one

The Systems Development Environment

Javier Larrea/AGE Fotostock

Chapter Objectives

After studying this chapter, you should be able to:

- Define information systems analysis and design.
- Discuss the modern approach to systems analysis and design that combines both process and data views of systems.
- Describe the role of the systems analyst in information systems development.
- Describe the information systems development life cycle (SDLC).
- List alternatives to the systems development life cycle, including a description of the role of computer-aided software engineering (CASE) tools in systems development.

Chapter Preview . . .

The key to success in business is the ability to gather, organize, and interpret information. Systems analysis and design is a proven methodology that helps both large and small businesses reap the rewards of utilizing information to its full capacity. As a systems analyst, the person in the organization most involved with systems analysis and design, you will enjoy a rich career path that will enhance both your computer and interpersonal skills.

The systems development life cycle (SDLC) is central to the development of an efficient information system. We will highlight four key SDLC steps: (1) planning and selection, (2) analysis, (3) design, and (4) implementation and operation. Be aware that these steps may vary in each organization, depending on its goals. The SDLC is illustrated in Figure 1-1. Each chapter of this book includes an updated version of the SDLC, highlighting which steps have been covered and which steps remain.

This text requires that you have a general understanding of computer-based information systems as provided in an introductory information systems course. This chapter previews systems analysis and lays the groundwork for the rest of the book.

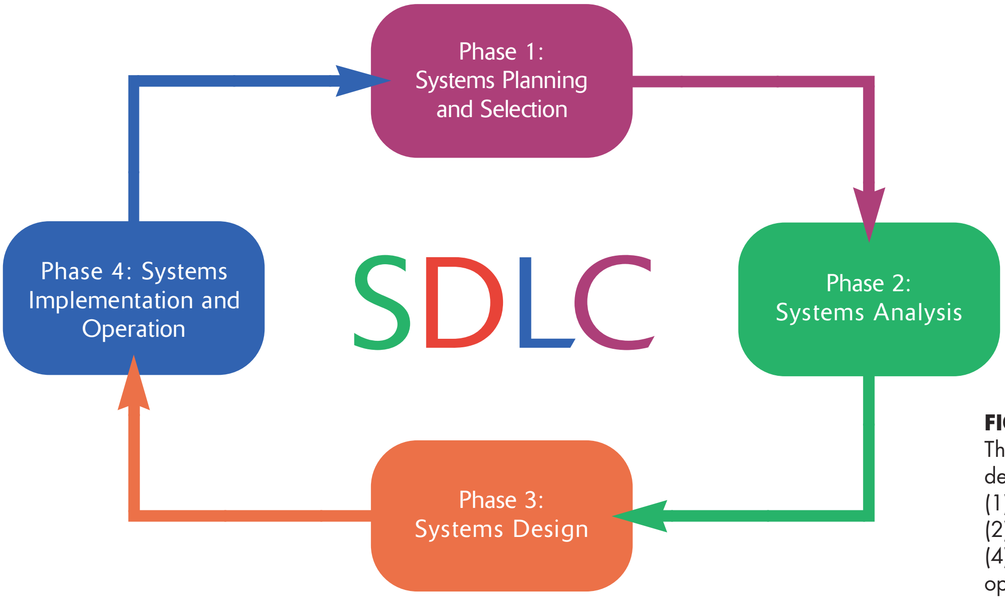


FIGURE 1-1
The four steps of the systems development life cycle (SDLC): (1) planning and selection, (2) analysis, (3) design, and (4) implementation and operation.

What Is Information Systems Analysis and Design?

Information systems analysis and design
The process of developing and maintaining an information system.

Information systems analysis and design is a method used by companies ranging from IBM to PepsiCo to Sony to create and maintain information systems that perform basic business functions such as keeping track of customer names and addresses, processing orders, and paying employees. The main goal of systems analysis and design is to improve organizational systems, typically through applying software that can help employees accomplish key business tasks more easily and efficiently. As a systems analyst, you will be at the center of developing this software. The analysis and design of information systems are based on:

- Your understanding of the organization's objectives, structure, and processes
- Your knowledge of how to exploit information technology for advantage

To be successful in this endeavor, you should follow a structured approach. The SDLC, shown in Figure 1-1, is a four-phased approach to identifying, analyzing, designing, and implementing an information system. Throughout this book, we use the SDLC to organize our discussion of the systems development process. Before we talk about the SDLC, we first describe what is meant by systems analysis and design.

Systems Analysis and Design: Core Concepts

Application software
Software designed to process data and support users in an organization. Examples include spreadsheets, word processors, and database management systems.

The major goal of systems analysis and design is to improve organizational systems. Often this process involves developing or acquiring **application software** and training employees to use it. Application software, also called a *system*, is designed to support a specific organizational function or process, such as inventory management, payroll, or market analysis. The goal of application software is to turn data into information. For example, software developed for the inventory department at a bookstore may keep track of the number of books in stock of the latest best seller. Software for the payroll department may keep track of the changing pay rates of employees. A variety of off-the-shelf application software can be purchased, including WordPerfect, Excel, and PowerPoint. However, off-the-shelf software may not fit the needs of a particular organization, and so the organization must develop its own product.

In addition to application software, the information system includes:

- The hardware and systems software on which the application software runs. Note that the systems software helps the computer function, whereas the application software helps the user perform tasks such as writing a paper, preparing a spreadsheet, and linking to the Internet.
- Documentation and training materials, which are materials created by the systems analyst to help employees use the software they've helped create.
- The specific job roles associated with the overall system, such as the people who run the computers and keep the software operating.
- Controls, which are parts of the software written to help prevent fraud and theft.
- The people who use the software in order to do their jobs.

The components of a computer-based information system application are summarized in Figure 1-2. We address all the dimensions of the overall system,

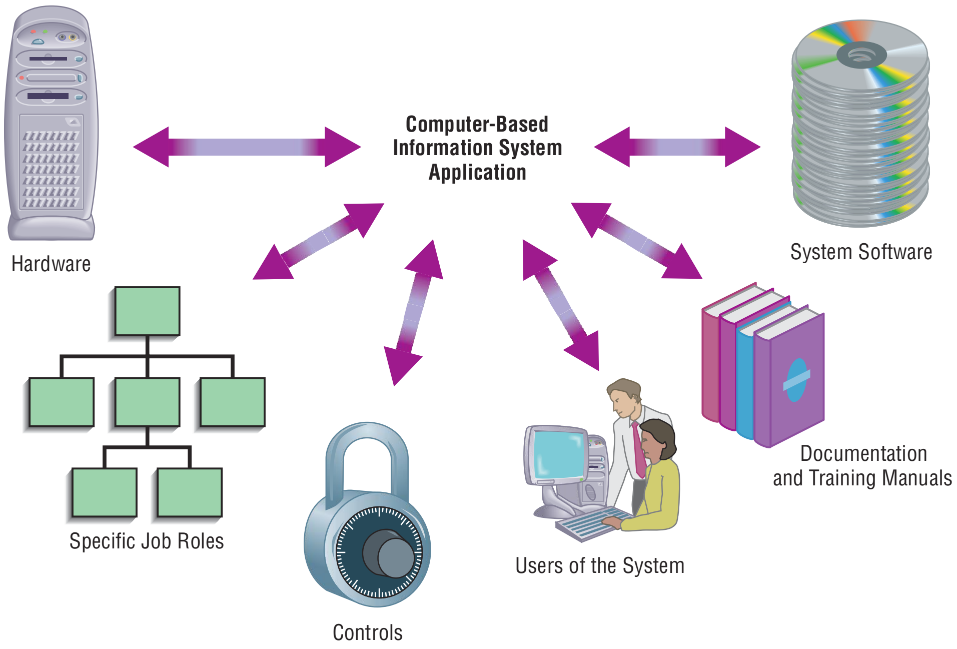


FIGURE 1-2
Components of a computer-based information system application.

with particular emphasis on application software development—your primary responsibility as a systems analyst.

Our goal is to help you understand and follow the software engineering process that leads to the creation of information systems. As shown in Figure 1-3, proven methodologies, techniques, and tools are central to software engineering processes (and to this book).

Methodologies are a sequence of step-by-step approaches that help develop your final product: the information system. Most methodologies incorporate several development techniques, such as direct observations and interviews with users of the current system.

Techniques are processes that you, as an analyst, will follow to help ensure that your work is well thought-out, complete, and comprehensible to others on your project team. Techniques provide support for a wide range of tasks, including conducting thorough interviews with current and future users of the information system to determine what your system should do, planning and managing the activities in a systems development project, diagramming how the system will function, and designing the reports, such as invoices, your system will generate for its users to perform their jobs.

Tools are computer programs, such as computer-aided software engineering (CASE) tools, that make it easy to use specific techniques. These three elements—methodologies, techniques, and tools—work together to form an organizational approach to systems analysis and design.

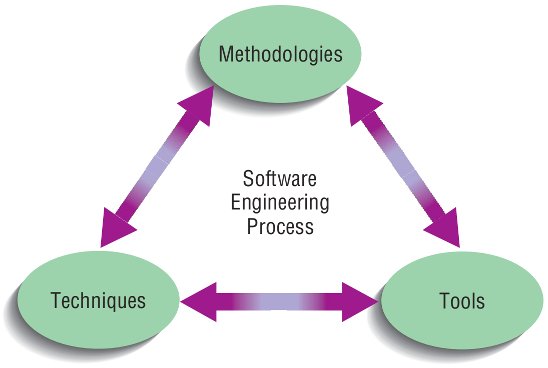


FIGURE 1-3
The software engineering process uses proven methodologies, techniques, and tools.

In the rest of this chapter, you will learn about approaches to systems development—the data- and process-oriented approaches. You will also identify the various people who develop systems and the different types of systems they develop. The chapter ends with a discussion of some of the methodologies, techniques, and tools created to support the systems development process. Before we talk more about computer-based information systems, let's briefly discuss what we mean by the word *system*.

Systems

The key term used most frequently in this book is *system*. Understanding systems and how they work is critical to understanding systems analysis and design.

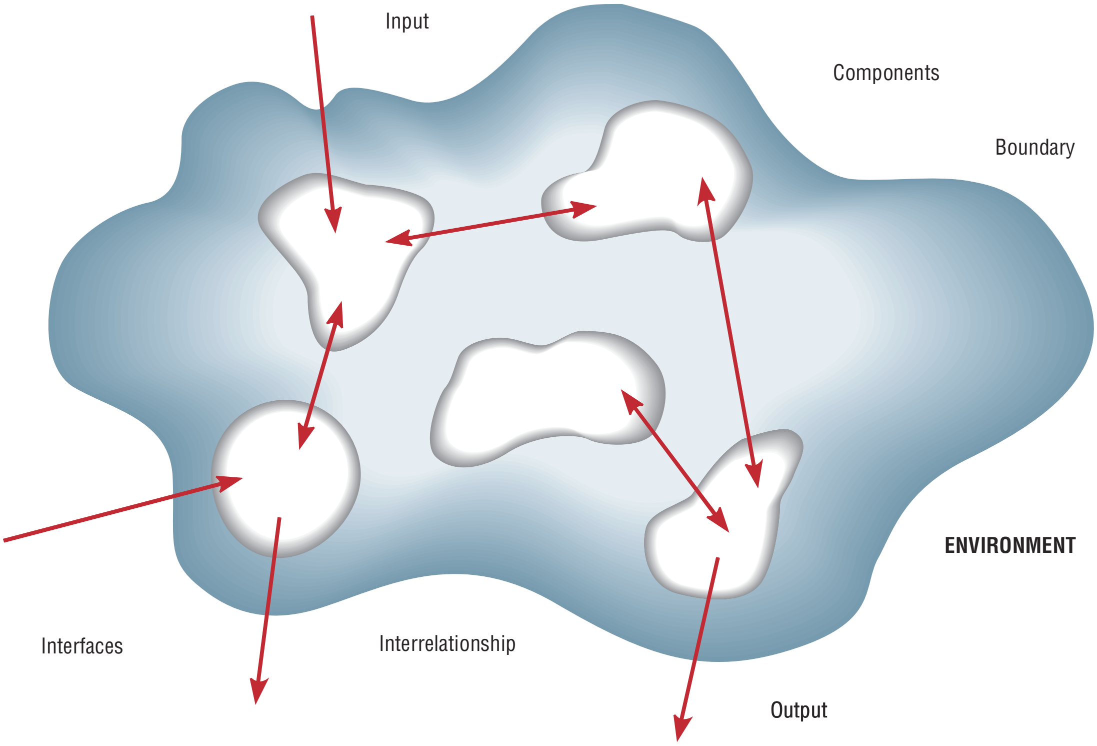
Definition of a System and Its Parts

System
A group of interrelated procedures used for a business function, with an identifiable boundary, working together for some purpose.

A **system** is an interrelated set of business procedures (or components) used within one business unit, working together for some purpose. For example, a system in the payroll department keeps track of checks, whereas an inventory system keeps track of supplies. The two systems are separate. A system has nine characteristics, seven of which are shown in Figure 1-4. A detailed explanation of each characteristic follows, but from the figure you can see that a system exists within a larger world, an environment. A boundary separates the system from its environment. The system takes input from outside, processes it, and sends the resulting output back to its environment. The arrows in the figure show this interaction between the system and the world outside of it.

1. Components
2. Interrelated components

FIGURE 1-4
Seven characteristics of a system.



3. Boundary
4. Purpose
5. Environment
6. Interfaces
7. Constraints
8. Input
9. Output

A system is made up of components. A **component** is either an irreducible part or an aggregate of parts, also called a *subsystem*. The simple concept of a component is very powerful. For example, just as with an automobile or a stereo system, with proper design, we can repair or upgrade the system by changing individual components without having to make changes throughout the entire system. The components are **interrelated;** that is, the function of one is somehow tied to the functions of the others. For example, the work of one component, such as producing a daily report of customer orders received, may not progress successfully until the work of another component is finished, such as sorting customer orders by date of receipt. A system has a **boundary,** within which all of its components are contained and which establishes the limits of a system, separating it from other systems. Components within the boundary can be changed, whereas systems outside the boundary cannot be changed. All of the components work together to achieve some overall **purpose** for the larger system: the system's reason for existing.

A system exists within an **environment**—everything outside the system's boundary that influences the system. For example, the environment of a state university includes prospective students, foundations and funding agencies, and the news media. Usually the system interacts with its environment. A university interacts with prospective students by having open houses and recruiting from local high schools. An information system interacts with its environment by receiving data (raw facts) and information (data processed in a useful format). Figure 1-5 shows how a university can be seen as a system. The points at which the system meets its environment are called **interfaces;** an interface also occurs between subsystems.

In its functioning, a system must face **constraints**—the limits (in terms of capacity, speed, or capabilities) to what it can do and how it can achieve its purpose within its environment. Some of these constraints are imposed inside the system (e.g., a limited number of staff available), and others are imposed by the environment (e.g., due dates or regulations). A system takes input from its environment in order to function. People, for example, take in food, oxygen, and water from the environment as input. You are constrained from breathing fresh air if you're in an elevator with someone who is smoking. Finally, a system returns output to its environment as a result of its functioning and thus achieves its purpose. The system is constrained if electrical power is cut.

Component
An irreducible part or aggregation of parts that makes up a system; also called a *subsystem*.

Interrelated
Dependence of one part of the system on one or more other system parts.

Boundary
The line that marks the inside and outside of a system and that sets off the system from its environment.

Purpose
The overall goal or function of a system.

Environment
Everything external to a system that interacts with the system.

Interface
Point of contact where a system meets its environment or where subsystems meet each other.

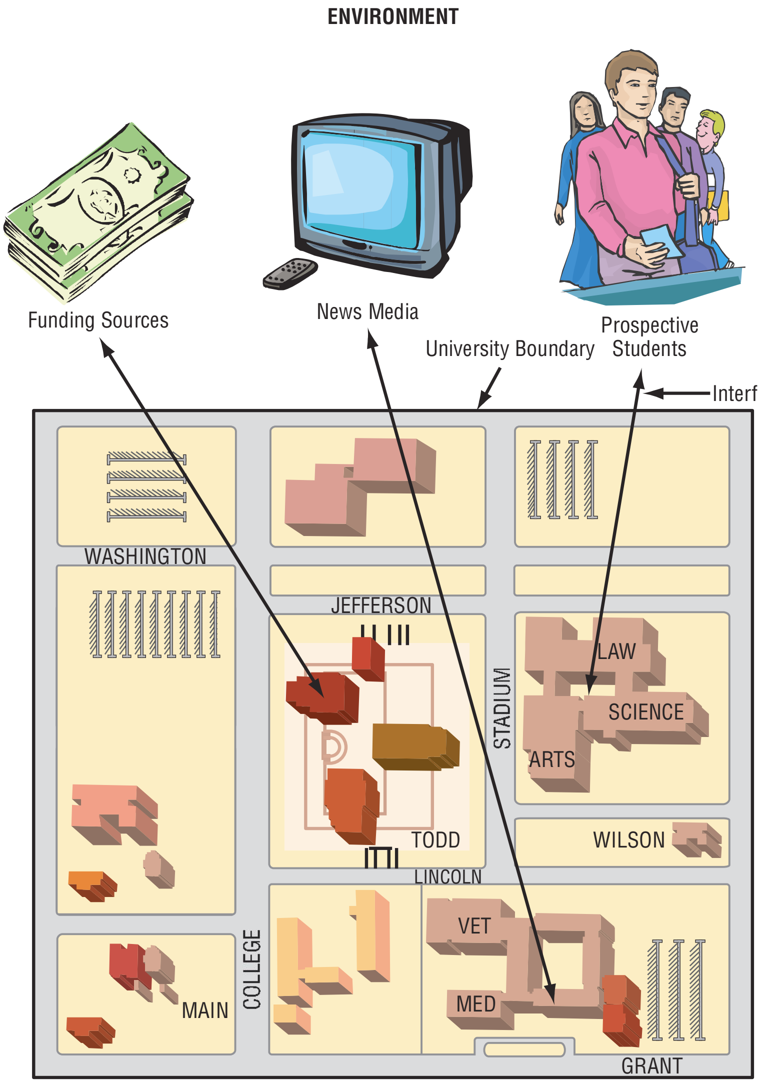
Constraint
A limit to what a system can accomplish.

Important System Concepts

Systems analysts need to know several other important systems concepts:

- Decomposition
- Modularity
- Coupling
- Cohesion

FIGURE 1-5
A university as a system.



Decomposition
The process of breaking the description of a system down into small components; also known as *functional decomposition*.

Decomposition is the process of breaking down a system into its smaller components. These components may themselves be systems (subsystems) and can be broken down into their components as well. How does decomposition aid understanding of a system? It results in smaller and less complex pieces that are easier to understand than larger, complicated pieces. Decomposing a system also allows us to focus on one particular part of a system, making it easier to think of how to modify that one part independently of the entire system. Decomposition is a technique that allows the systems analyst to:

- Break a system into small, manageable, and understandable subsystems
- Focus attention on one area (subsystem) at a time, without interference from other areas
- Concentrate on the part of the system pertinent to a particular group of users, without confusing users with unnecessary details
- Build different parts of the system at independent times and have the help of different analysts

Figure 1-6 shows the decomposition of a portable MP3 player. Decomposing the system into subsystems reveals the system's inner workings. You can decompose an MP3 player into at least three separate physical subsystems. (Note that decomposing the same MP3 player into *logical* subsystems would result in a different set of subsystems.) One subsystem, the battery, supplies the power for the entire system to operate. A second physical subsystem, the storage system, is made up of a hard drive that stores thousands of MP3 recordings. The third subsystem, the control subsystem, consists of a printed circuit board (PCB), with various chips attached, that controls all of the recording, playback, and access functions. Breaking the subsystems down into their components reveals even more about the inner workings of the system and greatly enhances our understanding of how the overall system works.

Modularity is a direct result of decomposition. It refers to dividing a system into chunks or modules of a relatively uniform size. Modules can represent a system simply, making it easier to understand and easier to redesign and rebuild. For example, each of the separate subsystem modules for the MP3 player in Figure 1-6 shows how decomposition makes it easier to understand the overall system.

Modularity
Dividing a system up into chunks or modules of a relatively uniform size.

Coupling means that subsystems are dependent on each other. Subsystems should be as independent as possible. If one subsystem fails and other subsystems are highly dependent on it, the others will either fail themselves or have problems functioning. Looking at Figure 1-6, we would say the components of a portable MP3 player are tightly coupled. The best example is the control system, made up of the printed circuit board and its chips. Every function the MP3 player can perform is enabled by the board and the chips. A failure in one part of the circuit board would typically lead to replacing the entire board rather than attempting to isolate the problem on the board and fix it. Even though repairing a circuit board in an MP3 player is certainly possible, it is typically not cost-effective; the cost of the labor expended to diagnose and fix the problem may be worth more than the value of the circuit board itself. In a home stereo system, the components are loosely coupled because the subsystems, such as the speakers, the amplifier, the receiver, and the CD player, are all physically separate and function independently. If the amplifier in a home stereo system fails, only the amplifier needs to be repaired.

Coupling
The extent to which subsystems depend on each other.

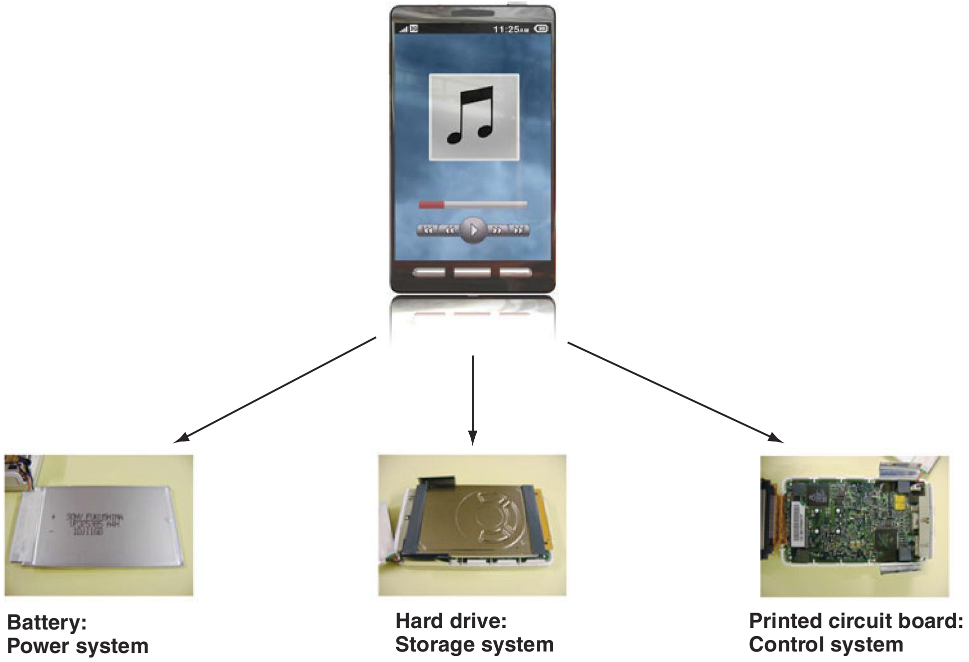


FIGURE 1-6
An MP3 player is a system with power supply, storage and control subsystems.

Sources: Shutterstock; ©Harald van Arkel/Chipmunk International.

Cohesion

The extent to which a system or subsystem performs a single function.

Cohesion is the extent to which a subsystem performs a single function. In the MP3 player example, supplying power is a single function.

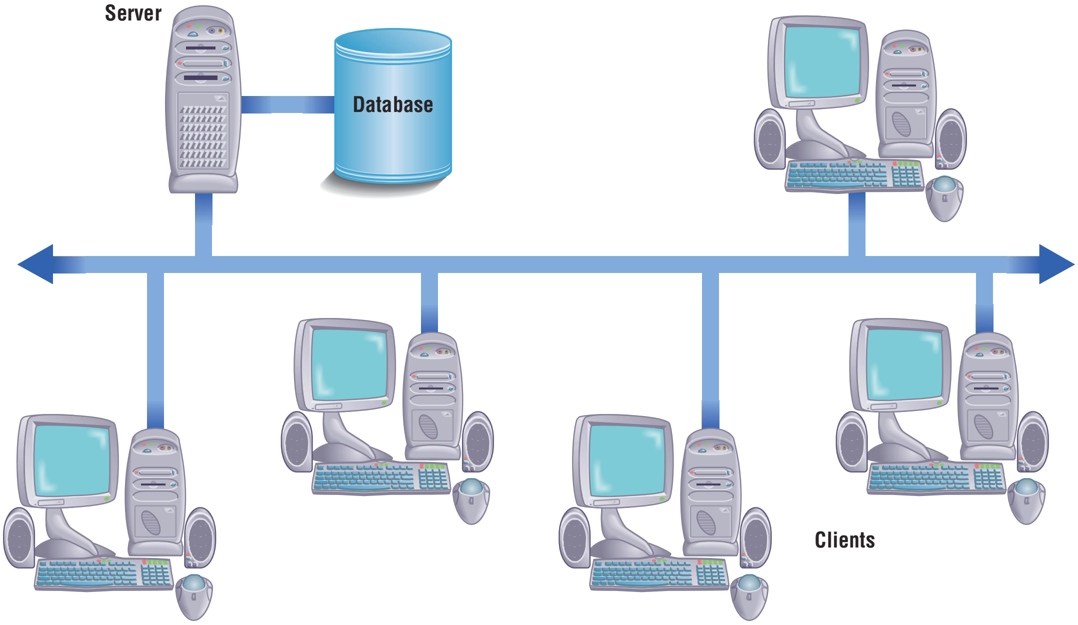
This brief discussion of systems should better prepare you to think about computer-based information systems and how they are built. Many of the same principles that apply to systems in general apply to information systems as well. In the next section, we review how the information systems development process and the tools that have supported it have changed over the decades.

A Modern Approach to Systems Analysis and Design

Today, systems development focuses on systems integration. Systems integration allows hardware and software from different vendors to work together in an application. It also enables existing systems developed in procedural languages to work with new systems built with visual programming environments. Developers use visual programming environments, such as Visual Basic, to design the user interfaces for systems that run on client/server platforms. In a client/server environment, some of the software runs on the server, a powerful computer designed to allow many people access to software and data stored on it, and some of the software runs on client machines. Client machines are the PCs you use at your desk at work. The database usually resides on the server. These relationships are shown in Figure 1-7. The Internet is also organized in a client/server format. With the browser software on your home PC, you can get files and applications from many different computers throughout the world. Your home PC is the client, and all of the Internet computers are servers.

Alternatively, organizations may purchase an enterprise-wide system from companies such as SAP (Systems, Applications, and Products in Data Processing) or Oracle. Enterprise-wide systems are large, complex systems that consist of a series of independent system modules. Developers assemble systems by choosing and implementing specific modules. Enterprise-wide systems usually contain software to support many different tasks in an organization rather than only one or two functions. For example, an enterprise-wide system may handle all human resources management, payroll, benefits, and retirement functions within a single, integrated system. It is, in fact, increasingly rare for organizations to develop systems in-house anymore. Chapter 2 will introduce you to the

FIGURE 1-7
The client/server model.



various sources of information systems technology. First, however, you must gain some insight into what your role will be in the systems development process.

Your Role in Systems Development

Although many people in organizations are involved in systems analysis and design, the **systems analyst** has the primary responsibility. A career as a systems analyst will allow you to have a significant impact on how your organization operates. This fast-growing and rewarding position is found in both large and small companies. IDC, a leading consulting group, predicts that growth in information technology (IT) employment will exceed 3 percent per year through at least 2013. The Bureau of Labor Statistics predicts additional increases in the numbers of IT jobs from 2004 to 2014. During this period, the professional IT workforce is projected to add more than 1 million new jobs in the United States. Information technology workers remain in demand.

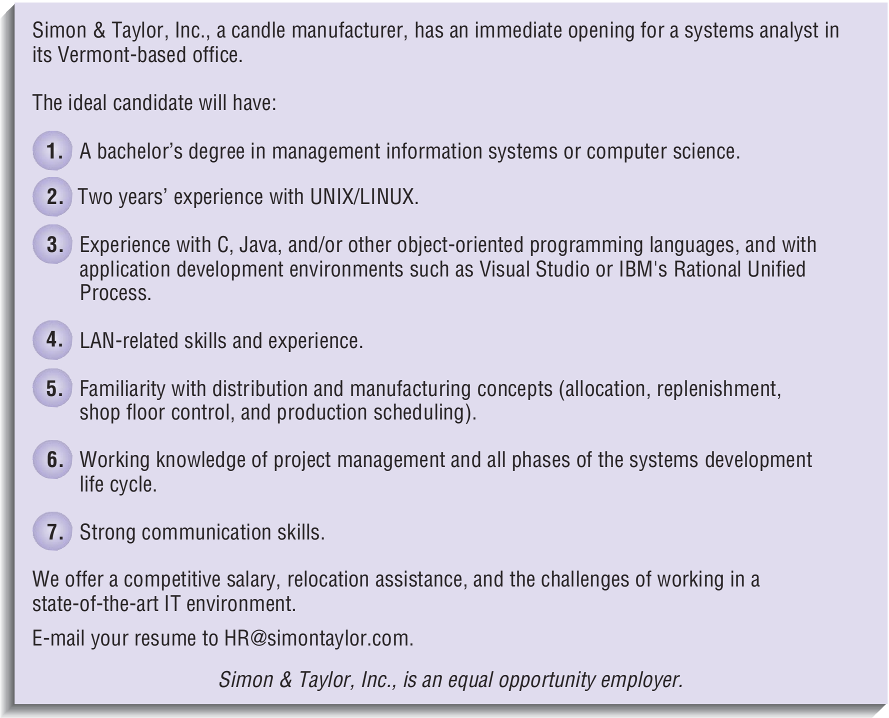
Systems analyst
The organizational role most responsible for the analysis and design of information systems.

The primary role of a systems analyst is to study the problems and needs of an organization in order to determine how people, methods, and information technology can best be combined to bring about improvements in the organization. A systems analyst helps system users and other business managers define their requirements for new or enhanced information services.

Systems analysts are key to the systems development process. To succeed as a systems analyst, you will need to develop four types of skills: analytical, technical, managerial, and interpersonal. Analytical skills enable you to understand the organization and its functions, to identify opportunities and problems, and to analyze and solve problems. One of the most important analytical skills you can develop is systems thinking, or the ability to see organizations and information systems as systems. Systems thinking provides a framework from which to see the important relationships among information systems, the organizations they exist in, and the environment in which the organizations themselves exist. Technical skills help you understand the potential and the limitations of information technology. As an analyst, you must be able to envision an information system that will help users solve problems and that will guide the system's design and development. You must also be able to work with programming languages such as C++ and Java, various operating systems such as Windows and Linux, and computer hardware platforms such as IBM and Mac. Management skills help you manage projects, resources, risk, and change. Interpersonal skills help you work with end users as well as with other analysts and programmers. As a systems analyst, you will play a major role as a liaison among users, programmers, and other systems professionals. Effective written and oral communication, including competence in leading meetings, interviewing end users, and listening, are key skills that analysts must master. Effective analysts successfully combine these four types of skills, as Figure 1-8 (a typical advertisement for a systems analyst position) illustrates.

Let's consider two examples of the types of organizational problems you could face as a systems analyst. First, you work in the information systems department of a major magazine company. The company is having problems keeping an updated and accurate list of subscribers, and some customers are getting two magazines instead of one. The company will lose money and subscribers if these problems continue. To create a more efficient tracking system, the users of the current computer system as well as financial managers submit their problem to you and your colleagues in the information systems department. Second, you work in the information systems department at a university, where you are called upon to address an organizational problem such as the mailing of student grades to the wrong addresses.

FIGURE 1-8
A job advertisement for a systems analyst.



Simon & Taylor, Inc., a candle manufacturer, has an immediate opening for a systems analyst in its Vermont-based office.

The ideal candidate will have:

1. A bachelor's degree in management information systems or computer science.
2. Two years' experience with UNIX/LINUX.
3. Experience with C, Java, and/or other object-oriented programming languages, and with application development environments such as Visual Studio or IBM's Rational Unified Process.
4. LAN-related skills and experience.
5. Familiarity with distribution and manufacturing concepts (allocation, replenishment, shop floor control, and production scheduling).
6. Working knowledge of project management and all phases of the systems development life cycle.
7. Strong communication skills.

We offer a competitive salary, relocation assistance, and the challenges of working in a state-of-the-art IT environment.

E-mail your resume to HR@simontaylor.com.

Simon & Taylor, Inc., is an equal opportunity employer.

When developing information systems to deal with problems such as these, an organization and its systems analysts have several options: They can go to an information technology services firm, such as Accenture or EDS, an HP Company, to have the system developed for them; they can buy the system off the shelf; they can implement an enterprise-wide system from a company such as SAP; they can obtain open-source software; or they can use in-house staff to develop the system. Alternatively, the organization can decide to outsource system development and operation. All of these options are discussed in detail in Chapter 2.

Developing Information Systems and the Systems Development Life Cycle

Organizations use a standard set of steps, called a **systems development methodology,** to develop and support their information systems. Like many processes, the development of information systems often follows a life cycle. For example, a commercial product, such as a Nike sneaker or a Honda car, follows a life cycle: It is created, tested, and introduced to the market. Its sales increase, peak, and decline. Finally, the product is removed from the market and is replaced by something else. The **systems development life cycle (SDLC)** is a common methodology for systems development in many organizations. It marks the phases or steps of information systems development: Someone has an idea for an information system and what it should do. The organization that will use the system decides to devote the necessary resources to acquiring it. A careful study is done of how the organization currently handles the work the system will support. Professionals develop a strategy for designing the new system, which is then either built or purchased. Once complete, the system is installed in the organization, and after proper training, the users begin to incorporate the new system into their daily work. Every organization uses a slightly different life-cycle model to model these steps, with anywhere from three to almost twenty identifiable phases. In this book, we highlight four SDLC

Systems development methodology
A standard process followed in an organization to conduct all the steps necessary to analyze, design, implement, and maintain information systems.

Systems development life cycle (SDLC)
The series of steps used to mark the phases of development for an information system.

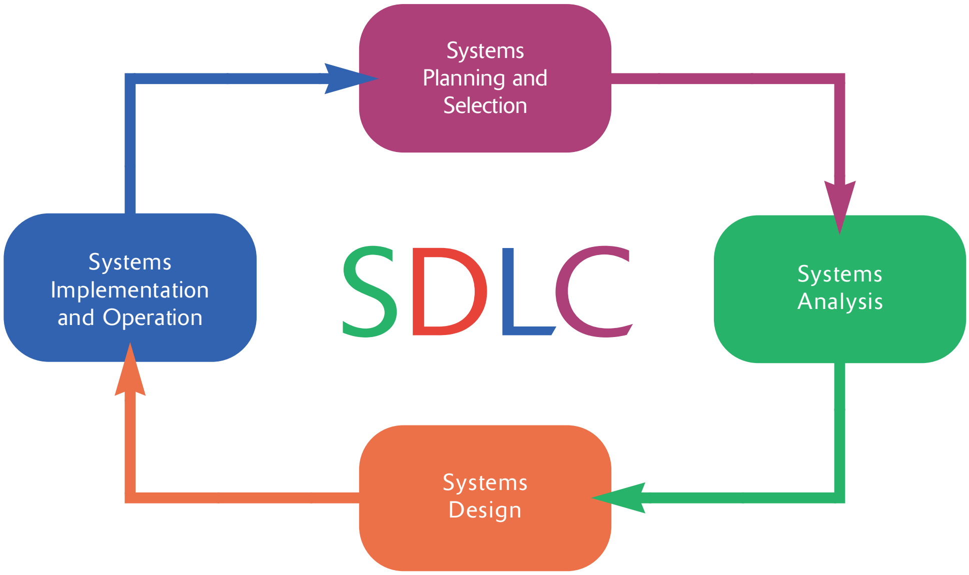

Systems Planning and Selection
Systems Analysis
Systems Design
Systems Implementation and Operation
SDLC

FIGURE 1-9
The systems development life cycle (SDLC).

steps: (1) planning and selection, (2) analysis, (3) design, and (4) implementation and operation (see Figure 1-9).

Although any life cycle appears at first glance to be a sequentially ordered set of phases, it actually is not. The specific steps and their sequence are meant to be adapted as required for a project. For example, in any given SDLC phase, the project can return to an earlier phase, if necessary. Similarly, if a commercial product does not perform well just after its introduction, it may be temporarily removed from the market and improved before being reintroduced. In the systems development life cycle, it is also possible to complete some activities in one phase in parallel with some activities of another phase. Sometimes the life cycle is iterative; that is, phases are repeated as required until an acceptable system is found. Some systems analysts consider the life cycle to be a spiral, in which we constantly cycle through the phases at different levels of detail, as illustrated in Figure 1-10. The circular nature of the life-cycle diagram in Figure 1-10 illustrates how the end of the useful life of one system

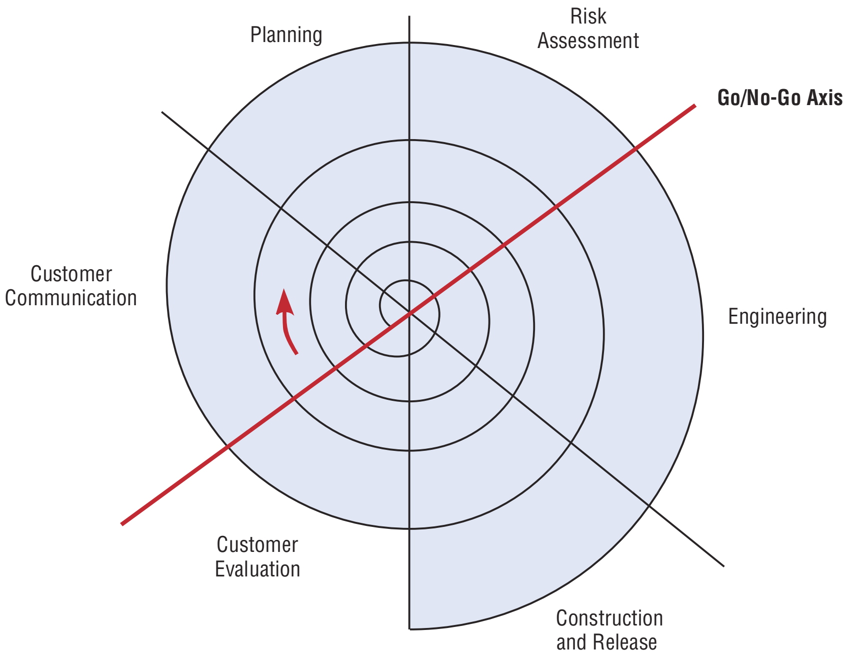


FIGURE 1-10
Evolutionary model SDLC.

leads to the beginning of another project that will replace the existing system altogether. However conceived, the systems development life cycle used in an organization is an orderly set of activities conducted and planned for each development project. The skills required of a systems analyst apply to all life-cycle models.

Every medium-to-large corporation, such as Wal-Mart, and every custom software producer, such as SAP, will have its own specific, detailed life cycle or systems development methodology in place. Even if a particular methodology does not look like a cycle, many of the SDLC steps are performed, and SDLC techniques and tools are used. This book follows a generic SDLC model, as illustrated in Figure 1-9. We use this SDLC as an example of methodology and a way to think about systems analysis and design. You can apply this methodology to almost any life cycle. As we describe this SDLC throughout the book, it becomes clear that each phase has specific outcomes and deliverables that feed important information to other phases. At the end of each phase (and sometimes within phases for intermediate steps), a systems development project reaches a milestone. Then, as deliverables are produced, they are often reviewed by parties outside the project team, including managers and executives.

Phase 1: Systems Planning and Selection

Systems planning and selection
The first phase of the SDLC, in which an organization's total information system needs are analyzed and arranged, and in which a potential information systems project is identified and an argument for continuing or not continuing with the project is presented.

The first phase in the SDLC, **systems planning and selection,** has two primary activities. First, someone identifies the need for a new or enhanced system. Information needs of the organization are examined, and projects to meet these needs are identified. The organization's information system needs may result from:

- Requests to deal with problems in current procedures
- The desire to perform additional tasks
- The realization that information technology could be used to capitalize on an existing opportunity

The systems analyst prioritizes and translates the needs into a written plan for the information systems (IS) department, including a schedule for developing new major systems. Requests for new systems spring from users who need new or enhanced systems. During the systems planning and selection phase, an organization determines whether resources should be devoted to the development or enhancement of each information system under consideration. A *feasibility study* is conducted before the second phase of the SDLC to determine the economic and organizational impact of the system.

The second task in the systems planning and selection phase is to investigate the system and determine the proposed system's scope. The team of systems analysts then produces a specific plan for the proposed project for the team to follow. This baseline project plan customizes the standardized SDLC and specifies the time and resources needed for its execution. The formal definition of a project is based on the likelihood that the organization's IS department is able to develop a system that will solve the problem or exploit the opportunity and determine whether the costs of developing the system outweigh the possible benefits. The final presentation to the organization's management of the plan for proceeding with the subsequent project phases is usually made by the project leader and other team members.

Phase 2: Systems Analysis

Systems analysis
Phase of the SDLC in which the current system is studied and alternative replacement systems are proposed.

The second phase of the systems development life cycle is **systems analysis.** During this phase, the analyst thoroughly studies the organization's current procedures and the information systems used to perform tasks such as general ledger, shipping, order entry, machine scheduling, and payroll. Analysis has several subphases. The first subphase involves determining the requirements of the system. In this subphase, you and other analysts work with users to determine what the users want from a proposed system. This subphase involves a careful study of any current systems, manual and computerized, that might be replaced or enhanced as part of this project. Next, you study the requirements and structure them according to their interrelationships, eliminating any redundancies. As part of structuring, you generate alternative initial designs to match the requirements. Then you compare these alternatives to determine which best meets the requirements within the cost, labor, and technical levels the organization is willing to commit to the development process. The output of the analysis phase is a description of the alternative solution recommended by the analysis team. Once the recommendation is accepted by the organization, you can make plans to acquire any hardware and system software necessary to build or operate the system as proposed.

Phase 3: Systems Design

The third phase of the SDLC is called **systems design.** During systems design, analysts convert the description of the recommended alternative solution into logical and then physical system specifications. You must design all aspects of the system from input and output screens to reports, databases, and computer processes.

Systems design
Phase of the SDLC in which the system chosen for development in systems analysis is first described independently of any computer platform, (logical design) and is then transformed into technology-specific details (physical design) from which all programming and system construction can be accomplished.

Logical design is not tied to any specific hardware and systems software platform. Theoretically, the system you design could be implemented on any hardware and systems software. Logical design concentrates on the business aspects of the system; that is, how the system will impact the functional units within the organization. Figure 1-11 shows both the logical design for a product and its physical design, side by side, for comparison. You can see from the comparison that many specific decisions had to be made to move from the logical model to the physical product. The situation is similar in information systems design.

In physical design, you turn the logical design into physical, or technical, specifications. For example, you must convert diagrams that map the origin, flow, and processing of data in a system into a structured systems design that can then be broken down into smaller and smaller units for conversion to instructions written in a programming language. You design the various parts of the system to perform the physical operations necessary to facilitate data capture, processing, and information output. During physical design, the analyst team decides which programming languages the computer instructions will be written in, which database systems and file structures will be used for the data, and which hardware platform, operating system, and network environment the system will run under. These decisions finalize the hardware and software plans initiated at the end of the analysis phase. Now you can acquire any new technology not already present in the organization. The final product of the design phase is the physical system specifications, presented in a form, such as a diagram or written report, ready to be turned over to programmers and other system builders for construction.

Phase 4: Systems Implementation and Operation

The final phase of the SDLC is a two-step process: **systems implementation and operation.** During systems implementation and operation, you turn system specifications into a working system that is tested and then put into use. Implementation includes coding, testing, and installation. During coding, programmers write the programs that make up the system. During testing, programmers and analysts test individual programs and the entire system in

Systems implementation and operation
Final phase of the SDLC, in which the information system is coded, tested, and installed in the organization, and in which the information system is systematically repaired and improved.

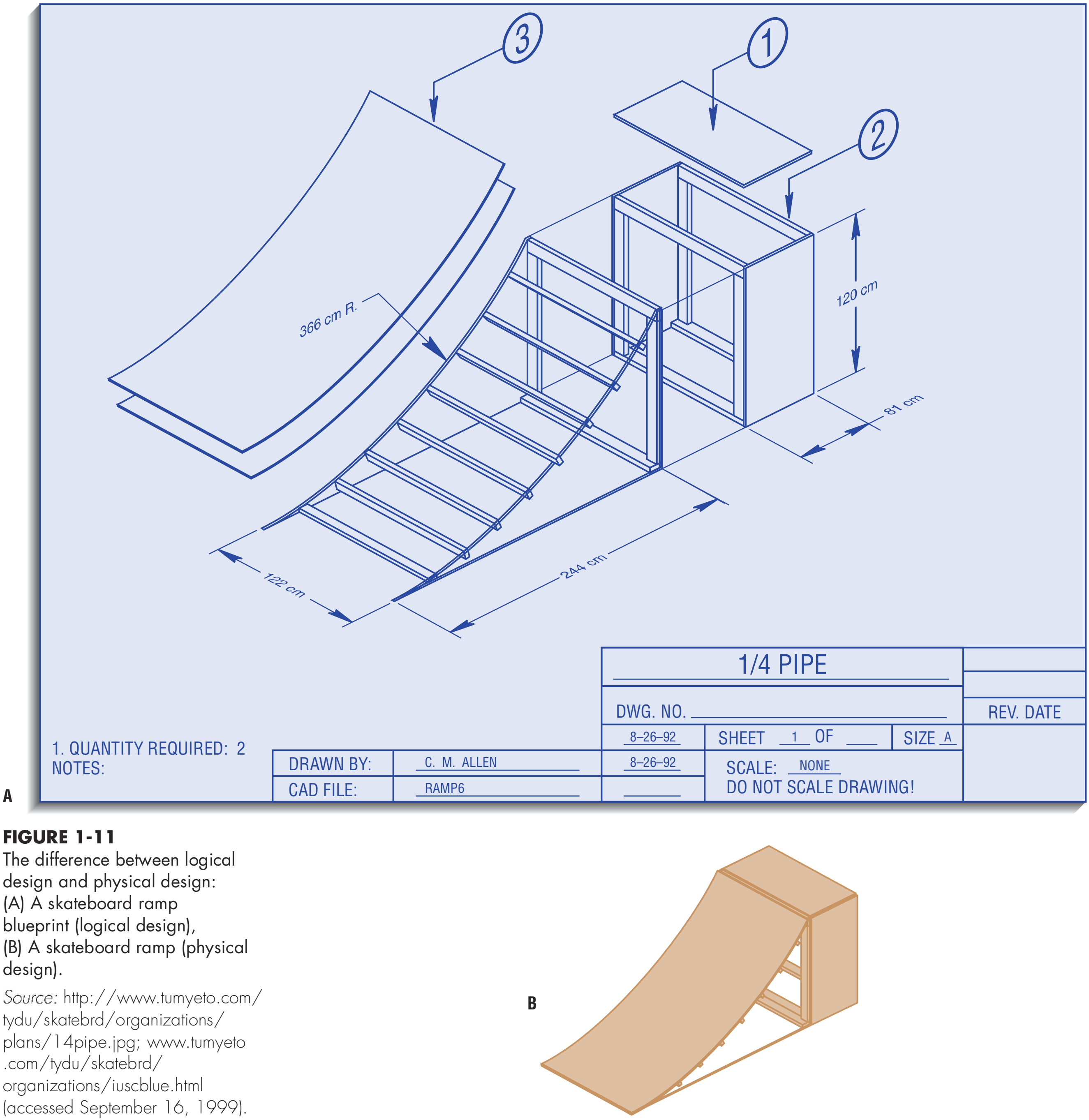


FIGURE 1-11
The difference between logical design and physical design:
(A) A skateboard ramp blueprint (logical design),
(B) A skateboard ramp (physical design).

Source: http://www.tumyeto.com/tydu/skatebrd/organizations/plans/14pipe.jpg; www.tumyeto.com/tydu/skatebrd/organizations/iuscblue.html (accessed September 16, 1999). Reprinted by permission of the International Association of Skateboard Companies.

order to find and correct errors. During installation, the new system becomes a part of the daily activities of the organization. Application software is installed, or loaded, on existing or new hardware; then users are introduced to the new system and trained. Begin planning for both testing and installation as early as the project planning and selection phase, because they both require extensive analysis in order to develop exactly the right approach.

Systems implementation activities also include initial user support such as the finalization of documentation, training programs, and ongoing user assistance. Note that documentation and training programs are finalized during implementation; documentation is produced throughout the life cycle, and training (and education) occurs from the inception of a project. Systems implementation can continue for as long as the system exists because ongoing user support is also part of implementation. Despite the best efforts of analysts, managers, and programmers, however, installation is not always a simple process. Many well-designed systems have failed because the installation process was faulty. Note that even a well-designed system can fail if implementation is not well managed. Because the management of systems implementation is usually done by the project team, we stress implementation issues throughout this book.

The second part of the fourth phase of the SDLC is operation. While a system is operating in an organization, users sometimes find problems with how it works and often think of improvements. During operation, programmers make the changes that users ask for and modify the system to reflect changing business conditions. These changes are necessary to keep the system running and useful. The amount of time and effort devoted to system enhancements during operation depends a great deal on the performance of the previous phases of the life cycle. Inevitably, the time comes when an information system is no longer performing as desired, when the costs of keeping a system running become prohibitive, or when an organization's needs have changed substantially. Such problems indicate that it is time to begin designing the system's replacement, thereby completing the loop and starting the life cycle over again.

The SDLC is a highly linked set of phases whose products feed the activities in subsequent phases. Table 1-1 summarizes the outputs or products of each phase based on the preceding descriptions. The subsequent chapters on the SDLC phases discuss the products of each phase and how they are developed.

Throughout the systems development life cycle, the systems development project itself needs to be carefully planned and managed. The larger the systems project, the greater the need for project management. Several project management techniques have been developed in the last quarter-century, and many have been improved through automation. Chapter 3 contains a more detailed treatment of project planning and management techniques.

TABLE 1-1: Products of the SDLC Phases

Phase	Products, Outputs, or Deliverables
Systems planning and selection	Priorities for systems and projects
	Architecture for data, networks, hardware, and IS management
	Detailed work plan for selected project
	Specification of system scope
	System justification or business case
Systems analysis	Description of current system
	General recommendation on how to fix, enhance, or replace current system
	Explanation of alternative systems and justification for chosen alternative
	Acquisition plan for new technology
Systems design	Detailed specifications of all system elements
Systems implementation and operation	Code
	Documentation
	Training procedures and support capabilities
	New versions or releases of software with associated updates to documentation, training, and support

Alternative Approaches to Development

Prototyping, computer-aided software engineering (CASE) tools, joint application design (JAD), rapid application development (RAD), participatory design (PD), and the use of Agile Methodologies represent different approaches that streamline and improve the systems analysis and design process from different perspectives.

Prototyping

Prototyping
Building a scaled-down version of the desired information system.

Designing and building a scaled-down but working version of a desired system is known as **prototyping.** A prototype can be developed with a CASE tool, a software product that automates steps in the systems development life cycle. CASE tools make prototyping easier and more creative by supporting the design of screens and reports and other parts of a system interface. CASE tools also support many of the diagramming techniques you will learn, such as data-flow diagrams and entity-relationship diagrams.

Figure 1-12 illustrates prototyping. The analyst works with users to determine the initial or basic requirements for the system. The analyst then quickly builds a prototype. When the prototype is completed, the users work with it and tell the analyst what they like and do not like about it. The analyst uses this feedback to improve the prototype and takes the new version back to the users. This iterative process continues until the users are relatively satisfied with what they have seen. The key advantages of the prototyping technique are: (1) it involves the user in analysis and design, and (2) it captures requirements in concrete, rather than verbal or abstract, form. In addition to being used as a stand-alone, prototyping may also be used to augment the SDLC. For example, a prototype of the final system may be developed early in analysis to help the analysts identify what users want. Then the final system is developed based on the specifications of the prototype. We discuss prototyping in greater detail in Chapter 5 and use various prototyping tools in Chapter 9 to illustrate the design of system outputs.

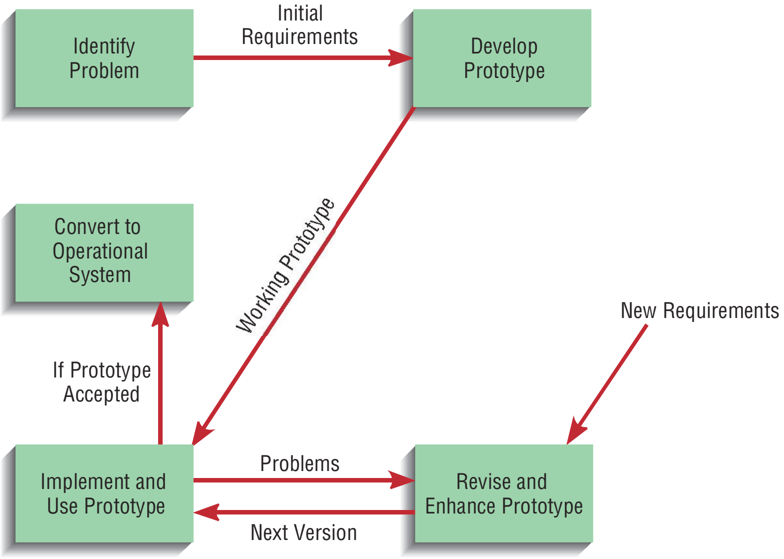
Computer-Aided Software Engineering (CASE) Tools

Computer-aided software engineering (CASE)
Software tools that provide automated support for some portion of the systems development process.

Computer-aided software engineering (CASE) refers to automated software tools used by systems analysts to develop information systems. These tools can be used to automate or support activities throughout the systems development process with the objective of increasing productivity and improving the overall quality of systems. CASE helps provide an engineering-type discipline to software

FIGURE 1-12
The prototyping method.

Source: Adapted from J. D. Naumann and A. M. Jenkins, "Prototyping: The New Paradigm for Systems Development," *MIS Quarterly* 6, no. 3 (1982): 29–44.



development and to the automation of the entire software life-cycle process, sometimes with a single family of integrated software tools. In general, CASE assists systems builders in managing the complexities of information system projects and helps ensure that high-quality systems are constructed on time and within budget.

Vendors of CASE products have "opened up" their systems through the use of standard databases and data-conversion utilities to share information across products and tools easier. An integrated and standard database called a **repository** is the common method for providing product and tool integration and has been a key factor in enabling CASE to manage larger, more complex projects easier and to seamlessly integrate data across various tools and products. The general types of CASE tools include:

- Diagramming tools that enable system process, data, and control structures to be represented graphically.
- Computer display and report generators that help prototype how systems "look and feel" to users. Display (or form) and report generators also make it easier for the systems analyst to identify data requirements and relationships.
- Analysis tools that automatically check for incomplete, inconsistent, or incorrect specifications in diagrams, forms, and reports.
- A central repository that enables the integrated storage of specifications, diagrams, reports, and project management information.
- Documentation generators that help produce both technical and user documentation in standard formats.
- Code generators that enable the automatic generation of program and database definition code directly from the design documents, diagrams, forms, and reports.

Repository
A centralized database that contains all diagrams, forms and report definitions, data structures, data definitions, process flows and logic, and definitions of other organizational and system components; it provides a set of mechanisms and structures to achieve seamless data-to-tool and data-to-data integration.

Joint Application Design

In the late 1970s, systems development personnel at IBM developed a new process for collecting information system requirements and reviewing system designs. The process is called **joint application design (JAD).** The idea behind JAD is to structure the requirements determination phase of analysis and the reviews that occur as part of the design. Users, managers, and systems developers are brought together for a series of intensive structured meetings run by a JAD session leader. By gathering the people directly affected by an IS in one room at the same time to work together to agree on system requirements and design details, time and organizational resources are better managed. Group members are more likely to develop a shared understanding of what the IS is supposed to do. JAD has become common in certain industries, such as insurance, and in specific companies, such as CIGNA. We discuss JAD in more detail in Chapter 5.

Joint application design (JAD)
A structured process in which users, managers, and analysts work together for several days in a series of intensive meetings to specify or review system requirements.

Rapid Application Development

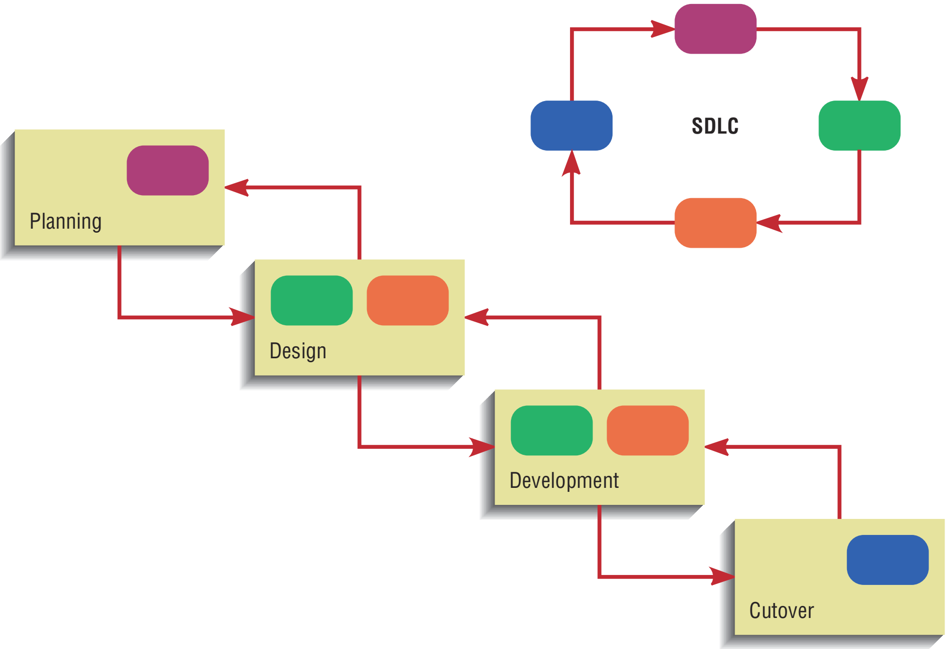
Prototyping, CASE, and JAD are key tools that support **rapid application development (RAD).** The fundamental principle of any RAD methodology is to delay producing detailed system design documents until after user requirements are clear. The prototype serves as the working description of needs. RAD involves gaining user acceptance of the interface and developing key system capabilities as quickly as possible. RAD is widely used by consulting firms. It is also used as an in-house methodology by firms such as the Boeing Company. RAD sacrifices computer efficiency for gains in human efficiency in rapidly building and rebuilding working systems. On the other hand, RAD methodologies can overlook important systems development principles, which may result in problems with systems developed this way.

Rapid application development (RAD)
Systems development methodology created to radically decrease the time needed to design and implement information systems.

RAD grew out of the convergence of two trends: the increased speed and turbulence of doing business in the late 1980s and early 1990s, and the ready availability of high-powered computer-based tools to support systems development and easy maintenance. As the conditions of doing business in a changing, competitive global environment became more turbulent, management in many organizations began to question whether it made sense to wait two to three years to develop systems that would be obsolete upon completion. On the other hand, CASE tools and prototyping software were diffusing throughout organizations, making it relatively easy for end users to see what their systems would look like before they were completed. Why not use these tools to address the problems of developing systems more productively in a rapidly changing business environment? So RAD was born.

As Figure 1-13 shows, the same phases followed in the traditional SDLC are also followed in RAD, but the phases are combined to produce a more streamlined development technique. Planning and design phases in RAD are shortened by focusing work on system functional and user interface requirements at the expense of detailed business analysis and concern for system performance issues. Also, usually RAD looks at the system being developed in isolation from other systems, thus eliminating the time-consuming activities of coordinating with existing standards and systems during design and development. The emphasis in RAD is generally less on the sequence and structure of processes in the life cycle and more on doing different tasks in parallel with each other and on using prototyping extensively. Notice also, that the iteration in the RAD life cycle is limited to the design and development phases, which is where the bulk of the work in a RAD approach takes place. Although it is possible in RAD to return to planning once design has begun, it is rarely done. Similarly, although it is possible to return to development from the cutover phase (when the system is turned over to the user), RAD is designed to minimize iteration at this point in the life cycle. The high level of user commitment and involvement throughout RAD implies that the system that emerges should be more readily accepted by the user community (and hence more easily implemented during cutover) than would a system developed using traditional techniques.

FIGURE 1-13
RAD systems development life cycle compared to standard SDLC.



Participatory Design

Developed in northern Europe, **participatory design (PD)** represents a viable alternative approach to the SDLC. One of the best-known companies that has used this approach is StatoilHydro, the Norwegian oil company. PD emphasizes the role of the user much more than do traditional North American techniques such as structured analysis and structured design. In some cases, PD may involve the entire user community in the development process. Each user has an equal voice in determining system requirements and in approving system design. In other cases, an elected group of users controls the process. These users represent the larger community, much as a legislature represents the needs and wants of the electorate. Typically, under PD, systems analysts work for the users. The organization's management and outside consultants provide advice rather than control. PD is partly a result of the roles of labor and management in the northern European workplace where labor is more organized, carries more clout, and is more intimately involved with technological changes than is true in North America.

Participatory design (PD)
A systems development approach that originated in northern Europe, in which users and the improvement of their work lives are the central focus.

Agile Methodologies

As you might imagine, many other approaches to systems analysis and design have been developed over the years. These approaches include eXtreme Programming, the Crystal family of methodologies, Adaptive Software Development, Scrum, and Feature Driven Development. In February 2001, many of the proponents of these alternative approaches met in Utah in the United States and reached a consensus on many of the underlying principles their various approaches contained. This consensus turned into a document they called "The Agile Manifesto" (see Appendix B for more detail). These **Agile Methodologies** share three key principles: (1) a focus on adaptive rather than predictive methodologies, (2) a focus on people rather than roles, and (3) a self-adaptive process. Adopting an adaptive rather than predictive methodology refers to the observation that engineering-based methodologies work best when the process and product are predictive. Software tends not to be as predictive as, say, a bridge, especially in today's turbulent business environment. More adaptive methodologies are needed, then, and the Agile Methodologies are based on the ability to adapt quickly. The focus on people rather than roles is also a criticism of engineering-based techniques, where people became interchangeable. An Agile approach views people as talented individuals, not people filling roles, each of whom has unique talents to bring to a development project. Finally, Agile Methodologies promote a self-adaptive software development process. As the methodologies are applied, they should also be adapted by a particular development team working on a particular project in a particular context. No single monolithic methodology effectively fits all developers on all projects at all times. You will learn much more about Agile Methodologies in Appendix B.

Agile Methodologies
A family of development methodologies characterized by short iterative cycles and extensive testing; active involvement of users for establishing, prioritizing, and verifying requirements; and a focus on small teams of talented, experienced programmers.

Key Points Review

1. **Define information systems analysis and design.**

   Systems analysis and design is the complex organizational process whereby computer-based information systems are developed and operated.

2. **Describe the role of the systems analyst in information systems development.**

   Systems analysts play a key organizational role in systems development. They act as liaisons between business users on one hand and technical personnel on the other. Analysts need to develop four sets of skills in order to succeed: analytical, technical, managerial, and interpersonal.

3. **Describe the information systems development life cycle (SDLC).**

   The SDLC used in this book has four major phases: (1) systems planning and selection, (2) systems analysis, (3) systems design, and (4) systems implementation and operation. In the first phase, which is planning and selection, analysts make detailed road maps of the system development project. In analysis, analysts work to solve the business problem being studied. In design, the solution to the problem is built. Finally, in the last phase, the system is given to users and kept running.

4. **List alternatives to the SDLC, including a description of the role of computer-aided software engineering (CASE) tools in systems development.**

   The alternative frameworks mentioned in this chapter are prototyping, joint application design (JAD), rapid application development (RAD), participatory design (PD), and Agile Methodologies. Using prototyping, analysts build a working model of the system. In JAD, analysts and users meet to solve problems and design systems. RAD decreases the time needed to design and implement information systems. In PD, the emphasis is on the user community. Agile Methodologies focus on adaptive rather than predictive methodologies, on people rather than roles. CASE tools represent the use of information technology to assist in the systems development process. They include diagramming tools, screen and report design tools, and other special-purpose tools. CASE tools help programmers and analysts do their jobs efficiently and effectively by automating routine tasks.

Key Terms Checkpoint

Here are the key terms from the chapter. The page where each term is first explained is in parentheses after the term.

1. **Agile Methodologies (p. 21)**
2. **Application software (p. 4)**
3. **Boundary (p. 7)**
4. **Cohesion (p. 10)**
5. **Component (p. 7)**
6. **Computer-aided software engineering (CASE) (p. 18)**
7. **Constraint (p. 7)**
8. **Coupling (p. 9)**
9. **Decomposition (p. 8)**
10. **Environment (p. 7)**
11. **Information systems analysis and design (p. 4)**
12. **Interface (p. 7)**
13. **Interrelated (p. 7)**
14. **Joint application design (JAD) (p. 19)**
15. **Modularity (p. 9)**
16. **Participatory design (PD) (p. 21)**
17. **Prototyping (p. 18)**
18. **Purpose (p. 7)**
19. **Rapid application development (RAD) (p. 19)**
20. **Repository (p. 19)**
21. **System (p. 6)**
22. **Systems analysis (p. 14)**
23. **Systems analyst (p. 11)**
24. **Systems design (p. 15)**
25. **Systems development life cycle (SDLC) (p. 12)**
26. **Systems development methodology (p. 12)**
27. **Systems implementation and operation (p. 15)**
28. **Systems planning and selection (p. 14)**

Match each of the key terms above with the definition that best fits it.

____ 1. The first phase of the SDLC in which an organization's total information system needs are analyzed and arranged, and in which a potential information systems project is identified and an argument for continuing or not continuing with the project is presented.

____ 2. The process of developing and maintaining an information system.

____ 3. A systems development approach that originated in northern Europe, in which users and the improvement of their work lives are the central focus.

____ 4. Software designed to process data and support users in an organization. Examples include spreadsheets, word processors, and database management systems.

____ 5. The organizational role most responsible for the analysis and design of information systems.

____ 6. A structured process in which users, managers, and analysts work together for several days in a series of intensive meetings to specify or review system requirements.

____ 7. Building a scaled-down version of the desired information system.

____ 8. A group of interrelated procedures used for a business function, with an identifiable boundary, working together for some purpose.

____ 9. An irreducible part or aggregation of parts that make up a system, also called a *subsystem*.

____ 10. Dependence of one part of the system on one or more other system parts.

____ 11. The line that marks the inside and outside of a system and that sets off the system from its environment.

_____ 12. The overall goal or function of a system.

_____ 13. Phase of the SDLC, in which the system chosen for development in systems analysis is first described independently of any computer platform and is then transformed into technology-specific details from which all programming and system construction can be accomplished.

_____ 14. Phase of the SDLC, in which the current system is studied and alternative replacement systems are proposed.

_____ 15. Everything external to a system that interacts with the system.

_____ 16. Point of contact where a system meets its environment or where subsystems meet each other.

_____ 17. A limit to what a system can accomplish.

_____ 18. Final phase of the SDLC, in which the information system is coded, tested, and installed in the organization, and in which the information system is systematically repaired and improved.

_____ 19. A standard process followed in an organization to conduct all the steps necessary to analyze, design, implement, and maintain information systems.

_____ 20. The series of steps used to mark the phases of development for an information system.

_____ 21. The process of breaking the description of a system down into small components; also known as *functional decomposition.*

_____ 22. Dividing a system up into chunks or modules of a relatively uniform size.

_____ 23. The extent to which subsystems depend on each other.

_____ 24. The extent to which a system or subsystem performs a single function.

_____ 25. Software tools that provide automated support for some portion of the systems development process.

_____ 26. A centralized database that contains all diagrams, forms and report definitions, data structures, data definitions, process flows and logic, and definitions of other organizational and system components; it provides a set of mechanisms and structures to achieve seamless data-to-tool and data-to-data integration.

_____ 27. Systems development methodology created to radically decrease the time needed to design and implement information systems.

_____ 28. Current approaches to systems development that focus on adaptive methodologies, people instead of roles, and an overall self-adaptive development process.

Review Questions

1. What is information systems analysis and design?
2. What is systems thinking? How is it useful for thinking about computer-based information systems?
3. What is decomposition? Coupling? Cohesion?
4. In what way are organizations systems?
5. List and explain the different phases in the systems development life cycle.
6. What is prototyping?
7. What are CASE tools? What is a CASE repository and how is it used?
8. What is JAD? What is participatory design?
9. What is RAD? How does it compare to the typical SDLC?
10. What are Agile Methodologies?

Problems and Exercises

1. Why is it important to use systems analysis and design methodologies when building a system? Why not just build the system in whatever way seems to be "quick and easy?" What value is provided by using an "engineering" approach?
2. Describe your university or college as a system. What is the input? The output? The boundary? The components? Their interrelationships? The constraints? The purpose? The interfaces? The environment? Draw a diagram of this system.
3. A car is a system with several subsystems, including the braking subsystem, the electrical subsystem, the engine, the fuel subsystem, the climate-control subsystem, and the passenger subsystem. Draw a diagram of a car as a system and label all of its system characteristics.
4. Your personal computer is a system. Draw and label a personal computer as a system as you did for a car in Problem and Exercise 3.
5. Choose a business transaction you undertake regularly, such as using an ATM machine, buying groceries at the supermarket, or buying a ticket for a university's basketball game. For this transaction, define the data, draw the data-flow diagram, and describe processing logic.
6. How is the joint application design (JAD) approach different from the participatory design (PD) approach developed in northern Europe? (You may

have to do some digging at the library to answer this question adequately.) What are the benefits in using these types of approaches in building information systems? What are the barriers?

7. How would you organize a project team of students to work with a small business client? How would you organize a project team if you were working for a professional consulting organization? How might these two methods of organization differ? Why?
8. How might prototyping be used as part of the SDLC?
9. Describe the difference in the role of a systems analyst in the SDLC versus prototyping.
10. Compare Figures 1-9 and 1-10. What similarities and differences do you see?

Discussion Questions

1. If someone at a party asked you what a systems analyst was and why anyone would want to be one, what would you say? Support your answer with evidence from this chapter.
2. Explain how a computer-based information system designed to process payroll is a specific example of a system. Be sure to account for all nine components of any system in your explanation.
3. How does the Internet, and more specifically the World Wide Web, fit into the picture of systems analysis and systems development drawn in this chapter?
4. What do you think systems analysis and design will look like in the next decade? As you saw earlier in the chapter, changes in systems development have been pretty dramatic in the past. A computer programmer suddenly transported from the 1950s to the 2000s would have trouble recognizing the computing environment that had evolved just fifty years later. What dramatic changes might occur in the next ten years?

Case Problems

1. Pine Valley Furniture

Alex Schuster began Pine Valley Furniture (PVF) as a hobby. Initially, Alex would build custom furniture in his garage for friends and family. As word spread about his quality craftsmanship, he began taking orders. The hobby has since evolved into a medium-sized business, employing more than fifty workers.

Over the years, increased demand has forced Alex to relocate several times, increase his sales force, expand his product line, and renovate Pine Valley Furniture's information systems. As the company began to grow, Alex organized the company into functional areas—manufacturing, sales, orders, accounting, and purchasing. Originally, manual information systems were used; however, as the business began to expand rapidly, a minicomputer was installed to automate applications.

In the beginning, a process-oriented approach was utilized. Each separate application had its own data files. The applications automated the manual systems on which they were modeled. In an effort to improve its information systems, PVF recently renovated its information systems, resulting in a company-wide database and applications that work with this database. Pine Valley Furniture's computer-based applications are primarily in the accounting and financial areas. All applications have been built in-house, and when necessary, new information systems staff is hired to support Pine Valley Furniture's expanding information systems.

a. How did PVF go about developing its information systems? Why do you think the company chose this option? What other options were available?
b. One option available to PVF was an enterprise-wide system. What features does an enterprise-wide system, such as SAP, provide? What is the primary advantage of an enterprise-wide system?
c. PVF will be hiring two systems analysts next month. Your task is to develop a job advertisement for these positions. Locate several Web sites or newspapers that have job advertisements for systems analysts. What skills are required?
d. What types of information systems are currently utilized at PVF? Provide an example of each.

2. Hoosier Burger

As college students in the 1970s, Bob and Thelma Mellankamp often dreamed of starting their own business. While on their way to an economics class, Bob and Thelma drove by Myrtle's Family Restaurant and noticed a "for sale" sign in the window.