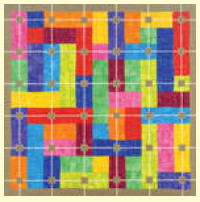
CHAPTER 1

Systems, Roles, and Development Methodologies

LEARNING OBJECTIVES

Once you have mastered the material in this chapter you will be able to:

1. **Recall the basic types of computer-based systems that a systems analyst needs to address.**
2. **Understand how users working in context with new technologies change the dynamics of a system.**
3. **Realize what the many roles of a systems analyst are.**
4. **Comprehend the fundamentals of three development methodologies: SDLC, the agile approach, and object-oriented systems analysis and design.**
5. **Understand what CASE tools are and how they help a systems analyst.**



Organizations have long recognized the importance of managing key resources such as people and raw materials. Information has now moved to its rightful place as a key resource. Decision makers now understand that information is not just a by-product of conducting business; rather, it fuels business and can be the critical factor in determining the success or failure of a business.

To maximize the usefulness of information, a business must manage it correctly, just as it manages other resources. Managers need to understand that costs are associated with the production, distribution, security, storage, and retrieval of all information. Although information is all around us, it is not free, and its strategic use for positioning a business competitively should not be taken for granted.

The ready availability of networked computers, along with access to the Internet and the Web, has created an information explosion throughout society in general and business in particular. Managing computer-generated information differs in significant ways from handling manually produced data. Usually there is a greater quantity of computer information to administer. Costs of organizing and maintaining it can increase at alarming rates, and users often treat it less skeptically than information obtained in different ways. This chapter examines the fundamentals of different kinds of information systems, the varied roles of systems analysts, and the phases in the systems development life cycle (SDLC) as they relate to Human–Computer Interaction (HCI) factors; it also introduces Computer-Aided Software Engineering (CASE) tools.

TYPES OF SYSTEMS

Information systems are developed for different purposes, depending on the needs of human users and the business. Transaction processing systems (TPS) function at the operational level of the organization; office automation systems (OAS) and knowledge work systems (KWS) support work at the knowledge level. Higher-level systems include management information systems (MIS) and decision support systems (DSS). Expert systems apply the expertise of decision makers to solve specific, structured problems. On the strategic level of management we find executive support systems (ESS). Group decision support systems (GDSS) and the more generally described computer-supported collaborative work systems (CSCWS) aid group-level decision making of a semistructured or unstructured variety.

The variety of information systems that analysts may develop is shown in Figure 1.1. Notice that the figure presents these systems from the bottom up, indicating that the operational, or lowest, level of the organization is supported by TPS, and the strategic, or highest, level of semistructured and unstructured decisions is supported by ESS, GDSS, and CSCWS at the top. This text uses the terms *management information systems, information systems* (IS), *computerized information systems,* and *computerized business information systems* interchangeably to denote computerized information systems that support the broadest range of user interactions with technologies and business activities through the information they produce in organizational contexts.

Transaction Processing Systems

Transaction processing systems (TPS) are computerized information systems that were developed to process large amounts of data for routine business transactions such as payroll and inventory. A TPS eliminates the tedium of necessary operational transactions and reduces the time once required to perform them manually, although people must still input data to computerized systems.

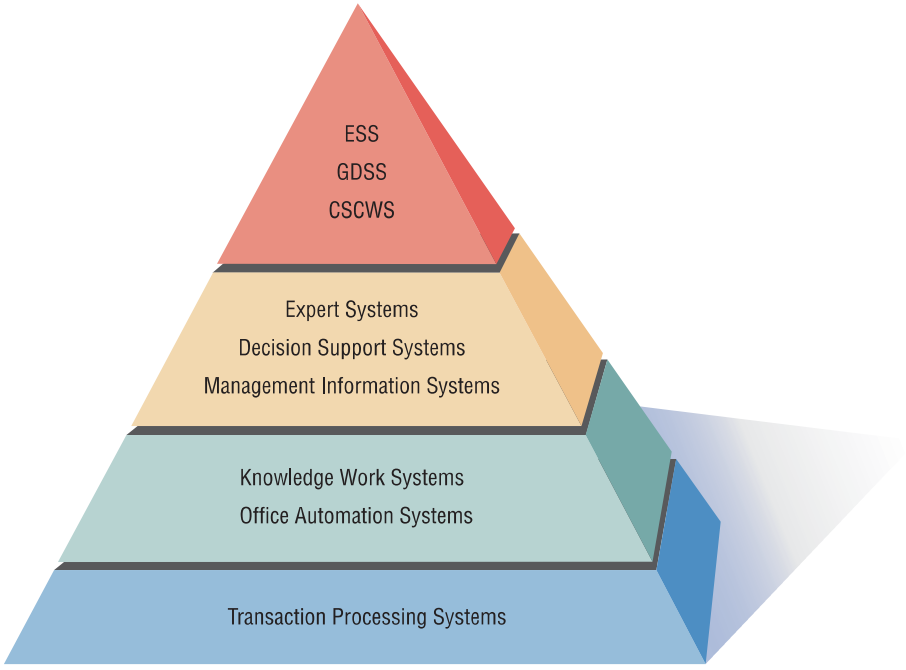
Transaction processing systems are boundary-spanning systems that permit the organization to interact with external environments. Because managers look to the data generated by the TPS for up-to-the-minute information about what is happening in their companies, it is essential to the day-to-day operations of business that these systems function smoothly and without interruption.

Office Automation Systems and Knowledge Work Systems

At the knowledge level of the organization are two classes of systems. Office automation systems (OAS) support data workers, who do not usually create new knowledge but rather analyze information to transform data or manipulate it in some way before sharing it with, or formally disseminating it throughout, the organization and, sometimes, beyond. Familiar aspects of OAS include

FIGURE 1.1

A systems analyst may be involved with any or all of these systems.



word processing, spreadsheets, desktop publishing, electronic scheduling, and communication through voice mail, email (electronic mail), and teleconferencing.

Knowledge work systems (KWS) support professional workers such as scientists, engineers, and doctors by aiding them in their efforts to create new knowledge (often in teams) and by allowing them to contribute it to their organization or to society at large.

Management Information Systems

Management information systems (MIS) do not replace transaction processing systems; rather, all MIS include transaction processing. MIS are computerized information systems that work because of the purposeful interaction between people and computers. By requiring people, software, and hardware to function in concert, management information systems support users in accomplishing a broader spectrum of organizational tasks than transaction processing systems, including decision analysis and decision making.

To access information, users of the management information system share a common database. The database stores both data and models that help the user interact with, interpret, and apply that data. Management information systems output information that is used in decision making. A management information system can also help integrate some of the computerized information functions of a business.

Decision Support Systems

A higher-level class of computerized information systems is decision support systems (DSS). DSS are similar to the traditional management information system because they both depend on a database as a source of data. A decision support system departs from the traditional management information system because it emphasizes the support of decision making in all its phases, although the actual decision is still the exclusive province of the decision maker. Decision support systems are more closely tailored to the person or group using them than is a traditional management information system. Sometimes they are discussed as systems that focus on business intelligence.

Artificial Intelligence and Expert Systems

Artificial intelligence (AI) can be considered the overarching field for expert systems. The general thrust of AI has been to develop machines that behave intelligently. Two avenues of AI research are (1) understanding natural language and (2) analyzing the ability to reason through a problem to its logical conclusion. Expert systems use the approaches of AI reasoning to solve the problems put to them by business (and other) users.

Expert systems are a very special class of information system that has been made practicable for use by business as a result of widespread availability of hardware and software such as personal computers (PCs) and expert system shells. An expert system (also called a knowledge-based system) effectively captures and uses the knowledge of a human expert or experts for solving a particular problem experienced in an organization. Notice that unlike DSS, which leave the ultimate judgment to the decision maker, an expert system selects the best solution to a problem or a specific class of problems.

The basic components of an expert system are the knowledge base, an inference engine connecting the user with the system by processing queries via languages such as structured query language (SQL), and the user interface. People called knowledge engineers capture the expertise of experts, build a computer system that includes this expert knowledge, and then implement it.

Group Decision Support Systems and Computer-Supported Collaborative Work Systems

Organizations are becoming increasingly reliant on groups or teams to make decisions together. When groups make semistructured or unstructured decisions, a group decision support system may afford a solution. Group decision support systems (GDSS), which are used in special rooms equipped in a number of different configurations, permit group members to interact with electronic support—often in the form of specialized software—and a special group facilitator. Group decision support systems are intended to bring a group together to solve a problem with the help of various supports such as polling, questionnaires, brainstorming, and scenario creation. GDSS software can be designed to minimize typical negative group behaviors such as lack of participation due to fear

of reprisal for expressing an unpopular or contested viewpoint, domination by vocal group members, and "group think" decision making. Sometimes GDSS are discussed under the more general term *computer-supported collaborative work systems* (CSCWS), which might include software support called groupware for team collaboration via networked computers. Group decision support systems can also be used in a virtual setting.

Executive Support Systems

When executives turn to the computer, they are often looking for ways to help them make decisions on the strategic level. Executive support systems (ESS) help executives organize their interactions with the external environment by providing graphics and communications technologies in accessible places such as boardrooms or personal corporate offices. Although ESS rely on the information generated by TPS and MIS, executive support systems help their users address unstructured decision problems, which are not application specific, by creating an environment that helps them think about strategic problems in an informed way. ESS extend and support the capabilities of executives, permitting them to make sense of their environments.

INTEGRATING TECHNOLOGIES FOR SYSTEMS

As users adopt new technologies, some of the systems analyst's work will be devoted to integrating traditional systems with new ones to ensure a useful context, as shown in Figure 1.2. This section describes some of the new information technologies systems analysts will be using as people work to integrate their ecommerce applications into their traditional businesses or as they begin entirely new ebusinesses.

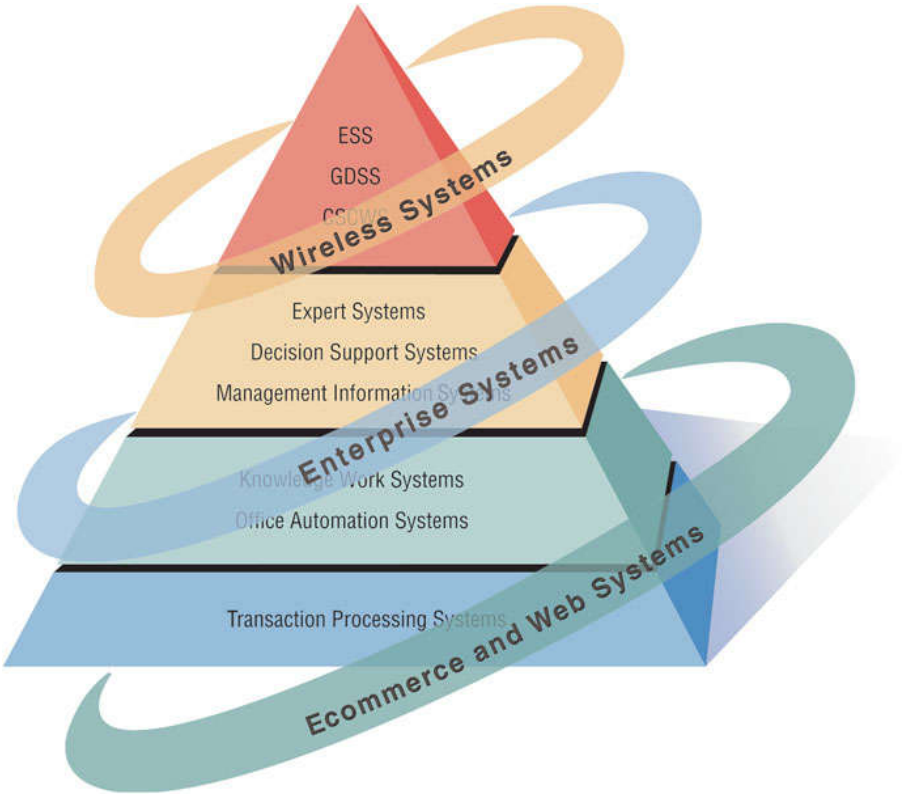
Ecommerce Applications and Web Systems

Many of the systems discussed here can be imbued with greater functionality if they are migrated to the World Wide Web or if they are originally conceived and implemented as Web-based technologies. There are many benefits to mounting or improving an application on the Web:

1. Increasing user awareness of the availability of a service, product, industry, person, or group.
2. The possibility of 24-hour access for users.

FIGURE 1.2

Systems analysts need to be aware that integrating technologies affect all types of users and systems.



3. Improving the usefulness and usability of the interface design.
4. Creating a system that can extend globally rather than remain local, thus reaching people in remote locations without worry of the time zone in which they are located.

Enterprise Systems

Many organizations envision potential benefits from the integration of many information systems existing on different management levels and within different functions. Some authors discuss integration as service-oriented architecture (SOA), which exists in layers. Enterprise systems would comprise the top layer. Enterprise systems, also called enterprise resource planning (ERP) systems, are designed to perform this integration. Instituting ERP requires enormous commitment and organizational change. Often systems analysts serve as consultants to ERP endeavors that use proprietary software. Popular ERP software includes that from SAP and Oracle. Some of these packages are targeted toward moving enterprises onto the Web. Typically, analysts as well as some users require vendor training, support, and maintenance to be able to properly design, install, maintain, update, and use a particular ERP package.

Systems for Wireless and Mobile Devices

Analysts are being asked to design a plethora of new systems and applications for adventurous users, including many for wireless and mobile devices such as the Apple iPhone, iPod, or the BlackBerry. In addition, analysts may find themselves designing standard or wireless communications networks for users that integrate voice, video, text messaging, and email into organizational intranets or industry extranets. Wireless ecommerce is referred to as mcommerce (mobile commerce).

Wireless local area networks (WLANs); wireless fidelity networks, called Wi-Fi; and personal wireless networks that bring together many types of devices under the standard called Bluetooth are all systems that you may be asked to design. In more advanced settings, analysts may be called on to design intelligent agents, software that can assist users with tasks in which the software learns users' preferences over time and then acts on those preferences. For example, in the use of pull technology, an intelligent agent would search the Web for stories of interest to the user, having observed the user's behavior patterns with information over time, and would conduct searches on the Web without continual prompting from the user.

Open Source Software

An alternative to traditional software development in which proprietary code is hidden from the users is called open source software (OSS). With OSS, the code, or computer instructions, can be studied, shared, and modified by many users and programmers. Rules of this community include the idea that any program modifications must be shared with all the people on the project.

Development of OSS has also been characterized as a philosophy rather than simply as the process of creating new software. Often those involved in OSS communities view it as a way to help societies change. Widely known open source projects include Apache for developing a Web server, the browser called Mozilla Firefox, and Linux, which is a Unix-like open source operating system.

However, it would be an oversimplification to think of OSS as a monolithic movement, and it does little to reveal what type of users or user analysts are developing OSS projects and on what basis. To help us understand the open source movement, researchers have recently categorized open source communities into four community types—ad hoc, standardized, organized, and commercial—along six different dimensions—general structure, environment, goals, methods, user community, and licensing. Some researchers argue that OSS is at a crossroads and that the commercial and community OSS groups need to understand where they converge and where the potential for conflict exists.

Open source development is useful for many applications running on diverse technologies, including handheld devices and communication equipment. Its use may encourage progress in creating standards for devices to communicate more easily. Widespread use of OSS may alleviate some of the severe shortages of programmers by placing programming tools in the hands of students in developing countries sooner than if they were limited to using proprietary packages, and it may lead to solving large problems through intense and extensive collaboration.

NEED FOR SYSTEMS ANALYSIS AND DESIGN

Systems analysis and design, as performed by systems analysts, seeks to understand what humans need to analyze data input or data flow systematically, process or transform data, store data, and output information in the context of a particular organization or enterprise. By doing thorough analysis, analysts seek to identify and solve the right problems. Furthermore, systems analysis and design is used to analyze, design, and implement improvements in the support of users and the functioning of businesses that can be accomplished through the use of computerized information systems.

Installing a system without proper planning leads to great user dissatisfaction and frequently causes the system to fall into disuse. Systems analysis and design lends structure to the analysis and design of information systems, a costly endeavor that might otherwise have been done in a haphazard way. It can be thought of as a series of processes systematically undertaken to improve a business through the use of computerized information systems. Systems analysis and design involves working with current and eventual users of information systems to support them in working with technologies in an organizational setting.

User involvement throughout the systems project is critical to the successful development of computerized information systems. Systems analysts, whose roles in the organization are discussed next, are the other essential component in developing useful information systems.

Users are moving to the forefront as software development teams become more international in their composition. This means that there is more emphasis on working with software users; on performing analysis of their business, problems, and objectives; and on communicating the analysis and design of the planned system to all involved.

New technologies also are driving the need for systems analysis. Ajax (Asynchronous JavaScript and XML) is not a new programming language, but a technique that uses existing languages to make Web pages function more like a traditional desktop application program. Building and redesigning Web pages that utilize Ajax technologies will be a task facing analysts. New programming languages, such as the open source Web framework, *Ruby on Rails*, which is a combination programming language and code generator for creating Web applications, will require more analysis.

ROLES OF THE SYSTEMS ANALYST

The systems analyst systematically assesses how users interact with technology and how businesses function by examining the inputting and processing of data and the outputting of information with the intent of improving organizational processes. Many improvements involve better support of users' work tasks and business functions through the use of computerized information systems. This definition emphasizes a systematic, methodical approach to analyzing—and potentially improving—what is occurring in the specific context experienced by users and created by a business.

Our definition of a systems analyst is necessarily broad. The analyst must be able to work with people of all descriptions and be experienced in working with computers. The analyst plays many roles, sometimes balancing several at the same time. The three primary roles of the systems analyst are consultant, supporting expert, and agent of change.

Systems Analyst as Consultant

The systems analyst frequently acts as a systems consultant to humans and their businesses and, thus, may be hired specifically to address information systems issues within a business. Such hiring can be an advantage because outside consultants can bring with them a fresh perspective that other people in an organization do not possess. It also means that outside analysts are at a disadvantage because an outsider can never know the true organizational culture. As an outside consultant, you will rely heavily on the systematic methods discussed throughout this text to analyze and design appropriate information systems for users working in a particular business. In addition, you will rely on information systems users to help you understand the organizational culture from others' viewpoints.

Systems Analyst as Supporting Expert

Another role that you may be required to play is that of supporting expert within a business for which you are regularly employed in some systems capacity. In this role the analyst draws on professional expertise concerning computer hardware and software and their uses in the business.

CONSULTING OPPORTUNITY 1.1

Healthy Hiring: Ecommerce Help Wanted

"You'll be happy to know that we made a strong case to management that we should hire a new systems analyst to specialize in ecommerce development," says Al Falfa, a systems analyst for the multioutlet international chain of Marathon Vitamin Shops. He is meeting with his large team of systems analysts to decide on the qualifications that their new team member should possess. Al continues, saying, "In fact, they were so excited by the possibility of our team helping to move Marathon into an ecommerce strategy that they've said we should start our search now and not wait until the fall."

Ginger Rute, another analyst, agrees, saying, "The demand for Web site developers is still outstripping the supply. We should move quickly. I think our new person should be knowledgeable in system modeling, JavaScript, C++, Rational Rose, and familiar with Ajax, just to name a few."

Al looks surprised at Ginger's long list of skills but then replies, "Well, that's certainly one way we could go. But I would also like to see a person with some business savvy. Most of the people coming out of school will have solid programming skills, but they should know about accounting, inventory, and distribution of goods and services, too."

The newest member of the systems analysis group, Vita Ming, finally breaks into the discussion. She says, "One of the reasons I chose to come to work with all of you was that I thought we all got along quite well together. Because I had some other opportunities, I looked very carefully at what the atmosphere was here. From what I've seen, we're a friendly group. Let's be sure to hire someone who has a good personality and who fits in well with us."

Al concurs, continuing, "Vita's right. The new person should be able to communicate well with us, and with business clients, too. We are always communicating in some way, through formal presentations, drawing diagrams, or interviewing users. If they understand decision making, it will make their job easier, too. Also, Marathon is interested in integrating ecommerce into the entire business. We need someone who at least grasps the strategic importance of the Web. Page design is such a small part of it."

Ginger interjects again with a healthy dose of practicality, saying, "Leave that to management. I still say the new person should be a good programmer." Then she ponders aloud, "I wonder how important UML will be?"

After listening patiently to everyone's wish list, one of the senior analysts, Cal Siem, speaks up, joking, "We'd better see if Superman is available!"

As the group shares a laugh, Al sees an opportunity to try for some consensus, saying, "We've had a chance to hear a number of different qualifications. Let's each take a moment and make a list of the qualifications we personally think are essential for the new ecommerce development person to possess. We'll share them and continue discussing until we can describe the person in enough detail to turn a description over to the human resources group for processing."

What qualifications should the systems analysis team be looking for when hiring their new ecommerce development team member? Is it more important to know specific languages or to have an aptitude for picking up languages and software packages quickly? How important is it that the person being hired has some basic business understanding? Should all team members possess identical competencies and skills? What personality or character traits are desirable in a systems analyst who will be working in ecommerce development?

This work is often not a full-blown systems project, but rather it entails a small modification or decision affecting a single department.

As the supporting expert, you are not managing the project; you are merely serving as a resource for those who are. If you are a systems analyst employed by a manufacturing or service organization, many of your daily activities may be encompassed by this role.

Systems Analyst as Agent of Change

The most comprehensive and responsible role that the systems analyst takes on is that of an agent of change, whether internal or external to the business. As an analyst, you are an agent of change whenever you perform any of the activities in the systems development life cycle (discussed in the next section) and are present and interacting with users and the business for an extended period (from two weeks to more than a year). An agent of change can be defined as a person who serves as a catalyst for change, develops a plan for change, and works with others in facilitating that change.

Your presence in the business changes it. As a systems analyst, you must recognize this fact and use it as a starting point for your analysis. Hence, you must interact with users and management (if they are not one and the same) from the very beginning of your project. Without their help you cannot understand what they need to support their work in the organization, and real change cannot take place.

If change (that is, improvements to the business that can be realized through information systems) seems warranted after analysis, the next step is to develop a plan for change along with the people who must enact the change. Once a consensus is reached on the change that is to be made, you must constantly interact with those who are changing.

As a systems analyst acting as an agent of change, you advocate a particular avenue of change involving the use of information systems. You also teach users the process of change, because changes in the information system do not occur independently; rather, they cause changes in the rest of the organization as well.

Qualities of the Systems Analyst

From the foregoing descriptions of the roles the systems analyst plays, it is easy to see that the successful systems analyst must possess a wide range of qualities. Many different kinds of people are systems analysts, so any description is destined to fall short in some way. There are some qualities, however, that most systems analysts seem to display.

Above all, the analyst is a problem solver. He or she is a person who views the analysis of problems as a challenge and who enjoys devising workable solutions. When necessary, the analyst must be able to systematically tackle the situation at hand through skillful application of tools, techniques, and experience. The analyst must also be a communicator capable of relating meaningfully to other people over extended periods of time. Systems analysts need to be able to understand humans' needs in interacting with technology, and they need enough computer experience to program, to understand the capabilities of computers, to glean information requirements from users, and to communicate what is needed to programmers. They also need to possess strong personal and professional ethics to help them shape their client relationships.

The systems analyst must be a self-disciplined, self-motivated individual who is able to manage and coordinate other people, as well as innumerable project resources. Systems analysis is a demanding career, but, in compensation, an ever-changing and always challenging one.

THE SYSTEMS DEVELOPMENT LIFE CYCLE

Throughout this chapter we have referred to the systematic approach analysts take to the analysis and design of information systems. Much of this is embodied in what is called the systems development life cycle (SDLC). The SDLC is a phased approach to analysis and design that holds that systems are best developed through the use of a specific cycle of analyst and user activities.

Analysts disagree on exactly how many phases there are in the SDLC, but they generally laud its organized approach. Here we have divided the cycle into seven phases, as shown in Figure 1.3. Although each phase is presented discretely, it is never accomplished as a separate step. Instead, several activities can occur simultaneously, and activities may be repeated.

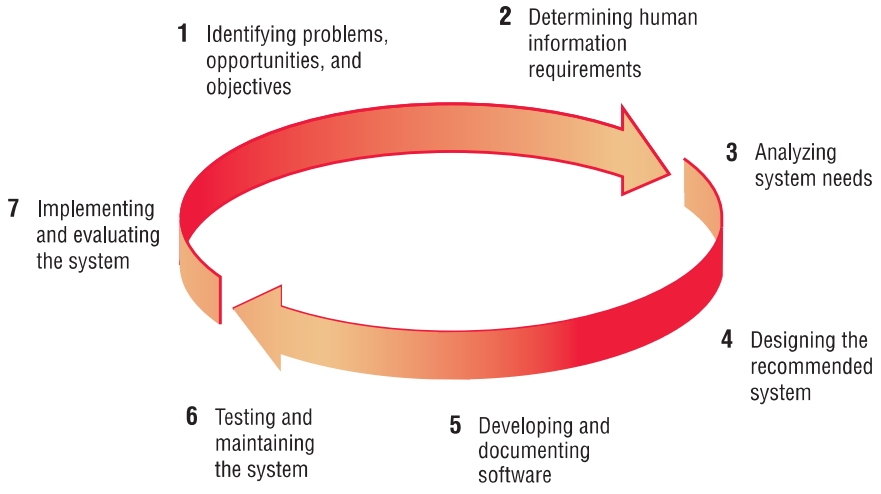


FIGURE 1.3
The seven phases of the systems development life cycle (SDLC).

Incorporating Human–Computer Interaction Considerations

In recent years, the study of human–computer interaction (HCI) has become increasingly important for systems analysts. Although the definition is still evolving, researchers characterize HCI as the "aspect of a computer that enables communications and interactions between humans and the computer. It is the layer of the computer that is between humans and the computer" (Zhang, Carey, Te'eni, & Tremaine, 2005, p. 518). Analysts using an HCI approach are emphasizing people rather than the work to be done or the IT that is involved. Their approach to a problem is multifaceted, looking at the "human ergonomic, cognitive, affective, and behavioral factors involved in user tasks, problem solving processes and interaction context" (Zhang, Carey, Te'eni, & Tremaine, 2005, p. 518). Human computer interaction moves away from focusing first on organizational and system needs and instead concentrates on human needs. Analysts adopting HCI principles examine a variety of user needs in the context of humans interacting with information technology to complete tasks and solve problems. These include taking into account physical or ergonomic factors; usability factors that are often labeled cognitive matters; the pleasing, aesthetic, and enjoyable aspects of using the system; and behavioral aspects that center on the usefulness of the system.

Another way to think about HCI is to think of it as a human-centered approach that puts people ahead of organizational structure or culture when creating new systems. When analysts employ HCI as a lens to filter the world, their work will possess a different quality than the work of those who do not possess this perspective.

Your career can benefit from a strong grounding in HCI. The demand for analysts who are capable of incorporating HCI into the systems development process keeps rising, as companies increasingly realize that the quality of systems and the quality of work life can both be improved by taking a human-centered approach at the outset of a project.

The application of human–computer interaction principles tries to uncover and address the frustrations that users voice over their use of information technology. These concerns include a suspicion that systems analysts misunderstand the work being done, the tasks involved, and how they can best be supported; a feeling of helplessness or lack of control when working with the system; intentional breaches of privacy; trouble navigating through system screens and menus; and a general mismatch between the system designed and the way users themselves think of their work processes.

Misjudgments and errors in design that cause users to neglect new systems or that cause systems to fall into disuse soon after their implementation can be eradicated or minimized when systems analysts adopt an HCI approach.

Researchers in HCI see advantages to the inclusion of HCI in every phase of the SDLC. This is a worthwhile approach, and we will try to mirror this by bringing human concerns explicitly into each phase of the SDLC. As a person who is learning systems analysis, you can also bring a fresh eye to the SDLC to identify opportunities for designers to address HCI concerns and ways for users to become more central to each phase of the SDLC. Chapter 14 is devoted to examining the role of the systems analyst in designing human-centered systems and interfaces from an HCI perspective.

Identifying Problems, Opportunities, and Objectives

In this first phase of the systems development life cycle, the analyst is concerned with correctly identifying problems, opportunities, and objectives. This stage is critical to the success of the rest of the project, because no one wants to waste subsequent time addressing the wrong problem.

The first phase requires that the analyst look honestly at what is occurring in a business. Then, together with other organizational members, the analyst pinpoints problems. Often others will bring up these problems, and they are the reason the analyst was initially called in. Opportunities are situations that the analyst believes can be improved through the use of computerized information systems. Seizing opportunities may allow the business to gain a competitive edge or set an industry standard.

Identifying objectives is also an important component of the first phase. The analyst must first discover what the business is trying to do. Then the analyst will be able to see whether some aspect of information systems applications can help the business reach its objectives by addressing specific problems or opportunities.

The people involved in the first phase are the users, analysts, and systems managers coordinating the project. Activities in this phase consist of interviewing user management, summarizing the

knowledge obtained, estimating the scope of the project, and documenting the results. The output of this phase is a feasibility report containing a problem definition and summarizing the objectives. Management must then make a decision on whether to proceed with the proposed project. If the user group does not have sufficient funds in its budget or wishes to tackle unrelated problems, or if the problems do not require a computer system, a different solution may be recommended, and the systems project does not proceed any further.

Determining Human Information Requirements

The next phase the analyst enters is that of determining the human needs of the users involved, using a variety of tools to understand how users interact in the work context with their current information systems. The analyst will use interactive methods such as interviewing, sampling and investigating hard data, and questionnaires, along with unobtrusive methods, such as observing decision makers' behavior and their office environments, and all-encompassing methods, such as prototyping.

The analyst will use these methods to pose and answer many questions concerning human-computer interaction (HCI), including questions such as, "What are the users' physical strengths and limitations?" In other words, "What needs to be done to make the system audible, legible, and safe?" "How can the new system be designed to be easy to use, learn, and remember?" "How can the system be made pleasing or even fun to use?" "How can the system support a user's individual work tasks and make them more productive in new ways?"

In the information requirements phase of the SDLC, the analyst is striving to understand what information users need to perform their jobs. At this point the analyst is examining how to make the system useful to the people involved. How can the system better support individual tasks that need doing? What new tasks are enabled by the new system that users were unable to do without it? How can the new system be created to extend a user's capabilities beyond what the old system provided? How can the analyst create a system that is rewarding for workers to use?

The people involved in this phase are the analysts and users, typically operations managers and operations workers. The systems analyst needs to know the details of current system functions: the who (the people who are involved), what (the business activity), where (the environment in which the work takes place), when (the timing), and how (how the current procedures are performed) of the business under study. The analyst must then ask why the business uses the current system. There may be good reasons for doing business using the current methods, and these should be considered when designing any new system.

Agile development is an object-oriented approach (OOA) to systems development that includes a method of development (including generating information requirements) as well as software tools. In this text it is paired with prototyping in Chapter 6. (There is more about object-oriented approaches in Chapter 10.)

If the reason for current operations is that "it's always been done that way," however, the analyst may wish to improve on the procedures. At the completion of this phase, the analyst should understand how users accomplish their work when interacting with a computer and begin to know how to make the new system more useful and usable. The analyst should also know how the business functions and have complete information on the people, goals, data, and procedures involved.

Analyzing System Needs

The next phase that the systems analyst undertakes involves analyzing system needs. Again, special tools and techniques help the analyst make requirement determinations. Tools such as data flow diagrams (DFD) to chart the input, processes, and output of the business's functions, or activity diagrams or sequence diagrams to show the sequence of events, illustrate systems in a structured, graphical form. From data flow, sequence, or other diagrams, a data dictionary is developed that lists all the data items used in the system, as well as their specifications.

During this phase the systems analyst also analyzes the structured decisions made. Structured decisions are those for which the conditions, condition alternatives, actions, and action rules can be determined. There are three major methods for analysis of structured decisions: structured English, decision tables, and decision trees.

At this point in the SDLC, the systems analyst prepares a systems proposal that summarizes what has been found out about the users, usability, and usefulness of current systems; provides cost-benefit analyses of alternatives; and makes recommendations on what (if anything) should be done. If one of the recommendations is acceptable to management, the analyst proceeds along

that course. Each systems problem is unique, and there is never just one correct solution. The manner in which a recommendation or solution is formulated depends on the individual qualities and professional training of each analyst and the analyst's interaction with users in the context of their work environment.

Designing the Recommended System

In the design phase of the SDLC, the systems analyst uses the information collected earlier to accomplish the logical design of the information system. The analyst designs procedures for users to help them accurately enter data so that data going into the information system are correct. In addition, the analyst provides for users to complete effective input to the information system by using techniques of good form and Web page or screen design.

Part of the logical design of the information system is devising the HCI. The interface connects the user with the system and is thus extremely important. The user interface is designed with the help of users to make sure that the system is audible, legible, and safe, as well as attractive and enjoyable to use. Examples of physical user interfaces include a keyboard (to type in questions and answers), onscreen menus (to elicit user commands), and a variety of graphical user interfaces (GUIs) that use a mouse or touch screen.

The design phase also includes designing databases that will store much of the data needed by decision makers in the organization. Users benefit from a well-organized database that is logical to them and corresponds to the way they view their work. In this phase the analyst also works with users to design output (either onscreen or printed) that meets their information needs.

Finally, the analyst must design controls and backup procedures to protect the system and the data, and to produce program specification packets for programmers. Each packet should contain input and output layouts, file specifications, and processing details; it may also include decision trees or tables, UML or data flow diagrams, and the names and functions of any prewritten code that is either written in-house or using code or other class libraries.

Developing and Documenting Software

In the fifth phase of the SDLC, the analyst works with programmers to develop any original software that is needed. During this phase the analyst works with users to develop effective documentation for software, including procedure manuals, online help, and Web sites featuring Frequently Asked Questions (FAQs), on Read Me files shipped with new software. Because users are involved from the beginning, phase documentation should address the questions they have raised and solved jointly with the analyst. Documentation tells users how to use software and what to do if software problems occur.

Programmers have a key role in this phase because they design, code, and remove syntactical errors from computer programs. To ensure quality, a programmer may conduct either a design or a code walkthrough, explaining complex portions of the program to a team of other programmers.

Testing and Maintaining the System

Before the information system can be used, it must be tested. It is much less costly to catch problems before the system is signed over to users. Some of the testing is completed by programmers alone, some of it by systems analysts in conjunction with programmers. A series of tests to pinpoint problems is run first with sample data and eventually with actual data from the current system. Often test plans are created early in the SDLC and are refined as the project progresses.

Maintenance of the system and its documentation begins in this phase and is carried out routinely throughout the life of the information system. Much of the programmer's routine work consists of maintenance, and businesses spend a great deal of money on maintenance. Some maintenance, such as program updates, can be done automatically via a vendor site on the Web. Many of the systematic procedures the analyst employs throughout the SDLC can help ensure that maintenance is kept to a minimum.

Implementing and Evaluating the System

In this last phase of systems development, the analyst helps implement the information system. This phase involves training users to handle the system. Vendors do some training, but oversight of training is the responsibility of the systems analyst. In addition, the analyst needs to plan for a smooth conversion from the old system to the new one. This process includes converting files from old formats to new ones, or building a database, installing equipment, and bringing the new system into production.

MAC APPEAL

At home and in our visits to university campuses and businesses around the world, we've noticed that students and organizations are increasingly showing an interest in Macs. Therefore, we thought it would add a little bit of interest to show some Mac options that a systems designer has. At the time we're writing this book, about one out of seven personal computers purchased in the United States is a Mac. Macs are quality Intel-based machines that run under a competent operating system and can also run Windows, so in effect everything that can be done on a PC can be done on a Mac. One way to run Windows is to boot directly into Windows (once it's installed); another is to use virtualization using software such as VM Fusion, which is shown in Figure 1.MAC.

Adopters of Macs have cited many reasons for using Macs including better security built into the Mac operating system, intelligent backup using the built-in time machine, the multitude of applications already included, the reliability of setup and networking, and the ability to sync Macs with other Macs and iPhones. The most compelling reason, we think, is the design itself.

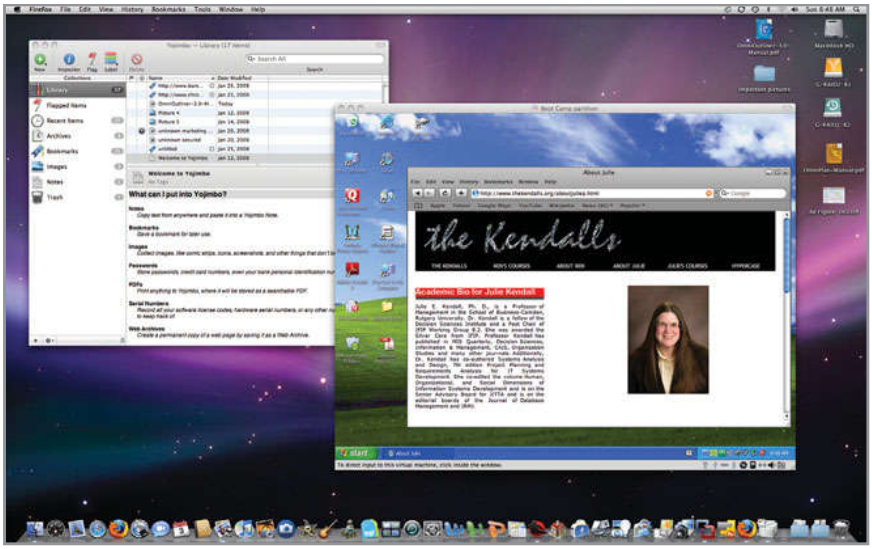


FIGURE 1.MAC

Running Windows on a Mac using Virtualization Software called VM Fusion.

Evaluation is included as part of this final phase of the SDLC mostly for the sake of discussion. Actually, evaluation takes place during every phase. A key criterion that must be satisfied is whether the intended users are indeed using the system.

It should be noted that systems work is often cyclical. When an analyst finishes one phase of systems development and proceeds to the next, the discovery of a problem may force the analyst to return to the previous phase and modify the work done there.

The Impact of Maintenance

After the system is installed, it must be maintained, meaning that the computer programs must be modified and kept up to date. Figure 1.4 illustrates the average amount of time spent on maintenance at a typical MIS installation. Estimates of the time spent by departments on maintenance have ranged from 48 to 60 percent of the total time spent developing systems. Very little time remains for new systems development. As the number of programs written increases, so does the amount of maintenance they require.

Maintenance is performed for two reasons. The first of these is to correct software errors. No matter how thoroughly the system is tested, bugs or errors creep into computer programs. Bugs

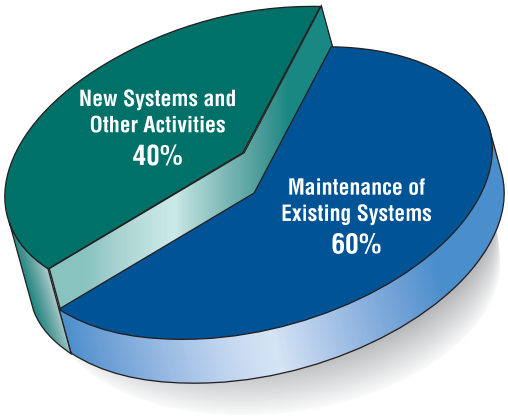


FIGURE 1.4

Some researchers estimate that the amount of time spent on system maintenance may be as much as 60 percent of the total time spent on systems projects.

in commercial PC software are often documented as "known anomalies," and are corrected when new versions of the software are released or in an interim release. In custom software (also called bespoke software), bugs must be corrected as they are detected.

The other reason for performing system maintenance is to enhance the software's capabilities in response to changing organizational needs, generally involving one of the following three situations:

1. Users often request additional features after they become familiar with the computer system and its capabilities.
2. The business changes over time.
3. Hardware and software are changing at an accelerated pace.

Figure 1.5 illustrates the amount of resources—usually time and money—spent on systems development and maintenance. The area under the curve represents the total dollar amount spent. You can see that over time the total cost of maintenance is likely to exceed that of systems development. At a certain point it becomes more feasible to perform a new systems study, because the cost of continued maintenance is clearly greater than that of creating an entirely new information system.

In summary, maintenance is an ongoing process over the life cycle of an information system. After the information system is installed, maintenance usually takes the form of correcting previously undetected program errors. Once these are corrected, the system approaches a steady state, providing dependable service to its users. Maintenance during this period may consist of removing a few previously undetected bugs and updating the system with a few minor enhancements. As time goes on and the business and technology change, however, the maintenance effort increases dramatically.

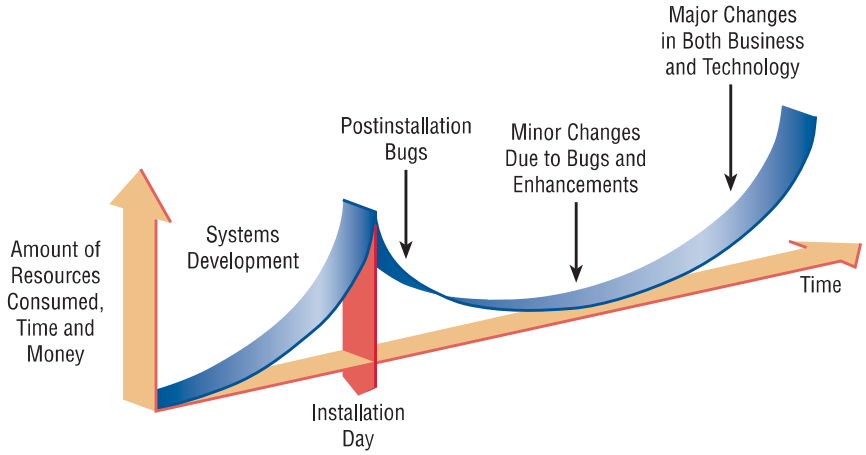


FIGURE 1.5

Resource consumption over the system life.

USING CASE TOOLS

Analysts who adopt the SDLC approach often benefit from productivity tools, called Computer-Aided Software Engineering (CASE) tools, that have been created explicitly to improve their routine work through the use of automated support. Analysts rely on CASE tools to increase productivity, communicate more effectively with users, and integrate the work that they do on the system from the beginning to the end of the life cycle.

Visible Analyst (VA) is one example of a CASE tool that enables systems analysts to do graphical planning, analysis, and design in order to build complex client/server applications and databases. Visible Analyst and another software product called Microsoft Visio allow users to draw and modify diagrams easily.

Analysts and users alike report that CASE tools afford them a means of communication about the system during its conceptualization. Through the use of automated support featuring onscreen output, clients can readily see how data flows and other system concepts are depicted, and they can then request corrections or changes that would have taken too much time with older tools.

Some analysts distinguish between upper and lower CASE tools. An upper CASE tool allows the analyst to create and modify the system design. All the information about the project is stored in an encyclopedia called the CASE repository, a large collection of records, elements, diagrams, screens, reports, and other information (see Figure 1.6). Analysis reports may be produced using the repository information to show where the design is incomplete or contains errors. Upper CASE tools can also help support the modeling of an organization's functional requirements, assist analysts and users in drawing the boundaries for a given project, and help them visualize how the project meshes with other parts of the organization.

Lower CASE tools are used to generate computer source code, eliminating the need for programming the system. Code generation has several advantages: (1) the system can be produced more quickly than by writing computer programs; (2) the amount of time spent on maintenance decreases with code generation; (3) code can be generated in more than one computer language, so it is easier to migrate systems from one platform to another; (4) code generation provides a cost-effective way of tailoring systems purchased from third-party vendors to the needs of the organization; and (5) generated code is free of computer program errors.

THE AGILE APPROACH

Although this text tends to focus on SDLC, the most widely used approach in practice, at times the analyst will recognize that the organization could benefit from an alternative approach. Perhaps a systems project using a structured approach has recently failed, or perhaps the organizational subcultures, composed of several different user groups, seem more in step with an alternative method. We cannot do justice to these methods in a small space; each deserves and has inspired its own books and research. By mentioning these approaches here, however, we hope to help you become aware that under certain circumstances, your organization may want to consider an alternative or supplement to structured analysis and design and to the SDLC.

The agile approach is a software development approach based on values, principles, and core practices. The four values are communication, simplicity, feedback, and courage. We recommend that systems analysts adopt these values in all projects they undertake, not just when adopting the agile approach.

In order to finish a project, adjustments often need to be made in project management. In Chapter 6 we will see that agile methods can ensure successful completion of a project by adjusting the important resources of time, cost, quality, and scope. When these four control variables are properly included in the planning, there is a state of balance between the resources and the activities needed to complete the project.

Taking development practices to the extreme is most noticeable when one pursues practices that are unique to agile development. In Chapter 6 we discuss four core agile practices: short releases, the 40-hour workweek, hosting an onsite customer, and using pair programming. At first glance these practices appear extreme, but as you will see, we can learn some important lessons from incorporating many of the values and practices of the agile approach into systems analysis and design projects.

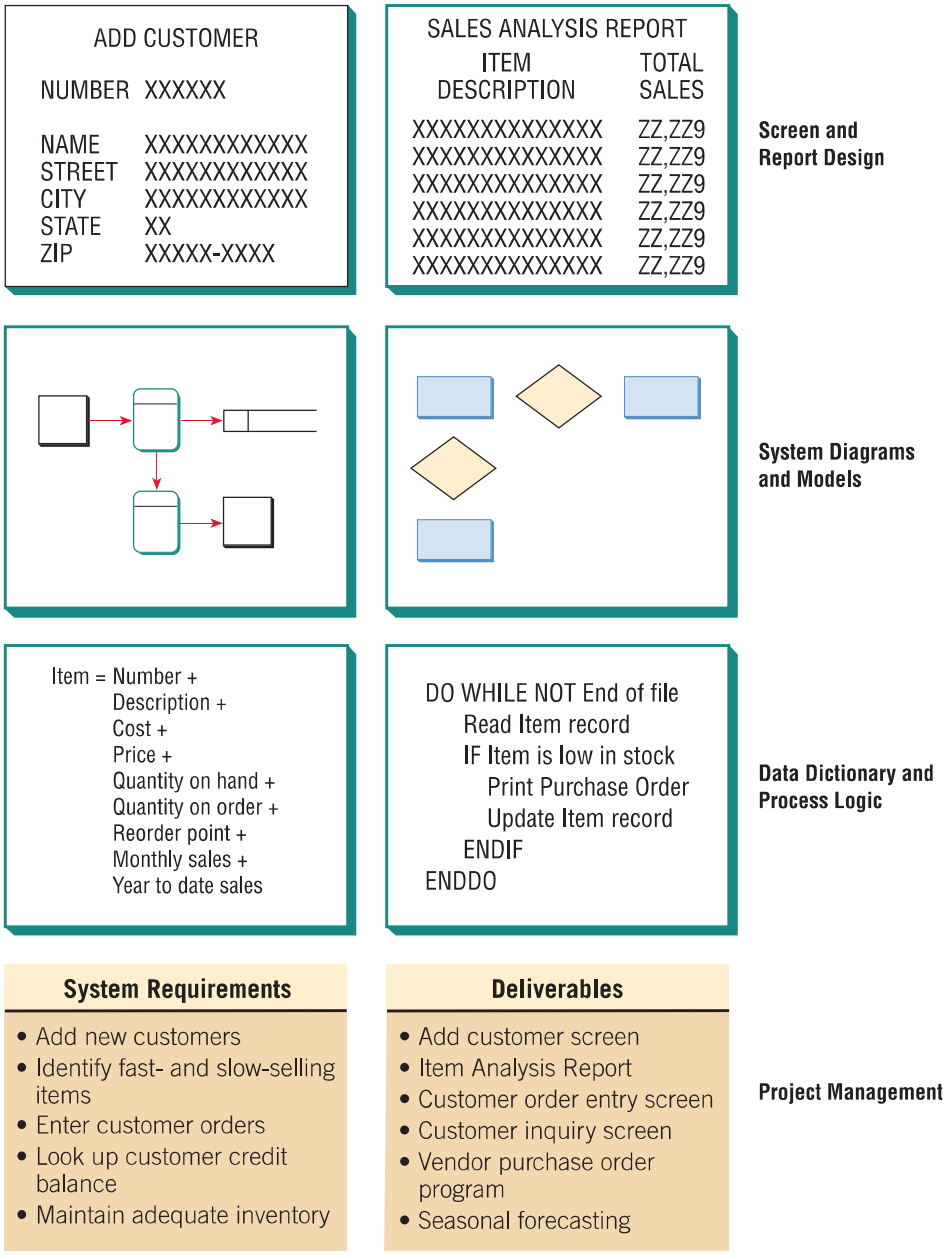


FIGURE 1.6
The repository concept.

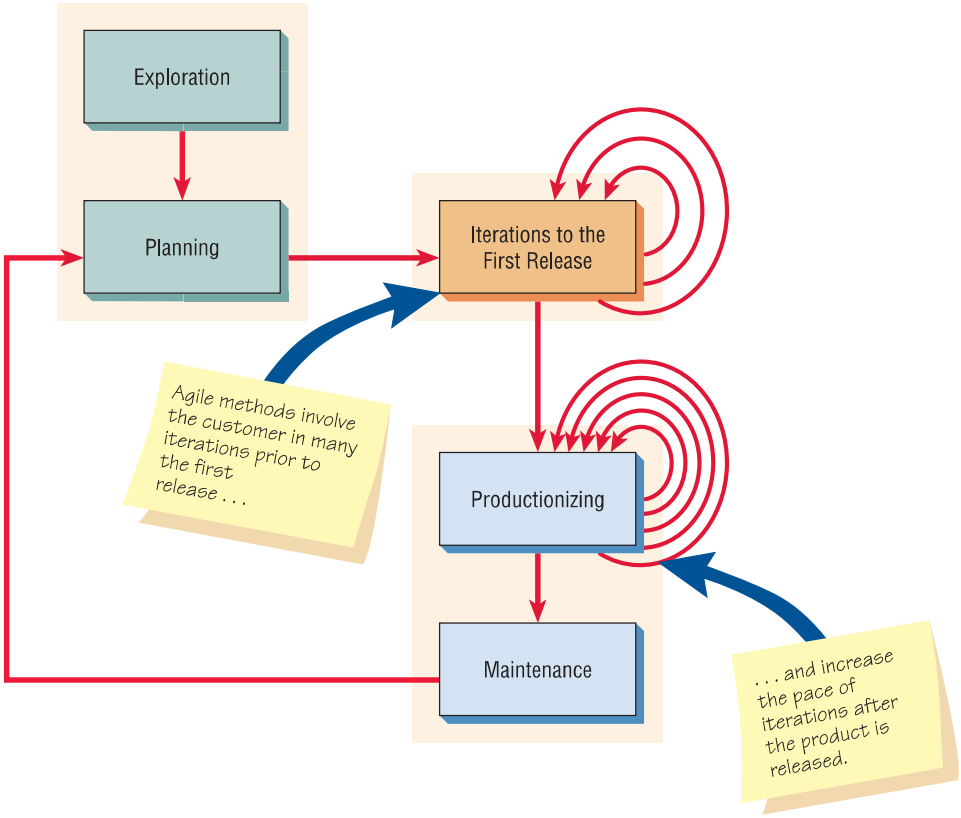
Developmental Process for an Agile Project

There are activities and behaviors that shape the way development team members and customers act during the development of an agile project. Two words that characterize a project done with an agile approach are interactive and incremental. By examining Figure 1.7 we can see that there are five distinct stages: exploration, planning, iterations to the first release, productionizing, and maintenance. Notice that the three red arrows that loop back into the "Iterations" box symbolize incremental changes created through repeated testing and feedback that eventually lead to a stable but evolving system. Also note that there are multiple looping arrows that feed back into the productionizing phase. These symbolize that the pace of iterations is increased after a product is released. The red arrow is shown leaving the maintenance stage and returning to the planning stage, so that there is a continuous feedback loop involving customers and the development team as they agree to alter the evolving system.

EXPLORATION. During exploration, you will explore your environment, asserting your conviction that the problem can and should be approached with agile development, assemble the team, and assess team member skills. This stage will take anywhere from a few weeks (if you already know

FIGURE 1.7

The five stages of the agile modeling development process show that frequent iterations are essential to successful system development.



your team members and technology) to a few months (if everything is new). You also will be actively examining potential technologies needed to build the new system. During this stage you should practice estimating the time needed for a variety of tasks. In exploration, customers also are experimenting with writing user stories. The point is to get the customer to refine a story enough so that you can competently estimate the amount of time it will take to build the solution into the system you are planning. This stage is all about adopting a playful and curious attitude toward the work environment, its problems, technologies, and people.

PLANNING. The next stage of the agile development process is called planning. In contrast to the first stage, planning may only take a few days to accomplish. In this stage you and your customers agree on a date anywhere from two months to half a year from the current date to deliver solutions to their most pressing business problems (you will be addressing the smallest, most valuable set of stories). If your exploration activities were sufficient, this stage should be very short.

The entire agile planning process has been characterized using the idea of a *planning game* as devised by Beck. The planning game spells out rules that can help formulate the agile development team's relationship with their business customers. Although the rules form an idea of how you want each party to act during development, they are not meant as a replacement for a relationship. They are a basis for building and maintaining a relationship.

So, we use the metaphor of a game. To that end we talk in terms of the goal of the game, the strategy to pursue, the pieces to move, and the players involved. The goal of the game is to maximize the value of the system produced by the agile team. In order to figure the value, you have to deduct costs of development, and the time, expense, and uncertainty taken on so that the development project could go forward.

The strategy pursued by the agile development team is always one of limiting uncertainty (downplaying risk). To do that they design the simplest solution possible, put the system into production as soon as possible, get feedback from the business customer about what's working, and adapt their design from there.

Story cards become the pieces in the planning game that briefly describe the task, provide notes, and provide an area for task tracking.

There are two main players in the planning game: the development team and the business customer. Deciding which business group in particular will be the business customer is not always

easy, because the agile process is an unusually demanding role for the customer to play. Customers decide what the development team should tackle first. Their decisions will set priorities and check functionalities throughout the process.

ITERATIONS TO THE FIRST RELEASE. The third stage in the agile development process is composed of iterations to the first release. Typically these are iterations (cycles of testing, feedback, and change) of about three weeks in duration. You will be pushing yourself to sketch out the entire architecture of the system, even though it is just in outline or skeletal form. One goal is to run customer-written functional tests at the end of each iteration. During the iterations stage you should also question whether the schedule needs to be altered or whether you are tackling too many stories. Make small rituals out of each successful iteration, involving customers as well as developers. Always celebrate your progress, even if it is small, because this is part of the culture of motivating everyone to work extremely hard on the project.

PRODUCTIONIZING. Several activities occur during this phase. In this phase the feedback cycle speeds up so that rather than receiving feedback for an iteration every three weeks, software revisions are being turned around in one week. You may institute daily briefings so everyone knows what everyone else is doing. The product is released in this phase, but may be improved by adding other features. Getting a system into production is an exciting event. Make time to celebrate with your teammates and mark the occasion. One of the watchwords of the agile approach, with which we heartily agree, is that it is supposed to be fun to develop systems!

MAINTENANCE. Once the system has been released, it needs to be kept running smoothly. New features may be added, riskier customer suggestions may be considered, and team members may be rotated on or off the team. The attitude you take at this point in the developmental process is more conservative than at any other time. You are now in a "keeper of the flame" mode rather than the playful one you experienced during exploration.

OBJECT-ORIENTED SYSTEMS ANALYSIS AND DESIGN

Object-oriented (O-O) analysis and design is an approach that is intended to facilitate the development of systems that must change rapidly in response to dynamic business environments. Chapter 10 helps you understand what object-oriented systems analysis and design is, how it differs from the structured approach of the SDLC, and when it may be appropriate to use an object-oriented approach.

Object-oriented techniques are thought to work well in situations in which complicated information systems are undergoing continuous maintenance, adaptation, and redesign. Object-oriented approaches use the industry standard for modeling object-oriented systems, called the unified modeling language (UML), to break down a system into a use case model.

Object-oriented programming differs from traditional procedural programming by examining objects that are part of a system. Each object is a computer representation of some actual thing or event. Objects may be customers, items, orders, and so on. Objects are represented by and grouped into classes that are optimal for reuse and maintainability. A class defines the set of shared attributes and behaviors found in each object in the class.

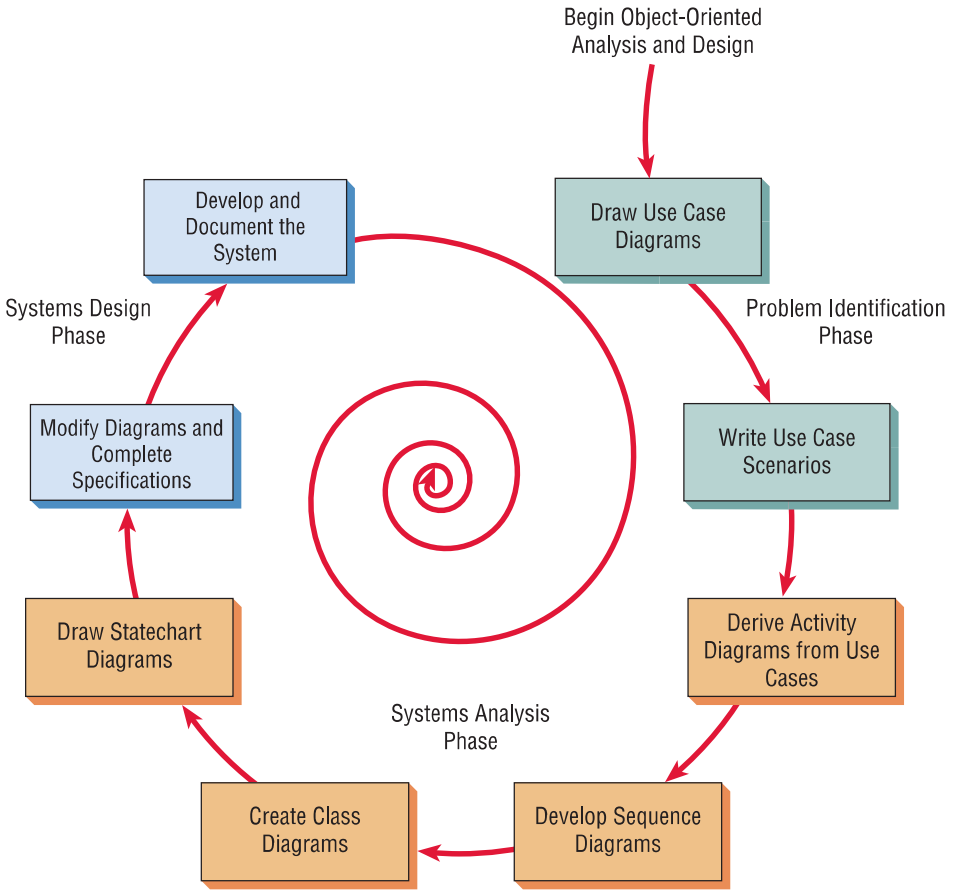
The phases in UML are similar to those in the SDLC. Since those two methods share rigid and exacting modeling, they happen in a slower, more deliberate pace than the phases of agile modeling. The analyst goes through problem and identification phases, an analysis phase, and a design phase as shown in Figure 1.8. Although much of the specifics are discussed in Chapters 2 and 10, the following steps give a brief description of the UML process.

1. Define the use case model.

   In this phase the analyst identifies the actors and the major events initiated by the actors. Often the analyst will start by drawing a diagram with stick figures representing the actors and arrows showing how the actors relate. This is called a use case diagram (Chapter 2) and it represents the standard flow of events in the system. Then an analyst typically writes up a use case scenario (Chapter 2), which describes in words the steps that are normally performed.
2. During the systems analysis phase, begin drawing UML diagrams.

   In the second phase (Chapter 10), the analyst will draw Activity Diagrams, which illustrate all the major activities in the use case. In addition, the analyst will create one or

FIGURE 1.8
The steps in the UML development process.



more sequence diagrams for each use case, which show the sequence of activities and their timing. This is an opportunity to go back and review the use cases, rethink them, and modify them if necessary.

3. Continuing in the analysis phase, develop class diagrams.
   The nouns in the use cases are objects that can potentially be grouped into classes. For example, every automobile is an object that shares characteristics with other automobiles. Together they make up a class.
4. Still in the analysis phase, draw statechart diagrams.
   The class diagrams are used to draw statechart diagrams, which help in understanding complex processes that cannot be fully derived by the sequence diagrams. The statechart diagrams are extremely useful in modifying class diagrams, so the iterative process of UML modeling continues.
5. Begin systems design by modifying the UML diagrams. Then complete the specifications.
   Systems design means modifying the existing system and that implies modifying the diagrams drawn in the previous phase. These diagrams can be used to derive classes, their attributes, and methods (methods are simply operations). The analyst will need to write class specifications for each class including the attributes, methods, and their descriptions. They will also develop methods specifications that detail the input and output requirements for the method, along with a detailed description of the internal processing of the method.
6. Develop and document the system.
   UML is, of course, a modeling language. An analyst may create wonderful models, but if the system isn't developed there is not much point in building models. Documentation is critical. The more complete the information you provide the development team through documentation and UML diagrams, the faster the development and the more solid the final production system.

Object-oriented methodologies often focus on small, quick iterations of development, sometimes called the spiral model. Analysis is performed on a small part of the system, usually starting

Choose	When
The Systems Development Life Cycle (SDLC) Approach	• systems have been developed and documented using SDLC • it is important to document each step of the way • upper-level management feels more comfortable or safe using SDLC • there are adequate resources and time to complete the full SDLC • communication of how new systems work is important
Agile Methodologies	• there is a project champion of agile methods in the organization • applications need to be developed quickly in response to a dynamic environment • a rescue takes place (the system failed and there is no time to figure out what went wrong) • the customer is satisfied with incremental improvements • executives and analysts agree with the principles of agile methodologies
Object-Oriented Methodologies	• the problems modeled lend themselves to classes • an organization supports the UML learning • systems can be added gradually, one subsystem at a time • reuse of previously written software is a possibility • it is acceptable to tackle the difficult problems first

FIGURE 1.9
How to decide which development method to use.

with a high-priority item or perhaps one that has the greatest risk. This is followed by design and implementation. The cycle is repeated with analysis of the next part, design, and some implementation, and it is repeated until the project is completed. Reworking diagrams and the components themselves is normal. UML is a powerful modeling tool that can greatly improve the quality of your systems analysis and design and the final product.

CHOOSING WHICH SYSTEMS DEVELOPMENT METHOD TO USE

The differences among the three approaches described earlier are not as big as they seem at the outset. In all three approaches, the analyst needs to understand the organization first (Chapter 2). Then the analyst or project team needs to budget their time and resources and develop a project proposal (Chapter 3). Next they need to interview organizational members and gather detailed data by using questionnaires (Chapter 4) and sample data from existing reports and observe how business is currently transacted (Chapter 5). The three approaches have all of these activities in common.

Even the methods themselves have similarities. The SDLC and object-oriented approaches both require extensive planning and diagramming. The agile approach and the object-oriented approach both allow subsystems to be built one at a time until the entire system is complete. The agile and SDLC approaches are both concerned about the way data logically moves through the system.

So given a choice to develop a system using an SDLC approach, an agile approach, or an object-oriented approach, which would you choose? Figure 1.9 provides a set of guidelines to help you choose which method to use when developing your next system.

SUMMARY

Information can be viewed as an organizational resource just as humans are. As such, it must be managed carefully, just as other resources are. The availability of affordable computer power to organizations has meant an explosion of information, and consequently, more attention must be paid to coping with the information generated.

Systems analysts recommend, design, and maintain many types of systems for users, including transaction processing systems (TPS), office automation systems (OAS), knowledge work systems (KWS), and management information systems (MIS). They also create decision-oriented systems for specific users. These

HYPERCASE® EXPERIENCE 1

"Welcome to Maple Ridge Engineering, what we call MRE. We hope you'll enjoy serving as a systems consultant for us. Although I've worked here five years in different capacities, I've just been reassigned to serve as an administrative aide to Snowden Evans, the head of the new Training and Management Systems Department. We're certainly a diverse group. As you make your way through the company, be sure to use all your skills, both technical and people oriented, to understand who we are and to identify the problems and conflicts that you think should be solved regarding our information systems."

"To bring you up to date, let me say that Maple Ridge Engineering is a medium-sized medical engineering company. Last year, our revenues exceeded $287 million. We employ about 335 people. There are about 150 administrative employees as well as management and clerical staff like myself; approximately 75 professional employees, including engineers, physicians, and systems analysts; and about 110 trade employees, such as drafters and technicians."

"There are four offices. You will visit us through HyperCase in our home office in Maple Ridge, Tennessee. We have three other branches in the southern United States as well: Atlanta, Georgia; Charlotte, North Carolina; and New Orleans, Louisiana. We'd love to have you visit when you're in the area."

"For now, you should explore HyperCase using either Firefox, Safari, or Microsoft Internet Explorer."

"To learn more about Maple Ridge Engineering as a company or to find out how to interview our employees, who will use the systems you design, and how to observe their offices in our company, you may want to start by going to the Web site found at www.pearsonhighered.com/kendall. Then click on the link labeled **HyperCase.** At the HyperCase display screen, click on **Start** and you will be in the reception room for Maple Ridge Engineering. From this point, you can start consulting right away."

This Web site contains useful information about the project as well as files that can be downloaded to your computer. There is a set of Visible Analyst data files, and another set of Visio data files that match HyperCase. They contain a partially constructed series of data flow diagrams, entity-relationship diagrams, UML diagrams, and repository information. The HyperCase Web site also contains additional exercises that may be assigned. HyperCase is designed to be explored, and you should not overlook any object or clue on a Web page.

include decision support systems (DSS), expert systems (ES), group decision support systems (GDSS), computer-supported collaborative work systems (CSCWS), and executive support systems (ESS). Many applications are either originating on, or moving to, the Web to support ecommerce and many other business functions.

Systems analysis and design is a systematic approach to identifying problems, opportunities, and objectives; to analyzing human and computer-generated information flows in organizations; and to designing computerized information systems to solve a problem. Systems analysts are required to take on many roles in the course of their work. Some of these roles are (1) an outside consultant to business, (2) a supporting expert within a business, and (3) an agent of change in both internal and external situations.

Analysts possess a wide range of skills. First and foremost, the analyst is a problem solver, someone who enjoys the challenge of analyzing a problem and devising a workable solution. Systems analysts require communication skills that allow them to relate meaningfully to many different kinds of people on a daily basis, as well as computer skills. Understanding and relating well to users is critical to their success.

Analysts proceed systematically. The framework for their systematic approach is provided in what is called the systems development life cycle (SDLC). This life cycle can be divided into seven sequential phases, although in reality the phases are interrelated and are often accomplished simultaneously. The seven phases are identifying problems, opportunities, and objectives; determining human information requirements; analyzing system needs; designing the recommended system; developing and documenting software; testing and maintaining the system; and implementing and evaluating the system.

The agile approach is a software development approach based on values, principles, and core practices. Systems that are designed using agile methods can be developed rapidly. Stages in the agile development process are exploration, planning, iterations to the first release, productionizing, and maintenance.

A third approach to systems development is called object-oriented analysis design. These techniques are based on object-oriented programming concepts that have become codified in UML, a standardized modeling language in which objects that are created include not only code about data but also instructions about the operations to be performed on the data. Key diagrams help analyze, design, and communicate UML-developed systems. These systems are usually developed as components and reworking the components many times is a normal activity in object-oriented analysis and design.

KEYWORDS AND PHRASES

agent of change
agile approach
agile methods
Ajax
artificial intelligence (AI)
bespoke software
Computer-Assisted Software Engineering (CASE)
CASE tools
computer-supported collaborative work systems (CSCWS)
decision support systems (DSS)
ecommerce applications
enterprise resource planning (ERP) systems
executive support systems (ESS)
expert systems
exploration phase
group decision support systems (GDSS)
human–computer interaction (HCI)
iterations to the first release phase
knowledge work systems (KWS)
maintenance phase
management information systems (MIS)
mcommerce (mobile commerce)
migrate systems
object-oriented (O-O) systems analysis and design
office automation systems (OAS)
open source software (OSS)
planning game
planning phase
productionizing phase
prototyping
rapid application development (RAD)
service-oriented architecture (SOA)
systems analysis and design
systems analyst
systems consultant
systems development life cycle (SDLC)
transaction processing systems (TPS)
unified modeling language (UML)

REVIEW QUESTIONS

1. Compare treating information as a resource to treating humans as a resource.
2. List the differences between OAS and KWS.
3. Define what is meant by MIS.
4. How does MIS differ from DSS?
5. Define the term *expert systems.* How do expert systems differ from decision support systems?
6. List the problems of group interaction that group decision support systems (GDSS) and computer-supported collaborative work systems (CSCWS) were designed to address.
7. Which is the more general term, CSCWS or GDSS? Explain.
8. Define the term *mcommerce.*
9. List the advantages of mounting applications on the Web.
10. What is the overarching reason for designing enterprise (or ERP) systems?
11. Provide an example of an open source software project.
12. List the advantages of using systems analysis and design techniques in approaching computerized information systems for business.
13. List three roles that the systems analyst is called upon to play. Provide a definition for each one.
14. What personal qualities are helpful to the systems analyst? List them.
15. List and briefly define the seven phases of the systems development life cycle (SDLC).
16. What are CASE tools used for?
17. What is the difference between upper and lower CASE tools?
18. Define what is meant by the agile approach.
19. What is the meaning of the phrase "the planning game"?
20. What are the stages in agile development?
21. Define the term *object-oriented analysis* and *design.*
22. What is UML?

SELECTED BIBLIOGRAPHY

Coad, P., and E. Yourdon. *Object-Oriented Analysis,* 2d ed. Englewood Cliffs, NJ: Prentice Hall, 1991.

Davis, G. B., and M. H. Olson. *Management Information Systems: Conceptual Foundation, Structure, and Development,* 2d ed. New York: McGraw-Hill, 1985.

Feller, J., P. Finnegan, D. Kelly, and M. MacNamara. "Developing Open Source Software: A Community-Based Analysis of Research." In IFIP International Federation for Information Processing, Vol. 208, *Social Inclusion: Societal and Organizational Implications for Information Systems.* Edited by E. Trauth, D. Howcroft, T. Butler, B. Fitzgerald, and J. DeGross, pp. 261–278. Boston: Springer, 2006.

Kendall, J. E., and K. E. Kendall. "Information Delivery Systems: An Exploration of Web Push and Pull Technologies." *Communications of AIS,* Vol. 1, Article 14, April 23, 1999.

Kendall, J. E., K. E. Kendall, and S. Kong. "Improving Quality Through the Use of Agile Methods in Systems Development: People and Values in the Quest for Quality." In *Measuring Information Systems Delivery Quality.* Edited by E. W. Duggan and H. Reichgelt, pp. 201–222. Hershey, PA: Idea Group Publishing, 2006.

Laudon, K. C., and J. P. Laudon. *Management Information Systems,* 11th ed. Upper Saddle River, NJ: Pearson Prentice Hall, 2010.

Verma, S. "Software Quality and the Open Source Process." In *Measuring Information Systems Delivery Quality.* Edited by E. W. Duggan and H. Reichgelt, pp. 284–303. Hershey, PA: Idea Group Publishing, 2006.

www.visible.com/Products/index.htm. Last accessed March 23, 2009.

Yourdon, E. *Modern Structured Analysis.* Englewood Cliffs, NJ: Prentice Hall, 1989.

Zhang, P., J. Carey, D. Te'eni, and M. Tremaine. "Integrating Human–Computer Interaction Development into the Systems Development Life Cycle: A Methodology." *Communications of the Association for Information Systems,* Vol. 15, 2005, pp. 512–543.

EPISODE 1

CPU CASE

ALLEN SCHMIDT, JULIE E. KENDALL, AND KENNETH E. KENDALL

The Case Opens

On a warm, sunny day in late October, Chip Puller parks his car and walks into his office at Central Pacific University. It felt good to be starting as a systems analyst, and he was looking forward to meeting the other staff.

In the office, Anna Liszt introduces herself. "We've been assigned to work as a team on a new project. Why don't I fill you in with the details, and then we can take a tour of the facilities?"

"That sounds good to me," Chip replies. "How long have you been working here?"

"About five years," answers Anna. "I started as a programmer analyst, but the last few years have been dedicated to analysis and design. I'm hoping we'll find some ways to increase our productivity," Anna continues.

"Tell me about the new project," Chip says.

"Well," Anna replies, "like so many organizations, we have a large number of microcomputers with different software packages installed on them. From what I understand, in the 1980s there were few personal computers and a scattered collection of software. This expanded rapidly in the 1990s, and now everyone uses computers. Some faculty members use more than one computer. The current system that is used to maintain software and hardware, which was originally quite useful, is now very outdated and quite overwhelmed."

"What about the users? Who should I know? Who do you think will be important in helping us with the new system?" Chip asks.

"You'll meet everyone, but there are key people I've recently met, and I'll tell you what I've learned so you'll remember them when you meet them.

"Dot Matricks is manager of all microcomputer systems at Central Pacific. We seem to be able to work together well. She's very competent. She'd really like to be able to improve communication among users and analysts."

"It will be a pleasure to meet her," Chip speculates.

"Then there's Mike Crowe, computer maintenance expert. He really seems to be the nicest guy, but way too busy. We need to help lighten his load. The software counterpart to Mike is Cher Ware. She's a free spirit, but don't get me wrong, she knows her job," Anna says.

"She could be fun to work with," Chip muses.

"Could be," Anna agrees. "You'll meet the financial analyst, Paige Prynter, too. I haven't figured her out yet."

"Maybe I can help," Chip says.

"Last, you should—I mean, you will—meet Hy Perteks, who does a great job running the Information Center. He'd like to see us be able to integrate our life cycle activities."

"It sounds promising," Chip says. "I think I'm going to like it here."

EXERCISES

E-1. From the introductory conversation Chip and Anna shared, which elements mentioned might suggest the use of CASE tools?