CHAPTER SIX OUTLINE

STUDENT LEARNING OUTCOMES

1. Define the traditional systems development life cycle (SDLC) and describe the seven major phases within it.
2. Compare and contrast the various component-based development methodologies.
3. Describe the selfsourcing process as an alternative to the traditional systems development life cycle.
4. Discuss the importance of prototypes and prototyping within any systems development methodology.
5. Describe the outsourcing environment and how outsourcing works.

PERSPECTIVES

WEB SUPPORT

www.mhhe.com/haag

- Best in computer resources and statistics
- Meta data
- Finding hosting services
- Searching for freeware and shareware
- Researching storefront software

SUPPORTING MODULES

XLM/F Building a Web Page with HTML
Extended Learning Module F provides hands-on instructions for building a Web page by writing the HTML (hypertext markup language) code. You'll learn how to work with headings; adjust text sizes, fonts, and colors; manipulate background colors and images; insert links to documents, other Web pages, and e-mail addresses; manipulate images; and insert both bulleted and numbered lists.

XLM/L Building Web Sites with FrontPage
Extended Learning Module L provides hands-on instructions for building a Web site using Microsoft's Web authoring software FrontPage. Like Extended Learning Module F, you'll learn how to incorporate such things as list, images, and links. You'll also learn much more in this module as FrontPage truly enhances your ability to create a Web page that takes advantage of many of today's exciting Web features.

XLM/M Programming in Excel with VBA
Extended Learning Module M covers the basics of learning how to write macros (short programs) in Excel using VBA, Visual Basic for Applications. It covers how to use the Visual Basic Editor (VBE), how to use the macro recorder, and how to write procedures, functions, if-then structures, and loops.

CHAPTER SIX

Systems Development

Phases, Tools, and Techniques

OPENING CASE STUDY: SAVING LIVES THROUGH SYSTEMS DEVELOPMENT AND INTEGRATION

The Centers for Disease Control and Prevention (CDC) in Atlanta, Georgia, tracks a wealth of information on everything from antimicrobial-resistant infections in hospitals to influenza outbreaks to terrorist biochemical attacks. Unfortunately, not all of that information is kept in an accessible manner for everyone who needs it and for every application software environment that needs to draw from it.

With offices all over the country, interacting with literally hundreds of state and local agencies, the CDC has volumes of information siloed on disparate servers and application software incapable of easily communicating with other application software. The CDC's IT initiative is to bring all this information together within a service-oriented architecture (SoA). An SoA is a holistic perspective of information and application software that focuses on ease of integration across all information repositories, application software, and hardware platforms.

While most businesses in the private sector use IT metrics such as ROI (return on investment), conversion rates, and click-throughs to measure the worth of a system, the CDC does so in terms of human life. For example, the CDC's BioSense initiative could provide early warning and critical information during an influenza pandemic. And that system needs to be integrated with those of pharmacies dispensing medications. In the early stages of an influenza outbreak, sales of acetaminophen might alert government officials to the threat even before people start showing up at hospitals.

The scope of the CDC's integration initiative is almost incomprehensible. The interrelationships and interactions in the natural, living world are complicated and astronomical in number. Foodborne illnesses, adverse drug reactions, hospital infections, and even EPA estimates of bacteria in rivers and stagnant ponds all relate to each other in some way. Most of these have their own dedicated IT systems for data gathering and analysis that must be integrated for the CDC so that its Public Health Information Network initiative can provide the right information to researchers and health care professionals across the country.

Systems development and the integration of all systems throughout an organization are of paramount importance in today's business world. It doesn't matter if you're trying to thwart an influenza pandemic or selling clothing on the Internet. You need integrated systems and accessible information to make the best decisions possible. Systems development is the focus of this chapter—how organizations go about the process of systems development, various methodologies for systems development, and how you, as an end user, can (and must) participate in the systems development process, and often, build systems for yourself. We live in a digital economy, and IT systems don't just magically appear. They must be thoughtfully and rigorously developed.[1]

Questions

1. All computers use a common binary base language. That being true, why is it so difficult to get computer systems to easily communicate with each other?
2. In systems development, prototyping is used to build a model of a proposed system. How have you used prototyping in your personal life to build something?
3. Outsourcing—going to another country for systems development—is big business. Why would the CDC *not* want to pursue outsourcing?

Introduction

Billions of dollars are spent each year on the acquisition, design, development, implementation, and maintenance of information systems. The ongoing need for safe, secure, and reliable systems solutions is a consequence of companies' increasing dependence on information technology to provide services and develop products, administer daily activities, and perform short- and long-term management functions.

Systems developers must ensure that all the business's needs and requirements are met when developing information systems, establish uniform privacy and security practices for system users, and develop acceptable implementation strategies for the new systems. This chapter focuses on the many factors that must come together to develop a successful information system.

You have three primary choices as to who will build your system (see Figure 6.1). First, you can choose ***insourcing,*** which involves in-house IT specialists within your organization to develop the system. Second, you can choose ***selfsourcing*** (also called ***end-user development***), which is the development and support of IT systems by end users (knowledge workers) with little or no help from IT specialists. Third, you can choose ***outsourcing,*** which is the delegation of specific work to a third party for a specified length of time, at a specified cost, and at a specified level of service.

As we introduce you to the systems development life cycle in the next section, we'll focus on insourcing and how the overall process works, key activities within each phase, roles you may play as an end user or knowledge worker, and opportunities you can capitalize on to ensure that your systems development effort is a success.

LEARNING OUTCOME 1

Insourcing and the Systems Development Life Cycle

The ***systems development life cycle (SDLC)*** is a structured step-by-step approach for developing information systems. It includes seven key phases and numerous activities within each (see Figure 6.2). This version of the SDLC is also referred to as a ***waterfall methodology***—a sequential, activity-based process in which one phase of the SDLC is followed by another, from planning through implementation (see Figure 6.2).

There are literally hundreds of different activities associated with each phase of the SDLC. Typical activities include determining budgets, gathering business requirements,

Figure 6.1

Insourcing, Selfsourcing, and Outsourcing

Insourcing

IT Specialists Within Your Organization

Selfsourcing

Knowledge Workers

Outsourcing

Another Organization

SDLC Phase	Activities
1. Planning	• Define the system to be developed • Set the project scope • Develop the project plan including tasks, resources, and timeframes
2. Analysis	• Gather the business requirements for the system
3. Design	• Design the technical architecture required to support the system • Design system models
4. Development	• Build the technical architecture • Build the database and programs
5. Testing	• Write the test conditions • Perform the testing of the system
6. Implementation	• Write detailed user documentation • Provide training for the system users
7. Maintenance	• Build a help desk to support the system users • Provide an environment to support system changes

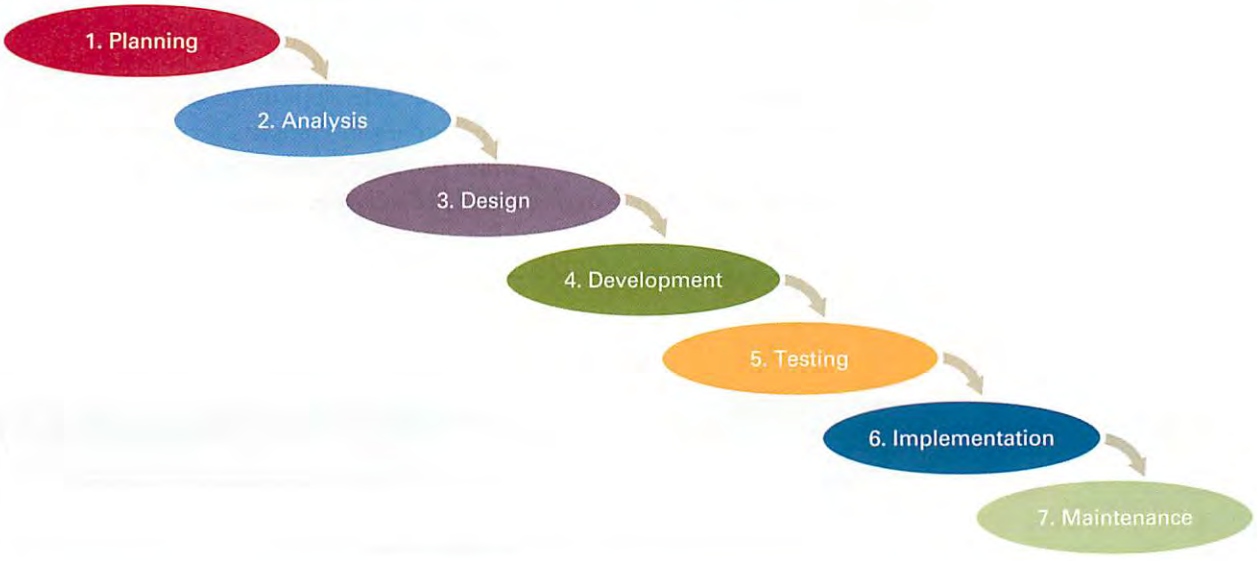


Figure 6.2
The Systems Development Life Cycle (SDLC), Phases and Activities, and the Waterfall Methodology

designing models, writing detailed user documentation, and project management. The activities you, as an end user, perform during each systems development project will vary depending on the type of system you're building and the tools you use to build it. Since we can't possibly cover them all in this brief introduction, we have chosen a few of the more important SDLC activities that you might perform on a systems development project as an end user.

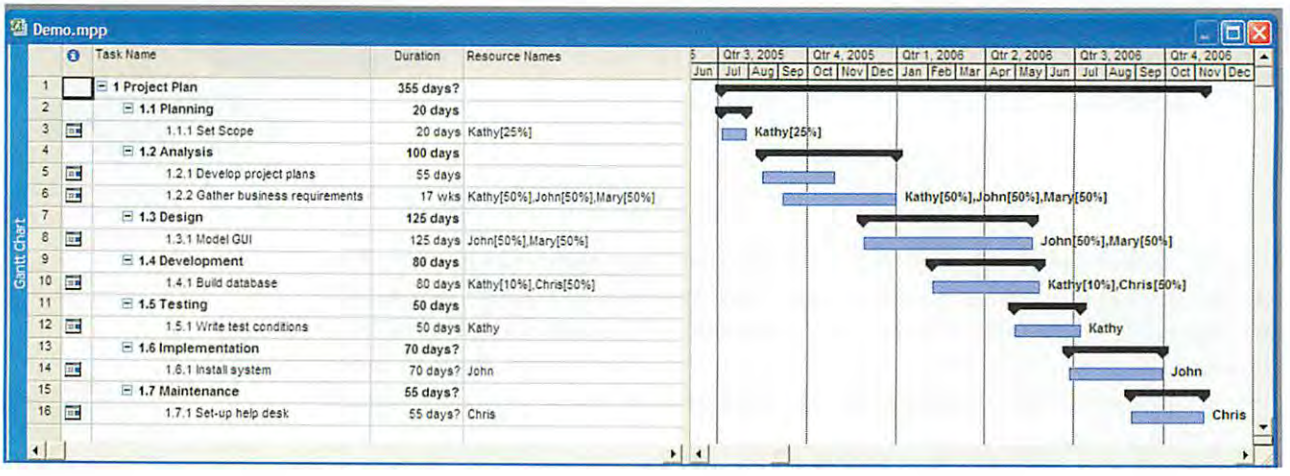
PHASE 1: PLANNING

During the ***planning phase*** of the SDLC you create a solid plan for developing your information system. The following are the three primary activities performed during the planning phase.

1. *Define the system to be developed:* You must identify and select the system for development or determine which system is required to support the strategic goals of your organization. Organizations typically track all the proposed systems and prioritize them based on business impact or critical success factors. A ***critical success factor (CSF)*** is simply a factor critical to your organization's success. This process allows your organization to strategically decide which systems to build.
2. *Set the project scope:* You must define the project's scope and create a project scope document for your systems development effort. The project scope clearly defines the high-level requirements. Scope is often referred to as the 10,000-foot view of the system or the most basic definition of the system. A ***project scope document*** is a written document of the project scope and is usually no longer than a paragraph. Project scoping is important for many reasons; most important it helps you avoid *scope creep* and *feature creep*. ***Scope creep*** occurs when the scope of the project increases beyond its original intentions. ***Feature creep*** occurs when developers (and end users) add extra features that were not part of the initial requirements.
3. *Develop the project plan:* You must develop a detailed project plan for your entire systems development effort. The ***project plan*** defines the *what, when,* and *who* questions of systems development including all activities to be performed, the individuals, or resources, who will perform the activities, and the time required to complete each activity. The project plan is the guiding force behind ensuring the on-time delivery of a complete and successful information system. Figure 6.3 provides a sample project plan. A ***project manager*** is an individual who is an expert in project planning and management, defines and develops the project plan, and tracks the plan to ensure that all key project milestones are completed on time. ***Project milestones*** represent key dates by which you need a certain group of activities performed. Either of the two *creeps* alluded to above can throw off a project plan.

Figure 6.3

A Sample Project Plan



PHASE 2: ANALYSIS

Once your organization has decided which system to develop, you can move into the analysis phase. The ***analysis phase*** of the SDLC involves end users and IT specialists working together to gather, understand, and document the business requirements for the proposed system. The following are the two primary activities you'll perform during the analysis phase.

1. *Gathering the business requirements:* ***Business requirements*** are the detailed set of end-user requests that the system must meet to be successful. The business requirements drive the entire system. A sample business requirement might state, "The CRM system must track all customer inquiries by product, region, and sales representative." The business requirement states what the system must do from the business perspective. Gathering business requirements is similar to performing an investigation. You must talk to everyone who has a claim in using the new system to find out what is required. An extremely useful way to gather business requirements is to perform a joint application development session. During a ***joint application development (JAD)*** session users and IT specialists meet, sometimes for several days, to define and review the business requirements for the system.
2. *Prioritize the requirements:* Once you define all the business requirements, you prioritize them in order of business importance and place them in a formal comprehensive document, the ***requirements definition document.*** The users receive the requirements definition document for their sign-off. ***Sign-off*** is the users' actual signatures indicating they approve all the business requirements. Typically, one of the first major milestones in the project plan is the users' sign-off on business requirements.

One of the key things to think about when you are reviewing business requirements is the cost to the company of fixing errors if the business requirements are unclear or inaccurate. An error found during the analysis phase is relatively inexpensive to fix; all you typically have to do is change a Word document. An error found during later phases, however, is incredibly expensive to fix because you have to change the actual system. Figure 6.4 displays how the cost to fix an error grows exponentially the later the error is found in the SDLC.

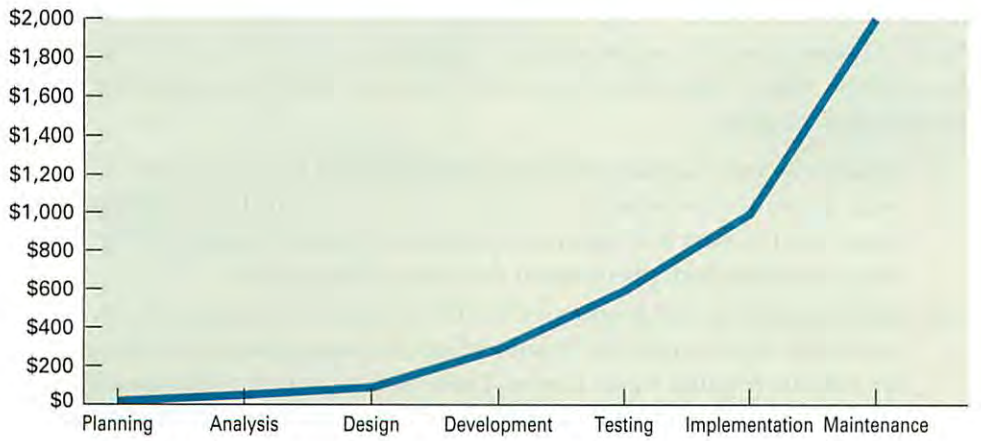


Figure 6.4

The Cost of Finding Errors

PHASE 3: DESIGN

The primary goal of the ***design phase*** of the SDLC is to build a technical blueprint of how the proposed system will work. During the analysis phase, end users and IT specialists work together to develop the business requirements for the proposed system from a logical point of view. That is, during analysis you document business requirements without respect to technology or the technical infrastructure that will support the system. As you move into design, the project team turns its attention to the system from a physical or technical point of view. You take the business requirements generated during the analysis phase and define the supporting technical architecture in the design phase. The following are the primary activities you'll perform during the design phase.

1. *Design the technical architecture:* The ***technical architecture*** defines the hardware, software, and telecommunications equipment required to run the system. Most systems run on a computer network with each employee having a workstation and the application software running on a server. The telecommunications requirements encompass access to the Internet and the ability for end users to connect remotely to the server. You typically explore several different technical architectures before choosing the final technical architecture.
2. *Design the system model:* Modeling is the activity of drawing a graphical representation of a design. You model everything you build including screens, reports, software, and databases (with E-R diagrams as we described in *Extended Learning Module C*). There are many different types of modeling activities performed during the design phase including a graphical user interface screen design.

It is at the point of the design phase in the SDLC that you, as an end user, begin to take a less active role in performing the various activities and divert your attention to "quality control." That is, IT specialists perform most of the functions in the design through maintenance phases. It is your responsibility to review their work, for example, verifying that the models of the screens, reports, software, and databases encapsulate all of the business requirements.

PHASE 4: DEVELOPMENT

During the ***development phase*** of the SDLC, you take all your detailed design documents from the design phase and transform them into an actual system. This phase marks the point at which you go from physical design to physical implementation. Again, IT specialists are responsible for completing most of the activities in the development phase. The following are the two main activities performed during the development phase.

1. *Build the technical architecture:* For you to build your system, you must first build the platform on which the system is going to operate. In the development phase, you purchase and implement equipment necessary to support the technical architecture you designed during the design phase.
2. *Build the database and programs:* Once the technical architecture is built, you initiate and complete the creation of supporting databases and writing the software required for the system. These tasks are usually undertaken by IT specialists, and it may take months or even years to design and create the databases and write all the software.

PHASE 5: TESTING

The ***testing phase*** of the SDLC verifies that the system works and meets all the business requirements defined in the analysis phase. Testing is critical. The following are the primary activities you'll perform during the testing phase.

1. *Write the test conditions:* You must have detailed test conditions to perform an exhaustive test. ***Test conditions*** are the detailed steps the system must perform along with the expected results of each step. The tester will execute each test condition and compare the expected results with the actual results to verify that the system functions correctly. Each time the actual result is different from the expected result, a "bug" is generated, and the system goes back to development for a "bug fix." A typical systems development effort has hundreds or thousands of test conditions. You must execute and verify all of these test conditions to ensure the entire system functions correctly.
2. *Perform the testing of the system:* You must perform many different types of tests when you begin testing your new system. A few of the more common tests include:
   - ***Unit testing***—tests individual units or pieces of code for a system.
   - ***System testing***—verifies that the units or pieces of code written for a system function correctly when integrated into the total system.
   - ***Integration testing***—verifies that separate systems can work together.
   - ***User acceptance testing (UAT)***—determines if the system satisfies the business requirements and enables users to perform their jobs correctly.

PHASE 6: IMPLEMENTATION

During the ***implementation phase*** of the SDLC you distribute the system to all the users and they begin using the system to perform their everyday jobs. The following are the two primary activities you'll perform during the implementation phase.

1. *Write detailed user documentation:* When you install the system, you must also provide employees with ***user documentation*** that highlights how to use the system. Users find it extremely frustrating to have a new system without documentation.
2. *Provide training for the system users:* You must also provide training for the users who are going to use the new system. You can provide several different types of training, and two of the most popular are online training and workshop training. ***Online training*** runs over the Internet or off a CD or DVD. Employees perform the training at any time, on their own computers, at their own pace. This type of training is convenient because they can set their own schedule to undergo the training. ***Workshop training*** is held in a classroom environment and is led by an instructor. Workshop training is most suitable for difficult systems for which employees need one-on-one time with an individual instructor.

You also need to choose the implementation method that best suits your organization, project, and employees to ensure a successful implementation. When you implement the new system, you have four implementation methods you can choose from:

1. ***Parallel implementation*** uses both the old and new systems until you're sure that the new system performs correctly.
2. ***Plunge implementation*** discards the old system completely and immediately uses the new system.

INDUSTRY PERSPECTIVE

FEDEX BELIEVES TESTING DOESN'T DELIVER

Testing is a critical step in any systems development effort. Just ask the people at FedEx. FedEx believes so strongly in testing, and that current testing methodologies are antiquated and don't work, it has commissioned the University of Memphis to research and study software testing and develop a new methodology that will deliver real results.

Called Systems Testing Excellence Program, the University of Memphis faculty and students in the FedEx Institute of Technology program at the university will attempt to develop testing methodologies that ensure that applications are of the highest quality while doing so in a shorter period of time than current testing methodologies. Using current methodologies, software code is written and turned over to a testing group; testers look for bugs and send those back to the development team. The development team must then assess the bug and determine if in fact it is a bug or if it results from an incorrect business requirement specification. Whatever the case, FedEx doesn't believe current testing methodologies work. According to Dave Miller, vice president of IT at FedEx, "We sort of believe that testing as a discipline has probably not kept up pace."

University students who stand out during the project may potentially be hired by FedEx to continue their work in the testing field.[2]

3. ***Pilot implementation*** has only a small group of people using the new system until you know it works correctly and then the remaining people are added to the system.
4. ***Phased implementation*** installs the new system in phases (e.g., accounts receivable, then accounts payable) until you're sure it works correctly and then the remaining phases of the new system are implemented.

PHASE 7: MAINTENANCE

Maintaining the system is the final phase of any systems development effort. During the ***maintenance phase*** of the SDLC, you monitor and support the new system to ensure it continues to meet the business goals. Once a system is in place, it must change as your business changes. Constantly monitoring and supporting the new system involves making minor changes (for example, new reports or information retrieval) and reviewing the system to be sure that it continues to move your organization toward its strategic goals. The following are the two primary activities you'll perform during the maintenance phase.

1. *Build a help desk to support the system users:* One of the best ways to support users is to create a help desk. A ***help desk*** is a group of people who respond to users' questions. Typically, users have a phone number for the help desk they call whenever they have issues or questions about the system. Providing a help desk that answers user questions is a terrific way to provide comprehensive support for users using new systems.
2. *Provide an environment to support system changes:* As changes arise in the business environment, you must react to those changes by assessing their impact on the system. It might well be that the system needs to be adapted or updated to meet the ever-changing needs of the business environment. If so, you must modify the system to support the new business environment.

Component-Based Development

Component-Based Development

LEARNING OUTCOME 2

The systems development life cycle you just read about is one of the oldest software development methodologies. In design and development (as well as the other phases), the SDLC takes a very singular view of the system under consideration and focuses solely on its development. That is, the SDLC does not really allow the development team to look around a software library and find existing code that can be reused for the new system under consideration. This has some tremendous disadvantages. Most notably, all software is written from scratch each time it is needed for each application. For example, there are probably many applications in the typical organization that have some sort of customer view screen and the ability to update a customer's information. However, within the traditional SDLC, the software that supports the customer view screen and the ability to update the information would be written each time for each application.

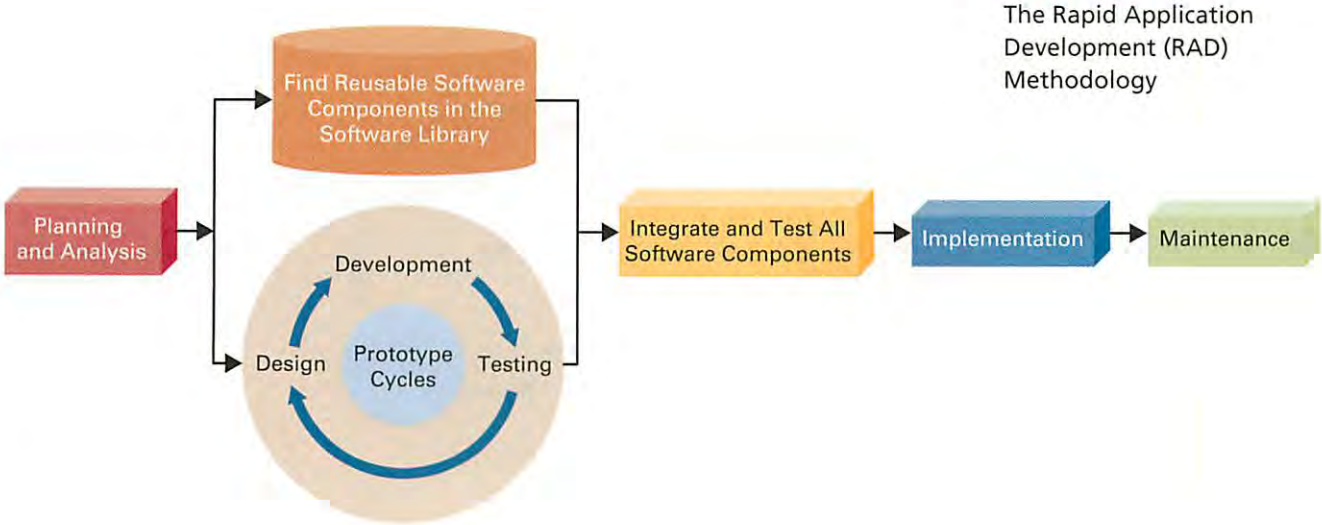
This has given rise to the notion of component-based development. ***Component-based development (CBD)*** is a general approach to systems development that focuses on building small self-contained blocks of code (components) that can be reused across a variety of applications within an organization. The goal here, for example, is to write the customer view screen and updating software only once, place it in a library of software components, and then allow software development teams to plug in that component (rather like the notion of *plug-and-play*) into whatever system needs to be developed.

Component-based development dramatically changes the systems development life cycle. It requires teams to (1) look through the software library for reusable code that already exists and (2) build new software in the form of components that can be reused later in other software development projects. In this approach, you can find new systems development methodologies being used including *rapid application development, extreme programming,* and *agile.*

RAPID APPLICATION DEVELOPMENT METHODOLOGY

The ***rapid application development (RAD)*** (also called ***rapid prototyping) methodology*** emphasizes extensive user involvement in the rapid and evolutionary construction of working prototypes of a system to accelerate the systems development process (see Figure 6.5).

Figure 6.5
The Rapid Application Development (RAD) Methodology



The fundamentals of RAD include:

1. Perform the planning and analysis phases in a similar manner as if you were using the traditional systems development life cycle.
2. Review the software library to determine if components already exist that can be used as part of the new system.
3. Create prototypes (i.e., working models of software components) that look and act like aspects of the desired system. Design, develop, and test the prototypes until they become fully functional software components.
4. Integrate the software components from the previous two steps and test them as a complete system.
5. Implement the new system, following many of the guidelines found in the traditional SDLC.
6. Provide ongoing support and maintenance.

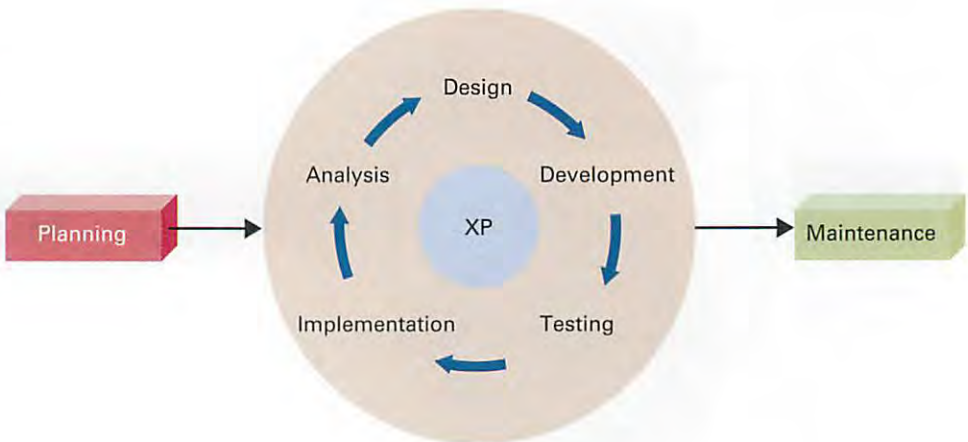
Overall, you want to actively involve end users in the analysis phase and in the iterative approach to the design, development, and testing of new software components. This end-user involvement and the use of prototyping tend to greatly accelerate the collecting of business requirements and the development of the software (i.e., software components). Moreover, if you can find reusable software components in the software library, the acceleration of the overall process is even greater. Prototyping is an essential part of the RAD methodology and we'll discuss it in more detail in a later section.

EXTREME PROGRAMMING METHODOLOGY

The ***extreme programming (XP) methodology*** breaks a project into tiny phases and developers cannot continue on to the next phase until the current phase is complete. XP is a lot like a jigsaw puzzle; there are many small pieces (i.e., software components). Individually, the pieces make no sense, but when they are combined together an organization can gain visibility into the entire system. The primary difference between the traditional SDLC and XP methodologies is that XP divides its phases into iterations. For example, the traditional SDLC approach develops the entire system, whereas XP develops the system in iterations (see Figure 6.6). Although not shown in Figure 6.6, the XP methodology, much like the RAD methodology, does rely heavily on reusing existing software components contained in a software library.

Figure 6.6

The Extreme Programming Methodology



GLOBAL PERSPECTIVE

MERCEDES-BENZ: BUILT-TO-ORDER TRUCKS FROM BUILT-TO-ORDER SOFTWARE

"Which driver's cabs and radios are available for a 24-ton rig?" In the past, such a question would have cost Mercedes-Benz dealers a lot of time leafing through big manuals. Today, the Mercedes-Benz Web site offers orientation to the multitude of equipment variations with the Mercedes-Benz customer advisory system (MBKS), developed by CAS Software AG.

The online program, called Truck Online Configurator (TOC), required much more development time than simply displaying the finished online systems for the automobile and transporter divisions with other data and image material. The TOC was given a completely new user guide that orients itself on the different demands of the Web site visitors in three ways: through technical features, trade solution, and transport task. For example, interested customers are able to compile their own vehicle preference by entering the required technical details, such as the type of drive, wheelbase, or engine performance needed. Or an interested customer can select by trade solution or transport task (where he or she simply tells the TOC that a truck is required for the transportation of frozen foods, for example) and a list of suitable models is offered.

CAS had only five months in which to realize this complex and extensive application, from the definition of the specialist requirements, to the technical specifications, right up to the development, testing, and installation of the software. This tight deadline required a risk-driven project management, which meant that individual phases of the project would overlap. While a few developers, together with the project managers of Mercedes-Benz, were clarifying the technical requirements of the individual functions, other parts of the TOC were already being implemented.

The CAS solution was virtually tailor-made for this project. The finished programs run in an extremely fast and stable manner. These were the central, technical requirements of Mercedes-Benz for the TOC: The company wanted an application which boasts maximum availability but with minimum response times.

Thanks to using the rapid application development (RAD) process, CAS was capable of creating very fast a high-quality application.[3]

Microsoft Corporation developed Internet Explorer and Netscape Communications Corporation developed Communicator using extreme programming. Both companies did a nightly compilation (called a *build*) of the entire project, bringing together all the current components. They established release dates and expended considerable effort to involve customers in each release. The extreme programming approach allowed both Microsoft and Netscape to manage millions of lines of code as specifications changed and evolved over time. Most important, both companies frequently held user design reviews and strategy sessions to solicit and incorporate user feedback.[4]

XP is a significant departure from traditional software development methodologies, and many organizations in different industries have developed successful software using it. One of the reasons for XP's success is that it stresses customer satisfaction. XP empowers developers to respond to changing customer and business requirements, even late in the systems development life cycle, and emphasizes teamwork. Managers, customers, and developers are all part of a team dedicated to delivering quality software. XP implements a simple, yet effective, way to enable group style development. The XP methodology supports quickly being able to respond to changing requirements and technology.

AGILE METHODOLOGY

The ***agile methodology,*** a form of XP, aims for customer satisfaction through early and continuous delivery of useful software components. Agile is similar to XP but with less focus on team coding and more on limiting project scope. An agile project sets a minimum number of requirements and turns them into a deliverable product. Agile means what it sounds like: fast and efficient; small and nimble; lower cost; fewer features; shorter projects.

The Agile Alliance, a group of software developers, has made its mission to improve software development processes. Its manifesto includes the following tenets:

- Satisfy the customer through early and continuous delivery of valuable software.
- Welcome changing requirements, even late in development.
- Business people and developers must work together daily throughout the project.
- Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
- The best architectures, requirements, and designs emerge from self-organizing teams.
- At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.[5]

SERVICE-ORIENTED ARCHITECTURE—AN ARCHITECTURE PERSPECTIVE

Regardless of which component-based methodology your organization chooses to use as its software development approach, it will most likely be employing a software architecture perspective called a *service-oriented architecture.* A ***service-oriented architecture (SOA or SoA)*** is a software architecture perspective that focuses on the development, use, and reuse of small self-contained blocks of code (called *services*) to meet all the application software needs of an organization. These *services* within the SoA architecture perspective are exactly the same as *components* in any of the component-based development methodologies.

An SoA is a high-level, holistic organizational approach to how your organization views and acts on all its software needs. If adopted, your organization would, in essence, be saying that all software will be developed and managed as a series of reusable services (blocks of code). From within your SoA perspective, you would then choose from among the different component-based development methodologies that support the concept of reusable services (i.e., components) for the development of specific systems. Those development methodologies would not include the traditional SDLC, but rather the approaches we just covered—RAD, XP, and agile.

SoA is growing rapidly in business importance and we'll cover this topic further in Chapter 7.

LEARNING OUTCOME 3

Selfsourcing (End-User Development)

What we want to look at now is how you, as a knowledge worker and end user, can go about developing your own systems, which we call *selfsourcing* or *end-user development.* Recall that ***selfsourcing (end-user development)*** is the development and support of IT systems by end users (knowledge workers) with little or no help from IT specialists. End users are individuals who will use a system, who, although skilled in their own domain,

are not IT or computer experts, and yet they know very well what they want from a system and are capable of developing such systems. Applications developed by end users support a wide range of decision-making activities and contribute to business processing in a wide range of tasks. Although certainly not on the level of enterprisewide enterprise resource planning systems, applications developed by end users are an important source for the organization's portfolio of information systems. The major tools for selfsourcing have been, and still continue to be, spreadsheets and database management systems and Web development.

Rapidly gaining in acceptance is the idea that selfsourcing can be a potent source of stress *relief* rather than a cause of stress. Rather than combating the trend toward end-user application development, IT staff should leverage it to offload solution building to end users. IT then frees its own scarce resources for complex, visible, infrastructure management tasks. A successful strategy for selfsourcing relies on two keys: (1) knowing which applications are good candidates and (2) providing end users with the right tools. After working through the selfsourcing process, we'll revisit the two keys.

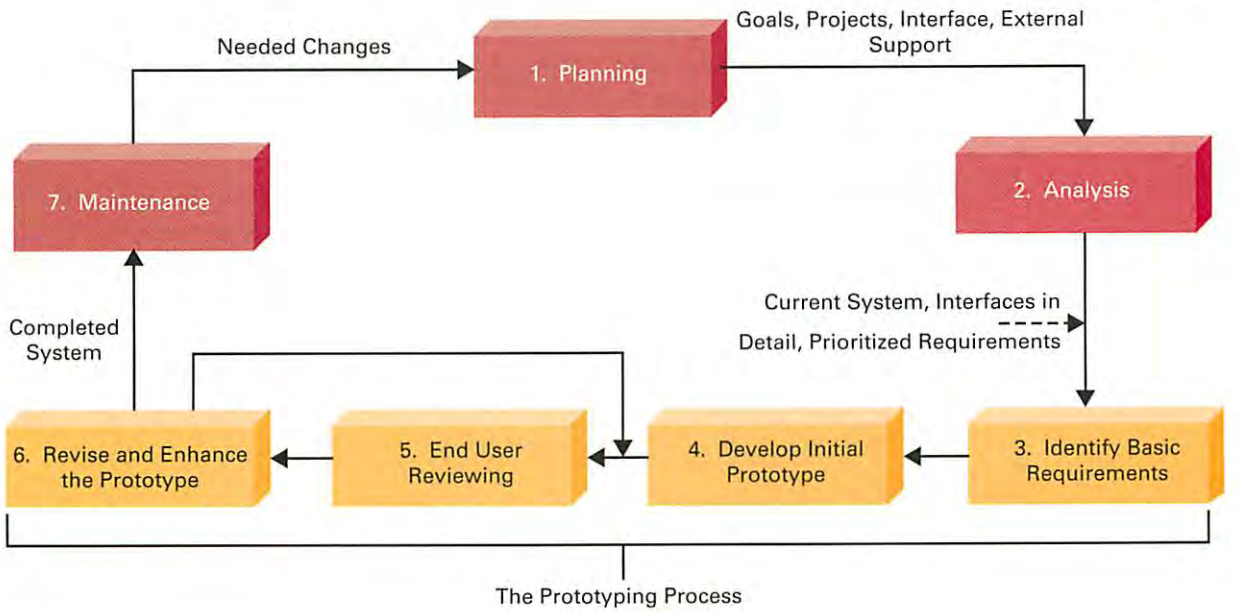
THE SELFSOURCING PROCESS

You can probably create many of the small end-user computing systems in a matter of hours, such as customizing reports, creating macros, building queries, and interfacing a letter in a word processing package with a customer database to create individualized mailings. More complicated systems, such as a student registration system or an employee payroll system, require that you follow the formal SDLC process during development.

In Figure 6.7, we've illustrated the selfsourcing process. As you can see, the selfsourcing process is similar to the phases in the SDLC. However, you should notice that the selfsourcing process encompasses prototyping (model building, steps 3 through 6), which we'll cover thoroughly in the next section. This is key—when you develop a system for yourself, you will most often go through the process of prototyping, continually building on and refining your model or prototype until the system is

Figure 6.7

The Selfsourcing Process



KEY TASKS IN SELFSOURCING

Planning

- Define system goals in light of organizational goals
- Create a project plan
- Identify any systems that require an interface
- Determine what type of external support you will require

Analysis

- Study and model the current system
- Understand the interfaces in detail
- Define and prioritize your requirements

Support

- Completely document the system
- Provide ongoing support

Figure 6.8
Key Tasks in Selfsourcing

complete. As you consider the key tasks in selfsourcing in Figure 6.8, we would alert you to several important issues.

ALIGNING YOUR SELFSOURCING EFFORTS WITH ORGANIZATIONAL GOALS When you first begin planning a system that you want to develop, you must consider it in light of your organization's goals. If you're considering developing a system for yourself that is counterintuitive to your organization's goals, then you should abandon it immediately. You also have to consider how you spend your time building systems carefully, since you are busy and your time is extremely valuable. It's important to remember that building a system takes time—your time.

DETERMINING WHAT EXTERNAL SUPPORT YOU WILL REQUIRE Some selfsourcing projects may involve support from IT specialists within your organization. These may be located within the IT department, as is the case when the IT function is located within its own area (i.e., top-down silo or matrix structure). They may also be located within your department, as is the case when the IT function is fully integrated across all units. Your in-house IT specialists are a valuable resource during the selfsourcing process. Don't forget about them and be sure to include them in the planning phase. The chances of building a successful system increase greatly when you have both end users and IT specialists working together.

DOCUMENTING THE SYSTEM ONCE COMPLETE Even if you're developing a system just for yourself, you still need to document how it works. When you get promoted, other people will come in behind you and probably use the system you developed and might even make changes to it. For this reason, you must document how your system works from a technical point of view as well as create an easy-to-read user's manual.

PROVIDE ONGOING SUPPORT When you develop a system through selfsourcing, you must be prepared to provide your own support and maintenance. Since you are the primary owner and developer of the system, you're solely responsible for ensuring the system continues to function properly and continues to meet all the changing business requirements. You must also be prepared to support other end users who use your system, as they will be counting on you to help them learn and understand the system you developed. The systems development process doesn't end with implementation: It continues on a daily basis with support and maintenance.

THE ADVANTAGES OF SELFSOURCING

- *Improves requirements determination*—During insourcing, end users tell IT specialists what they want. In selfsourcing, end users essentially tell themselves

what they want. Potentially, this greatly improves the chances of thoroughly understanding and capturing all the business requirements and thus the prospect of success for the new system.

- *Increases end user participation and sense of ownership*—No matter what you do, if you do it yourself, you always take more pride in the result. The same is true when developing an IT system through selfsourcing. If end users know that they own the system because they developed it and now support it, they are more apt to participate actively in its development and have a greater sense of ownership.
- *Increases speed of systems development*—Many small systems do not lend themselves well to insourcing and the traditional SDLC. These smaller systems may suffer from "analysis paralysis" because they don't require a structured step-by-step approach to their development. In fact, insourcing may be slower than selfsourcing for smaller projects.
- *Reduces the invisible backlog*—Literally no organization has all the resources to develop every system that end users need. If end users can take on the development of some of the smaller systems, the end result is the reduction of the backlog of systems that the organization needs to develop. The ***invisible backlog*** is the list of all systems that an organization needs to develop but—because of the prioritization of systems development needs—never get funded because of the lack of organizational resources.

POTENTIAL PITFALLS AND RISKS OF SELFSOURCING

- *Inadequate end user expertise leads to inadequately developed systems*—Many selfsourcing systems are never completed because end users lack the real expertise with IT tools to develop a complete and fully working system. This seem like no big deal, since the system couldn't have been that important if the people who needed it never finished developing it. But that's not true. If end users devote time to the selfsourcing process and never complete the system, that time is wasted time.
- *Lack of organizational focus creates "privatized" IT systems*—Many selfsourcing projects are done outside the IT systems plan for an organization, meaning there may be many private IT systems that do not interface with other systems and that contained uncontrolled and duplicated information. Such systems serve no meaningful purpose in an organization and can only lead to more problems.
- *Insufficient analysis of design alternatives leads to subpar IT systems*—Some end users jump to immediate conclusions about the hardware and software they should use without carefully analyzing all the possible alternatives. If this happens, end users may develop systems whose components are inefficient.
- *Lack of documentation and external support leads to short-lived systems*—When end users develop a system, they often forgo documentation of how the system works and fail to realize that they can expect little or no support from IT specialists. All systems—no matter who develops them—must change over time. End users must realize that anticipating those changes is their responsibility and making those changes will be easier if they document their system well.

WHICH APPLICATIONS FOR IT TO OFFLOAD

The following checklist helps IT staff to determine which applications are in IT's domain and which are good candidates for selfsourcing. IT delivers maximum value

to the enterprise by focusing on high-cost, high-return applications with the following characteristics:

- Infrastructure-related
- Mission-critical ERP, CRM, SCM, business intelligence and e-business
- Support large numbers of concurrent users, for example, call center applications

Other applications may be good candidates for selfsourcing.

THE RIGHT TOOL FOR THE JOB

Requirements for selfsourcing tools and enterprise development tools (for IT specialists) are quite different. Ease of use is paramount for selfsourcing development tools. That's because end users are not skilled programmers and might use the development tools so infrequently that they can forget commands that aren't intuitive. Therefore, end users must have development tools that:

- *Are easy to use:* This is essential for rapid, low-cost development. For application programs, specific characteristics of ease-of-use include: simple data entry, error checking for values in lists and ranges, easy report generation (e.g., drag and drop), and ease of Web publishing.
- *Support multiple platforms:* To minimize support requirements, end users should select one or two development tools that run on the range of hardware platforms and operating systems within the organization.
- *Offer low cost of ownership:* Cost factors include not only the tool's purchase price, but also training time, speed of application development, and required skill level.
- *Support a wide range of data types:* By its very nature, data is dynamic. Therefore, the toolset should support all the features normally found in database management system products.

LEARNING OUTCOME 4

Prototyping

Prototyping is the process of building a model that demonstrates the features of a proposed product, service, or system. A ***prototype,*** then, is simply a model of a proposed product, service, or system. If you think about it, people prototype all the time. Automobile manufacturers build prototypes of cars to demonstrate and test safety features, aerodynamics, and comfort. Building contractors construct models of homes and other structures to show layouts and fire exits.

In systems development, prototyping can be an invaluable tool for you. Prototyping is an iterative process in which you build a model from basic business requirements, have users review the prototype and suggest changes, and further refine and enhance the prototype to include suggestions. Especially, prototyping is a dynamic process that allows end users to see, work with, and evaluate a model and suggest changes to that model to increase the likelihood of success of the proposed system. Prototyping is an invaluable tool in the component-based development methodologies (RAD, XP, and agile), selfsourcing, and insourcing.

You can use prototyping to perform a variety of functions in the systems development process:

- *Gathering requirements:* Prototyping is a great requirements gathering tool. You start by simply prototyping the basic system requirements. Then you allow end

users to add more requirements (information and processes) as you revise the prototype.

- *Helping determine requirements:* In many systems development projects, end users aren't sure what they really want. They simply know that the current system doesn't meet their needs. In this instance, you can use prototyping to help end users determine their exact requirements.
- *Proving that a system is technically feasible:* Let's face it, there are some things to which you cannot apply technology. And knowing whether you can is often unclear when defining the scope of the proposed system. If you're uncertain about whether something can be done, prototype it first. A prototype you use to prove the technical feasibility of a proposed system is a ***proof-of-concept prototype.***
- *Selling the idea of a proposed system:* Many people resist changes in IT. The current system seems to work fine, and they see no reason to go through the process of developing and learning to use a new system. In this case, you have to convince them that the proposed system will be better than the current one. Because prototyping is relatively fast, you won't have to invest a lot of time to develop a prototype that can convince people of the worth of the proposed system. A prototype you use to convince people of the worth of a proposed system is a ***selling prototype.***

THE PROTOTYPING PROCESS

Prototyping is an excellent tool in systems development. Most often, IT specialists (insourcing) use prototyping in the SDLC to form a technical system blueprint. In self-sourcing, however, you can often continue to refine the prototype until it becomes the final system. The prototyping process for either case is almost the same up to a point; only the result differs. Figure 6.9 (on the next page) illustrates the difference between insourcing and selfsourcing prototyping. Regardless of who does the prototyping, the prototyping process involves four steps:

1. *Identify basic requirements:* During the first step, you gather the basic requirements for a proposed system. These basic requirements include input and output information desired and, perhaps, some simple processes. At this point, you're typically unconcerned with editing rules, security issues, or end-of-period processing (for example, producing W-2s for a payroll system at the end of the year).
2. *Develop initial prototype:* Having identified the basic requirements, you then set out to develop an initial prototype. Most often, your initial prototype will include only user interfaces, such as data entry screens and reports.
3. *End user reviewing:* Step 3 starts the truly iterative process of prototyping. When end users first initiate this step, they evaluate the prototype and suggest changes or additions. In subsequent returns to step 3 (after step 4), they evaluate new versions of the prototype. It's important to involve as many end users as possible during this iterative process. This will help resolve any discrepancies in such areas as terminology and operational processes.
4. *Revise and enhance the prototype:* The final sequential step in the prototyping process is to revise and enhance the prototype according to any end user suggestions. In this step, you make changes to the current prototype and add any new requirements. Next, you return to step 3 and have the end users review the new prototype; then step 4 again, and so on.

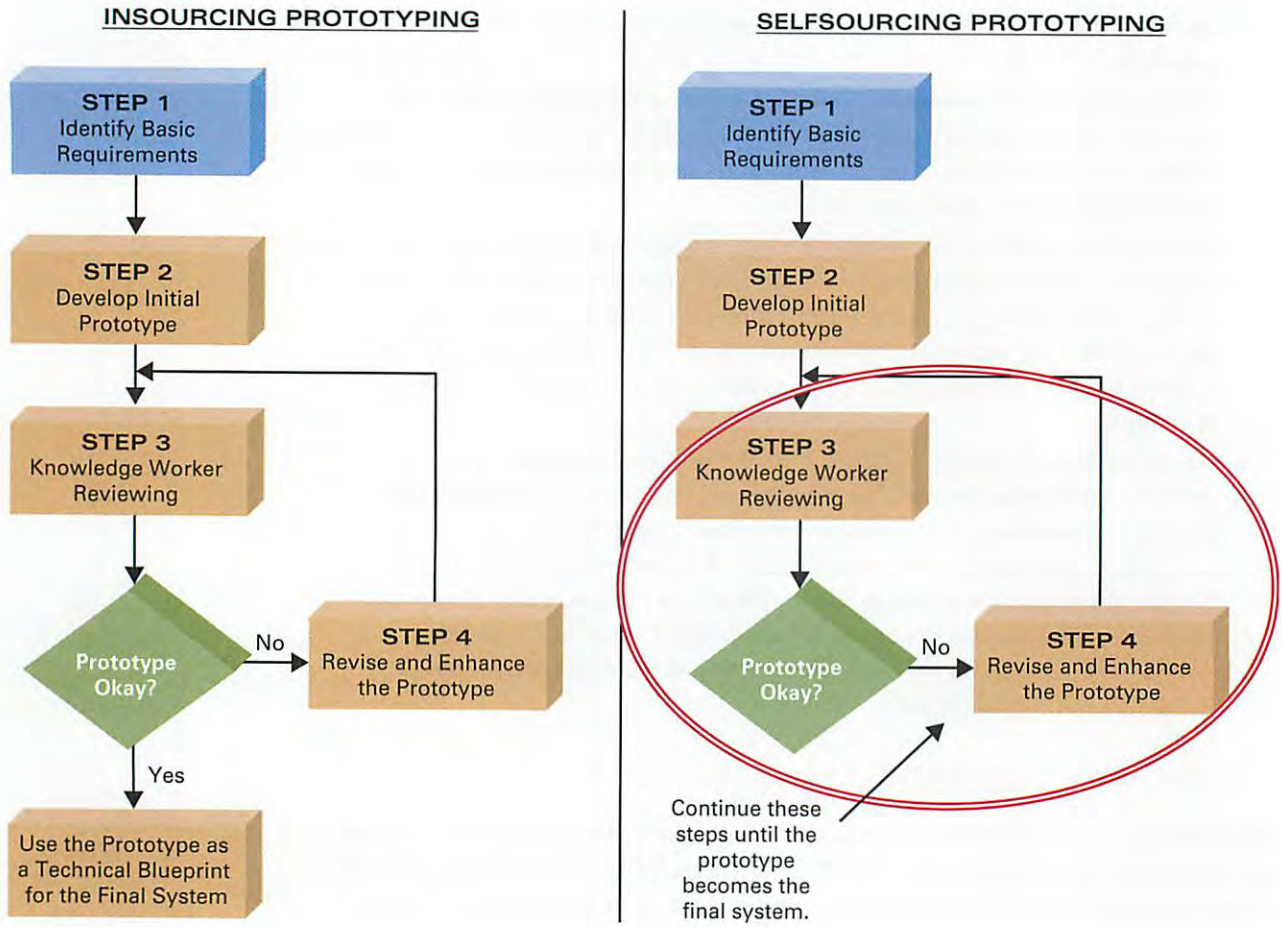


Figure 6.9

Prototyping Steps for Insourcing and Selfsourcing

For either insourcing or selfsourcing, you continue the iterative processes of steps 3 and 4 until end users are happy with the prototype. What happens to the prototype after that, however, differs.

During selfsourcing, you're most likely to use the targeted application software package or application development tool to develop the prototype. This simply means that you can continually refine the prototype until it becomes the final working system. For example, if you choose to develop a simple CRM application using Microsoft Access, you can prototype many of the operational features using Microsoft Access development tools. Because you develop these prototypes using the targeted application development environment, your prototype can eventually become the final system. This also holds true for component-based development methodologies.

That process is not necessarily the same when insourcing and using the traditional SDLC. Most often, IT specialists develop prototypes using special prototyping development tools. Many of these tools don't support the creation of a final system—you simply use them to build prototypes. Therefore, the finished prototype becomes a blueprint or technical design for the final system. In the appropriate stages of the SDLC, IT specialists will implement the prototype in another application development environment better suited to the development of production systems.

THE ADVANTAGES OF PROTOTYPING

- *Encourages active end user participation*—First and foremost, prototyping encourages end users to actively participate in the development process. As opposed to interviewing and reviewing documentation, prototyping allows end users to see and work with models of a proposed system.
- *Helps resolve discrepancies among end users*—During the prototyping process, many end users participate in defining the requirements for and reviewing the prototype. The word *many* is key. If several end users participate in prototyping, you'll find it's much easier to resolve any discrepancies the end users may encounter.
- *Gives end users a feel for the final system*—Prototyping, especially for user interfaces, provides a feel for how the final system will look and work. When end users understand the look and feel of the final system, they are more likely to see its potential for success.
- *Helps determine technical feasibility*—Proof-of-concept prototypes are great for determining the technical feasibility of a proposed system.
- *Helps sell the idea of a proposed system*—Finally, selling prototypes can help break down resistance barriers. Many people don't want new systems because the old ones seem to work just fine, and they're afraid the new system won't meet their expectations and work properly. If you provide them with a working prototype that proves the new system will be successful, they will be more inclined to buy into it.

THE DISADVANTAGES OF PROTOTYPING

- *Leads people to believe the final system will follow shortly*—When a prototype is complete, many people believe that the final system will follow shortly. After all, they've seen the system at work in the form of a prototype—how long can it take to bring the system into production? Unfortunately, it may take months or even years. You need to understand that the prototype is only a model, not the final system missing only a few simple bells and whistles.
- *Gives no indication of performance under operational conditions*—Prototypes seldom take all operational conditions into consideration. This problem surfaced at the Department of Motor Vehicles in a state on the East Coast. During prototyping, the system, which handled motor vehicle and driver registration for the entire state, worked fine for 20 workstations at two locations. When the system was finally installed for all locations (which included more than 1,200 workstations), the system spent all its time just managing communication traffic; it had absolutely no time to complete any transaction. This is potentially the most significant drawback to prototyping. You must prototype operational conditions as well as interfaces and processes.
- *Leads the project team to forgo proper testing and documentation*—You must thoroughly test and document all new systems. Unfortunately, many people believe they can forgo testing and documentation when using prototyping. After all, they've tested the prototype; why not use the prototype as the documentation for the system? Don't make that mistake.

INDUSTRY PERSPECTIVE

THE KEY TO PROTOTYPING MAY BE IN VISUALIZATION

Using high-fidelity prototyping and simulation, software developers are now able to greatly accelerate the already fast and efficient process of prototyping. These sorts of visual definition capabilities can be found in prototyping products published by the likes of Compuware, Borland, and Serena. According to Forrester analyst Carey Schwaber, these tools "let you construct upfront a fairly accurate simulation of what the application's going to look like down the road. Instead of describing what you want, you work with a business analyst or usability engineer to construct it. It serves as a visual contract between business and development."

According to Ron Beck, director of software development at Bally Technologies, his company has had great success using Serena's Composer, a definition modeling and prototyping tool. Bally offers many IT-related products to the casino and gaming industry. One such product is Bally's latest enterprise accounting product. According to Ron, Composer has greatly reduced the time it takes to map casino-specific processes to the accounting software. In a short amount of time, it gives users an understanding of how the accounting software will handle functions unique to the gaming industry.

As Ron explains it, "We used to waste so much time on mapping requirements due to lack of communication. Now we're able to model the interface and interactions between systems and users to see what works."[6]

LEARNING OUTCOME 5

Outsourcing

The third choice as to who will build your IT systems—beyond insourcing (in-house IT specialists) and selfsourcing (end users)—is outsourcing. ***Outsourcing*** is the delegation of specific work to a third party for a specified length of time, at a specified cost, and at a specified level of service. With competitive pressures to cut costs and reduce time-to-market, many organizations are looking to outsource their IT systems development (not to mention ongoing operation, maintenance, and support). The Outsourcing Research Council recently completed a study indicating that human resources (HR) is the top outsourcing area for many companies. Fifty percent of the companies surveyed said they were already outsourcing some or all of their payroll processing and another 38 percent said they were considering it.[7]

Energizer, the world's largest manufacturer of batteries and flashlights, outsourced its HR operations to ADP, one of the top HR outsourcing companies. Energizer currently has more than 3,500 employees and 2,000 retired employees who all require multiple HR IT-related services. ADP provides Energizer with centralized call centers, transaction-processing services, and Web-based employee self-service systems. Energizer's vice president of Human Resources, Peter Conrad, stated, "ADP was clearly the most capable, and offered the kind of one-stop shopping our company was looking for." For several of the systems provided by ADP employee usage topped over 80 percent in the first six months the systems were active.[8]

The main reasons behind the rapid growth of the outsourcing industry include the following:

- **Globalization:** As markets open worldwide, competition heats up. Companies may engage outsourcing service providers to deliver international services. And outsourcing service providers may be located throughout the globe.

- **The Internet:** Barriers to entry, such as the lack of capital, are dramatically reduced in the world of e-business. New competitors enter the market daily.
- **Growing economy and low unemployment rate:** Building a competitive workforce domestically is much harder and more expensive.
- **Technology:** Technology is advancing at such an accelerated rate that companies often lack the resources, workforce, or expertise to keep up.
- **Deregulation:** As private industries such as telecommunications and energy deregulate, markets open and competition increases.

IT outsourcing today represents a significant opportunity for your organization to capitalize on the intellectual resources of other organizations by having them take over and perform certain business functions in which they have more expertise than IT specialists in your company. Information technology outsourcing enables organizations to keep up with market and technology advances with less strain on human and financial resources and more assurance that the IT infrastructure will keep pace with evolving business practices. IT outsourcing for software development can take one of four forms (see Figure 6.10):

1. Purchasing existing software.
2. Purchasing existing software and paying the publisher to make certain modifications.
3. Purchasing existing software and paying the publisher for the right to make modifications yourself.
4. Outsourcing the development of an entirely new and unique system for which no software exists.

In these instances, we're not talking about personal productivity software you can buy at a local computer store. We're talking about large software packages that may cost millions of dollars. For example, every organization has to track financial information, and there are several different systems they can purchase that help them perform this activity. Have you ever heard of Oracle Financials? This is a great system your organization can buy that tracks all the organizational financial information. If Oracle Financials is exactly what you need (i.e., it meets all your business requirements), then you act on option 1.

Figure 6.10

Major Forms of Outsourcing Systems Development

If it meets most of your needs, you can act on either option 2 or 3. However, if you have a need for a unique suite of software that doesn't exist in the commercial market, you need to act on option 4. Let's explore that process.

THE OUTSOURCING PROCESS

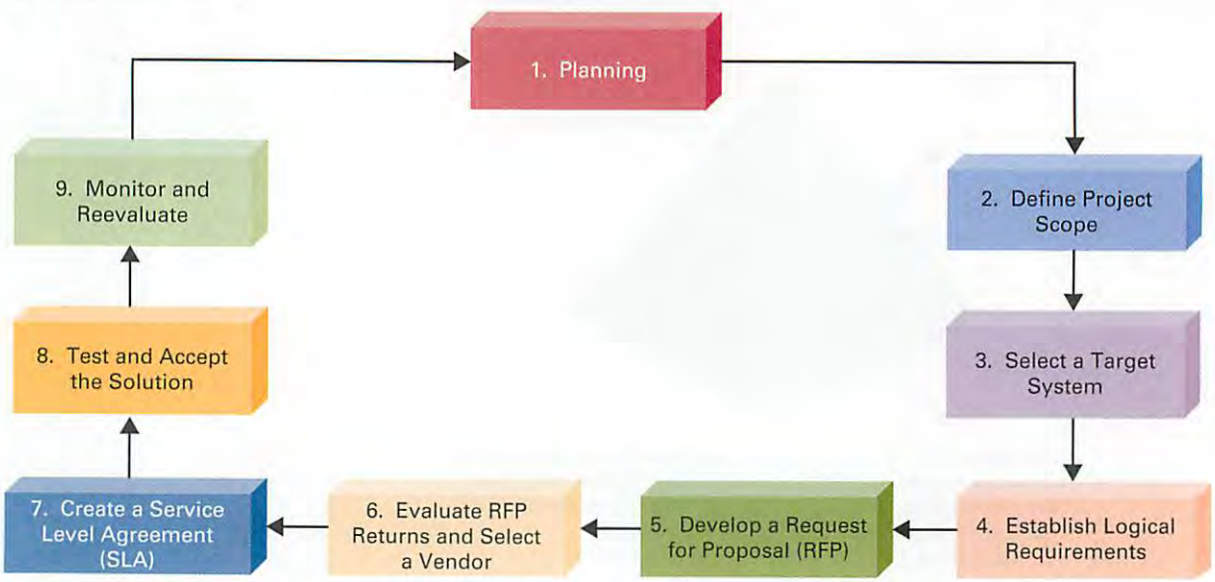
The outsourcing process is both similar to and quite different from the systems development life cycle. It's different in that you turn over much of the design, development, testing, implementation, and maintenance steps to another organization. It's similar in that your organization begins with planning and defining the project scope (see Figure 6.11). It's during one of these first two phases of the process that your organization may come to understand that it needs a particular system but cannot develop it in-house. If so, that proposed system can be outsourced. That is step 3, of the process, in Figure 6.10, the selection of a target system for outsourcing. Below, we briefly describe the remaining steps in the outsourcing process, starting with step 4.

ESTABLISH LOGICAL REQUIREMENTS Regardless of your choice of insourcing or outsourcing, you must still perform the analysis phase—especially the primary activity of gathering the business requirements for the proposed system. Remember that identification of the business requirements drives the entire systems development effort; if the business requirements are not accurate or complete, there is no way the system will be successful. If you choose to outsource, part of gathering the business requirements becomes step 5, your *request for proposal.*

DEVELOP A REQUEST FOR PROPOSAL (RFP) Outsourcing involves telling another organization what you want. What you want is essentially the logical requirements for a proposed system, and you convey that information by developing a request for proposal. A ***request for proposal (RFP)*** is a formal document that describes in excruciating detail your logical requirements for a proposed system and invites outsourcing organizations (which we'll refer to as *vendors*) to submit bids for its development. An RFP is one of

Figure 6.11

The Outsourcing process



the two most important documents in the outsourcing process. (The other is a service level agreement and we'll not debate which is the most important.) For systems of great size, your organization may create an RFP that's hundreds of pages long and requires months of work to complete.

It's vitally important that you take all the time you need to create a complete and thorough RFP. Eventually, your RFP will become the foundation for a legal and binding contract into which your organization and the vendor will enter. At a minimum, an RFP includes key information such as an overview of your organization, underlying business processes that the proposed system will support, a request for a detailed development time frame, and a request for a statement of detailed outsourcing costs.

All this information is vitally important to both your organization and the vendors. For your organization, the ability to develop a complete and thorough RFP means that you completely understand what you have and what you want. For the vendors, a complete and thorough RFP makes it easier to propose a system that will meet most, if not all, your needs.

EVALUATE RFP RETURNS AND SELECT A VENDOR Your next activity in outsourcing (step 6 in Figure 6.11) is to evaluate the RFP returns and choose a vendor. You perform this evaluation of the RFP returns according to a scoring mechanism you identified in the RFP. This is not a simple process. No two vendors will ever provide RFP returns in the same format, and the RFP returns you receive are usually longer than the RFP itself.

CREATE A SERVICE LEVEL AGREEMENT (SLA) Once you've chosen a vendor, a lengthy legal process follows. Outsourcing is serious business, and serious business between two organizations almost always requires a lot of negotiating and the use of lawyers. Eventually, your organization has to enter into a legal and binding contract that very explicitly states the features of the proposed system, the exact system costs, the time frame for development, acceptance criteria, criteria for breaking the contract for nonperformance or noncompliance, and postdevelopment metrics for activities such as maintenance, support, operational performances, and so on. This legal and binding contract is often called a *service level agreement,* which we'll discuss in more detail after explaining the remaining steps in the outsourcing process.

RFPs and SLAs

TEST AND ACCEPT THE SOLUTION As with all systems, testing and accepting the solution are crucial. Once a vendor installs the new system, it's up to you and your organization to test the entire system before accepting it. You'll need to develop detailed test plans and test conditions that test the entire system. This alone may involve months of running and testing the new system while continuing to operate the old one (the parallel method).

When you "accept" a solution, you're saying that the system performs to your expectations and that the vendor has met its contractual obligations thus far according to the service level agreement. Accepting a solution involves granting your sign-off on the system, which releases the vendor from any further development efforts or modifications to the system.

MONITOR AND REEVALUATE Just like the systems you develop using the SDLC, systems you obtain through outsourcing need constant monitoring and reevaluation. In outsourcing, you also have to reassess your working relationship with the vendor. Is the vendor providing maintenance when you need it and according to the service level agreement? Does the system really perform the stated functions? Do month-end and

year-end processes work according to your desires? Does the vendor provide acceptable support if something goes wrong? These are all important questions that affect the ultimate success of your outsourcing efforts.

THE SERVICE LEVEL AGREEMENT

When you contract the services of an outsourcing organization (i.e., vendor), you will most likely do so using a service level agreement. Broadly, a ***service level agreement (SLA)*** is a formal contractually obligated agreement between two parties. Within different environments, an SLA takes on different meanings. In the context of systems development, an SLA defines the work to be done, the time frame, the metrics that will be used to measure the success of the systems development effort, and the costs. Most SLAs are business oriented and void of detailed technical specifications. Technical specifications are included in a supporting document (similar to a contract addendum) called a ***service level specification (SLS)*** or ***service level objective (SLO).***

If the vendor further agrees to provide postdevelopment maintenance and support, then the SLA would outline in detail the terms of the maintenance and support activities, their costs, and—again—key metrics by which the success of those activities will be measured. The metrics you include in an SLA are the real key, and we'll describe and define such metrics in Chapter 7 when discussing how to create quantifiable measures of the value of an IT system.

GEOPOLITICAL OUTSOURCING OPTIONS

In a geopolitical sense, furthermore, there are three types of outsourcing:

1. ***Onshore outsourcing*** is the process of engaging another company in the same country for services. Much of the current jargon relating to outsourcing is based on the United States as the reference point. Thus, "onshore" outsourcing typically means contracting a U.S. company to provide business services.
2. ***Nearshore outsourcing*** is contracting an outsourcing arrangement with a company in a nearby country. Often, this country will share a border with the native country. Again, this term is often used with the United States as the frame of reference. In this case, nearshore outsourcing will take place in either Canada or in Mexico, usually.
3. ***Offshore outsourcing*** is contracting with a company that is geographically far away. For many companies, certain IT services, such as application development, maintenance, and help desk support, fall within the category of functions that are ideal for offshore outsourcing.

Although onshore and nearshore outsourcing are important forms in the outsourcing industry, when outsourcing is spoken about nowadays, it is usually *offshore* outsourcing that is being referenced.

OFFSHORE OUTSOURCING From a humble beginning as a mere cost-cutting concept, offshore outsourcing has gradually moved ahead and established itself as a successful business model by rendering not only cost-effective but also sophisticated and highly efficient quality services. According to International Data Corporation (IDC), U.S.-based companies tripled their offshore outsourcing spending from $5.5 billion in 2000 to more than $17.6 billion in 2005. The offshore outsourcing trend has overcome all barriers of political turmoil, language problems, and culture differences, and has proved that no matter in which part of the world your outsourcer resides, what really matters is industry-standard, high-quality service together with decisive cost advantage.[9]

INDUSTRY PERSPECTIVE

THE BANKING INDUSTRY BANKS ON OFFSHORE OUTSOURCING

According to a recent study by the consulting firm Deloitte, banks throughout the world, especially the United States, will dramatically increase their IT budgets over the next three years devoted to acquiring technology services abroad in the form of offshore outsourcing. Currently, banks' offshore outsourcing expenditures represent 6 percent of the banking industry's $44 billion total annual IT budget. That percentage is expected to rise to 30 percent by 2010.

According to the study, "Among larger institutions in particular, offshoring is not one available cost-cutting strategy, it's become a necessity." Banks are offshoring a variety of IT-related services ranging from low-level application maintenance to more sophisticated enterprisewide infrastructure and application development.

Offshoring IT-related services and application development can produce significant savings. For example, in India programmers are paid anywhere from 40 to 80 percent less than their U.S. counterparts. Indeed, many outsourcers in India are reaping the rewards from the banking industry. Wipro reported that banking industry offshoring accounted for 21 percent of its total annual revenue of $2.39 billion. TCS, a Wipro rival in India, reported that banking industry offshoring accounted for 38 percent of its $2.97 billion total revenue in 2006.[10]

Since the mid-1990s, major U.S. companies have been sending significant portions of their software development work offshore—primarily to vendors in India, but also to vendors in China, Eastern Europe (including Russia), Ireland, Israel, and the Philippines. The big selling point for offshore outsourcing to these countries is "good work done cheap." A programmer who earns $63,000 per year in the United States is paid as little as $5,000 per year overseas (see Figure 6.12). Companies can easily realize cost savings of 30 to 50 percent through offshore outsourcing and still get the same, if not better, quality of service.

Stories about U.S. companies outsourcing work offshore to India have been reported for years; however, it is becoming increasingly apparent that Romania, Bulgaria, Russia, China, Ghana, the Philippines, and dozens of other countries are also clamoring for and getting business from the United States. The reality is that offshore outsourcing is a growing trend. According to a recent study from Meta Group, the worldwide offshore outsourcing market will grow 20 percent annually through 2008. Meta also claims that offshoring growth will outpace outsourcing in general and predicts that the average enterprise will offshore 60 percent of application development by 2008 or 2009.[11]

Figure 6.12

Typical Salary Ranges for Computer Programmers

Country	Salary Range per Year
China	$5,000–9,000
India	$6,000–10,000
Philippines	$6,500–11,000
Russia	$7,000–13,000
Ireland	$21,000–28,000
Canada	$25,000–50,000
United States	$60,000–90,000

What types of functions or projects make good candidates for offshore outsourcing? Data conversions and system migrations with well-defined requirements and specifications and minimal end-user interaction with the development team are typical projects taken offshore. Naturally, the company must be willing to allow its application code to be located offsite during development. Application development projects are also good offshore candidates. From a SDLC perspective, offshore work is most beneficial in the development and testing phases where end-user interaction is limited, and the task is well defined. For stable applications, most maintenance activities can be performed remotely so application maintenance is also a good candidate for offshore outsourcing. With the right communication infrastructure and a clear understanding of your company's business language requirements, call center or help desk functions can also be moved offshore.

THE ADVANTAGES AND DISADVANTAGES OF OUTSOURCING

Making the decision to outsource may be critical one way or the other to your organization's success. Thus far in our discussion of outsourcing, we've alluded to some advantages and disadvantages of outsourcing. Following is a summary of the major advantages and disadvantages of outsourcing the systems development process, in order to help you make the important outsourcing decision.

ADVANTAGES Your organization may benefit from outsourcing because outsourcing allows you to:

- *Focus on unique core competencies:* By outsourcing systems development efforts that support noncritical business functions, your organization can focus on developing systems that support important, unique core competencies.
- *Exploit the intellect of another organization:* Outsourcing allows your organization to obtain intellectual capital by purchasing it from another organization. Often you won't be able to find individuals with all the expertise required to develop a system. Outsourcing allows you to find those individuals with the expertise you need to get your system developed and implemented.
- *Better predict future costs:* When you outsource a function, whether systems development or some other business function, you know the exact costs.
- *Acquire leading-edge technology:* Outsourcing allows your organization to acquire leading-edge technology without having to acquire technical expertise and bear the inherent risks of choosing the wrong technology.
- *Reduce costs:* Outsourcing is often seen as a money saver for organizations. Reducing costs is one of the important reasons organizations outsource.
- *Improve performance accountability:* Outsourcing involves delegating work to another organization at a specified level of service. Your organization can use this specified level of service to guarantee that it gets exactly what it wants from the vendor.

DISADVANTAGES Outsourcing may *not* be a beneficial option for you because it:

- *Reduces technical know-how for future innovation:* Outsourcing is a way of exploiting the intellect of another organization, so it can also mean that your organization will no longer possess that expertise internally. If you outsource because you don't have the necessary technical expertise today, you'll probably have to outsource for the same reason tomorrow.

- *Reduces degree of control:* Outsourcing means giving up control. No matter what you choose to outsource, you are in some way giving up control over that function.
- *Increases vulnerability of your strategic information:* Outsourcing systems development involves telling another organization what information you use and how you use that information. In doing so, you could be giving away strategic information and secrets.
- *Increases dependency on other organizations:* As soon as you start outsourcing, you immediately begin depending on another organization to perform many of your business functions.

Summary: Student Learning Outcomes Revisited

1. **Define the traditional systems development life cycle (SDLC) and describe the seven major phases within it.** The ***systems development life cycle (SDLC)*** is a structured step-by-step approach for developing information systems. The seven major phases within it include:
   - ***Planning***—creating a solid plan for developing your information system
   - ***Analysis***—gathering, understanding, and documenting the business requirements
   - ***Design***—building a technical blueprint of how the proposed system will work
   - ***Development***—taking all the design documents and transforming them into an actual system
   - ***Testing***—verifying that the system works and meets all the business requirements
   - ***Implementation***—distributing and using the new system
   - ***Maintenance***—monitoring and supporting the new system
2. **Compare and contrast the various component-based development methodologies.** Component-based methodologies (CBD) include:
   - ***Rapid application development (RAD or rapid prototyping)***—extensive user involvement in the rapid and evolutionary construction of working prototypes of a system to accelerate the systems development process
   - ***Extreme programming (XP)***—breaks a project into tiny phases, with each phase focusing on a small aspect of the system that eventually becomes a component or small software module
   - ***Agile***—aims for customer satisfaction through early and continuous delivery of useful software components
3. **Describe the selfsourcing process as an alternative to the traditional systems development life cycle.** ***Selfsourcing (end-user development)*** is the development and support of IT systems by end users with little or no help from IT specialists. While the traditional SDLC uses in-house IT specialists to develop a system, selfsourcing has the user developing his or her own system. The user typically prototypes the system using the targeted application software environment and can thus continually refine and enhance the prototype until it becomes the final working system.
4. **Discuss the importance of prototypes and prototyping within any systems development methodology.** ***Prototyping*** is the process of building a model (i.e., ***prototype***) that demonstrates the features of a proposed product, service, or system. Prototyping can be used effectively to gather requirements, help determine requirements when they are unknown, prove that a system is technically feasible (***proof-of-concept prototype***), and sell the idea of a proposed system (***selling prototype***).
5. **Describe the outsourcing environment and how outsourcing works.** ***Outsourcing*** is the delegation of specific work to a third party for a specified length of time, at a specified cost, and at a specified level of service. Outsourcing is growing today because of globalization, the Internet, a growing economy and low unemployment rate,

technology, and deregulation. Everything from food service to payroll services to call centers is being outsourced. In the outsourcing process, you target a system for outsourcing and build a ***request for proposal (RFP)*** that invites vendors to bid for its development. You choose a vendor and enter into an agreement called a ***service level agreement (SLA)*** that states exactly what the vendor is going to do. In the end, you test and accept the solution from the vendor and begin using the system.

CLOSING CASE STUDY ONE

GETTING ON THE RIGHT TRACK AT GENERAL MOTORS

"Our data tells us that the vehicle owners that don't have a satisfactory dealership repair experience are only half as likely to buy that model car again." Think about the ramifications of that statement. If a vehicle owner has a poor experience with something as simple as an oil change, that person is only half as likely to spend $30,000 or more on that model of car again. That is quite the return on investment (ROI) for providing a good experience for an oil change that costs about $50.

Bryan Burkhardt, global director of retail inventory management for General Motors (GM) service and parts operations, made the above statement. Bryan very clearly understands the relationship between after-sales and service and retaining the loyalty of a customer. Unfortunately, left to their own devices, most parts managers at any of GM's 7,000 North American dealerships overstock too many of the more commonly sold parts and seldom have those on hand that are infrequently purchased. As Bryan explains, there is "not enough breadth of parts, but way too much depth on the ones they do have."

That "not enough breadth of parts" means that GM repair shops have been providing customers with a satisfactory repair experience only 67 percent of the time. Bryan and his team set out to change that and implemented a new inventory management system, upping the satisfactory repair experience to 96 percent of the time. The new inventory management system is a centralized, Web-based system that tracks inventory levels in real time. If the quantity-on-hand for any part ever falls below 5, the system notifies the parts manager and automatically routes an order for parts replenishment to one of 16 national parts distribution centers. The system can even accommodate regional differences and keep more of certain parts on hand for dealerships in a given region of the United States, such as more windshield wipers for dealerships in the northwest during the spring. All told, the system tracks over 500,000 GM parts from 4,000 different suppliers.

One of the biggest challenges facing Bryan and his team is that each of the dealerships is allowed to choose its own dealership management system, resulting in the use of 28 different systems. So far, the new centralized inventory management system has been certified to work with only 6 of those systems. Because of the increase in productivity and inventory efficiencies, GM is working with the new inventory management system to provide certification for more of the dealership-centric systems in the hope that all systems will be certified and all dealerships converted to the new inventory management system by the end of 2007. As the new inventory management system is certified to work with another dealership management system, GM pilots the system with a few of the dealerships interested in making the conversion. When the system is determined to work correctly, the remaining dealerships are encouraged to make the conversion.

The new inventory management system is yielding both efficiency and effectiveness. From an efficiency point of view, the new system improves inventory turnover by about 11 percent. It also reduces the time parts managers spend on reviewing inventory and ordering parts from 90 minutes per day to just 10 to 15 minutes per day.

But the most important results are being realized in the area of effectiveness, that is, customer-centric measures such as satisfaction. As Bryan explains it, "At heart, it's about enhancing the ownership experience." After all, providing a satisfactory repair experience doubles the chances that the automobile owner will buy that same model again. In short, customer retention is key.[12]

Questions

1. In implementing the new inventory management system and converting existing users (dealerships) to it, what sort of implementation method is GM using? In your opinion, why is this the most appropriate method? If you had to choose a different implementation method, what would it be and why?
2. Why do you believe that General Motors has allowed its 7,000 North American dealerships to choose different dealership management systems? What are the advantages to allowing this freedom of choice? What are the disadvantages to such an approach?
3. The new inventory management system provides a satisfactory repair experience 96 percent of the time, up from 67 percent. While that increase is both strong and good, is 96 percent really that good? Why or why not? What quantitative approach would you use to justify system enhancements to improve the 96 percent to 98 percent? To 100 percent?
4. If you refer back to Chapter 4 and decision support systems, how would you characterize the decision-making process of determining how many parts to keep on hand? Is that mainly a recurring decision or a nonrecurring decision? Is that mainly a structured decision or a nonstructured decision? For both the latter questions, justify your answers.
5. The new inventory management system reduces the overall costs of parts inventory and, at the same time, increases customer satisfaction. How is this an example of a bottom-line initiative? How is this an example of a top-line initiative? If necessary, refer back to Chapter 1.

CLOSING CASE STUDY TWO

SHOULD AN ORGANIZATION OUTSOURCE SECURITY?

It may seem like an odd question, one that gets an immediate "no," but think again. Many businesses realize that they do not possess the expertise, time, or money to detect, identify, isolate, and stop the hundreds of hackers, viruses, worms, and other malcontents that daily bombard most IT systems. Moreover, most organizations wouldn't identify "security" as a unique core competency, so there is some argument for outsourcing it. The following statistics come from a survey performed by InformationWeek/Accenture Global Information Security. They identify the percentage of respondents who use an outsourcer for some form of security.

- Firewall management—29 percent
- Intrusion detection management—25 percent
- Messaging protection—18 percent
- Security strategy development—15 percent
- Security governance—11 percent
- End device security—11 percent

The numbers show that many companies are willing to outsource IT-related security.

BOILING SPRINGS SAVINGS BANK

Most likely, the number one reason why companies outsource security functions is because they lack the resources and expertise to handle it themselves. According to Ken Emerson, director of strategic planning and CIO at Boiling Springs Savings Bank in New Jersey, "In the security world, it's a game of catch-up. I couldn't possibly throw enough resources at it internally." Ken contracted Perimeter Internetworking to provide security for e-mail and handle the function of intrusion detection and prevention. As Ken explained it further, "I didn't feel like I had the necessary knowledge on my staff, especially with the rapidly growing volume of spam."

But before hiring Perimeter, Ken thought about his customers who rely on Boiling Springs to keep their money safe. So, he did a background check on Perimeter and learned that it had passed the Statement of Auditing Standards No. 70, an in-depth audit of a service provider's control activities. He also found that

none of the other security firms he was evaluating had received that sort of certification.

Perimeter electronically linked to Boiling Springs's systems and monitored all e-mail traffic and intrusion attempts. It even found a worm on a specific Boiling Springs PC and notified the bank so it could shut down the infected computer.

KETTERING MEDICAL CENTER NETWORK

Kettering Medical Center Network, a group of 50 health care facilities in the Dayton, Ohio, area, turned over some of its IT security to Symantec. Specifically, Kettering contracted Symantec to analyze all of its data collected by Check Point Technologies and Cisco Systems firewalls. The focus here was to protect remote physicians' offices that are on the network. These types of remote access areas are prime targets for infiltrating a much larger network. As Bob Burritt, IS network and technology manager at Kettering, explained it, "We need to be concerned if someone is trying to do a port scan against our systems or if our network contains ad bots or spy bots trying to communicate out."

If someone did succeed in penetrating Kettering's network and shutting down the system, the results would be catastrophic. Not only could doctors and other health care professionals not communicate with each other and share information, Kettering would lose approximately $1 million a day if it couldn't bill patients or its health care partners or collect fees.

Boiling Springs Savings Bank and Kettering Medical Center Network are just two examples of the many companies that are effectively outsourcing some portion of IT security. Overall among U.S. companies, 25 percent are now outsourcing some aspect of IT security.[13,14]

Questions

1. If you were developing a new system using the traditional systems development life cycle (SDLC), at what point would you need to identify that you needed to outsource some aspect of IT security?
2. In reference to the first question, how would you continue with the in-house systems development effort and, at the same time, carry on the process of outsourcing IT security with another company?
3. Boiling Springs Savings Bank did a background check on Perimeter before hiring it. Search the Web for resources than can help an organization do background checks on IT security firms. What did you find? Did you find a couple of Web sites or certification organizations that offer some guarantee of IT security firms? If so, whom did you find?
4. Turning over IT security to an outside organization is tantamount to giving another organization complete access to all your systems and information. What stipulations would you include in a service level agreement with an IT security outsourcer to ensure that it didn't exploit the openness of your systems and steal strategic and sensitive information?
5. Do some research on the Web for companies that specialize in IT security outsourcing besides Perimeter and Symantec. Whom did you find? Do they seem to be reputable? Do they include a list of clients you can contact for references?

Key Terms and Concepts

Agile methodology, 286
Analysis phase, 279
Business requirement, 279
Component-based development (CBD), 283
Critical success factor (CSF), 278
Design phase, 280
Development phase, 280
Extreme programming (XP) methodology, 284
Feature creep, 278
Help desk, 282
Implementation phase, 281
Insourcing, 276
Integration testing, 281
Invisible backlog, 289
Joint application development (JAD), 279
Maintenance phase, 282
Nearshore outsourcing, 298
Offshore outsourcing, 298
Online training, 281
Onshore outsourcing, 298
Outsourcing, 276
Parallel implementation, 281
Phased implementation, 282
Pilot implementation, 282
Planning phase, 278
Plunge implementation, 281
Project manager, 278
Project milestone, 278
Project plan, 278
Project scope document, 278
Proof-of-concept prototype, 291
Prototype, 290
Prototyping, 290

Rapid application development (RAD) (rapid prototyping) methodology, 283
Request for proposal (RFP), 296
Requirements definition document, 279
Scope creep, 278
Self sourcing (end-user development), 276
Selling prototype, 291
Service level agreement (SLA), 298
Service level objective (SLO), 298
Service level specification (SLS), 298
Service-oriented architecture (SOA or SoA), 286
Sign-off, 279
Systems development life cycle (SDLC), 276
System testing, 281
Technical architecture, 280
Test condition, 281
Testing phase, 281
Unit testing, 281
User acceptance testing (UAT), 281
User documentation, 281
Waterfall methodology, 276
Workshop training, 281

Short-Answer Questions

1. What are the three primary groups of people who undertake the systems development process?
2. What is the systems development life cycle?
3. What are scope creep and feature creep?
4. How do the four implementation methods differ?
5. What is component-based development?
6. How are component-based development and a service-oriented architecture related?
7. Why do organizations prototype?
8. What are the advantages of selfsourcing?
9. What is the difference between a selling prototype and a proof-of-concept prototype?
10. What is the role of a service level agreement (SLA) in outsourcing?
11. What are the three geopolitical forms of outsourcing?

Assignments and Exercises

1. **REQUEST FOR PROPOSAL** A request for proposal (RFP) is a formal document that describes in detail your logical requirements for a proposed system and invites outsourcing organizations to submit bids for its development. Research the Web and find three RFP examples. Briefly explain in a one-page document what each RFP has in common and how each RFP is different.
2. **MAKING THE WHO DECISION** Complete the table below by answering yes, no, or maybe in the columns of insource, selfsource, and outsource for each systems development condition listed on the left.

	INSOURCE	SELFSOURCE	OUTSOURCE
The system will support a unique core competency			
Cost is an overriding consideration			
Time-to-market is critical			
You possess the necessary expertise			
Organizational control of the system is critical			
The system will support a common business function			
Gaining and having technical expertise is part of your strategic plan			
The system will support only a very few users			

3. **YOUR RESPONSIBILITIES DURING EACH STEP IN THE SDLC** During insourcing, you have many responsibilities because you're a business process expert, liaison to the customer, quality control analyst, and manager of other people. According to which step of the SDLC you're in, your responsibilities may increase or decrease. In the table below, determine the extent to which you participate in each SDLC step according to your four responsibilities. For each row you should number the SDLC steps 1 through 7, with a 1 indentifying the step in which your responsibility is the greatest and a 7 identifying the step in which your responsibility is the least.

	SDLC STEP						
	PLANNING	ANALYSIS	DESIGN	DEVELOPMENT	TESTING	IMPLEMENTATION	MAINTENANCE
Business process expert							
Liaison to the customer							
Quality control analyst							
Manager of other people							

4. **CONSTRUCTION AND THE SDLC** The systems development life cycle is often compared to the activities in the construction industry. Fill in the following chart listing some of the activities performed in building a house and how they relate to the different SDLC steps.

SDLC	Activities for Building a Home
Planning	
Analysis	
Design	
Development	
Testing	
Implementation	
Maintenance	

Discussion Questions

1. Why is it important to develop a logical model of a proposed system before generating a technical architecture? What potential problems would arise if you didn't develop a logical model and went straight to developing the technical design?

2. If you view systems development as a question-and-answer session, another question you could ask is, "Why do organizations develop IT systems?" Consider what you believe to be the five most important reasons organizations develop IT systems. How do these reasons

relate to topics in the first five chapters of this book?

3. Your company has just decided to implement a new financial system. Your company's financial needs are almost the same as those of all the other companies in your industry. Would you recommend that your company purchase an existing system or build a custom system? Would you recommend your company use end-user development or outsource the new system?
4. There are seven phases in the systems development life cycle. Which one do you think is the hardest? Which one do you think is the easiest? Which one do you think is the most important? Which one do you think is the least important? If you had to skip one of the phases, which one would it be and why?
5. You are talking with another student who is complaining about having to learn about the systems development life cycle, because he is not going to work in an IT department. Would you agree with this student? What would you say to him to convince him that learning about the systems development life cycle is important no matter where he works?
6. A company typically has many systems it wants to build, but unfortunately it usually doesn't have the resources to build all the systems. How does a company decide which systems to build?
7. People often think that a system is complete once it is implemented. Is this true? What happens after a system is implemented? What can you do to ensure the system continues to meet the knowledge workers' needs?

CHAPTER PROJECTS

Group Projects

- Executive Information System Reporting: Political Campaign Finance Consultants (p. 466)
- Using Relational Technology to Track Projects: Foothills Construction (p. 469)
- Assessing the Value of Outsourcing Information Technology: Creating Forecasts (p. 472)
- Creating a Decision Support System: Buy versus Lease (p. 476)

e-Commerce Projects

- Best in Computer Resources and Statistics (p. 488)
- Meta Data (p. 489)
- Finding Hosting Services (p. 493)
- Searching for Shareware and Freeware (p. 498)
- Researching Storefront Software (p. 498)