Planning and developing large information systems is akin to a multilevel chess game—it is a very demanding task for even the most experienced professionals. Information systems tend to grow in complexity as they grow in size. Systems development is the entire set of activities needed to construct an information systems solution to a business problem or opportunity. This chapter discusses a variety of approaches to systems development. We begin with planning. Next we present the concept of the systems development life cycle (SDLC), which provides the framework for all activities in the development process. Third, we discuss other methods for systems development that organizations may use in standalone fashion or in conjunction with the SDLC, including prototyping, rapid application development, and object-oriented development. The next section examines alternatives to in-house development, including end-user development, acquisition of systems from vendors, and use of application service providers (ASPs). The chapter closes with a look at Internet- and intranet-based systems development.
EVEN IN TOUGH TIMES, IT PROJECTS AND PEOPLE REMAIN A PRIORITY FOR BOISE CASCADE

With revenue down about 5 percent, Boise Cascade Corp. was having a difficult year, but that didn’t mean the $7.81 billion forest-products and office-supplies company would be shelving important IT projects, said company chairman and CEO George Harad. In fact, the Boise, Idaho, company increased its IT budget slightly for the year, to about $60 million, and the IT department had several initiatives in the works. The company is convinced its IT projects will help it achieve two key goals: increased efficiency and greater responsiveness to customers. In an industry that generally takes a wait-and-see approach to IT, Boise Cascade stands out for its early identification and pursuit of strategic IT projects it believes will yield measurable improvements in its business.

The Business Problem

Among the projects in progress is One Boise, an initiative to unify databases used by Boise Cascade’s office-products-distribution business. The project reflects the company’s overall philosophy of keeping its data and IT systems and processes “clean and simple,” as Bob Egan, VP of IT, describes it. The company tackles IT projects methodically, from the ground up, so that it has a solid base on which to build top-level user applications and services, which are more flexible as a result. For instance, the underlying organization of its databases and its company-wide use of applications and shared utilities mean Boise Cascade can more easily change its user interfaces and make applications available via the Web. “We’re conservative in approach but not in outcomes,” Egan says.

Another tactic of the IT department is to work with executives from all the company’s business units. For instance, for each major application it implements, the IT staff works with a senior-level executive from the affected operating unit to identify priorities and ensure that the new technologies will be cost effective.

In the IT department itself, Egan’s own priorities are clear: He treats Boise’s IT staff as a top asset. The department maintains a healthy attitude about work-life balance, including an emphasis on 40-hour work weeks, with rare exceptions for major projects. The biggest project the department took on in the past year was an overhaul of Boise Cascade’s network services. The project included implementing new directory services based on Microsoft Active Directory, putting in new firewalls, installing a virtual private network, and building a subnetwork for the e-commerce sites it operates for its paper and wood-products divisions.

At the same time, Boise upgraded the groupware tools it uses, including Lotus Notes, Exchange 2000, and NetMeeting. It also installed quality-of-service software that will let it monitor and control how the company uses its network bandwidth. The upgrade, whose goal was to prepare for anticipated increases in traffic from wireless devices (such as pagers, handhelds, notebooks equipped with wireless modems, and wireless LANs) and e-commerce, gives Boise a load-balanced network for its e-commerce sites and hosted applications. In the future, it will supply authentication and security services that will provide essential access controls as the company establishes more connections among its network, the Internet, and customer networks.

The IT Solution

Egan cascade. com
The company has managed to minimize IT turnover rate to an unusually low 3 percent annually for the past five years, has achieved its major IT development goals, and is achieving the business goals that the IT supports. In implementing e-commerce systems for its major businesses, Boise Cascade set out to provide customers with Web-based, real-time access to the same product-ordering and delivery systems Boise Cascade uses internally. Customers can do some custom product configuration, create and save catalog templates, customize their views of product data, and arrange for deliveries, Egan says. “All the business logic needed for custom product configuration and delivery is available over the Web,” adds Egan, “and three-fourths of customer inquiries can be handled through self-service over the Web.”

Source: John Rendleman and Diane Rezendes Khirallah, informationweek.com (September 17, 2001).

Because information systems are so critical to competitive advantage in so many companies, their timely and careful development are very high priorities. Just how much of a priority becomes more clear when development budgets are threatened in tough economic circumstances. This strong prioritization of IT shows that Boise Cascade understands the major role IT plays in achievement of its goals of increased efficiency and enhanced customer relations. In addition, the company also has an enlightened perspective on how to manage its IT personnel, who represent crucial knowledge resources that are not easily duplicated or substituted for. The case also demonstrates the increasing importance of Web and e-commerce application development, paying benefits of increased customer service and customer retention. Finally, we see that the firm anticipates that mobile computing will be an important aspect of how it interacts with its customers, and how this new communication channel should interface with its other e-commerce platforms.

14.1 INFORMATION SYSTEMS PLANNING

Planning an information system doesn’t start with bits and bytes, mips, or a Web site. Rather, it starts with gaining a holistic perspective on what the firm aims to achieve and how it will do so. Systems development is the entire set of activities needed to construct an information systems solution to a business problem or opportunity. A key component is information systems planning, which begins with the strategic plan of the organization, as shown in Figure 14.1. The organization’s strategic plan states the firm’s overall mission, the goals that follow from the mission, and the broad steps necessary to reach these goals. A key input into the organization’s strategic plan is an assessment of the current state of the organization, in which the current performance of the firm is compared to the previous strategic plan. The mission states what the organization ideally wants to become or to create at some future point in time. The strategic planning process matches the organization’s objectives and resources to its changing markets and opportunities.

The organizational strategic plan and the existing IT architecture provide the inputs in developing the information systems strategic plan. The IT architecture delineates the way an organization’s information resources should be used to accomplish its mission. It encompasses both technical and managerial aspects of information resources. The technical aspects include hardware and operating systems, networking, data and data management systems, and applications software. The managerial
aspects specify how managing the IS department will be accomplished, how functional area managers will be involved, and how IS decisions will be made.

**The IS Strategic Plan**

The **IS strategic plan** is a set of long-range goals that describe the IT architecture and major IS initiatives needed to achieve the goals of the organization. The IS strategic plan must meet three objectives:

- **It must be aligned with the organization’s strategic plan.**
- **It must provide for an IT architecture that enables users, applications, and databases to be seamlessly networked and integrated.**
- **It must efficiently allocate IS development resources among competing projects, so the projects can be completed on time, within budget, and have the required functionality.**

The IS strategic plan states the mission of the IS department, which defines the department’s underlying purpose. The mission helps to answer questions relating to three major issues:

- **Efficiency.** Does the IS function help the organization reach its goals with minimum resources?
- **Effectiveness.** Does the IS function help the functional area managers (and executives) do the right things?
- **Competitiveness.** Does the IS function engage in projects that will enhance the organization’s competitive position?

The mission of the IS department requires a great deal of input from all of the organization’s functional area managers, and often from higher organizational officers as...
This input will help to define the appropriate role of the IS department in accomplishing the organization’s goals.

**The IS Operational Plan**

The IS strategic plan may require a new IT architecture, or the existing IT architecture may be sufficient. In either case, the IS strategic plan leads to the **IS operational plan**, which is a clear set of projects that will be executed by the IS department and by functional area managers in support of the IS strategic plan.

A typical IS operational plan contains the following elements:

- **Mission:** The mission of the IS function
- **IS environment:** A summary of the information needs of the functional areas and of the organization as a whole
- **Objectives of the IS function:** The IS function’s current best estimate of its goals
- **Constraints on the IS function:** Technological, financial, and personnel limitations on the IS function
- **Long-term systems needs:** A summary of the systems needed by the company and the IS projects selected to reach organizational goals
- **Short-range plan:** A list of present projects, and a detailed plan of projects to be developed or continued during the current year

**Before you go on . . .**

1. Where does information systems planning begin?
2. What is the relationship of the IS strategic plan to the organization’s strategic plan?

## 14.2 The Traditional Systems Development Life Cycle (SDLC)

The **systems development life cycle (SDLC)** is the traditional systems development method used by most organizations today. The SDLC is a structured framework that consists of sequential processes by which information systems are developed. As shown in Figure 14.2, these include systems investigation, systems analysis, systems design, programming, testing, implementation, operation, and maintenance. These processes, in turn, consist of well-defined tasks. Some of these tasks are present in most projects, whereas others are present in only certain types of projects. That is, large projects typically require all the tasks, whereas smaller development projects may require only a subset of the tasks.

Other models for the SDLC may contain more or fewer than the eight stages we present here. The flow of tasks, however, remains largely the same, regardless of the number of stages. In the past, developers used the **waterfall approach** to the SDLC, in which tasks in one stage were completed before the work proceeded to the next stage. Today, systems developers go back and forth among the stages as necessary.

Systems development projects produce desired results through team efforts. Development teams typically include users, systems analysts, programmers, and
Users are employees from all functional areas and levels of the organization who will interact with the system, either directly or indirectly. Direct interaction means that users will make hands-on use of the system, and indirect interaction means that users will use the outputs from the system. **Systems analysts** are information systems professionals who specialize in analyzing and designing information systems. **Programmers** are information systems professionals who modify existing computer programs or write new computer programs to satisfy user requirements. **Technical specialists** are experts on a certain type of technology, such as databases or telecommunications. All people who are affected by changes in information systems (users and managers, for example) are known as **systems stakeholders**, and are typically involved by varying degrees and at various times in the systems development.

In the remainder of this section we will look at each of the processes in the SDLC.

### Systems Investigation

Systems development professionals agree that the more time invested in understanding the business problem to be solved, in understanding technical options for systems, and in understanding problems that are likely to occur during development, the greater the chance of actually successfully solving the (correct) problem. For these reasons, systems investigation begins with the business problem. Problems (and opportunities) often require not only understanding them from the internal point of view, but also seeing them as organizational partners (suppliers or customers) would see them. Another useful perspective is that of competitors. How have they responded to similar situations, and what outcomes and additional opportunities have materialized? Creativity and out-of-the-box thinking can pay big dividends when isolated problems can be recognized as systemic failures whose causes cross organizational boundaries. Once these perspectives can be gained, those involved can also begin to better see the true scope of the project and propose possible solutions. Then an initial assessment of these proposed system solutions can begin.
Feasibility studies. The next task in the systems investigation stage is the feasibility study. The feasibility study determines the probability of success of the proposed systems development project and assesses the project’s technical, economic, and behavioral feasibility. The feasibility study is critically important to the systems development process because, done properly, the study can prevent organizations from making costly mistakes (like creating systems that will not work, will not work efficiently, or that people can’t or won’t use). The various feasibility analyses also give the stakeholders an opportunity to decide what metrics to use to measure how a proposed system (and later, a completed system) meets their various objectives.

Technical feasibility. Technical feasibility determines if the hardware, software, and communications components can be developed and/or acquired to solve the business problem. Technical feasibility also determines if the organization’s existing technology can be used to achieve the project’s performance objectives.

Economic feasibility. Economic feasibility determines if the project is an acceptable financial risk and if the organization can afford the expense and time needed to complete the project. Economic feasibility addresses two primary questions: Do the benefits outweigh the costs of the project? Can the project be completed as scheduled?

Three commonly used methods to determine economic feasibility are breakeven analysis, return on investment (ROI), and net present value (NPV). Breakeven analysis determines the amount of time required for the cumulative cash flow from a development project to equal its initial and ongoing investment. Return on investment is the ratio of the net cash inflows from a project divided by the cash outflows of the project. The net present value is the net amount by which project savings exceed project expenses, after allowing for the cost of capital and the time value of money. The cost of capital is the minimum desired rate of return on an investment and is the average cost of funds used to finance the operations of the business. NPV also takes the time value of money into account. The time value of money means that a sum received at a future date is not worth as much as a sum received today, because a sum of money you have today can be invested to earn interest. (For more details on how to compute NPV, see the Web site.)

Determining return on investment in IT projects is rarely straightforward, but it often is essential. Part of the difficulty stems from the fact that the proposed system or technology may be “cutting edge,” and there may be no previous evidence of what sort of ROI is to be expected. IT’s About Business Box 14.1 gives a good example of both the organizational necessity and the inherent challenges of ROI analysis of IT projects.

Behavioral feasibility. Behavioral feasibility addresses the human issues of the project. All systems development projects introduce change into the organization, and people generally fear change. In fact, employees may overtly or covertly resist a new system. Overt resistance may take the form of sabotaging the new system (e.g., entering data incorrectly) or deriding the new system to anyone who will listen. Covert resistance typically occurs when employees quietly refuse to use the new system. They simply do their jobs using their old methods.

A primary manifestation of change brought about by the introduction of new systems is changes in organizational information flows. These changes affect the information gatekeepers, who are the stakeholders with responsibility for information that is important to the organization or one of its units. Because “knowledge is power,” these gatekeepers want to remain in control of collecting, processing, and disseminating this information.
A more positive and pragmatic concern of behavioral feasibility is assessing the skills and training needs that often accompany a new information system. In some organizations, a proposed system may require mathematical or linguistic skills beyond what the workforce currently possesses. In others, a workforce may simply need additional skill building rather than remedial education. Behavioral feasibility is as much about “can they use it” as it is about “will they use it.”

After the feasibility analysis, a “Go/No-Go” decision is reached. The functional area manager for whom the system is to be developed and the project manager sign off on the decision. If the decision is “No-Go,” the project is put on the shelf until conditions are more favorable, or the project is discarded. If the decision is “Go,” then the systems development project proceeds and the systems analysis phase begins.

---

**Box 14.1 Sears demands ROI analysis for hand-held computer system**

Dennis Honan, an IS executive at retailer Sears, Roebuck & Co., is a veteran of what he calls the company’s “ROI culture.” Honan, VP of Information Systems for Sears’s Home Services business, got approval to spend some $20 million to equip the unit’s 14,000-person service staff with handheld PCs. The overriding goal of the project was to improve the efficiency of Sears’s service technicians. Not only were they to be given handheld computers, but the devices would also be linked by wireless WANs to Sears’s databases. Honan projected an average 6 to 8 percent gain in the technicians’ productivity, mainly because the setup would let them request price estimates, check availability for appliance parts, place orders, receive software upgrades, and get job-schedule updates from wherever they were working. That, in turn, would let technicians complete more calls a day.

Also, when customers cancel or reschedule service calls—something that happens up to 100 times a day in some districts—technicians and dispatchers could learn about the changes and make schedule adjustments almost immediately. In the past, they’d be paged, have to find a pay phone, then wait for instructions. “Here was an opportunity to computerize everything, eliminate paper service orders, and have the ability to communicate almost instantaneously with the technicians,” says Vince Accardi, director of process management.

The project sounded good enough to go, but presenting a formal ROI analysis was “absolutely essential” to the approval process, Honan says. A dd s Joseph Smialowski, Sears’s senior VP and CIO and a key player in the approval of all types of investments at the retailer, “All our projects—whether it’s opening new stores or buying new systems—have to compete for the capital that’s available. There are really no projects that can slip through without going through a quantitative analysis.”

To justify the handheld PC initiative, Home Services managers used a cost–benefit measure to determine net annual savings. To illustrate the longer-term benefits of the investment, they also calculated the net present value (NPV) of cash flows over a five-year period.

Home Services presented the expected benefits in terms of expected annual savings for Sears. The project proposal then went through a multilevel evaluation process: first within Home Services, and next at the Company’s strategic planning level. The plan was evaluated for technical soundness, accuracy of the cost estimates, and to see if it fit Sears’s business model and enterprise architecture.

The proposal then went to Sears’s finance committee, which includes the company’s CEO, the chief financial officer (CFO), the CIO, and two business presidents. They approved the project. It was then rolled out, first in test markets, then district by district.


**Questions**

1. IS projects at Sears are not exempt from demanding ROI analysis. Why would they be, or why should they be, at any other business organization?

2. Given the enormous expense of the project—which is just one small part of the overall IT infrastructure—what would you estimate the yearly IT budget to be at a firm like Sears?
As noted above, although originally conceived to be discrete consecutive steps, current thinking holds that certain activities in the SLDC should be repeated as necessary. The most common of these are feasibility analyses, often repeated at the ends of the analysis, design, programming, and testing phases. The reason is simply that the accuracy of any feasibility study substantially increases as developers learn more about all aspects of the project.

Systems Analysis

Once a development project has the necessary approvals from all participants, the systems analysis stage begins. Systems analysis is the examination of the business problem that the organization plans to solve with an information system. This stage defines the business problem, identifies its causes, specifies the solution, and identifies the information requirements that the solution must satisfy. Understanding the business problem requires understanding the various processes involved. These can often be quite complicated and interdependent. Analysts have a variety of tools that support this analysis. (For an example of a modern process modeling tool and how it is used, see the case about the city of Austin, Texas, at the book’s Web site.)

Organizations have three basic solutions to any business problem: (1) Do nothing and continue to use the existing system unchanged. (2) Modify or enhance the existing system. (3) Develop a new system. The main purpose of the systems analysis stage is to gather information about the existing system, to determine which of the three basic solutions to pursue, and to determine the requirements for an enhanced or new system. The end product (the “deliverable”) of this stage is a set of systems requirements.

Arguably the most difficult task in systems analysis is to identify the specific information requirements that the system must satisfy. Information requirements specify what information, how much information, for whom, when, and in what format. Systems analysts use many different techniques to obtain the information requirements for the new system. These techniques include structured and unstructured interviews with users, and direct observation. Structured interviews have questions written in advance. In unstructured interviews, the analyst does not have predefined questions but uses experience to elicit the problems of the existing system from the user. With direct observation, analysts observe users interacting with the existing system.

In developing information requirements, analysts must be careful not to let any preconceived ideas they have interfere with their objectivity. Further, analysts must be unobtrusive, so that users will interact with the system as they normally would.

There are problems associated with eliciting information requirements, regardless of the method used by the analyst. First, the business problem may be poorly defined. Second, the users may not know exactly what the problem is, what they want, or what they need. Third, users may disagree with each other about business procedures or even about the business problem. Finally, the problem may not be information related, but may require other solutions, such as a change in management or additional training.

The systems analysis stage produces the following information:

- Strengths and weaknesses of the existing system
- Functions that the new system must have to solve the business problem
- User information requirements for the new system

Armed with this information, systems developers can proceed to the systems design stage.
Systems Design

Systems analysis describes what a system must do to solve the business problem, and systems design describes how the system will accomplish this task. The deliverable of the systems design phase is the technical design that specifies the following:

- System outputs, inputs, and user interfaces
- Hardware, software, databases, telecommunications, personnel, and procedures
- How these components are integrated

This output is the set of system specifications.

Systems design encompasses two major aspects of the new system:

- **Logical systems design** states what the system will do, with abstract specifications.
- **Physical systems design** states how the system will perform its functions, with actual physical specifications.

Logical design specifications include the design of outputs, inputs, processing, databases, telecommunications, controls, security, and IS jobs. Physical design specifications include the design of hardware, software, database, telecommunications, and procedures. For example, the logical telecommunications design may call for a wide-area network connecting the company’s plants. The physical telecommunications design will specify the types of communications hardware (e.g., computers and routers), software (e.g., the network operating system), media (e.g., fiber optics and satellite), and bandwidth (e.g., 100 Mbps).

When both these aspects of system specifications are approved by all participants, they are “frozen.” That is, once the specifications are agreed upon, they should not be changed. However, users typically ask for added functionality in the system (called **scope creep**), for several reasons. First, as users more clearly understand how the system will work and what their information and processing needs are, they see additional functions that they would like the system to have. Also, as time passes after the design specifications are frozen, business conditions often change, and users ask for added functionality. Because scope creep is expensive, project managers place controls on changes requested by users. These controls help to prevent **runaway projects**—systems development projects that are so far over budget and past deadline that they must be abandoned, typically with large monetary loss.

Programming

Systems developers utilize the design specifications to acquire the software needed for the system to meet its functional objectives and solve the business problem. As discussed in Chapter 4, organizations may buy the software or construct it in-house.

Although many organizations tend to purchase packaged software, many other firms continue to develop custom software in-house. For example, Wal-Mart and Eli Lilly build practically all their software in-house. The chief benefit of custom development is systems that are better suited than packaged applications to an organization’s new and existing business processes. For many organizations, custom software is more expensive than packaged applications. However, if a package does not closely fit the company needs, the savings are often diluted when the information systems staff or consultants must extend the functionality of the purchased packages.

If the organization decides to construct the software in-house, then **programming** involves the translation of the design specifications into computer code. This process can be lengthy and time-consuming, because writing computer code remains as much
an art as a science. Large systems development projects can require hundreds of thousands of lines of computer code and hundreds of computer programmers. In such projects, programming teams are used. These teams often include functional area users to help the programmers focus on the business problem at hand.

In an attempt to add rigor (and some uniformity) to the programming process, programmers use structured programming techniques. These techniques improve the logical flow of the program by decomposing the computer code into modules, which are sections of code (subsets of the entire program). This modular structure allows for more efficient and effective testing, because each module can be tested by itself. These structured programming techniques include the following restrictions:

- Each module has one, and only one, function.
- Each module has only one entrance and one exit. That is, the logic in the computer program enters a module in only one place and exits in only one place.
- There are no GO TO statements allowed.

For example, a flowchart for a simple payroll application might look like the one shown in Figure 14.3. The figure shows the only three types of structures that are used in structured programming: sequence, decision, and loop. In the sequence structure, program statements are executed one after another until all the statements in the sequence have been executed. The decision structure allows the logic flow to branch, depending on certain conditions being met. The loop structure enables the software to
execute the same program, or parts of a program, until certain conditions are met (e.g., until the end of the file is reached, or until all records have been processed).

As already noted, structured programming enforces some standards about how program code is written. This approach and some others were developed not only to improve programming, but also to standardize how a firm’s various programmers do their work. This helps ensure that all the code developed by different programmers will work together. Even with these advances, however, programming can be difficult to manage. IT’s About Business 14.2 gives an example of these sorts of challenges and current approaches for dealing with them.

**Testing**

Thorough and continuous testing occurs throughout the programming stage. Testing checks to see if the computer code will produce the expected and desired results under certain conditions. Testing requires a large amount of time, effort, and expense to do properly. However, the costs of improper testing, which could possibly lead to a system that does not meet its objectives, are enormous.

Testing is designed to detect errors (“bugs”) in the computer code. These errors are of two types: syntax errors and logic errors. **Syntax errors** (e.g., a misspelled word or a misplaced comma) are easier to find and will not permit the program to run. **Logic errors** permit the program to run, but result in incorrect output. Logic errors are more

---

**Box 14.2: Belk Inc. tracks development progress, reaps rewards**

With limited management resources available and the pressure to deploy new business solutions quickly, measuring productivity in systems development often becomes a low priority for many organizations. IT departments at smaller companies in particular seem reluctant to institute policies to track the performance of development projects. Some companies, however, have been forced to adopt productivity measurement methods, and are reaping rewards for doing so.

For example, national retailer Belk Inc. had to adopt productivity metrics as a means of reducing devastating system failures. Conda Lashley, the veteran IT consultant that Belk hired, was used to nursing client organizations through crashes that periodically downed their systems. But nothing had prepared Lashley for the failure rate at Belk. Soon after joining the company as senior VP for systems development, Lashley discovered that Belk’s batch systems went down an astounding 800 times a month. The Charlotte, North Carolina, outfit, a private company with estimated annual revenue of $1.7 billion, paid a heavy price for the constant bandaging: In 1997, Belk spent $1.1 million of its $30 million IT budget on unplanned maintenance.

To steady the systems, Lashley instituted a series of tracking measures. Programmers began logging their time. Required software functions were carefully counted in application development projects. Belk compared its cycle time, defect rates, and productivity with competitors’ figures. And systems managers were required to draw up blueprints for reducing the crashes—with the results reviewed in their performance evaluations.

The transition to tracking the IT department’s performance was painful but worthwhile. Belk’s systems became more stable—monthly disruptions dropped to 480 incidents, a figure Lashley hopes to slash by another 30 percent. Unplanned maintenance costs also have been brought under control, with initial cuts in unplanned maintenance expenses of $800,000.

**Questions**

1. Why do you think IT departments are reluctant to use productivity measures, beyond the basic reason of resource scarcity?
2. What is it about the software development process that makes its progress difficult to measure or estimate?
3. What do you think about the notion that one cannot truly understand something that one cannot measure?
difficult to detect, because the cause is not obvious. The programmer must follow the flow of logic in the program to determine the source of the error in the output.

As software increases in complexity, the number of errors increases, making it almost impossible to find them all. This situation has led to the idea of "good enough software," software that developers release knowing that errors remain in the code. However, the developers feel that the software will still meet its functional objectives. That is, they have found all the show-stopper bugs, errors that will cause the system to shut down or will cause catastrophic loss of data.

Implementation

Implementation is the process of converting from the old system to the new system. Organizations use four major conversion strategies: parallel, direct, pilot, and phased.

In a parallel conversion process, the old system and the new system operate simultaneously for a period of time. That is, both systems process the same data at the same time, and the outputs are compared. This type of conversion is the most expensive, but also the least risky. Most large systems have a parallel conversion process to lessen the risk.

In a direct conversion process, the old system is cut off and the new system is turned on at a certain point in time. This type of conversion is the least expensive, but the most risky if the new system doesn’t work as planned. Few systems are implemented using this type of conversion, due to the risk involved.

The pilot conversion process introduces the new system in one part of the organization, such as in one plant or in one functional area. The new system runs for a period of time and is assessed. After the new system works properly, it is introduced in other parts of the organization.

The phased conversion process introduces components of the new system, such as individual modules, in stages. Each module is assessed, and, when it works properly, other modules are introduced, until the entire new system is operational.

Operation and Maintenance

After conversion, the new system will operate for a period of time, until (like the old system it replaced) it no longer meets its objectives. Once the new system’s operations are stabilized, audits are performed during operation to assess the system’s capabilities and determine if it is being used correctly.

Systems need several types of maintenance. The first type is debugging the program, a process that continues throughout the life of the system. The second type is updating the system to accommodate changes in business conditions. Examples here include adjusting to new governmental regulations and managing the Y2K problem. These corrections and upgrades usually do not add any new functionality; they are necessary in order for the system to continue meeting its objectives. The third type of maintenance adds new functionality to the system. This process involves adding new features to the existing system without disturbing its operation.

Before you go on . . .

1. What are the basic steps in the traditional systems development life cycle?
2. What is the purpose of structured programming?
3. What are the common strategies for implementing new systems?
Most organizations use the traditional systems development life cycle because it has three major advantages: control, accountability, and error detection. An important issue in systems development is that the later in the development process that errors are detected, the more expensive they are to correct. The structured sequence of tasks and milestones in the SDLC thus makes error detection easier and saves money in the long run.

However, the SDLC does have disadvantages. By its structured nature, it is relatively inflexible. It is also time-consuming, expensive, and discourages changes to user requirements once they have been established. Development managers who must develop large, enterprisewide applications therefore find it useful to mix and match development methods and tools in order to reduce development time, complexity, and costs. These methods and tools include prototyping, rapid application development, integrated computer-assisted software engineering (ICASE) tools, and object-oriented development. Although all these methods and tools can reduce development time, none can consistently deliver in all cases. They are perhaps best considered as options to complement or replace portions of the SLDC. This section discusses each of these methods and tools.

**Prototyping**

Using the prototyping approach, systems developers first obtain only a general idea of user requirements. That is, the developers do not try to obtain a complete set of user specifications for the system and do not plan to develop the system all at once. Instead, they quickly develop a prototype, which either contains parts of the new system of most interest to the users, or is a small-scale working model of the entire system. The prototype is given to the users, who are able to use it and make suggestions for improving it. The developers review the prototype with the users and use the suggestions to refine the prototype. This process continues through several iterations until either the users approve the system or it becomes apparent that the system cannot meet users’ needs. If the system is viable, the developers can use the prototype on which to build the full system. Developing screens that a user will see and interact with is a typical use of prototyping. (See the book’s Web site for a model that shows the prototyping process.)

The main advantage of prototyping is that this approach speeds up the development process. In addition, prototyping gives users the opportunity to clarify their information requirements as they review iterations of the new system. Prototyping is useful in the development of decision support systems and executive information systems, where user interaction is particularly important.

Prototyping also has disadvantages. It can largely replace the analysis and design stages of the SDLC in some projects. As a result, systems analysts may not produce adequate documentation for the programmers. This lack of documentation can lead to problems after the system becomes operational and needs maintenance. In addition, prototyping can result in an excess of iterations, which can consume the time that prototyping should be saving.

**Joint Application Design (JAD)**

Joint application design (JAD) is a group-based method for collecting user requirements and creating system designs. JAD is most often used within the systems analysis and systems design stages of the SDLC.
In the traditional SDLC, systems analysts interview or directly observe potential users of the new information system individually to understand each user’s needs. The analysts will obtain many similar requests from users, but also many conflicting requests. The analysts must then consolidate all requests and go back to the users to resolve the conflicts, a process that usually requires a great deal of time.

In contrast to the SDLC requirements analysis, JAD has a group meeting in which all users meet simultaneously with analysts. During this meeting, all users jointly define and agree upon systems requirements. This process saves a tremendous amount of time.

The JAD approach to systems development has several advantages. First, the group process involves more users in the development process while still saving time. This involvement leads to greater support for, and acceptance of, the new system and can produce a system of higher quality. This involvement also may lead to easier implementation of the new system and lower training costs.

The JAD approach also has disadvantages. First, it is very difficult to get all users to the JAD meeting. For example, large organizations may have users literally all over the world; to have all of them attend a JAD meeting would be prohibitively expensive. Second, the JAD approach has all the problems caused by any group process (e.g., one person can dominate the meeting, some participants may be shy and not contribute in a group setting, or some participants may sit back and let others do the work). To alleviate these problems, JAD sessions usually have a facilitator, who is skilled in systems analysis and design as well as in managing group meetings and processes.

**Rapid Application Development (RAD)**

**Rapid application development (RAD)** is a systems development method that can combine JAD, prototyping, and integrated CA SE tools (described below) to rapidly produce a high-quality system. RAD is an iterative approach similar to prototyping, in which requirements, designs, and the system itself are developed with sequential refinements. RAD and prototyping are iterative and emphasize speed of development. However, prototyping typically uses specialized languages, such as fourth-generation languages (4GLs) and screen generators, whereas RAD packages include different tools with similar capabilities. With RAD tools, developers enhance and extend the initial version through multiple iterations until it is suitable for operational use. The tools work together as part of an integrated package. RAD produces functional components of a final system, rather than limited-scale versions.

With RAD, users are intensively involved early in the development process. Initially, JAD sessions are used to collect system requirements. ICASE tools (discussed next) are then used to quickly structure requirements and develop prototypes. As the prototypes are developed and refined, users review them in additional JAD sessions.

Rapid application development methods and tools enable systems developers to build applications faster, such as systems where the user interface is important or systems that involve rewriting legacy applications. Typical RAD packages include:

- **Graphical user development environment:** The ability to create many aspects of an application by drag-and-drop applications.
- **Reusable components:** A library of common, standard “objects” such as buttons and dialog boxes. The developer drags and drops these items into the application.
- **Code generator:** After the developer drags and drops components into the design, the package automatically writes computer programs to implement the reports, input screens, buttons, and dialog boxes.
• **Programming language:** Such as Visual Basic or C++. This package includes an integrated development environment (IDE) for creating, testing, and debugging computer code.

The main advantage of **RAD** is the active involvement of users in the development process. Active user involvement means that the new system has a better chance of meeting user needs, eases implementation of the new system, and can reduce training costs. **RAD** also speeds the development process, reduces development costs, and can create applications that are easier to maintain and modify.

Like other methods, **RAD** does have disadvantages. The method’s accelerated approach to systems analysis may result in systems with limited functionality and flexibility for change. The system, therefore, may not be able to respond to changing business conditions and may have a limited useful life span. **RAD**’s accelerated development process may produce systems that are not of the highest quality. **RAD** packages can result in an endless iterative process. **RAD** packages do provide features that enable developers to document the system, but developers may ignore these features in the accelerated process.

Like many methodologies, **RAD** can be used to develop enterprisewide applications. See the book’s Web site for an example of **RAD** in this role in a large corporation.

**Integrated Computer-Assisted Software Engineering (ICASE) Tools**

**Computer-aided software engineering (CASE)** tools automate many of the tasks in the **SDLC**. The tools used to automate the early stages of the **SDLC** (systems investigation, analysis, and design) are called **upper CASE** tools. The tools used to automate later stages in the **SDLC** (programming, testing, operation and maintenance) are called **lower CASE** tools. **CASE** tools that provide links between upper **CASE** and lower **CASE** tools are called **integrated CASE (ICASE)** tools. (For more on **CASE** tools, see the Web site.)

**CASE** tools provide advantages and disadvantages for systems developers. These tools can produce systems with a longer effective operational life that more closely meet user requirements. **CASE** tools can speed up the development process and result in systems that are more flexible and adaptable to changing business conditions. Finally, systems produced using **CASE** tools typically have excellent documentation.

On the other hand, **CASE** tools can produce initial systems that are more expensive to build and maintain. **CASE** tools do require more extensive and accurate definition of user needs and requirements. Also, **CASE** tools are difficult to customize and may be difficult to use with existing systems.

**Object-Oriented Development**

**Object-oriented development** is based on a fundamentally different view of computer systems than that found in traditional **SDLC** development approaches. Traditional approaches provide specific step-by-step instructions in the form of computer programs, in which programmers must specify every procedural detail. These programs usually result in a system that performs the original task but may not be suited for handling other tasks, even when the other tasks involve the same real-world entities. For example, a billing system will handle billing but probably will not be adaptable to handle mailings for the marketing department or generate leads for the sales force,
even though the billing, marketing, and sales functions all use similar data such as customer names, addresses, and current and past purchases.

An object-oriented (OO) system begins not with the task to be performed, but with the aspects of the real world that must be modeled to perform that task. Therefore, in the example above, if the firm has a good model of its customers and its interactions with them, this model can be used equally well for billings, mailings, and sales leads.

**Advantages of the object-oriented approach.** The OO approach to software development offers many advantages:

1. It reduces the complexity of systems development and leads to systems that are easier and quicker to build and maintain, because each object is relatively small and self-contained.
2. It improves programmers’ productivity and quality. Once an object has been defined, implemented, and tested, it can be reused in other systems.
3. Systems developed with the OO approach are more flexible. These systems can be modified and enhanced easily, by changing some types of objects or by adding new types.
4. The OO approach allows the systems analyst to think at the level of the real-world systems (as users do) and not at the level of the programming language. The basic operations of an enterprise change much more slowly than the information needs of specific groups or individuals. Therefore, software based on generic models (which the OO approach is) will have a longer life span than programs written to solve specific, immediate problems.
5. The OO approach is also ideal for developing Web applications.
6. The OO approach depicts the various elements of an information system in user terms (i.e., business or real-world terms), and therefore, the users have a better understanding of what the new system does and how it meets its objectives.

The OO approach does have disadvantages. OO systems, especially those written in Java, generally run more slowly than those developed in other programming languages. Also, many programmers have little skill and experience with OO languages, necessitating retraining.

**Object-oriented analysis and design (OOA & D).** The development process for an object-oriented system begins with a feasibility study and analysis of the existing system. At this point, systems developers identify the objects in the new system. The object is the fundamental element in OOA & D. It represents a tangible real-world entity, such as a customer, bank account, student, or course. Objects have properties. For example, a customer has an identification number, name, address, account number(s), and so on. Objects also contain the operations that can be performed on their properties. For example, customer object’s operations may include obtain-account-balance, open-account, withdraw-funds, and so on.

Therefore, OOA & D analysts define all the relevant objects needed for the new system, including their properties (called data values) and their operations (called behaviors). The analysts then model how the objects interact to meet the objectives of the new system. In some cases, analysts can reuse existing objects from other applications (or from a library of objects) in the new system, saving time spent coding. In most cases, however, even with object reuse, some coding will be necessary to customize the objects and their interactions for the new system.
The information systems department usually employs the methods discussed in the 
two previous sections. These methods require highly skilled employees and are com- 
plex, resulting in a backlog in application development and a relatively high failure 
rate. Therefore, in-house software development is not always a feasible solution. Or-
organizations consider alternatives to in-house development for various reasons: The IS 
staff may not be large enough, The IS staff may not have necessary skills (e.g., Web 
applications), the IS staff may have too many systems development projects sched-
uled, or the IS staff may exhibit poor performance due to turnover or rapid changes in 
technology.

The four main methods for developing systems outside the information systems 
department are end-user development, external acquisition, use of application service 
providers, and outsourcing.

**End-User Development**

For the first two decades of computer applications in the business world (early 1950s 
to mid-1970s), users worked in a mainframe computing environment and were totally 
dependent on the IS staff. Since that time, an enormous shift has taken place, with 
users now making direct, hands-on use of computers to solve their business problems, 
a phenomenon called **end-user computing**. Users are also performing more and more 
**end-user development** of their own systems.

Today, IS professionals still manage and control the hardware, software, data-
bases, and networks needed to support the modern computing environment. Profes-
sional IS programmers still develop and maintain complex software systems, while 
users will continue to do more ad-hoc programming.

Many factors are driving the trends toward increased end-user computing and 
end-user development—increasingly powerful hardware at declining cost, increasingly 
diverse software capabilities, a demand for the IS projects and accompanying backlog 
in IS departments, and ability of a computer-literate workforce to develop small ap-
lications at little extra cost. Many of these factors represent continuing trends, mean-
ing that end-user computing and end-user development will become even more 
important in the future. For more on end-user computing, see the book’s Web site.

**External Acquisition of Software**

The choice between developing proprietary software in-house and purchasing existing 
software is called the **make-or-buy decision**. Proprietary application software gives 
the organization exactly what it needs and wants, as well as a high level of control in

---

**Before you go on . . .**

1. What are some common disadvantages of the SDLC approach?
2. What alternative analysis and design methodologies does RAD typically 
encompass?
3. How is object-oriented systems development fundamentally different from the 
SDLC approach?
the development process. In addition, the organization has more flexibility in modifying the software during the development process to meet new requirements. On the other hand, proprietary software requires a large amount of resources (time, money, personnel) that the in-house staff may have trouble providing. The large quantity of resources needed for this software increases the risk of making the software in-house.

The initial cost of off-the-shelf software is often lower because the software development firm can spread the cost over a number of customers. There is lower risk that the software will fail to meet the firm's business needs, because the software can be examined prior to purchase. The software should be of high quality, because many customers have used and helped debug it. However, buying off-the-shelf software may mean that an organization has to pay for features and functions that are not needed. The software may lack necessary features, causing the buyer to have to make expensive modifications to customize the package. Finally, the buyer's particular IT infrastructure may differ from what the software was designed for, and require some additional modification to run properly. (Advantages of external acquisition of software are discussed at the Web site.)

Criteria that may be used to select an application package to purchase include:

- Cost and financial terms
- Upgrade policy and cost
- Vendor's reputation and availability for help
- Vendor's past customers (check references)
- Ease of Internet interface
- Availability and quality of documentation
- Necessary hardware and networking resources
- Required training (does vendor provide?)
- Security
- Speed of learning for developers/users
- Graphical presentation
- Data management capabilities

Application Service Providers (ASPs)

Application service providers (ASPs) provide applications to organizations on a subscription basis. The packaged application is not sold or licensed to the organization and does not reside on the user’s system. Instead, it is hosted on the ASP’s data center and is accessed remotely by the customer. Although gaining tremendous popularity, the concept is not new, having been offered by EDS, IBM, and others for over 30 years.

Customers today typically access applications over an Internet-based or carrier-operated virtual private network. The growth in ASP popularity comes from the increased availability and reliability of such networks, the widespread use of Web browsers as a client interface, and the increased difficulty and cost of installing licensed or purchased software on one’s own network. Many ASPs offer basic services consisting of access to standard versions of applications, and also offer more complex services including upgrades and updates, network and server monitoring, and basic help-desk support.

A application service is part of a general trend away from selling software toward providing software services. Such service can range from simple applications all the way through customer relationship planning systems to enterprise resource planning systems. For an example of the how one company uses the services of an ASP to connect with suppliers, see IT's About Business 14.3. Although there are numerous independent ASPs, many famous large firms such as Compaq, Pricewaterhouse-Coopers, Intel, Oracle, SAP, and others also provide application service (often
through collaboration with one or more other firms). Using ASPs (rather than the old model of licensing or purchase) is also part of a shift toward outsourcing, discussed below.

**Outsourcing**

Information technology is now an integral organizational resource. However, IT is not the primary business of many organizations. Core competencies, the areas in which an organization performs best and that represent its competitive advantage, are in other functions, such as manufacturing, logistics, or services. IT is complex, expensive, and constantly changing, making it difficult for organizations to manage. Organizations may not be able to manage their information technology as well as firms that specialize in managing IT. For these organizations, outsourcing IT may be the best strategy.

**Outsourcing**, in its broadest sense, is the purchase of any product or service from another company. For example, all major automobile companies outsource the manufacture of many components, such as sound systems and air conditioners. Also, many firms outsource their cafeterias to food service companies, and many organizations outsource their entire human resources departments to outside firms. In general, companies outsource the products and services they do not want to or are unable to produce themselves.
Information systems departments have outsourced computer hardware, telecommunications services, and systems software (such as operating systems) for some time. These departments also purchase end-user software (e.g., Microsoft Office) because there is no reason to reinvent tools that a software company specializing in these products can provide more cheaply.

Recently, information technology has involved hiring outside organizations to perform functions that in the past have been performed internally by information systems departments. Common areas for outsourcing have included maintaining computer centers and telecommunications networks. Some companies, however, outsource most of the IT functions—including systems and applications development—leaving only a very small internal information systems department. This department develops IS plans and negotiates with the vendors performing the outsourced functions.

Firms that provide outsourcing cite numerous benefits that establish their claim that they can provide IT services at 10 to 40 percent lower cost, with higher quality. They note these reasons:

- **Hardware economies of scale.** With multiple customers, outsourcers can use more cost-efficient larger computers or obtain discounts on volume purchases of hardware. They also can operate their computers with less excess capacity because peak loads from different customers will not all occur at the same time.

- **Staffing economies of scale.** A larger customer base also makes it possible for outsourcers to hire highly skilled, specialized technical personnel whose salaries would be hard to justify in smaller IS groups.

- **Specialization.** Providing computer services is one of the core competencies of the outsourcing firm, rather than an incidental part of its business.

- **Tax benefits.** Organizations can deduct outsourcing fees from current income, in contrast to depreciating computer hardware purchases over three to five years.

However, outsourcing can create problems for companies, which include:

- **Limited economies of scale.** Although outsourcers can negotiate larger discounts on hardware, the advantage may not be significant, especially over a five-year life of a mainframe.

- **Staffing.** Typically, former employees, rather than the highly skilled vendor staff, serve customers. In some cases, the outsourcer shifts the better former employees to other accounts.

- **Lack of business expertise.** In addition to losing former employees to other accounts, the remaining staff members tend to become more technically oriented and have less knowledge of the business issues in the customer’s industry.

- **Contract problems.** Some customers fail to adequately specify service levels in their contract with the outsourcer and so must pay excess fees for services not in the contract, or for volumes greater than the averages written into the contract.

- **Internal cost reduction opportunities.** Organizations can achieve many of the cost savings of outsourcing by improving their own IT management. For example, these firms can achieve economies of scale by consolidating multiple data centers into one location.

Outsourcing can benefit public as well as private organizations. For example government IT organizations are beginning to adopt outsourcing practices. Federal agencies, states, and local municipalities are hiring outside services firms for more than just specific system-integration projects. Some are outsourcing parts of their
daily IT operations such as desktop and LAN management. Others are even more aggressive, turning over major IT functions to outside companies.

**EXAMPLE**

**Brinker looks offshore for Web app solutions** When $3 billion Dallas restaurant owner Brinker International wanted to improve its gift-card program at more than 850 restaurants a few years ago, it looked offshore to Cognizant Technology Solutions to find a service provider that could turn a good idea into a good product under a tight deadline. Recently, Robert Hess, Brinker’s director of software services, said his company has its sights set on developing a new Web-based application that will create a competitive advantage by aggregating data from its restaurants and making that information available via the Web. And it has again tapped Cognizant Technology Solutions to get the project done on time and on budget. In the highly competitive restaurant business, Brinker’s philosophy is that the key to improving each restaurant’s performance lies in understanding point-of-sale and inventory data collected at each location. Hess says the number of developers he needed to complete the project, as well as the urgency of the project, led him to outsource much of the work to Cognizant. About 70 percent of the development work is done at facilities in India, keeping costs down, while the rest is done in conjunction with Brinker personnel, primarily in Dallas.

There are a variety of guidelines to help organizations as they outsource some or all of their IT function. These guidelines include:

- **Short-period contracts.** Outsourcing contracts are often written for five-to ten-year terms. Because IT and the competitive environment change so rapidly, it is probable that some of the terms will not be in the customer’s best interests after five years. If a longer-term contract is used, it needs to include adequate mechanisms for negotiating revisions where necessary.

- **Subcontracting.** Vendors may subcontract some of the service to other vendors. The contract should give the customer some control over the circumstances, including choice of vendors, and any subcontract arrangements.

- **Selective outsourcing.** This strategy is used by many organizations that prefer not to outsource the majority of their IT functions, but rather to outsource only certain areas (such as network management).

For another outsourcing example and other suggestions for successful outsourcing, see the book’s Web site.

As we can see from this section and from Sections 14.2 and 14.3, there are a number of ways to develop information systems. Each of these represents a basic approach, but there can be a number of variations. Given that no two organizations or systems are exactly alike or affected by the same constraints, systems developers face a complex decision when choosing a development methodology. Also, they may switch from one methodology to another for different parts of a large system, depending on a variety of considerations. To better illustrate how such a decision might be made, Manager’s Checklist 14.1 summarizes the advantages and disadvantages of these methodologies.

Even the appropriate systems development methodology cannot ensure success in all cases. Organizational factors such as culture, bureaucracy, and budget can make or break a development project. See the book’s Web site for an example of an organization faced with a variety of challenges to successful systems development.
<table>
<thead>
<tr>
<th>Methodology</th>
<th>Advantages</th>
<th>Disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Traditional Systems Development (SDLC)</strong></td>
<td>• Forces staff to be systematic by going through every step in a structured process.</td>
<td>• May produce excessive documentations</td>
</tr>
<tr>
<td></td>
<td>• Ensures quality by maintaining standards.</td>
<td>• Users are often unwilling or unable to study the specifications they approve.</td>
</tr>
<tr>
<td></td>
<td>• Has lower probability of missing important issues in collecting user requirements.</td>
<td>• Takes too long to go from the original ideas to a working system.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Users have trouble describing requirements for a proposed system.</td>
</tr>
<tr>
<td><strong>Prototyping</strong></td>
<td>• Helps clarify user requirements.</td>
<td>• May encourage inadequate problem analysis.</td>
</tr>
<tr>
<td></td>
<td>• Helps verify the feasibility of the design.</td>
<td>• Not practical with large number of users.</td>
</tr>
<tr>
<td></td>
<td>• Promotes genuine user participation in the development process.</td>
<td>• User may not give up the prototype when the system is completed.</td>
</tr>
<tr>
<td></td>
<td>• Promotes close working relationship between systems developers and users.</td>
<td>• May generate confusion about whether or not the information system is complete and maintainable.</td>
</tr>
<tr>
<td></td>
<td>• Works well for ill-defined problems.</td>
<td>• System may be built quickly, which may result in lower quality.</td>
</tr>
<tr>
<td></td>
<td>• May produce part of the final system.</td>
<td></td>
</tr>
<tr>
<td><strong>Joint Application Development (JAD)</strong></td>
<td>• Easy for senior management to understand.</td>
<td>• System often narrowly focused, which limits future evolution, flexibility, and adaptability to changing business conditions.</td>
</tr>
<tr>
<td></td>
<td>• Provides needed structure to the user requirements collection process.</td>
<td>• System may be built quickly, which may result in lower quality.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Rapid Application Development (RAD)</strong></td>
<td>• Active user involvement in analysis and design stages.</td>
<td>• Very difficult to train analysts and programmers on the OO approach.</td>
</tr>
<tr>
<td></td>
<td>• Easier implementation due to user involvement.</td>
<td>• Limited use of common objects and classes.</td>
</tr>
<tr>
<td><strong>Object-Oriented Development (OO)</strong></td>
<td>• Integration of data and processing during analysis and design should lead to higher-quality systems.</td>
<td>• Creates lower-quality systems because an amateur does the programming.</td>
</tr>
<tr>
<td></td>
<td>• Reuse of common objects and classes makes development and maintenance easier.</td>
<td></td>
</tr>
<tr>
<td><strong>End-User Development</strong></td>
<td>• Bypasses the information systems department and avoids delays</td>
<td></td>
</tr>
</tbody>
</table>
Web browsers and Internet communications use open, nonproprietary standards, making it easy to adapt them to any operating system and to any personal computer hardware. Open standards eliminate most of the incompatibility problems and integration difficulties that have always been problems for systems developers.

The Web browser is a highly intuitive and nearly universal interface that is easy to learn to use. The hyperlinks feature of Web pages represents an extremely powerful capability for organizing information into a usable and accessible form. It requires less skill to develop Web pages in hypertext markup language (HTML), or using Web page development tools, than it takes to write code in programming languages such as C or COBOL. This simplicity means that applications can be developed rapidly. Im-

### Before you go on . . .

1. Why do companies seek alternatives to having their IS department develop systems?
2. What are the advantages and disadvantages to end-user developed systems?
3. What factors are considered in a make-or-buy decision?

### Manager's Checklist 14.1

<table>
<thead>
<tr>
<th>External Acquisition of Application Package or Services</th>
<th>End-User Development</th>
</tr>
</thead>
<tbody>
<tr>
<td>Software exists and can be tried out.</td>
<td>User controls the application and can change it as needed.</td>
</tr>
<tr>
<td>Software has been used for similar problems in other organizations.</td>
<td>Directly meets user requirements.</td>
</tr>
<tr>
<td>Reduces time spent for analysis, design, and programming.</td>
<td>Increased user acceptance of new system.</td>
</tr>
<tr>
<td>Has good documentation that will be maintained.</td>
<td>Frees up IT resources and may reduce application development backlog.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Advantages</th>
<th>Disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td>May eventually require consulting and maintenance assistance from the IT department.</td>
<td>Controlled by another company that has its own priorities and business considerations.</td>
</tr>
<tr>
<td>System may not have adequate documentation.</td>
<td>Package’s limitations may prevent desired business processes.</td>
</tr>
<tr>
<td>Poor quality control.</td>
<td>May be difficult to get needed enhancements if other companies using the package do not need those enhancements.</td>
</tr>
<tr>
<td>System may not have adequate interfaces to existing systems.</td>
<td>Lack of intimate knowledge about how the software works and why it works that way.</td>
</tr>
</tbody>
</table>

### 14.5 BUILDING INTERNET AND INTRANET APPLICATIONS

Web browsers and Internet communications use open, nonproprietary standards, making it easy to adapt them to any operating system and to any personal computer hardware. Open standards eliminate most of the incompatibility problems and integration difficulties that have always been problems for systems developers.

The Web browser is a highly intuitive and nearly universal interface that is easy to learn to use. The hyperlinks feature of Web pages represents an extremely powerful capability for organizing information into a usable and accessible form. It requires less skill to develop Web pages in hypertext markup language (HTML), or using Web page development tools, than it takes to write code in programming languages such as C or COBOL. This simplicity means that applications can be developed rapidly. Im-
Implementations are easy because most users already know how to work with a browser. Although there are some major potential security problems, the simplicity of the browser and the HTML language reduce the risks of failure in development.

These considerations suggest that developing Internet (and intranet) applications should be fast and inexpensive.

**An Internet and Intranet Development Strategy**

Because the Web browser technology is so new yet so simple, most organizations do not use the SDLC approach for Internet/intranet development. Much of the activity has been on a low-budget, experimental basis to gain experience with the technologies. However, most organizations have now gained some experience, so it is appropriate for them to start implementing more formal planning and development policies.

The first planning issue is to identify the objectives for organizational Web sites. These should align with and support organizational strategies. The objectives will vary depending on whether the sites are on: (1) the Internet, which represents the organization to the general public; (2) an extranet, for use with business partners; or (3) an intranet, to serve the needs of organizational employees.

This initial planning also needs to adequately cover infrastructure requirements as well as security and legal issues. Organizational servers and communications links need to have capabilities consistent with the importance of the site to the organization’s activities and business strategies. If the site is critical, it needs to have enough capacity to avoid breakdowns or delays during peak periods. The organization also needs to have specialized personnel (e.g., Webmasters) to operate and maintain Web sites, and it may want to establish a steering committee to develop and monitor compliance with policies regarding organizational sites.

Security provisions need to be appropriate to the intended use of each site, with adequate firewalls (see Chapter 15) to protect data and programs, and with mechanisms to protect the security of customer transactions. The legal department needs to address potential liability issues related to the sites and set policies to prevent improper use of intellectual property.

After the planning issues have been resolved, the organization can identify and prioritize potential projects. Management may choose to fund the more promising applications, leaving the others to be developed later or by end users. The steering committee should monitor end-user projects to make sure that they comply with organizational standards in relation to communications with external customers, and with internal security requirements.

**JAVA—A Promising Tool**

Internet and intranet Web pages are coded primarily in HTML, a simple language that is most useful for displaying static content to viewers. HTML has very limited capabilities for interacting with viewers or for providing information that is continually being updated. It is not suitable for collecting information, such as names and addresses, for providing animation, or for changing information such as stock quotes. To do these types of things it is necessary to add programs written in some form of programming language to the HTML for a Web site.

As discussed in Chapter 4, Java is relatively new, but it has already established itself as the most important programming language for putting extra features into Web pages. Java was specifically designed to work over networks: Java programs can be sent from a Web server over the Internet and then run on the computer that is...
viewing the Web page. It has numerous security features to prevent these downloaded programs from damaging files or creating other problems on the receiving computer.

Java is an object-oriented language, so the concepts of object-oriented development are relevant to its use. However, the Java Web page programs, called applets, need to be relatively small to avoid delays in transmitting them over the Internet. Java programs run more slowly than other languages, such as C, which is another reason to keep them small. Because Java programs tend to be small, it is not necessary that Java developers use the very formal development methodologies appropriate for large system projects. Prototyping is probably the most suitable approach for developing Java applets, because it provides for a high level of interaction between the developers and users in regard to the critical issues of the appearance and ease-of-use of the Web page.

Before you go on . . .

1. Why is creating Internet-based systems often easier than other types of systems?
2. Why does the Java language play so important a role in Internet-based systems?

FOR THE ACCOUNTING MAJOR
A accounting personnel help perform the cost–benefit analyses on proposed projects to assess their economic feasibility. They may also monitor ongoing project costs to keep the budget on track. Both of these business functions are traditionally data intensive. Because of this, their information systems are continually being developed and refined. A accounting personnel undoubtedly will find themselves involved with systems development at various points throughout their careers.

FOR THE FINANCE MAJOR
Finance personnel are frequently involved with the financial issues that accompany any large-scale systems development project. They need to stay abreast of the emerging techniques used to determine project costs and return on investment. Due to the intensity of data and information in their various functions, finance departments themselves are also common recipients of new systems. Finance personnel who are acquainted with systems development methodologies are better equipped to assist in getting the right system developed in the right way.

FOR THE MARKETING MAJOR
Marketing, in most organizations, is becoming data- and information-intensive, so the marketing function is also a hotbed of systems development. Marketing personnel, like their cohorts in other functional areas, will find themselves participating on systems development teams. This will involve development of in-house systems, but it increasingly means aiding in the development of systems—such as Internet and WWW-based systems—that reach out directly from the organization to the customer.
Describe the information systems planning process.

Information systems planning begins with the strategic plan of the organization, which states the firm’s overall mission, the goals that follow from the mission, and the broad steps necessary to reach these goals. The organizational strategic plan and the existing IT architecture provide the inputs in developing the information systems strategic plan. The IS strategic plan is a set of long-range goals that describe the IT architecture and major IS initiatives needed to achieve the goals of the organization. The IS strategic plan states the mission of the IS department, which defines its underlying purpose. The IS strategic plan may require a new IT architecture, or the existing IT architecture may be sufficient. In either case, the IS strategic plan leads to the IS operational plan, which is a clear set of projects that will be executed by the IS/IT department and by functional area managers in support of the IS strategic plan.

Discuss the concept of a systems development life cycle (SDLC).

The systems development life cycle is the traditional systems development method used by most organizations today. The SDLC is a structured framework that consists of distinct sequential processes: systems investigation, systems analysis, systems design, programming, testing, implementation, operation, and maintenance. These processes, in turn, consist of well-defined tasks. Some of these tasks are present in most projects, while others are present in only certain types of projects. That is, smaller development projects may require only a subset of the tasks, and large projects typically require all tasks.

Discuss the advantages and disadvantages of the traditional development, prototyping, rapid application development, object-oriented development, and end-user development life cycles.

Development managers who must develop large applications find it useful to mix and match development methods and tools in order to reduce development time, costs, and complexity. Also, they may switch from one methodology to another for different parts of a large system, depending on a variety of considerations.

FOR THE PRODUCTION/OPERATIONS MANAGEMENT MAJOR

Participation on development teams is also a common role for production/operations people. Manufacturing is becoming increasingly computer controlled and integrated with other allied systems, from design to logistics to inventory control to customer support. Every link in this value chain requires expert input to develop effective systems that can seamlessly integrate with other parts of the larger, enterprisewide system.

FOR THE HUMAN RESOURCES MAJOR

The human resources department is closely involved with several aspects of the systems development project. New systems may require terminating employees, hiring new employees, or changing job descriptions, tasks that are handled by the human resources department. The organization may hire consultants for the development project, and the human resources department handles contracts with these consultants. If the organization outsources the entire development project, the human resources department may still handle some of the HR-related contractual issues with the other company.
Manager’s Checklist 14.2 summarizes the specific advantages and disadvantages of the various alternative methods of systems development.

4 Identify the advantages and disadvantages of CASE tools.

The advantages of CASE tools are that they can produce systems with a longer effective operational life that more closely meet user requirements, can speed up the development process and result in systems that are more flexible and adaptable to changing business conditions, and can produce systems with excellent documentation. The disadvantages are that CASE tools can produce initial systems that are more expensive to build and maintain, require more extensive and accurate definition of user needs and requirements, are difficult to customize, and may be difficult to use with existing systems.

5 Evaluate the alternatives to in-house systems development.

In-house systems development requires highly skilled employees to undertake a complex process, which results in a backlog in application development and a relatively high failure rate. Organizations may sometimes find it preferable to purchase already-existing applications packages than to develop them. The three main methods for developing systems outside the information systems department are end-user development, external acquisition, and outsourcing. In addition, application service providers (ASPs) are becoming increasingly popular. The advantages and disadvantages of these three alternatives are summarized in Manager’s Checklist 14.2.

6 Discuss the key features of Internet and intranet development.

Internet and intranet development uses Web browsers with open, nonproprietary standards, making it easy to adapt to any operating system and to any personal computer hardware. Web browsers are nearly universal and the interface is easy to learn to use, so applications can be developed rapidly. The simplicity of the browser and the HTML language reduce the risks of failure in development, so most organizations do not use the SDLC approach for Internet/intranet development. But they should have specialized Webmasters to operate and maintain Web sites, and these sites must have adequate security. Internet and intranet development often uses the Java programming language and applets.

INTERACTIVE LEARNING SESSION

Go to the Web site (or CD) and access Chapter 14: Information Systems Development and read the case presented. You will be presented with a business problem which will require you to query a database for certain information. You will be able to construct SQL statements to obtain the needed information. As you construct your SQL statements, you will see what information results and decide if it meets your requirements. You may then change your SQL statements as necessary to obtain further (or different) information.

DISCUSSION QUESTIONS

1. Why is it important for everyone to have a basic understanding of the systems development process?
2. Should prototyping be used on every systems development project? Why or why not?
3. What can be done to prevent a runaway systems development project?
4. What are the characteristics of structured programming? Why is structured programming so important?
5. How can an organization control end-user development? Should an organization be strict in this control or loose? Support your answer.
The Business Problem

Lehman Brothers, founded in 1850, provides a wide menu of research, distribution, trading, and financing services to businesses, institutions, governments, and high-net-worth individual investors. For several years, the firm has been building client/server applications that use a Visual Basic client to access databases and a custom-developed UNIX analytics engine through Common Object Request Broker Architecture (CORBA). This aging architecture provided a lot of functionality to Lehman’s traders and operations, but as market volatility increased, the firm needed more current market intelligence. The fat-client application didn’t scale well and required too much support.

Lehman Brothers Inc. set itself this goal: Make it possible for financial traders to provide clients with an immediate sense of how their portfolios are doing, as well as the tools to better advise clients on various investment scenarios based on projected market changes. But Robert Okin, the Lehman executive heading the project, didn’t want to sacrifice the current application’s strengths. “We were asked to extend the reach of our app, yet maintain the rich client interface,” says Okin, Lehman’s VP of structured credit trading technology.

The IT Solution

“Embrace and extend” is a marketing phrase that characterizes what the New York global investment bank is accomplishing, using commercial software components from Infragistics Inc. Graphical user interface components are being used to retrofit an aging client/server application for doing portfolio management and scenario analysis. With the Infragistics toolkit, developers can deliver the look and feel of Microsoft Office, Outlook, or Windows 2000 (Windows XP is in the offing). The presentation components include grids, charts, Outlook bars, calendaring components, and data explorers, which deliver the familiar look and feel of Microsoft applications. Using these off-the-shelf components, developers can more quickly deliver a professional-looking application with maximum code reuse.

PROBLEM-SOLVING ACTIVITIES

1. Develop a hybrid systems development methodology, using the SDLC as the framework, and adding prototyping, joint application design, rapid application development, and CASE tools where appropriate. Point out the advantages of your new methodology.

2. Develop guidelines for deciding when a system should be scrapped and a new system developed. Include cost, maintenance, and systems effectiveness in your answer.

3. Research the viability of using the Java language as a basis for an enterprisewide IS.

INTERNET ACTIVITIES

1. Use an Internet search engine to obtain information on CASE and ICASE tools. Select several vendors and compare and contrast their offerings.

2. Use the World Wide Web to find consulting firms that specialize in systems analysis and design. Compare the size, reputation/history, and degree of specialization.

3. Use the Web to learn about analysis and design of intranets. What sort of sites have the most (and the most useful) information?

TEAM ACTIVITIES AND ROLE PLAYING

1. Divide into groups, with each group visiting a local company (include your university). At each firm, study the systems development process. Find out the methodology or methodologies used by each organization and the type of application each methodology applies. Prepare a report and present it to the class.

2. As a group, design an information system for a startup business of your choice. Describe your chosen systems development methodologies, and justify your choices of hardware, software, telecommunications support, and other aspects of the proposed system.

REAL-WORLD CASE

Lehman Brothers Traders Are Instantly Informed Thanks to Java-based Applications
The Results  Lehman had been depending on analytics engines that ran at the end of the day to value a trader’s portfolio. It was only on the following morning that traders could see where they stood. The Infragistics components let Lehman create a front-end trading application that hooks into its existing middleware. This lets traders access the most current company and market data. “The Java-based version we’re working on now is going to allow traders to bring up their portfolios in real time as they enter trades,” Okin says. “They can also play around with the various [market dynamics] that affect the valuation of those trades.” When traders click on the page, Sun Microsystems’ Java WebStart application-management tool in the background brings up the application on the user’s desktop. Java WebStart lets users launch applications simply by clicking on a Web-page link.

Lehman was able to bring the application from planning to working code within three months with four developers—a month of specification and design and two months of coding put them close to beta testing. Certainly, the fact that they already had a client/server application in place that delivered significant functionality, as well as the server-based middleware interfaced with their back-end systems and real-time data, gave the Lehman developers a solid platform upon which to build. Okin credits the use of commercial components for letting Lehman deliver the application sooner than if it had needed to build a JavaServer Page screen that could handle more than 50 deals, and provide the rich client server capabilities that one expects from a desktop application. A future goal is to support mobility among traders so they can access the application while traveling.

Questions
1. What is the significance of the type of legacy system in place at Lehman Brothers?
2. Why would the company prefer to go with commercially available components instead of developing its own in-house?
3. Why would Java be important for a system that supports remote access? (Hint: The Web.)