

The Systems Development Environment

Learning Objectives

After studying this chapter, you should be able to

- 1.1 define *information systems analysis and design*,
- 1.2 describe the information systems development life cycle (SDLC),
- 1.3 explain computer-aided software engineering (CASE) tools,

- 1.4 describe the Agile Methodologies and eXtreme Programming, and
- 1.5 explain object-oriented analysis and design and the Rational Unified Process (RUP).

Introduction

Information systems analysis and design is a complex, challenging, and stimulating organizational process that a team of business and systems professionals uses to develop and maintain computer-based information systems. Although advances in information technology continually give us new capabilities, the analysis and design of information systems is driven from an organizational perspective. An organization might consist of a whole enterprise, specific departments, or individual work groups. Organizations can respond to and anticipate problems and opportunities through innovative use of information technology. Information systems analysis and design is therefore an organizational improvement process. Systems are built and rebuilt for organizational benefits. Benefits result from adding value during the process of creating, producing, and supporting the organization's products and services. Thus the analysis and design of information systems is based on your understanding of the organization's objectives, structure, and processes, as well as your knowledge of how to exploit information technology for advantage.

In the current business environment, the Internet, especially the World Wide Web, has been firmly integrated into an organization's way of doing business. Although you are probably most familiar with marketing done on the web and web-based retailing sites, such as eBay or Amazon.com, the overwhelming majority of business use of the web is business-to-business applications.

These applications run the gamut of everything businesses do, including transmitting orders and payments to suppliers, fulfilling orders and collecting payments from customers, maintaining business relationships, and establishing electronic marketplaces where businesses can shop online for the best deals on resources they need for assembling their products and services. Although the Internet seems to pervade business these days, it is important to remember that many of the key aspects of business—offering a product or service for sale, collecting payment, paying employees, maintaining supplier and client relationships—have not changed. Understanding the business and how it functions is still the key to successful systems development, even in the fast-paced, technology-driven environment that organizations find themselves in today.

Careers in information technology (IT) present a great opportunity for you to make a significant and visible impact on business. The demand for skilled information technology workers is growing. According to the US Bureau of Labor Statistics, the professional IT workforce will grow by more than 22 percent between 2010 and 2020 (Thibodeau, 2012). The fastest growth will come for software developers (32 percent) and database administrators (31 percent). One particular aspect of the information technology industry, cloud computing, created almost 14 million technology and related jobs between 2012 and 2015 (McDougall, 2012). Annual revenues from

Information systems analysis and design

The complex organizational process whereby computer-based information systems are developed and maintained.

cloud computing will be over \$1.1 trillion (USD) starting that year. And the growth will be global, with the number of cloud computing jobs in Brazil increasing by 186 percent, the number of jobs in China and India almost doubling, and growth in cloud-related jobs increasing by 66 percent in the United States. (See more about cloud computing in Chapter 2.) With the challenges and opportunities of dealing with rapid advances in technology, it is difficult to imagine a more exciting career choice than information technology, and systems analysis and design is a big part of the IT landscape. Furthermore, analyzing and designing information systems will give you the chance to understand organizations at a depth and breadth that might take many more years to accomplish in other careers.

Application software

Computer software designed to support organizational functions or processes.

An important (but not the only) result of systems analysis and design is **application software**, software designed to support a specific organizational function or process, such as inventory management, payroll, or market analysis. In addition to application software, the total information system includes the hardware and systems software on which the application software runs, documentation and training materials, the specific job roles associated with the overall system, controls, and the people who use the software along with their work methods. Although we will address all of these various dimensions of the overall system, we will emphasize application software development—your primary responsibility as a systems analyst.

In the early years of computing, analysis and design was considered an art. Now that the need for systems and software has become so great, people in industry and academia have developed work methods that make analysis and design a disciplined process. Our goal is to help you develop the knowledge and skills needed to understand and follow such software engineering processes. Central to software engineering processes (and to this book) are various methodologies, techniques, and tools that have been developed, tested, and widely used over the years to assist people like you during systems analysis and design.

Methodologies are comprehensive, multiple-step approaches to systems development that will guide your work and influence the quality of your final product—the information system. A methodology adopted by an organization will be consistent with its general management style (e.g., an organization’s orientation toward consensus management will influence its choice of systems development methodology). Most methodologies incorporate several development techniques.

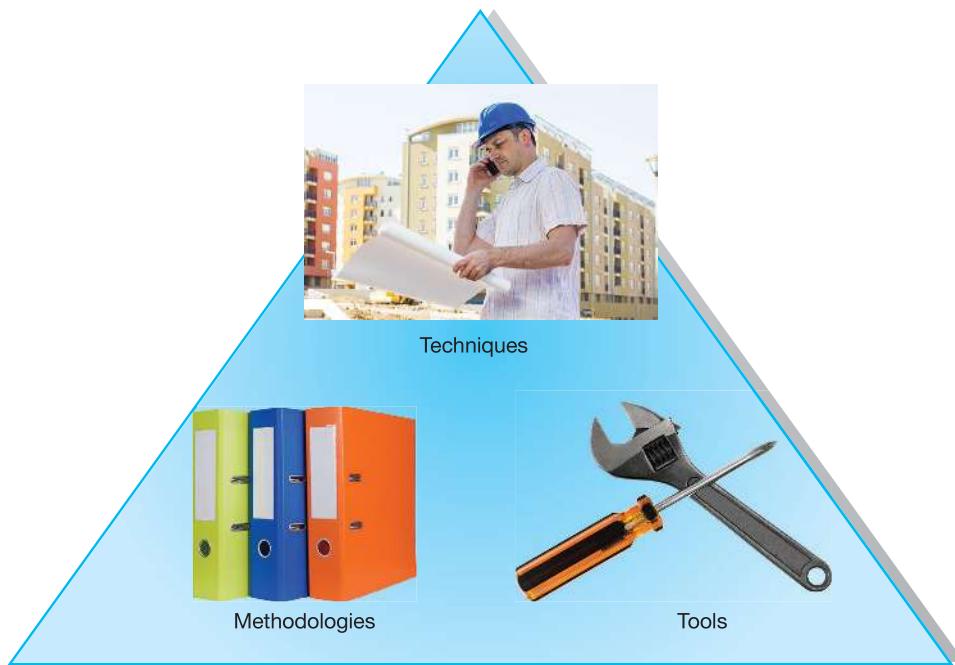
Techniques are particular processes that you, as an analyst, will follow to help ensure that your work is well thought out, complete, and comprehensible to others on your project team. Techniques provide support for a wide range of tasks, including conducting thorough interviews to determine what your system should do, planning and managing the activities in a systems development project, diagramming the system’s logic, and designing the reports your system will generate.

Tools are typically computer programs that make it easy to use and benefit from techniques and to faithfully follow the guidelines of the overall development methodology. To be effective, techniques and tools must both be consistent with an organization’s systems development methodology. Techniques and tools must make it easy for systems developers to conduct the steps called for in the methodology. These three elements—methodologies, techniques, and tools—work together to form an organizational approach to systems analysis and design (see Figure 1-1).

Systems analyst

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

Although many people in organizations are responsible for systems analysis and design, in most organizations the **systems analyst** has the primary responsibility. When you begin your career in systems development, you will most likely begin as a systems analyst or as a programmer with some systems analysis responsibilities. 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. As such, a systems analyst is an agent of change and innovation.

**FIGURE 1-1**

An organizational approach to systems analysis and design is driven by methodologies, techniques, and tools

Sources: Top: Mitarart/Fotolia; Left: Lev/Fotolia; Right: PaulPaladin/Fotolia

In the rest of this chapter, we will examine the systems approach to analysis and design. You will learn how systems analysis and design has changed over the decades as computing has become more central to business. You will learn about the systems development life cycle, which provides the basic overall structure of the systems development process and of this book. This chapter ends with a discussion of some of the methodologies, techniques, and tools created to support the systems development process.

A MODERN APPROACH TO SYSTEMS ANALYSIS AND DESIGN

The analysis and design of computer-based information systems began in the 1950s. Since then, the development environment has changed dramatically, driven by organizational needs as well as by rapid changes in the technological capabilities of computers. In the 1950s, development focused on the processes the software performed. Because computer power was a critical resource, efficiency of processing became the main goal. Computers were large, expensive, and not very reliable. Emphasis was placed on automating existing processes, such as purchasing or paying, often within single departments. All applications had to be developed in machine language or assembly language, and they had to be developed from scratch because there was no software industry. Because computers were so expensive, computer memory was also at a premium, so system developers conserved as much memory as possible for data storage.

The first procedural, or third-generation, computer programming languages did not become available until the beginning of the 1960s. Computers were still large and expensive, but the 1960s saw important breakthroughs in technology that enabled the development of smaller, faster, less expensive computers—minicomputers—and the beginnings of the software industry. Most organizations still developed their applications from scratch using their in-house development staff. Systems development was more an art than a science. This view of systems development began to change in the 1970s, however, as organizations started to realize how expensive it was to develop customized information systems for every application. Systems development came to be more

disciplined as many people worked to make it more like engineering. Early database management systems, using hierarchical and network models, helped bring discipline to the storage and retrieval of data. The development of database management systems helped shift the focus of systems development from processes first to data first.

The 1980s were marked by major breakthroughs in computing in organizations, as microcomputers became key organizational tools. The software industry expanded greatly as more and more people began to write off-the-shelf software for microcomputers. Developers began to write more and more applications in fourth-generation languages, which, unlike procedural languages, instructed a computer on what to do instead of how to do it. Computer-aided software engineering (CASE) tools were developed to make systems developers' work easier and more consistent. As computers continued to get smaller, faster, and cheaper, and as the operating systems for computers moved away from line prompt interfaces to windows- and icon-based interfaces, organizations moved to applications with more graphics. Organizations developed less software in-house and bought relatively more from software vendors. The systems developer's job went through a transition from builder to integrator.

The systems development environment of the late 1990s focused on systems integration. Developers used visual programming environments, such as PowerBuilder or Visual Basic, to design the user interfaces for systems that run on client/server platforms. The database, which may be relational or object-oriented, and which may have been developed using software from firms such as Oracle, Microsoft, or Ingres, resided on the server. In many cases, the application logic resided on the same server. Alternatively, an organization may have decided to purchase its entire enterprise-wide system from companies such as SAP AG 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. Starting in the middle years of the 1990s, more and more systems development efforts focused on the Internet, especially the web.

Today there is continued focus on developing systems for the Internet and for firms' intranets and extranets. As happened with traditional systems, Internet developers now rely on computer-based tools to speed and simplify the development of web-based systems. Many CASE tools directly support web application development. More and more, systems implementation involves a three-tier design, with the database on one server, the application on a second server, and client logic located on user machines. Another important development is the move to wireless system components. Wireless devices can access web-based applications from almost anywhere. Finally, the trend continues toward assembling systems from programs and components purchased off the shelf. In many cases, organizations do not develop the application in-house. They don't even run the application in-house, choosing instead to use the application on a per-use basis by accessing it through the cloud.

DEVELOPING INFORMATION SYSTEMS AND THE SYSTEMS DEVELOPMENT LIFE CYCLE

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 traditional methodology used to develop, maintain, and replace information systems.

Most organizations find it beneficial to 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 follows a life cycle in that it is created, tested, and introduced to the market. Its sales increase, peak, and decline. Finally, the product is removed from the market and replaced by something else. The **systems development life cycle (SDLC)** is a common methodology for systems development in many organizations; it features several phases that mark the progress of the systems analysis and design effort. Every textbook author and information systems development organization uses a slightly different life-cycle model, with anywhere from 3 to almost 20 identifiable phases.

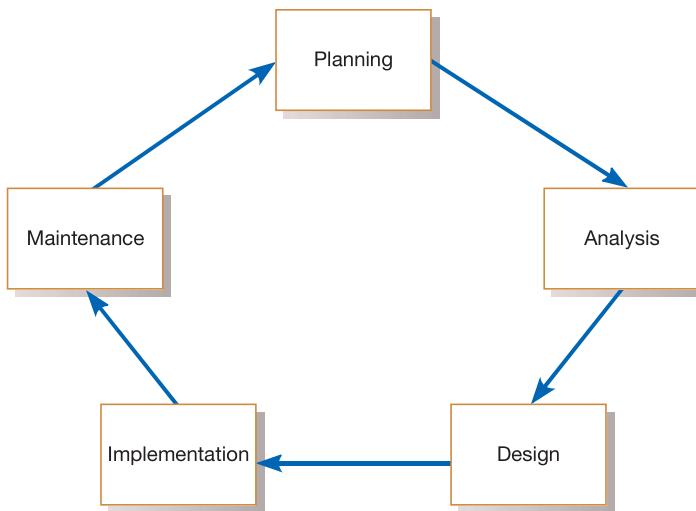


FIGURE 1-2
Systems development life cycle

The life cycle can be thought of as a circular process in which the end of the useful life of one system leads to the beginning of another project that will develop a new version or replace an existing system altogether (see Figure 1-2). At first glance, the life cycle appears to be a sequentially ordered set of phases, but it is not. The specific steps and their sequence are meant to be adapted as required for a project, consistent with management approaches. 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 SDLC, 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 people consider the life cycle to be a spiral, in which we constantly cycle through the phases at different levels of detail (see Figure 1-3). 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. Software is the most obvious end product of the life cycle; other essential outputs include documentation about the system and how it was developed, as well as training for users.

Every medium to large corporation and every custom software producer will have its own specific life cycle or systems development methodology in place

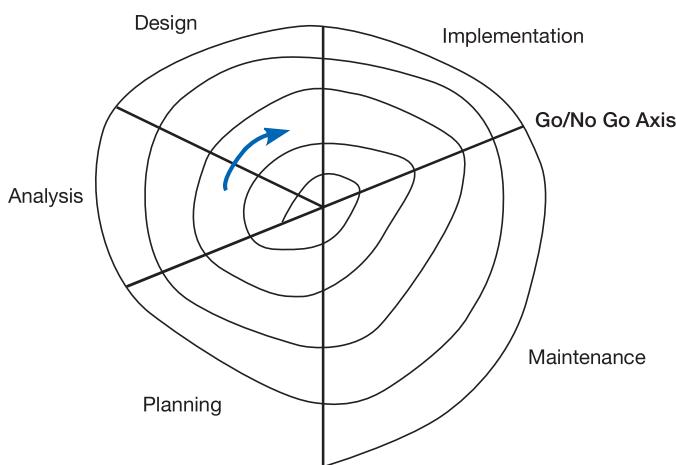


FIGURE 1-3
Evolutionary model

FIGURE 1-4

U.S. Department of Justice's systems development life cycle

(Source: Diagram based on <http://www.justice.gov/archive/jmd/irm/lifecycle/ch1.htm#para1.2>)

Top to bottom: haveseen/Shutterstock; Kruwt/Fotolia; Bedrin/Shutterstock; Pressmaster/Shutterstock; pilot39/Fotolia; Sozajiten; Elnur/Fotolia; rtguest/Shutterstock; michaeljung/Shutterstock; AleksaStudio/Shutterstock



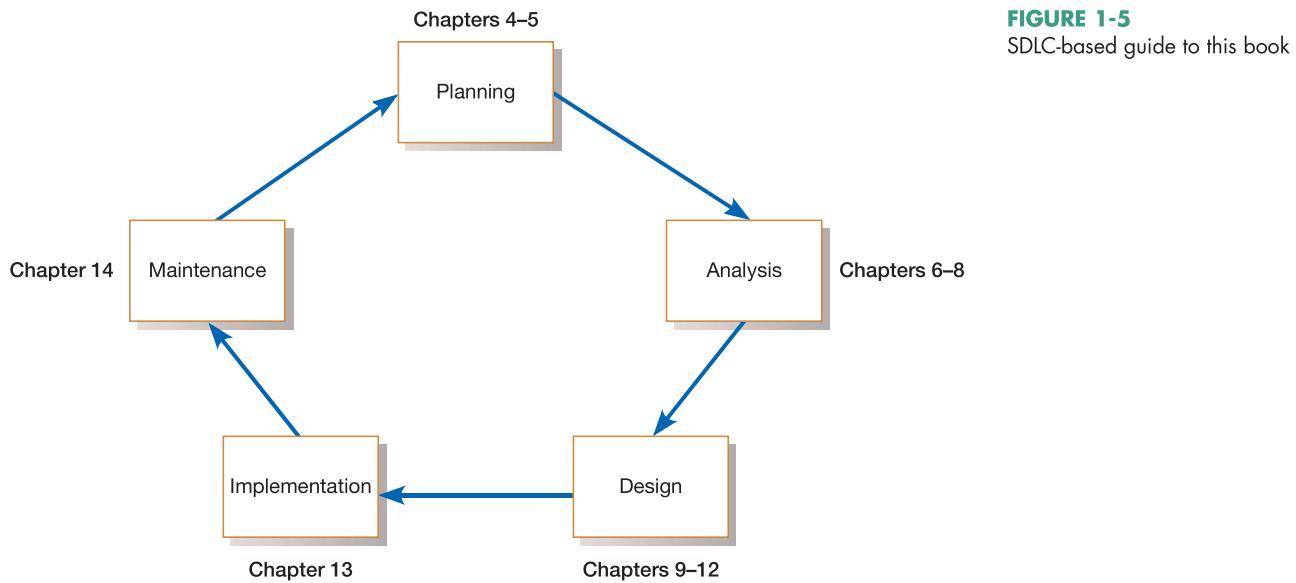
(see Figure 1-4). Even if a particular methodology does not look like a cycle, and Figure 1-4 does not, you will probably discover that many of the SDLC steps are performed and SDLC techniques and tools are used. Learning about systems analysis and design from the life-cycle approach will serve you well no matter which systems development methodology you use.

When you begin your first job, you will likely spend several weeks or months learning your organization's SDLC and its associated methodologies, techniques, and tools. In order to make this book as general as possible, we follow a rather generic life-cycle model, as described in more detail in Figure 1-5. Notice that our model is circular. We use this SDLC as one example of a methodology but, more important, as a way to arrange the topics of systems analysis and design. Thus, what you learn in this book, you can apply to almost any life cycle you might follow. As we describe this SDLC throughout the book, you will see that each phase has specific outcomes and deliverables that feed important information to other phases. At the end of each phase, a systems development project reaches a milestone and, as deliverables are produced, they are often reviewed by parties outside the project team. In the rest of this section, we provide a brief overview of each SDLC phase. At the end of the section, we summarize this discussion in a table that lists the main deliverables or outputs from each SDLC phase.

The first phase in the SDLC is **planning**. In this phase, someone identifies the need for a new or enhanced system. In larger organizations, this recognition may be part of a corporate and systems planning process. Information needs of the organization as a whole are examined, and projects to meet these needs are proactively identified. The organization's information system needs may result from requests to deal with problems in current procedures, from the desire to perform additional

Planning

The first phase of the SDLC in which an organization's total information system needs are identified, analyzed, prioritized, and arranged.



tasks, or from the realization that information technology could be used to capitalize on an existing opportunity. These needs can then be prioritized and translated into a plan for the information systems department, including a schedule for developing new major systems. In smaller organizations (as well as in large ones), determination of which systems to develop may be affected by ad hoc user requests submitted as the need for new or enhanced systems arises, as well as from a formalized information planning process. In either case, during project identification and selection, an organization determines whether resources should be devoted to the development or enhancement of each information system under consideration. The outcome of the project identification and selection process is a determination of which systems development projects should be undertaken by the organization, at least in terms of an initial study.

Two additional major activities are also performed during the planning phase: the formal, yet still preliminary, investigation of the system problem or opportunity at hand and the presentation of reasons why the system should or should not be developed by the organization. A critical step at this point is determining the scope of the proposed system. The project leader and initial team of systems analysts also produce a specific plan for the proposed project the team will follow using the remaining SDLC steps. 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 information systems 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 benefits it could provide. The final presentation of the business case for proceeding with the subsequent project phases is usually made by the project leader and other team members to someone in management or to a special management committee with the job of deciding which projects the organization will undertake.

The second phase in the SDLC is **analysis**. During this phase, the analyst thoroughly studies the organization's current procedures and the information systems used to perform organizational tasks. Analysis has two subphases. The first is requirements determination. In this subphase, analysts work with users to determine what the users want from a proposed system. The requirements determination process usually involves a careful study of any current systems, manual and computerized, that might be replaced or enhanced as part of the project. In the second part of analysis, analysts study the requirements and structure them according to their

FIGURE 1-5
SDLC-based guide to this book

Analysis

The second phase of the SDLC in which system requirements are studied and structured.

Design

The third phase of the SDLC in which the description of the recommended solution is converted into logical and then physical system specifications.

Logical design

The part of the design phase of the SDLC in which all functional features of the system chosen for development in analysis are described independently of any computer platform.

Physical design

The part of the design phase of the SDLC in which the logical specifications of the system from logical design are transformed into technology-specific details from which all programming and system construction can be accomplished.

Implementation

The fourth phase of the SDLC, in which the information system is coded, tested, installed, and supported in the organization.

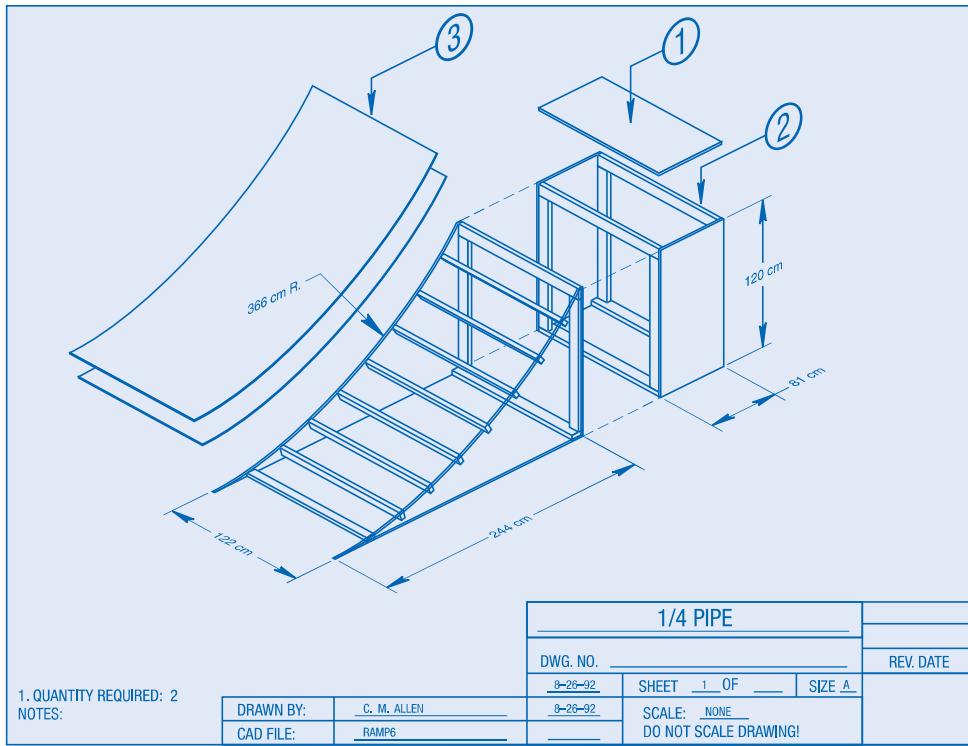
interrelationships and eliminate any redundancies. The output of the analysis phase is a description of (but not a detailed design for) the alternative solution recommended by the analysis team. Once the recommendation is accepted by those with funding authority, the analysts can begin to make plans to acquire any hardware and system software necessary to build or operate the system as proposed.

The third phase in the SDLC is **design**. During design, analysts convert the description of the recommended alternative solution into logical and then physical system specifications. The analysts must design all aspects of the system, from input and output screens to reports, databases, and computer processes. The analysts must then provide the physical specifics of the system they have designed, either as a model or as detailed documentation, to guide those who will build the new system. That part of the design process that is independent of any specific hardware or software platform is referred to as **logical design**. Theoretically, the system could be implemented on any hardware and systems software. The idea is to make sure that the system functions as intended. Logical design concentrates on the business aspects of the system and tends to be oriented to a high level of specificity.

Once the overall high-level design of the system is worked out, the analysts begin turning logical specifications into physical ones. This process is referred to as **physical design**. As part of physical design, analysts design the various parts of the system to perform the physical operations necessary to facilitate data capture, processing, and information output. This can be done in many ways, from creating a working model of the system to be implemented to writing detailed specifications describing all the different parts of the system and how they should be built. In many cases, the working model becomes the basis for the actual system to be used. During physical design, the analyst team must determine many of the physical details necessary to build the final system, from the programming language the system will be written in, to the database system that will store the data, to the hardware platform on which the system will run. Often the choices of language, database, and platform are already decided by the organization or by the client, and at this point these information technologies must be taken into account in the physical design of the system. The final product of the design phase is the physical system specifications in a form ready to be turned over to programmers and other system builders for construction. Figure 1-6 illustrates the differences between logical and physical design.

The fourth phase in the SDLC is **implementation**. The physical system specifications, whether in the form of a detailed model or as detailed written specifications, are turned over to programmers as the first part of the implementation phase. During implementation, analysts 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. Sometimes the code is generated by the same system used to build the detailed model of the system. During testing, programmers and analysts test individual programs and the entire system in order to find and correct errors. During installation, the new system becomes part of the daily activities of the organization. Application software is installed, or loaded, on existing or new hardware, and users are introduced to the new system and trained. Testing and installation should be planned for as early as the project initiation and planning phase; both testing and installation require extensive analysis in order to develop exactly the right approach.

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. 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,

**FIGURE 1-6**

Difference between logical design and physical design
(a) A skateboard ramp blueprint (logical design)

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



(b) A skateboard ramp (physical design)

installation is not always a simple process. Many well-designed systems have failed because the installation process was faulty. Even a well-designed system can fail if implementation is not well managed. Because the project team usually manages implementation, we stress implementation issues throughout this book.

The fifth and final phase in the SDLC is **maintenance**. When a system (including its training, documentation, and support) is operating in an organization, users sometimes find problems with how it works and often think of better ways to perform its functions. Also, the organization's needs with respect to the system change over time. In maintenance, programmers make the changes that users ask for and modify the system to reflect evolving business conditions. These changes are necessary to keep the system running and useful. In a sense, maintenance is not a separate phase but a repetition of the other life cycle phases required to study and implement the needed changes. One might think of maintenance as an overlay on the life cycle rather than as a separate phase. The amount of time and effort devoted to maintenance depends a great deal on the performance of the previous phases of the life cycle. There inevitably comes a time, however, when an information system is no longer performing as desired, when maintenance costs 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

Maintenance

The final phase of the SDLC, in which an information system is systematically repaired and improved.

TABLE 1-1 Products of SDLC Phases

Phase	Products, Outputs, or Deliverables
Planning	Priorities for systems and projects; an architecture for data, networks, and selection hardware, and information systems management are the result of associated systems Detailed steps, or work plan, for project Specification of system scope and planning and high-level system requirements or features Assignment of team members and other resources System justification or business case
Analysis	Description of current system and where problems or opportunities exist, with a general recommendation on how to fix, enhance, or replace current system
Design	Explanation of alternative systems and justification for chosen alternative Functional, detailed specifications of all system elements (data, processes, inputs, and outputs) Technical, detailed specifications of all system elements (programs, files, network, system software, etc.) Acquisition plan for new technology
Implementation	Code, documentation, training procedures, and support capabilities
Maintenance	New versions or releases of software with associated updates to documentation, training, and support

and starting the life cycle over again. Often the distinction between major maintenance and new development is not clear, which is another reason maintenance often resembles the life cycle itself.

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 in-text descriptions. The chapters on the SDLC phases will elaborate on the products of each phase as well as on how the products are developed.

Throughout the SDLC, the systems development project itself must be carefully planned and managed. The larger the systems project, the greater the need for project management. Several project management techniques have been developed over the past decades, and many have been made more useful through automation. Chapter 3 contains a more detailed treatment of project planning and management techniques. Next, we will discuss some of the criticisms of the SDLC and present alternatives developed to address those criticisms. First, however, we will introduce you to a specialized SDLC that focuses on security during development.

A SPECIALIZED SYSTEMS DEVELOPMENT LIFE CYCLE

Although the basic SDLC provides an overview of the systems development process, the concept of the SDLC can also be applied to very specific aspects of the process. As mentioned previously, the maintenance phase can be described in terms of the SDLC. Another example of a specialized SDLC is Microsoft's Security Development Lifecycle (SDL) (see <http://www.microsoft.com/security/sdl/default.aspx> for details). The Security Development Lifecycle is depicted in Figure 1-7. First note how the five basic phases of the development life cycle (in green) are not exactly the same as the five phases of the SDLC we will use in this book. Three of the five phases are almost identical to the phases in our SDLC. The Microsoft SDL starts with "requirements," which is similar to "analysis"; this is followed by the design phase, which is followed by implementation. Our life cycle starts with planning and ends with maintenance. Both of these phases are peculiar to systems development in an organizational context, where