book teach that technique.

- *Process management.* Process management (REC) (also called *methodology management*) defines the standards for applying the methodology to a project. Process management techniques are taught in Part Five, Module A, "Project and Process Management."
- *Project management.* Project management (REC) builds on process management by applying the methodology to specific projects in the form of schedule planning, staffing and supervision, progress reporting, management of expectations, budgeting, and schedule management. Project management techniques are taught in Part Five, Module A, "Project and Process Management."
- *Presentation skills.* The project charter and any verbal presentations of the project and plan obviously require presentation skills (REC). Presentation skills are taught in Part Five, Module E, "Interpersonal Skills and Communications."

Steps The following steps are suggested to complete this activity.

1. (REC) Review the original plan.

2. (REQ) Review the system models, problems and opportunities, cause-effect analyses, system improvement objectives, and scope. Ask yourself two questions:
   a. Has the scope of the project significantly expanded?
   b. Are the problems, opportunities, or objectives more difficult to solve than originally anticipated?
3. (REC) Estimate time required for each project activity in the next phase—the definition phase.
4. (REC) If necessary, refine baseline estimates for the overall project plan.
5. (OPT) If the answer is yes, renegotiate scope, schedule, and/or budget with the system owner group. (See Part Five, Module A, "Project and Process Management," for a simple expectations management instrument.)

(REC) Activity: Present Findings and Recommendations

As with the survey phase, the study phase findings and recommendations should be communicated to all affected personnel. The format may be a report, a verbal presentation, or an inspection by an auditor or peer group (called a walkthrough).

Purpose The purpose of this activity is to communicate the project and goals to all staff. The report or presentation, if developed, is a consolidation of the activities' documentation.

Roles Ideally, the activity should be facilitated by a *business analyst*. Alternatively, the activity could be facilitated by a systems analyst or the project manager. Other roles are defined as follows:

- System owner roles (defined in the previous activity).
  - Executive sponsor (OPT).
  - User managers (REC).
  - System managers (REC).
  - Project manager (REQ).
- System user roles.
  - Business analysts (REC).
  - All direct and indirect users (REC).
- Systems analysts.
  - Any systems analysts assigned to the project (REC).
- System designers are not typically involved in this activity.
- System builders are not typically involved in this activity.

Prerequisites (Inputs) This activity is triggered by the completion of the system objectives or project plan activity. The inputs include the *system models* (REC), the *cause-effect analysis* (REC), the *system improvement objectives and constraints*, and the *revised project plan* (REC) generated by the prior activities.

Deliverables (Outputs) The key deliverable of this activity is the *detailed study findings* (REC). It usually includes a feasibility update and the revised project plan. A representative outline for a written report is illustrated in Figure 4.13.

Applicable Techniques The following techniques are applicable to this activity. For each activity, we indicate which chapters or modules of the book teach that technique.

- *Interpersonal skills.* Once again, good interpersonal skills (REQ) are essential to this activity. These include persuasion, sales (of ideas), writing, and speaking. Applicable interpersonal skills are taught in Part Five, Module E, "Interpersonal Skills and Communications."

FIGURE 4.13 *Detailed Study Report Outline*

Analysis of the Current _____ System

I. Executive summary (approximately 2 pages)
   - A. Summary of recommendation
   - B. Summary of problems, opportunities, and directives
   - C. Brief statement of system improvement objectives
   - D. Brief explanation of report contents

II. Background information (approximately 2 pages)
   - A. List of interviews and facilitated group meetings conducted
   - B. List of other sources of information that were exploited
   - C. Description of analytical techniques used

III. Overview of the current system (approximately 5 pages)
   - A. Strategic implications (if the project is part of or impacts an existing information systems strategic plan)
   - B. Models of the current system
     1. Interface model (showing project scope)
     2. Data model (showing project scope)
     3. Geographic models (showing project scope)
     4. Process model (showing functional decomposition only)

IV. Analysis of the current system (approximately 5–10 pages)
   - A. Performance problems, opportunities, and cause-effect analysis
   - B. Information problems, opportunities, and cause-effect analysis
   - C. Economic problems, opportunities, and cause-effect analysis
   - D. Control problems, opportunities, and cause-effect analysis
   - E. Efficiency problems, opportunities, and cause-effect analysis
   - F. Service problems, opportunities, and cause-effect analysis

V. Detailed recommendations (approximately 5–10 pages)
   - A. System improvement objectives and priorities
   - B. Constraints
   - C. Project plan
     1. Scope reassessment and refinement
     2. Revised master plan
     3. Detailed plan for the definition phase

VI. Appendixes
   - A. Any detailed system models
   - B. (other documents as appropriate)

Steps The following steps are suggested to complete this activity.

1. (REQ) Review the deliverables of all prior activities.
2. (OPT) Write the detailed study findings.
3. (REQ) Present the findings to the system owners. Be prepared to defend recommendations, address issues and controversies, and answer questions. One of the following decisions must be made:
   - Authorize the project to continue, as is, to the definition phase.
   - Adjust the scope, cost, and/or schedule for the project and then continue to the definition phase.
   - Cancel the project due to either (1) lack of resources to further develop the system, (2) realization that the problems and opportunities are simply not as important as anticipated, or (3) realization that the benefits of the new system are not likely to exceed the costs.
4. (REC) Present the findings to all affected staff.

This concludes the study phase. With some level of approval of the system owners, the project can now proceed to the definition phase.

THE DEFINITION PHASE OF SYSTEMS ANALYSIS

Many analysts make a critical mistake after completing the study of the current information system. The temptation at that point is to begin looking at alternative solutions, particularly technical solutions. The most frequently cited error in new information systems is illustrated in the statement, "Sure the system works, and it is technically impressive, but it just doesn't do what we wanted (or needed) it to do."

Did you catch the key word? The key word is *what*. Analysts are frequently so preoccupied with the *technical* solution that they inadequately define the *business* requirements for that solution. The definition phase answers the question, *What does the user need and want from a new system?* The definition phase is critical to the success of any new information system! In different methodologies the definition phase might be called the *requirements analysis* or *logical design* phase.

Can you ever skip the definition phase? Absolutely not! New systems will always be evaluated, first and foremost, on whether or not they fulfilled business objectives and requirements regardless of how impressive or complex the technological solution might be!

Here again, your information systems building blocks (Figure 4.14) can serve as a useful framework for identifying what information systems requirements need to be defined. Notice that we are still concerned with the system users' perspectives. Also notice that for most of the building blocks, it is possible to draw graphical models to document the requirements for a new and improved system. This information represents the system users' view of the new application's requirements.

Figure 4.15 illustrates the typical activities of the definition phase. Once again, let's examine the activities conducted during this phase, the individuals' roles in each activity, the inputs and outputs, and techniques that are commonly used to complete each activity.

Activity: Outline Business Requirements

The initial activity of the definition phase is to *outline business requirements*. Effective writers outline their publications. This activity serves the same purpose. The foundation for this activity was established in the study phase when we identified system improvement objectives. This activity translates those objectives into inputs, outputs, and processes that will be specified in greater detail during subsequent activities. Rarely will this activity identify *all* the business requirements—no more than an initial outline for a written publication will include all headings and subheadings for a final publication. But the outline will frame your thinking as you proceed. Neither completeness nor perfection is a goal. The entire activity can be completed in as little as an hour.

Purpose The purpose of this activity is to identify, in general terms, the business requirements for a new or improved information system. A classic *input-process-output* framework should prove sufficient to structure the activity.

Roles The activity is facilitated by a business analyst or systems analyst. Other roles are defined as follows:

- System owner roles (most were defined in the previous activity).
  - User managers (REQ)—User managers played a key role in defining system improvement objectives. In this activity, they are especially helpful in identifying the outputs (information) to be produced by a system.
  - Project manager (OPT).
- System user roles.
  - Business analysts (REC)—Because business analysts are actually users on loan from the business community, they are well suited to the task of identifying high-level requirements, especially the inputs and processes required in the system.

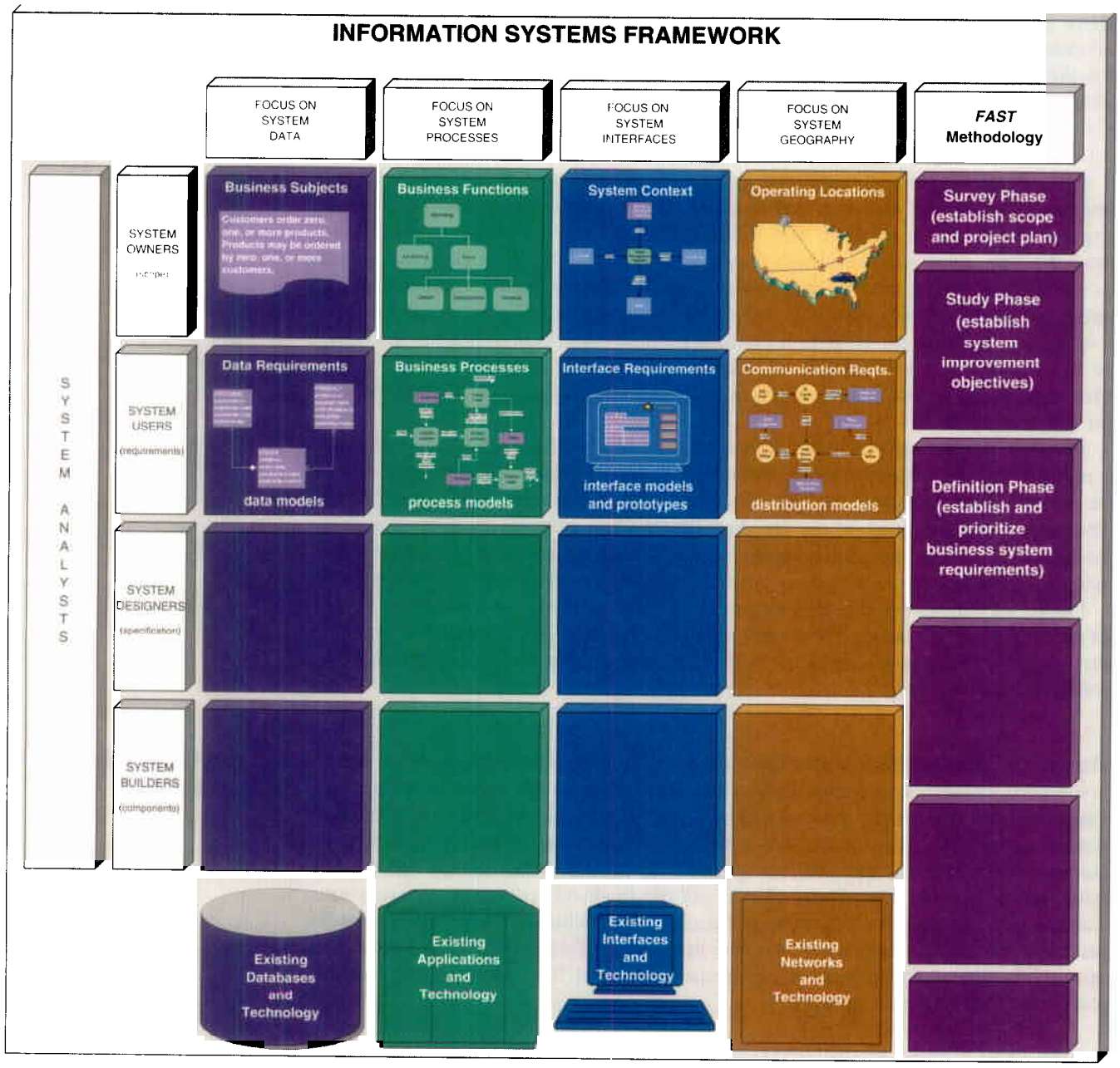


FIGURE 4.14 *Building Blocks for the Definition Phase*

- Appropriate direct and indirect users (OPP)—Be careful! If too many users are involved in this activity, the team can easily become overly preoccupied with details.

- Systems analysts.
  - System architects (REC)—We introduce this role here and define it to be systems analysts who are skilled in the modeling tools used to document and verify business system requirements. Although models are not a deliverable of this activity, this activity begins to capture the facts that will be used to construct models.
- System designers are not typically involved in this activity.
- System builders are not typically involved in this activity.

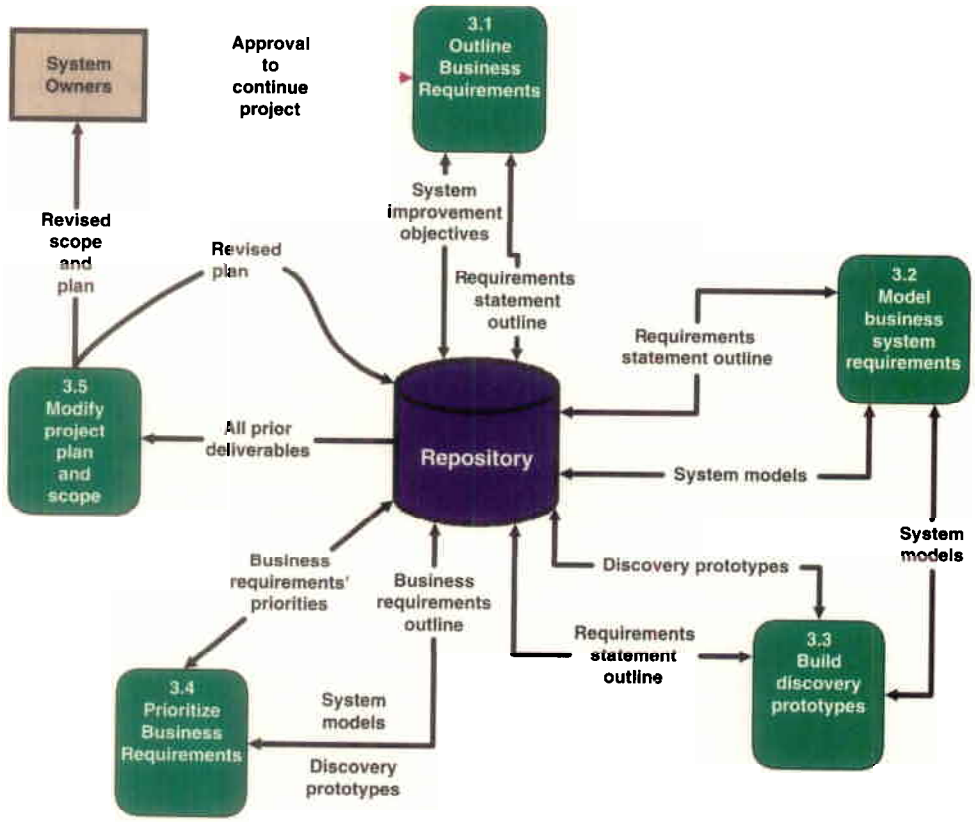


FIGURE 4.15
Activity Diagram for the Definition Phase

Prerequisites (Inputs) This activity is triggered by approval from the system owners to continue the project into the definition phase. The key input is the *system improvement objectives* (REQ) from the study phase. Of course, any and all relevant information from the study phase should be available for reference as needed.

Deliverables (Outputs) The only deliverable of this activity is a *requirements statement outline* (REQ). Various formats can work. In its simplest format, the outline could be divided into four logical sections: (1) the original list of objectives, (2) inputs, (3) processes, and (4) outputs. We prefer a slightly more complex format that associates inputs, processes, and outputs with each individual system improvement objective.

Applicable Techniques The following techniques are applicable to this activity. For each activity, we indicate which chapters or modules of the book teach that technique.

- *Joint application development.* The preferred technique for rapidly outlining business system requirements is joint application development (REC). The requisite analysis can usually be completed in less than one-half a working day. JAD techniques and conflict resolution are explored in Part Five, Module D, "Joint Application Development."
- *Interpersonal skills.* Good interpersonal skills are necessary to maintain a focus on the requirements. These skills are explored in Part Five, Module E, "Interpersonal Skills and Communications."

Steps The following steps are suggested to complete this activity.

1. (REQ) Review and refine the system improvement objectives.
2. (REQ) For each objective:
   a. Identify and document any business events or inputs to which the system must respond. Briefly define each event or input, but do not define the specific data content of any input.
   b. Identify and document any special business policies, processing, or decisions that must be made to adequately respond to each event or input.
   c. Identify and document the normal business outputs or responses to the aforementioned business events or inputs.
   d. Identify and document any information that must be produced or made available.
3. (REC) Compare the system improvement objectives and requirements against the original problem statements from the study phase. Are you still solving the original problems or is the scope of the project growing? Increased scope is not necessarily wrong; however, an appropriate adjustment of expectations (particularly schedule and budget) may eventually become necessary.

Activity: Model Business System Requirements

Identifying requirements is only the tip of the iceberg. We must also find a way to express the requirements such that they can be verified and communicated to both business and technical audiences. Technicians must understand requirements so they can transform them into appropriate technical solutions. Business users must understand requirements so they can prioritize the needs and justify the expenditures for any technical solution. Best current practice suggests that we *model business system requirements*. Recall that system models are pictures (similar to flowcharts) that express requirements or designs.

The best systems analysts can develop models that provide no hint of how the system will or might be implemented. This is called logical or essential system modeling.

> **Logical models** depict what a system is or what a system must do—*not* how the system will be implemented. Because logical models depict the *essence* of the system, they are sometimes called **essential models.**

Logical models express business requirements—sometimes referred to as the **logical design**—as opposed to the technical solution, which is called the *physical design*. In theory, by focusing on the logical design of the system, the project team will (1) appropriately separate business concerns from technical solutions, (2) be more likely to conceive and consider new and different ways to improve business processes, and (3) be more likely to consider different, alternative technical solutions (when the time comes for physical design).

Purpose The purpose of this activity is to model business system requirements such that they can be verified by system users and subsequently understood and transformed by system designers into a technical solution. In a sense, the system models bridge the communication gap that inevitably exists between business and technical staffs.

Roles The activity is facilitated by a *systems analyst*. Other roles are defined as follows:

- System owner roles.
  - User managers (REC).
  - Project manager (REC).
- System user roles.
  - Business analysts (REC).
  - Appropriate direct and indirect users (REQ).

- Systems analysts.
- System architects (REQ).
- System designers are not recommended since they tend to talk in technical terms that intimidate and frustrate the users and user managers.
- System builders are not typically involved in this activity. On the other hand, programmers who are skilled in user interface construction might be invited to observe the activity as a preface to constructing rapid prototypes of user interfaces for later activities.

Prerequisites (Inputs) This activity is usually triggered by completion of the *requirements statement outline* (REC); however, it often begins as part of the same group meeting. Because system modeling is a mature technique, there may already exist many system models or templates in the corporate repository. These models may be very general or very detailed, depending on their source.

It is also possible to *reverse engineer* some system models directly from existing databases and program libraries. The resulting models tend to be very physical, meaning technology-oriented. The systems analyst would need to translate those physical system models into logical system models.

Deliverables (Outputs) The deliverables of this activity are the *system models* (REQ). Your information systems framework identifies the need for four system models:

- DATA—All systems capture and store data. **Data models** (such as the entity relationship diagram shown earlier in Figure 4.4) are used to model the data requirements for many new systems. These data models eventually serve as the starting point for designing databases.
- PROCESSES—All systems perform work. **Process models** (such as the data flow diagram shown earlier in Figure 4.2) are frequently used to model the work flow through business systems. These process models serve as a starting point for designing computer applications and programs.
- INTERFACES—No system exists in isolation from either people, other systems, or other organizations. Actually, **interface models** such as context diagrams (Figure 4.16) are drawn in the study phase, but they might be expanded to show more detail in this definition phase activity. Context diagrams depict net inputs to the system, their sources, net outputs from the system, their destinations, and shared databases. These interface models serve as the basis for designing user and system interfaces.
- GEOGRAPHY—Because today's business and information systems have greater geographic breadth, systems analysts are exploring ways to model geographical locations. These **distribution models** serve as a starting point for designing the communication systems for distributing the data, processes, and interfaces to the various geographical locations. We'll defer any samples of a distribution model to Chapter 7.

As suggested earlier in this chapter, many systems analysts are now experimenting with **object models** as an alternative to data and process models.

Most systems analysts use computer-aided systems engineering (CASE) software to construct and maintain system models and their underlying documentation. Most of the system models illustrated in this book were developed with a CASE tool called *System Architect*. An inexpensive, student edition of this CASE tool is available from Irwin along with a tutorial workbook. Most CASE tools support system modeling; therefore, gaining experience with a tool such as *System Architect* is easily transferable to organizations that use any other CASE tool.

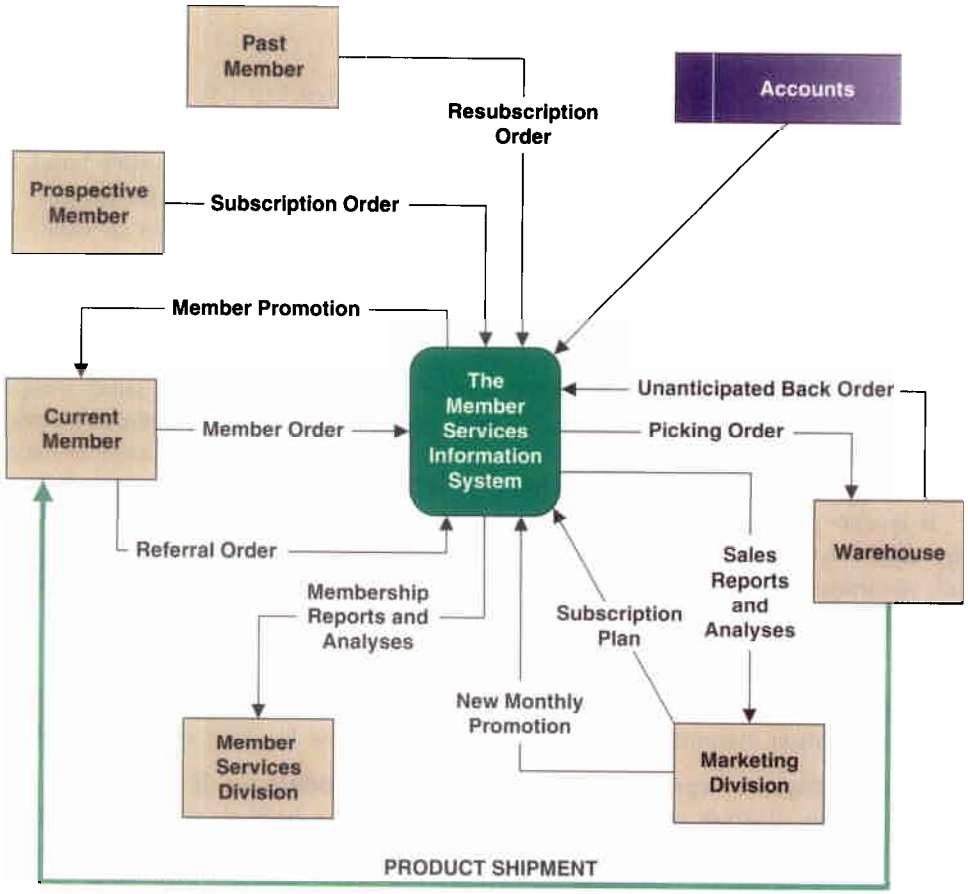


FIGURE 4.16
Interface Model (also Called a Context Diagram)

Applicable Techniques The following techniques are applicable to this activity. For each activity, we indicate which chapters or modules of the book teach that technique.

- *Data modeling*. Data modeling is the most popular technique for expressing the business requirements for data that will be stored in a system's database. This technique is extensively covered in Chapter 5, "Data Modeling."
- *Process modeling*. Arguably, process modeling is the oldest and most widely practiced technique for expressing business process requirements, work flow, inputs, and outputs. This technique is taught in Chapter 6, "Process Modeling." Chapter 6 also teaches you how to draw the interface model or context diagram.
- *Distribution modeling*. Distribution modeling is the least mature of the system modeling techniques covered in this textbook; however, its importance is growing as organizations seek ways to express the business geography to be supported by a system. Chapter 7 teaches distribution modeling.
- *Object modeling*. Object modeling has been in development for many years; however, it is only now emerging as a successor to traditional data and process modeling. Object modeling is being driven by the growing use of object technology and object-oriented analysis methods to exploit that technology. Chapter 8 teaches object modeling using the latest object modeling standards.
- *Fact-finding*. You can't build models without facts. These techniques are taught in Part Five, Module B, "Fact-Finding and Information Gathering."

- *Joint application development.* JAD has become the most popular technique for quickly constructing system models in direct cooperation with system owners and system users. JAD techniques merge the model construction and verification into the same meetings to accelerate the project. JAD is taught in Part Five, Module D, "Joint Application Development."

Steps The following steps are suggested to complete this activity.

1. (REC) Review the system improvement objectives and requirements statement outline.
2. (REQ) Collect or retrieve any system models that may have been developed in prior projects. High-level system models may have been created as part of an information strategy planning project or business process redesign project. Detailed models may have been created as part of prior application development projects. In either case, existing models are typically stored in the corporate repository. Many organizations have formal checkout/check-in procedures for using and updating existing system models.
3. (OPT) If the appropriate CASE technology is available, consider reverse engineering existing databases or applications into physical system models. Then translate those physical models into more business-friendly logical system models. The value of this step depends on the quality and value of the databases and applications to be reverse engineered. Many systems are so old or poorly designed that the value of reverse engineering is questionable.
4. (REQ) Draw the interface model. The interface model establishes the scope and boundary for the entire project.
5. (REQ) Depending on your modeling strategy of choice:
   a. If you practice *structured analysis:*
      i. (REQ) Construct and verify the process models.
      ii. (REC) Construct and verify data models.
      iii. (REC) Synchronize process and data models. This synchronization ensures that the models are consistent and compatible with one another.
      iv. (OPT) Construct and verify distribution models.
   b. If you practice *information engineering* (REC):
      i. (REQ) Construct and verify data models.
      ii. (REQ) Construct, verify, and synchronize the process models.
      iii. (REC) Construct and verify the distribution models.
   c. If you practice *object-oriented analysis* (the leading edge):
      i. (REC) Identify use cases. Use cases are an object method that connects objects to familiar business events. Use cases are taught in the object modeling chapter.
      ii. (REQ) Construct and verify object models. Several popular object model standards exist.

[OPT] Activity: Build Discovery Prototypes

An alternative or complementary approach to system modeling of business requirements is to *build discovery prototypes*. You may recall the following definitions:

Prototyping is the act of building a small-scale, representative, or working model of the users' requirements to discover or verify the users' requirements.

Prototyping is typically used in the requirements definition phase to establish user interface requirements (as was shown in Figure 4.14, INTERFACE column, SYSTEM USER row). But these prototypes also help analysts and programmers identify detailed business requirements that show up on the screens. For this reason, user interface prototypes are often called discovery prototypes.

Discovery prototypes are simple mock-ups of screens and reports that are intended to help systems analysts discover requirements. The discovered requirements would normally be added to system models. A synonym is requirements prototypes.

Although discovery prototyping is optional, it is frequently applied to systems development projects, especially in those cases when the parties involved are having difficulty developing or completing system models. The philosophy is that the users will recognize their requirements when they see them.

Prototypes are developed using **fourth-generation languages** (4GLs), most of which include **rapid application development** (RAD) facilities for quickly "painting" screens, forms, and reports. Examples of such languages include *Powerbuilder* and *Focus;* however, most third-generation languages are evolving to include the same RAD facilities (e.g., *Visual BASIC* and *Delphi,* which is based on Pascal). Furthermore, most PC-based database management systems now include rapid application development facilities that can be used to prototype inputs and outputs (e.g., *Access, Visual dBASE,* and *Paradox*).

Purpose The purpose of this optional activity is to (1) establish user interface requirements, and (2) discover detailed data and processing requirements interactively with users through the rapid development of sample inputs and outputs.

Roles The activity is facilitated by the *business analyst* or *systems analyst*. Other roles are defined as follows:

- System owners usually do not elect to participate unless they are also system users.
- System user roles.
  - Business analyst (REC)—The first time, a phrase is added to define the role. After that, role name only unless some special explanation is required.
  - Direct system users.
- Systems analyst roles—Systems analysts facilitate, observe, and assist this activity. It should be recognized that many systems analysts have the skills necessary to play the system designer and builder roles described below.
- System designer roles.
  - User interface specialist (OPT)—If prototypes are to evolve into the final system, a user interface specialist is suggested. User interface specialists are system designers who have become skilled in the standards or guidelines of a particular graphical user interface (such as *MacOS* or *Windows 95*). They are also responsible for local interface standards that ensure that all internally developed applications exhibit the same look and feel.
- System builder roles.
  - Prototyper (REQ)—This is a systems analyst or programmer who is skilled in the use of rapid application development (RAD) technology for the purpose of developing discovery prototypes.
  - Programmer (OPT)—It is not the purpose to write complete application programs at this stage of the project; however, the current trend is toward using the same prototyping languages to rapidly complete the development of the applications. Consequently, programmers may get useful insights from observing this activity.

Prerequisites (Inputs) This activity is not triggered by any event. It occurs, as deemed appropriate or desirable, during the system modeling activity. Thus, it occurs in parallel with system modeling. It uses the *system requirements outline* (REC). It can also use any *system models* (REC) as they are developed.

Deliverables (Outputs) The deliverables of this activity are *discovery prototypes* (REQ) of selected inputs and outputs. These prototypes are shared with the system modeling activity since they can be very helpful in refining the system models. The models are also stored in the repository since they can eventually evolve into the final production system.

Applicable Techniques The following techniques are applicable to this activity. For each activity, we indicate which chapters or modules of the book teach that technique.

- *Prototyping*. Prototyping (REQ) is predominantly considered to be a design technique because it is based on design and construction of actual program components. For that reason, we teach it in the system design unit of this textbook (Chapters 12–14).
- *Technology*. It is not the intent of this book to teach specific technologies. There are many RAD technologies that are well suited to prototyping. Clearly, the actual use of prototyping will require an investment in learning the technology to be used. Fortunately, most of today's visual programming languages are easy to learn and use as prototyping tools. (It will take a much greater knowledge of these languages to complete the application's development beyond the prototype.)

Steps The following steps are suggested to complete this activity.

1. (REC) Review the system improvement objectives and requirements statement outline.
2. (REC) Study any system models that may have been developed.
3. (OPT) Working directly with system users, construct a simple, single-user prototype of the database and load it with some sample data. Do not become preoccupied with data editing and perfection.
4. (REQ) Working directly with the system users, construct input prototypes for each business event. Do not worry about input editing, system security, etc.; the focus is completely on business requirements. Do not spend too much time on any one input since this stage does not develop the final system.
5. (REQ) Working directly with system users, construct output prototypes for each business output. Do not worry about whether the data are real or whether or not they make sense. Focus on identifying the columns, totals, and graphs the users are seeking. If you built a sample database in step 3 and used step 4 to collect data for that database, you can probably use that database prototype to quickly generate sample reports.
6. (REC) Return to the system modeling activity to formalize the requirements that have been discovered through the above prototyping steps.

Activity: Prioritize Business Requirements

We stated earlier that the success of a systems development project can be measured in terms of the degree to which business requirements are met. But actually, all requirements are not equal. If a project gets behind schedule or over budget, it may be useful to recognize which requirements are more important than others. Systems can be built in versions. Early versions should deliver the most important requirements.

Thus, given the proposed system requirements, models, and prototypes defined earlier, the system owner and systems analyst should *prioritize business requirements*. The actual priorities should be specified jointly by the system owners and users. Prioritization of business requirements also enables a popular technique called timeboxing.

Timeboxing is a technique that develops larger fully functional systems in versions. The development team selects the smallest subset of the system that, if fully implemented, will return immediate value to the system owners and users. That subset is developed, ideally with a time frame of six to nine months or less. Subsequently, value-added versions of the system are developed in similar time frames.

Timeboxing requires that the priorities be clearly understood.

Purpose The purpose of this activity is to prioritize business requirements for a new system. It is not the intent of this activity to eliminate any business requirement—only to recognize that some requirements can be implemented later than others.

Roles The activity is facilitated by the *business analyst* or *project manager*. Other roles are defined as follows:

- System owner roles.
  - Executive sponsor (OPT)—Most executive sponsors will empower their user managers (below) to make these decisions.
  - User managers (REQ)—The reason should be obvious. If management strongly disagrees with priorities, the project could lose support.
  - Project manager (REC).
- System user roles.
  - Business analyst (REC)—A great choice for facilitator of this activity since business analysts come from the user community and have the trust of that community.
  - Appropriate direct and indirect system users (REQ).
- Good systems analysts listen to discussion and answer questions during this activity. User buy-in to priorities is critical to the political feasibility of any new system if a systems analyst or project manager facilitates this activity.
- System designers are not typically involved in this activity because they tend to influence priorities for technical, nonbusiness reasons.
- System builders are not typically involved in this activity.

Prerequisites (Inputs) This activity can begin in parallel with the other definition phase activities. The only inputs are business requirements as expressed in the updated *business requirements outline* (REQ), *system models* (REC), and *discovery prototypes* (OPT).

Deliverables (Outputs) While some priorities can be set as requirements are established, the priorities can never be finalized until all the business requirements have been identified. The deliverables of this activity are *business requirements' priorities* (REQ) as recorded in the repository.

Applicable Techniques There are no special techniques for prioritizing requirements. A simple but effective technique is described as follows:

- Is the business requirement mandatory? In other words, is it something the system owner must have? Careful! There is a temptation to label too many requirements as mandatory. By definition, if a system does not include a mandatory requirement, that system cannot fulfill its purpose. Perform the following test on any suspected mandatory requirements: Rank them. If you can rank them, they are not mandatory. You should not be able to rank requirements that you absolutely must have. All mandatory requirements are essential to the first version of the system!

- Is the business requirement desirable but not essential? Desirable requirements are things the user eventually wants; however, the early versions of the system can provide value without them. Unlike mandatory requirements, desirable requirements can and should be ranked.
- Is the business requirement optional? This is a catchall category for those features and capabilities that you could live without indefinitely. Although these would be nice to have, they are not really requirements. These requirements can also be ranked.

The priorities resulting from this analysis will allow the system owner and systems analyst to make intelligent decisions if cost and schedule become constrained.

Steps The following steps are suggested to complete this activity.

1. (REQ) For each system input and output, categorize it as mandatory, optional, or desirable.
2. (REQ) For each desirable requirement above, rank it with respect to the other desirable requirements. Make note of any dependencies that exist between requirements (e.g., requirement 5 depends on the implementation of requirement 2).
3. (REQ) For each optional requirement, rank it with respect to the other optional requirements. Make note of any dependencies that exist between requirements.
4. (OPT) Define system versions. A recommended scheme follows:
   a. (REC) Version one consists of all mandatory requirements.
   b. (REC) Versions two through x consist of logical groupings of desirable requirements.
   c. (REC) Optional requirements are usually added to versions as time permits or deferred to maintenance releases of the system. Many such requirements are for new reports. Today, users can be given relatively simple technology (e.g., query tools and report generators) to fulfill many such requirements on their own.

Activity: Modify the Project Plan and Scope

Here again, recall that project scope is a moving target. Now that we've identified business system requirements, we should step back and redefine our understanding of the project scope and adapt our project plan accordingly. The team must consider the possibility that the new system may be larger than originally expected. If so, they must adjust the schedule, budget, or scope accordingly.

Purpose The purpose of this activity is to (1) modify the project plan to reflect changes in scope that have become apparent during requirements definition, and (2) secure approval to continue the project into the next phase. (Work may have already started on the configuration or design phases; however, the decisions still require review.)

Roles The activity is facilitated by the *project manager*. Other roles are defined as follows:

- System owner roles.
  - Executive sponsor (REQ)—As the final spending authority, the sponsor must approve project continuation.
  - User managers (REQ)—The system belongs to these managers; therefore, their input is crucial.
  - Project manager (REQ)—The project manager must make any changes to the schedule and budget.

- System user roles.
  - Business analysts (OPT).
- Other systems analysts are not usually involved in this activity.
- System designer roles.
  - Database administrator (REC)—Given the data model, the database administrator will reassess time to design and implement the database.
  - Network administrator (REC)—Given the geographic model, the network administrator will reassess the time to design, expand, or implement the network.
  - Application administrator (REC)—Given the process and data, or object models, the application administrator will reassess the time to design, construct, and implement the application programs.
- System builders are not involved in this activity.

Prerequisites (Inputs) This activity is triggered by initial completion of the *system models* (REQ), *discovery prototypes* (OPT), and the *business requirements priorities* (REC).

Deliverables (Outputs) The deliverable of this activity is a revised *project plan* (REC) that covers the remainder of the project. Additionally, a detailed *configuration plan* (OPT) and *design plan* (OPT) could be produced.

Applicable Techniques The following techniques are applicable to this activity. For each activity, we indicate which chapters or modules of the book teach that technique.

- *Process management.* Process management (REC) (also called methodology management) defines the standards for applying the methodology to a project. Process management techniques are taught in Part Five, Module A, "Project and Process Management."
- *Project management.* Project management (REC) builds on process management by applying the methodology to specific projects in the form of schedule planning, staffing and supervision, progress reporting, management of expectations, budgeting, and schedule management. Project management techniques are taught in Part Five, Module A, "Project and Process Management."
- *Presentation skills.* The project charter and any verbal presentations of the project and plan obviously require presentation skills (REC). Presentation skills are taught in Part Five, Module E, "Interpersonal Skills and Communications."

Steps The following steps are suggested to complete this activity.

1. (REC) Review the original plan.
2. (REQ) Review the up-to-date business requirements outline, system models, discovery prototypes, and business requirements' priorities. Ask yourself two questions:
   a. Has the scope of the project significantly expanded?
   b. Are the requirements more substantial than originally anticipated?
3. (REC) Estimate the time required for each project activity in the next phase—the design phase.
4. (REC) If necessary, refine baseline estimates for the overall project plan.
5. (OPT) If the answer is yes, then renegotiate scope, schedule, and/or budget with the system owner group. (See Part Five, Module A, "Project and Process Management," for a simple expectations management instrument.)

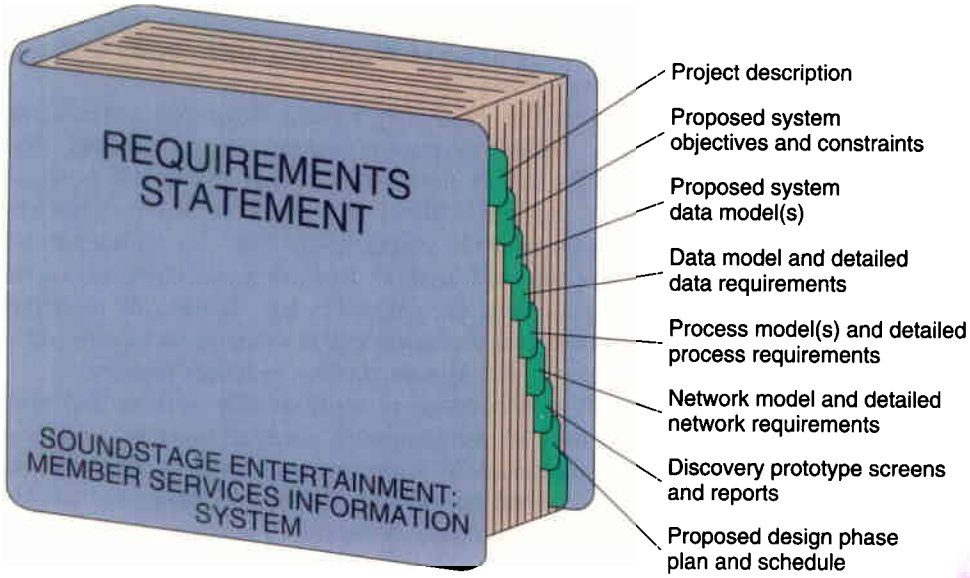


FIGURE 4.17
A Hard Copy Requirements Statement

Some Final Words about System Requirements

A consolidation of all system models, discovery prototypes, and supporting documentation is sometimes called a **requirements statement**. All elements of the requirements statement are stored in the repository, but most systems analysts find it useful to keep a printed copy of that documentation for reference and reporting. This hard-copy requirements statement might be organized as shown in Figure 4.17. It is not recommended that typical users be exposed to this rather imposing document. The contents of the requirements statement are best presented to users in subsets (for example, the data models and associated documentation).

Finally, the definition phase is now complete—or is it? It was once popular to freeze the business requirements before beginning the system design and construction phases. But today's economy has become increasingly fast paced. Businesses are measured on their ability to adapt to rapidly changing requirements and opportunities. Information systems can be no less responsive. *FAST* does not freeze the requirements statement. Nor does the methodology call for a gala presentation of requirements to the business community. Such an event might be construed to impart a false sense of completeness or perfection. The definition phase does not end with a bang. It quietly and efficiently transitions into the configuration and design phases. As new business requirements come up, they will be acknowledged and documented in the system models and repository. But they will also be carefully assessed to determine their impact on scope, schedule, and budget. System owners, when promptly supplied with new information, will ultimately decide whether the new requirements are mandatory, desirable, or optional and then make the decision on their inclusion.

THE NEXT GENERATION OF REQUIREMENTS ANALYSIS

Predicting the future of requirements analysis is not easy, but we'll make an attempt. CASE technology will continue to improve making it easier to model system requirements. Two CASE technologies will lead the charge. First, CASE tools will include object modeling to support emerging object-oriented analysis techniques. Some CASE tools will be purely object-oriented. but we believe that the demand for other types of modeling support (data modeling for databases and process modeling for business processes and work flow), and the need to integrate these models with object models, will place a premium on comprehensive CASE tools that can support many types of models. Second, the reverse engineering technology in CASE tools will improve our ability to more quickly generate first-draft system models from existing databases and application programs.

CASE technology and RAD technology will continue to complement one another. Indeed, we expect the tools to interoperate in both directions to simplify both system modeling and discovery prototyping.

Object-oriented analysis is poised to eventually replace structured analysis and information engineering as the method of choice among systems analysts. This change will not be as rapid as the object purists would like, but it will occur all too rapidly for a generation of systems analysts who were skilled in the older methods. There is a grand opportunity for young talent who are well educated and skilled in the use of object-oriented analysis (and design); however, career opportunities will remain strong for database specialists who know data modeling and information engineering. Also, process modeling is enjoying something of a renaissance thanks to the popularity of business process redesign projects.

One thing will *not* change! We will continue to need systems analysts and business analysts who understand how to fundamentally analyze business problems and define logical business requirements as a preface to system design. But we will all have to do that with increased speed and accuracy to meet the accelerated systems development schedules required in tomorrow's faster-paced economy.

WHERE DO YOU GO FROM HERE?

This chapter provided a detailed overview of the systems analysis phases of a project. You are now ready to learn some of the systems skills introduced in this chapter. Because system modeling is the most popular systems analysis technique, we recommend you complete Chapters 5 to 8. Each of these chapters teaches a different system modeling technique that you can immediately apply to projects.

The order of the system modeling chapters is flexible; however, we do recommend Chapter 5, "Data Modeling," first. All applications have databases, and data modeling is an essential skill for database development. Also, it is easier to synchronize early data models with later process models than vice versa.

This would also be an appropriate time to develop some of the soft skills of systems analysis. For example, models cannot be constructed without business facts and ideas. Part Five contains two modules that can help you learn about getting facts and ideas from system owners and users. Specifically, Module B, "Fact-Finding and Information Gathering," presents the traditional ways to collect facts and ideas. Module D, "Joint Application Development," presents a more contemporary approach based on facilitated group meetings. If your course requires a live project engagement with real managers and users, one or both modules will prove invaluable to your later system modeling efforts.

For those of you who have completed a systems analysis course already, this chapter was a review. For you, we suggest you merely review the system modeling chapters and proceed to Chapter 9, "Systems Design."

SUMMARY

1. Formally, systems analysis is the dissection of a system into its component pieces. As a problem-solving phase, it precedes systems synthesis or systems design. With respect to information systems development, systems analysis is the survey and planning of a project, the study and analysis of the existing system, and the definition of business requirements for the new system.
2. The results of systems analysis are stored in a repository for use in later phases and projects.
3. There are several popular or emerging strategies for systems analysis. These techniques can be used in combination with one another.
   a. Modern structured analysis, a technique that focuses on processes.
   b. Information engineering, a technique that focuses on data and strategic planning.
   c. Prototyping, a technique that focuses on building small-scale prototypes of solutions.
   d. Joint application development, a technique that focuses on facilitated group meetings with both technicians and users.
   e. Business process redesign, a technique that focuses on simplifying and streamlining fundamental business processes before applying information technology to those processes.
   f. Object-oriented analysis, a technique that integrates the concerns of data and processes to create objects that can be easily adapted and reused.
4. Each phase of systems analysis (survey, study, and definition) can be understood in the context of the information system building blocks: DATA, PROCESSES, INTERFACES, and GEOGRAPHY.
5. The purpose of the survey phase is to determine the worthiness of the project and to create a plan to complete those projects deemed worthy. To accomplish the survey phase objectives, the systems analyst will work with the system owner, system users, IS manager, and other IS staff to: *(a)* survey problems, opportunities, and solutions; *(b)* negotiate project scope; *(c)* plan the project; and *(d)* present the project. The deliverable for the survey phase is an oral or written project feasibility assessment that must go to a decision-making body, commonly referred to as the steering committee. A steering committee is a decision-making body that prioritizes potential information systems projects. Not all projects require this evaluation. For example, projects initiated by formal systems planning have already been justified by the planning team and management. In such cases the project skips directly to the study phase.
6. The purpose of the study phase is to answer the questions: Are the problems really worth solving? and Is a new system really worth building? To answer these questions, the study phase thoroughly analyzes the alleged problems and opportunities first identified in the survey phase. To complete the study phase, the analyst will continue to work with the system owner, system users, and other IS management and staff. The systems analyst and appropriate participants will: *(a)* optionally, model the current system; *(b)* optionally, analyze business processes; *(c)* analyze problems and opportunities; *(d)* establish system improvement objectives and constraints; *(e)* modify project scope and the plan; and *(f)* present the findings and recommendations.
7. The purpose of the definition phase is to identify what the new system is to do without the consideration of technology; in other words, to define the business requirements for a new system. As in the survey and study phases, the analyst actively works with system users and owners as well as other IS professionals to complete the definition phase. To complete the definition phase, the analyst and appropriate participants will: *(a)* outline business requirements; *(b)* model the business system requirements; *(c)* optionally, build discovery prototypes; *(d)* prioritize the business system requirements; and *(e)* modify the project plan and scope.
8. System models are a key deliverable in systems analysis. The project team typically builds:
   a. Data models for the business database requirements.
   b. Process models or object models for the business process requirements to be programmed.
   c. Interface models to demonstrate how the system should interact with both system users and other systems (including external organizations).
   d. Distribution models for the business geography to be supported by a network.

KEY TERMS

activity diagram, p. 129
business area analysis, p. 124
business process redesign, p. 126
constraint, p. 146
data model, p. 155
discovery prototypes, p. 158
distribution model, p. 155
essential model, p. 154
feasibility prototyping, p. 125
fourth-generation language, p. 158
information engineering, p. 123
information strategy planning, p. 123
interface model, p. 155
joint application development, p. 125
logical design, p. 121
logical model, p. 154
model, p. 122
model-driven development, p. 122
modern structured analysis, p. 122
object model, p. 155
object-oriented analysis, p. 126

objective, p. 145
process model, p. 155
prototyping, pp. 125, 157
rapid application development, p. 125
repository, p. 121
requirements prototyping, p. 125
requirements statement, p. 163
steering body, p. 135
systems analysis (classical), p. 120
systems analysis (contemporary), p. 121
systems synthesis, p. 120
timeboxing, p. 160

REVIEW QUESTIONS

1. What is the difference between systems analysis and systems synthesis?
2. What is the difference between systems analysis and logical design?
3. What role does a repository play in systems analysis?
4. What is the difference between modern structured analysis and information engineering? Which approach is most popular at the time of this book's publication?
5. What is model-driven development?
6. What is prototyping? Is it an alternative to model-driven development? Why or why not?
7. What is business process redesign?
8. What is object-oriented analsysis? How is it similar to, and different from, modern structured analysis and information engineering?
9. Identify and briefly describe the purpose of the three systems analysis phases.
10. What important question is addressed during the survey phase? What are the fundamental objectives of the survey phase? How might the information systems building blocks be used to identify the general level of understanding required for fulfilling these objectives?
11. What important question is addressed during the definition phase? What are the fundamental objectives of the definition phase? How might the information systems building blocks be used to identify what requirements must be specified to fulfill these objectives?
12. What is the end product of each systems analysis phase? Explain the purpose and content of each of these products.

PROBLEMS AND EXERCISES

1. How do the classical definitions of systems analysis and systems synthesis relate to contemporary systems analysis and design?
2. What triggers the survey phase? Do all systems projects go through a survey phase? Explain your answer.
3. Differentiate between the survey and study phases of the systems analysis process.
4. What are the fundamental objectives of the study phase? Explain how the information systems building blocks aid in identifying the required level of understanding to be obtained in the study phase.
5. What is the value of modeling during the systems analysis phases? What types of models might be developed during the survey, study, and definition phases?
6. Explain how the joint application development (JAD) technique can be used during the definition phase. Explain why prototypes and system models should complement one another.

PROJECTS AND RESEARCH

1. How might the PIECES framework introduced in Chapter 3 be used in the study phase of systems analysis? Use the PIECES framework to evaluate your local course registration system. Do you see problems or opportunities? (Alternative: Substitute any system with which you are familiar.)
2. Interview a systems analyst or systems manager in the information services unit of a local organization. How do they initiate projects? What type of report or document do they produce to justify a project in the early stages of development?
3. Interview a systems analyst in a local organization. What techniques do they use to ensure that the problems stated by their users are worthy of solution? Do they quantify that worth? If so, how? If not, why not?
4. Interview a systems manager in a local organization. What strategy do they use to verify and document business requirements for a system? Which of the strategies described in this book most closely correspond to their strategies? How do they teach these strategies to their analysts? How do their users react to these strategies? How do they enforce the use of their strategies? What new