SYSTEMS ANALYSIS AND DESIGN IN A CHANGING WORLD

Satzinger | Jackson | Burd

SIXTH EDITION

COURSE TECHNOLOGY
CENGAGE Learning

Systems Analysis and Design in a Changing World, Sixth Edition

John W. Satzinger, Robert B. Jackson, Stephen D. Burd

Publisher: Joe Sabatino

Senior Acquisitions Editor: Charles McCormick Jr.

Senior Product Manager: Kate Mason

Development Editor: Kent Williams

Editorial Assistant: Courtney Bavaro

Marketing Director: Keri Witman

Marketing Manager: Adam Marsh

Senior Marketing Communications Manager: Libby Shipp

Marketing Coordinator: Suellen Ruttkay

Design Direction, Production Management, and Composition: PreMediaGlobal

Media Editor: Chris Valentine

Senior Art Director: Stacy Jenkins Shirley

Cover Designer: Itzhack Shelomi

Cover Credit: ©iStock Photo

Manufacturing Coordinator: Julio Esperas

Library of Congress Control Number: 2011940028

ISBN-13: 978-1-111-53415-8

ISBN-10: 1-111-53415-2

Instructor Edition:

ISBN-13: 978-1-111-53284-0

ISBN-10: 1-111-53284-2

Course Technology
20 Channel Center Street
Boston, MA 02210
USA

Screenshots for this book were created using Microsoft Visio® and Project®, and were used with permission from Microsoft.

Printed in the United States of America
1 2 3 4 5 6 7 15 14 13 12 11

1

From Beginning to End: An Overview of Systems Analysis and Design

Chapter Outline

- Software Development and Systems Analysis and Design
- Systems Development Life Cycle
- Introduction to Ridgeline Mountain Outfitters
- Iterative Development
- Developing RMO's Tradeshow System
- Where You Are Headed—The Rest of This Book

Learning Objectives

After reading this chapter, you should be able to:

- Describe the purpose of systems analysis and design in the development of information systems
- Describe the characteristics of iterative systems development
- Explain the six core processes of the Systems Development Life Cycle
- Identify key documents that are used in planning a project
- Identify key diagrams used in systems analysis and systems design
- Explain the utility of identifying use cases in systems development
- Explain the utility of identifying object classes in systems development

Software Development and Systems Analysis and Design

Computers are everywhere today, and microchips impact every part of our lives. We live in a world not only of ubiquitous computing but of pervasive communication and connectivity. An incredibly large part of our everyday lives depends on computer chips, connection links, and application software.

You have grown up in this world of high technology. You use smartphones, laptops, iPads, notepads, electronic game equipment, and so on. Your mobile devices provide daily (if not hourly) text messages, tweets, videos, snapshots, Internet access, games, and much more. Many of you have already developed your own application software or you have a friend who has written applications for laptops, smartphones, iPads, or Facebook. Some of you have taken programming classes; others of you have taught yourself how to write computer application programs. Given that we live in this world of high-tech gadgets, we might ask, "What is systems analysis and design, and why is it important?" How does the development of new technology and new application software utilize systems analysis and design? In other words, what role does systems analysis and design play in the development of high-tech solutions and applications?

computer application (app) a computer software program that executes on a computing device to carry out a specific function or set of related functions

information system a set of interrelated computer components that collects, processes, stores, and provides as output the information needed to complete business tasks

First, let us provide two important definitions. A **computer application** is a computer software program that executes on a computing device to carry out a specific function or set of related functions. Sometimes, *computer application* is shortened to **app** (such as an iPhone app or a Facebook app). An **information system** is a set of interrelated computer components that collects, processes, stores (usually in a database), and provides as output the information needed to complete business tasks. Although these terms are sometimes used synonymously, an application usually refers to only the computer software involved, whereas an information system may include the software, the database, and even the related manual processes. Examples of computer applications include browsers that access the Internet to play games or calendaring apps. **Figure 1-1** shows a typical mobile digital device.

Why is systems analysis and design important in the development of information systems? To answer that question, let us consider an analogous

FIGURE **1-1**
Typical mobile digital device

situation: the art and science of creating a beautiful building. In this scenario, there is the buyer who has the vision, the builder who will construct the building, and the architect who serves as the bridge between the buyer and the builder. The architect helps the buyer develop the vision but must also communicate the building's specifications to the builder. In doing so, the architect uses various tools to first capture the vision from the buyer and then provide the builder with instructions—including such tools as line drawings, blueprints, to-scale models, detail specifications, and even on-site inspections.

Just as a builder doesn't start construction without plans, programmers don't just sit down and start writing program code. They need someone (maybe themselves) to function like an architect—planning, capturing the vision, understanding details, specifying needs—before writing the code and verifying that it satisfies the vision. The software architect has to be able to understand and capture the vision of the persons funding the project. Usually, we call this person a *systems analyst*. In situations where you are the programmer as well as the analyst, it might be easy to keep track of the details without writing them down. However, in today's world, with some system development teams distributed worldwide, you may only be responsible for part of the programming, with the rest handled by team members around the world. In a distributed team situation, it is much more important to have written documents to assist in understanding, capturing, explaining, and specifying the software application.

In a nutshell, systems analysis and design (SA&D) is about providing the tools and techniques to you, the developer, so you can understand the need (business need), capture the vision, define a solution, communicate the vision and the solution, build the solution and direct others in building the solution, confirm that the solution meets the need, and launch the solution application.

Included in SA&D are all the skills, steps, guidelines, and tools that support and lead up to the actual programming of the system. SA&D includes such "soft" skills as interviewing and talking to users as well as such "hard" (more technical) skills as detailing specifications and designing solutions. Many of the technical skills are associated with creating models that capture specifications or define solutions. In this book, you will learn all these skills as well as how they work together to develop an information system.

systems analysis those activities that enable a person to understand and specify what the new system should accomplish

Let us conclude this section with a few more definitions. **Systems analysis** consists of those activities that enable a person to understand and specify what the new system should accomplish. The operative words here are "understand" and "specify." Systems analysis is much more than simply a brief statement of the problem. For example, a customer management system must keep track of customers, register products, monitor warranties, and keep track of service levels, among many other functions—all of which have myriad details. Systems analysis describes in detail the "what" that a system must do to satisfy the need or to solve the problem.

systems design those activities that enable a person to define and describe in detail the system that solves the need

Systems design consists of those activities that enable a person to describe in detail the system that solves the need. The operative word in this case is "solves." In other words, systems design describes "how" the system will work. It specifies in detail all the components of the solution system and how they work together to provide the desired solution.

Systems Development Life Cycle

project a planned undertaking that has a beginning and an end, and that produces some definite result

Initial development of a new system is usually done as a project. What this means is that the activities required to develop a new system are identified, planned, organized, and monitored. We can think of a **project** as a planned undertaking that has a beginning and an end and produces some definite result. Some projects are very formal, whereas others are so informal they can barely be recognized as projects.

Systems Development Life Cycle (SDLC) the entire process consisting of all the activities required to build, launch, and maintain an information system

To manage a project with analysis, design, and other development activities, you need a project management framework to guide and coordinate the work of the project team. The **Systems Development Life Cycle (SDLC)** identifies all the activities required to build, launch, and maintain an information system. Normally, the SDLC includes all the activities that are part of systems analysis, systems design, programming, testing, and maintaining the system as well as other project management processes that are required to successfully launch and deploy the new information system.

There are many approaches to the SDLC and many variations for projects that have various needs. However, there is a core set of processes that is always required, even though there is also an incredible number of variations of these core processes—in how each process is planned and executed and in how the processes are combined into a project. Here are six core processes required in the development of any new application:

1. Identify the problem or need and obtain approval to proceed.
2. Plan and monitor the project—what to do, how to do it, and who does it.
3. Discover and understand the details of the problem or the need.
4. Design the system components that solve the problem or satisfy the need.
5. Build, test, and integrate system components.
6. Complete system tests and then deploy the solution.

information systems development process the actual approach used to develop a particular information system

There are many ways to implement these six core processes of the SDLC. An **information systems development process** is the actual approach used to develop a particular information system. Most information systems you will develop are conceived and built to solve organizational problems, which are usually very complex, thus making it difficult to plan and execute a development project. In fact, many projects end up being much larger than expected—often resulting in late deliveries that are over budget. During the last 10 years, several new information systems development processes have been developed to enhance project success. One of the newer and more effective ones is called **Agile Development**. The basic philosophy of Agile Development is that neither team members nor the user completely understands the problems and complexities of a new system, so the project plan and the execution of the project must be responsive to unanticipated issues. It must be agile and flexible. It must have procedures in place to allow for, anticipate, and even embrace changes and new requirements during the development process.

Agile Development an information systems development process that emphasizes flexibility to anticipate new requirements during development

Maybe the best way to understand these concepts is to see the way they play out in a complete example. That is the objective of this chapter, "From Beginning to End: An Overview of Systems Analysis and Design." Here, we will use a fairly small information system application to show you all six core processes (as much as is feasible in a textbook, anyway). We will illustrate one way to organize the various activities into an actual working project; in other words, we will show you one version of an information system development process. By going all the way through a very simple project, you will more easily learn and understand the complex concepts provided in the rest of the text. Our project involves Ridgeline Mountain Outfitters, a retailer and manufacturer of clothing for all types of outdoor activities.

Introduction to Ridgeline Mountain Outfitters

Ridgeline Mountain Outfitters (RMO) is a large retail company that specializes in clothing and related accessories for all types of outdoor and sporting activities. By the 2010s, the Rocky Mountain and Western states had seen tremendous growth in recreation activities, and with the increased interest in outdoor sports, the market for winter and summer sports clothes had exploded. Skiing, snowboarding, mountain biking, water skiing, jet skiing, river running, jogging,

hiking, ATV biking, camping, mountain climbing, rappelling—all had seen a tremendous increase in interest in these states. People needed appropriate sports clothes for these activities, so RMO expanded its line of sportswear to respond to this market. It also added a line of high-fashion activewear and accessories to round out its offerings to the expanding market of active people.

The company's growth charted an interesting history of mail-order, brick-and-mortar, and online sales. RMO got its start by selling to local clothing stores in the Park City, Utah, area. In the late-1980s and early-1990s, it began selling directly to customers by using catalogs with mail-in and telephone orders. It opened its first store in 1994 and soon expanded to 10 retail outlets throughout the West. Last year, retail store revenue was $67 million, telephone and mail-order revenues were $10 million, and Web-based sales were $200 million. Most sales continue to be in the West, although the market in several areas of the eastern United States and Canada is growing.

RMO produces its own line of outdoor and sportswear clothing. However, in order to offer a complete range of outdoor clothing in its retail outlets, it also sells other brands of outdoor and sportswear clothing. In addition, other types of clothing and accessories, such as footwear, leather apparel, and specialty sportswear, are available in the retail stores and through the online store.

Figure 1-2 shows a sample of the catalogs that RMO mails out. Although mail-order and telephone sales are small, receiving the catalog encourages customers to go online to make purchases, so RMO continues to produce and mail abbreviated versions of its catalogs.

Figure 1-3 illustrates a typical order page from the online system.

Trade Shows

In order to keep its product line current and popular, RMO's purchasing agents attend apparel and fabric trade shows around the world. RMO purchasers have a good track record of predicting what products will be good sellers. In addition, RMO is always watching for new products and accessories that will enable it to expand its product line appropriately.

When purchasing agents attend a trade show, they frequently find various products that they want to add to the spring, summer, or winter apparel offering.

FIGURE 1-2
RMO winter catalog

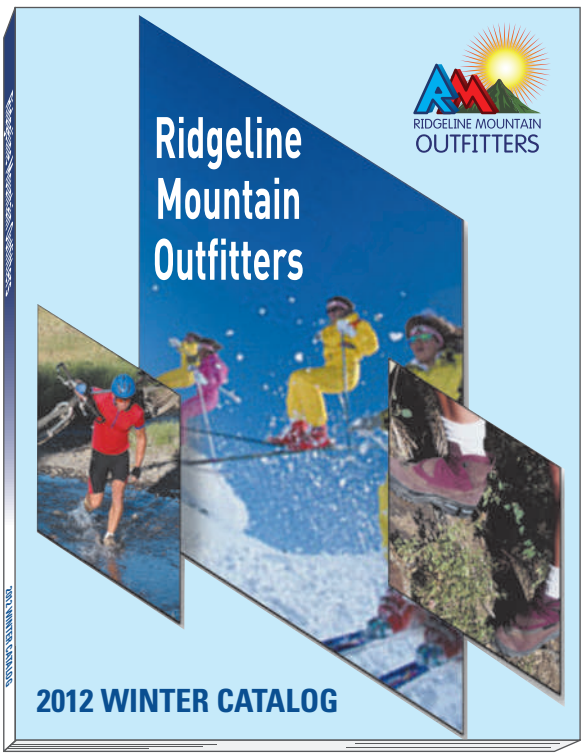
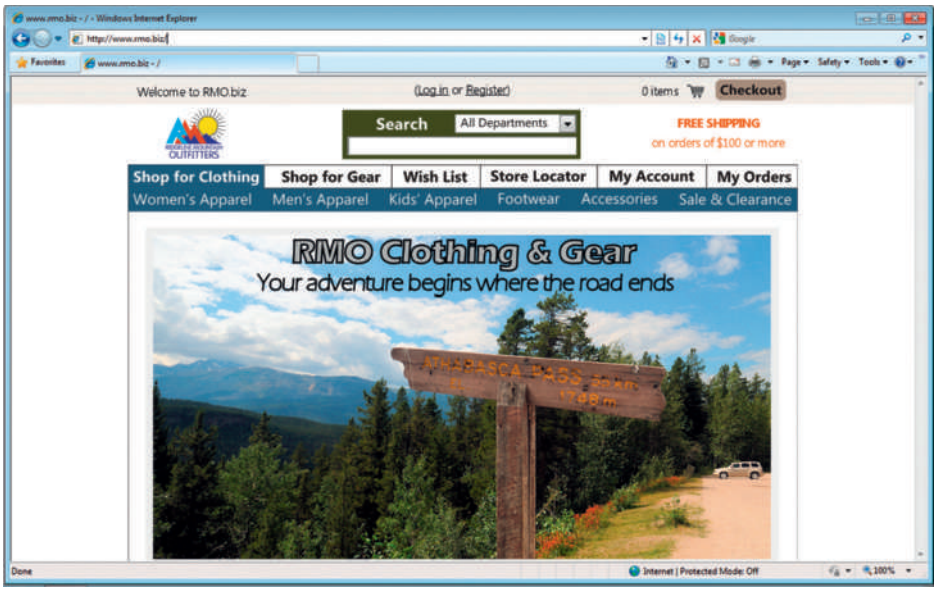


FIGURE 1-3
RMO sample online ordering page



In the past, when RMO buyers wanted to place an order, they would exchange contact information with the seller at the trade show and upon returning to the home office would then follow up via e-mails and phone calls to formulate a contract and make a purchase order. However, to expedite orders, RMO has now initiated a project to develop a system for collecting and tracking information about its suppliers and the new products it adds to its merchandise offerings.

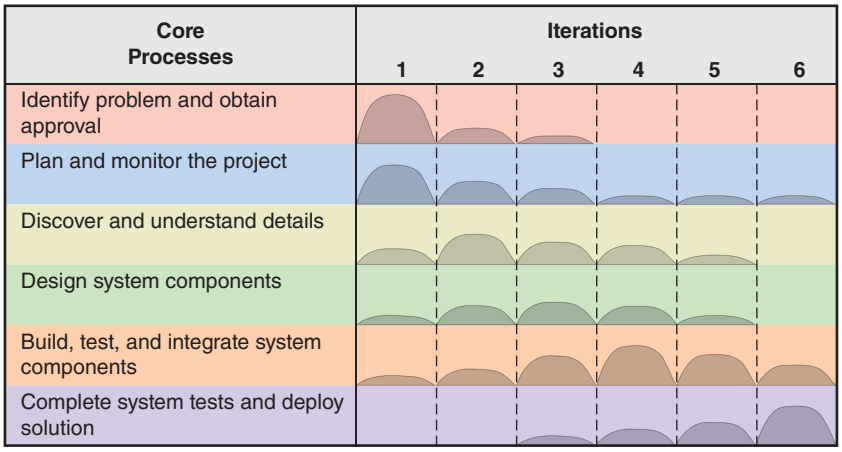
Iterative Development

iterative development an approach to system development in which the system is "grown" piece by piece through multiple iterations

Iterative development is an approach to system development in which the system is "grown" in an almost organic fashion. Core components are developed first and then additional components are added. It is called "iterative" because the six core development processes are repeated over and over again to add additional functionality to the overall system. In other words, there is one big project, which consists of many mini-projects, and the information system is grown piece by piece.

Figure 1-4 illustrates how an iterative Agile project might be executed. This is a sample diagram. Real projects might be quite different. Across the figure, you see six iterations. An iteration is like a mini-project in that it has a completed result and a constrained time frame. Often, it will last two to four weeks. Down the left side of the figure, you see the six core processes. The rounded

FIGURE 1-4
The six core processes, with iterations for a typical project



mounds inside the graph represent the amount of effort for that core process during that iteration. The amount of area under a curve is an approximate indication of the amount of effort expended within an iteration with regard to a particular core process. For example, in Figure 1-4, Iteration 1 appears to primarily focus on identifying the problem and planning the project. Lesser amounts of discovery, design, and build and test may also be done. For this iteration, nothing is done with regard to deploying the system.

There are several benefits to iterative development. For one, portions of the often system can sometimes be deployed sooner. If there are core functions that provide basic support, these can be deployed in an early iteration. A second benefit is that by taking a small portion and developing it first, many tough problems can be addressed early in the project. Many of today's systems are so large and complex that it is impossible to remember and understand everything. By focusing on only a small portion, the requirements are fewer and easier to grasp and solve. Finally, developing a system in iterations makes the entire development process much more flexible and able to address new requirements throughout the project.

A key element of iterative development is choosing a part of the solution system that can be done in two to four weeks. During one iteration, often all the core development processes are involved, including programming and system-wide testing, so the result is a part of the working system, even though it may only have a portion of the functionality that is ultimately required.

Developing RMO's Tradeshow System

We will organize our sample project—the RMO Tradeshow System—for the first iteration, and our goal is to have the iteration last six days. However, our primary objective is to introduce you to the concepts and techniques of the six core processes. Therefore, in some instances, we may go a little deeper into a core process than we might do on the first iteration of a real project. It is a little unrealistic to complete an entire iteration with all the necessary details in only six days, but it should be a good learning experience. There will not be a one-to-one correspondence with the six SDLC core processes and the six days of the project, but we will include all the SDLC core processes within the project.

At the end of this chapter, we have provided a small case project that has also been divided into a six-day project. Each day of the project has several possible assignments that can be completed to help you solidify your understanding of systems analysis and design and the six core processes.

Most new applications require a project with several iterations. In the first iteration, there are usually three major objectives. The first objective is to get project approval. The second objective is to get a clear picture of the system's overall vision—all the major functions and data requirements. The third objective is to determine the detail specifications and develop a solution for one portion of the system (i.e., actually analyze, design, build, and test one part of the system).

In our project, we will touch on all these objectives. We will show an example of a System Vision Document and then develop one portion of the overall system. We have constrained the scope of the new system so we can complete it in one iteration. It should be noted that the division of this project into days and daily activities is somewhat arbitrary. There are numerous ways to partition and organize the work. The following organization is quite workable, but it is not the only way to organize the project.

Pre-Project Activities

Before the project actually begins, the head of RMO's purchasing department works with a systems analyst to identify and document the specific business need as well as define a specific project objective. RMO's management reviews the primary project objective and provides budget approval. Every organization has to give budget approval before a project can start. Some organizations have

a formal process to get a project approved; other organizations have a less formal process. Normally, there are two goals an organization must decide on to get a project off the ground:

- Identify the problem and document the objective of the solution system. (Core Process 1)
- Obtain approval to commence the project. (Core Process 1)

System Vision Document

As with all new projects within RMO, a System Vision Document is developed to identify the benefits to the company and the functional capabilities that will be included in the system. Frequently, this is done in two steps: developing a preliminary statement of benefits and then adding estimates of specific dollar costs and dollar benefits. **Figure 1-5** is the System Vision Document for this project.

FIGURE 1-5
Tradeshow System Vision Document

System Vision Document
RMO Tradeshow System

RIDGELINE MOUNTAIN OUTFITTERS

Problem Description

Trade shows have become an important information source for new products, new fashions, and new fabrics. In addition to the large providers of outdoor clothing and fabrics, there are many smaller providers. It is important for RMO to capture information about these suppliers while the trade show is in progress. It is also important to obtain information about specific merchandise products that RMO plans to purchase. Additionally, if quality photographs of the products can be obtained while at the trade show, then the creation of online product pages is greatly facilitated.

It is recommended that a new system be developed and deployed so field purchasing agents can communicate more rapidly with the home office about suppliers and specific products of interest. This system should be deployed on portable equipment.

System Capabilities

The new system should be capable of:

- Collecting and storing information about the manufacturer/wholesaler (suppliers)
- Collecting and storing information about sales representatives and other key personnel for each supplier
- Collecting information about products
- Taking pictures of products (and/or uploading stock images of products)
- Functioning as a stand-alone without connection
- Connecting via Wi-Fi (Internet) and transmitting data
- Connecting via telephone and transmitting data

Business Benefits

It is anticipated that the deployment of this new system will provide the following business benefits to RMO:

- Increase timely communication between trade show attendees and home office, thereby improving the quality and speed of purchase order decisions
- Maintain correct and current information about suppliers and their key personnel, thereby facilitating rapid communication with suppliers
- Maintain correct and rapid information and images about new products, thereby facilitating the development of catalogs and Web pages
- Expedite the placing of purchase orders for new merchandise, thereby catching trends more rapidly and speeding up product availability

As described earlier, RMO needs a portable system that can be used by its purchasing agents as they attend various product and clothing fabric trade shows. The system needs to fulfill two major requirements. First, it has to have the functionality to capture information about suppliers and products. Second, it needs to be able to communicate with the home office systems, and because these trade shows are held in various venues around the world, various methods of connectivity are needed.

Preliminary investigation considered various equipment options, including laptop computers, iPad computing devices, and smartphones. Even though smartphones appeared to have the best connection options, the small size made viewing the details of photographs somewhat difficult; the iPad and other similar portable devices with advanced technology also appear to be viable options. However, due to the similarity of the smartphones and tablets, it seems feasible to develop an application that will execute on either device. Each purchasing agent could use his or her preferred device.

Toward the end of the pre-project activities, a meeting is held involving all the key persons, including a representative of executive management. The decision is made to move ahead with the project and budget the necessary funds.

Day 1 Activities

RMO—Supplier Information Subsystem

The project actually begins with Day 1, which is essentially a planning day. Usually, the first activity is the project team reviewing the System Vision Document and verifying that the preliminary work is still valid. It reviews the scope of the project to become familiar with the problem to be solved, then it plans the iterations and activities for the remainder of the project. The second SDLC core process—planning the project—includes business analysis and project management activities. All these topics will be treated in depth in later chapters. These activities are completed on Day 1:

- Determine the major components (functional areas) that are needed. (Core Process 2)
- Define the iterations and assigning each functional area to an iteration. (Core Process 2)
- Determine team members and responsibilities. (Core Process 2)

Planning the Overall Project and the Project Iterations

Myriad details need to be considered in a project plan. For our project, we will only focus on the bare essentials. We will describe project planning more elaborately in later chapters.

The project team meets with the users to review the overall business need and the objectives of the new system. The System Vision Document serves as the starting point for these discussions. As is often the case, the list of System Capabilities provides the foundation information for determining the overall project plan. The first step is to divide the system into several subsystems or components. A **subsystem** is simply a portion of the overall system. Based on the list of System Capabilities, the project team identifies these functional subsystems:

subsystem an identifiable and partitioned portion of an overall system

- Supplier Information subsystem
- Product Information subsystem

The Supplier Information subsystem will collect and maintain information about the manufacturers or wholesalers and the contract people that work for them. The Product Information subsystem will capture information about the various products, including detailed descriptions and photographs.

The next step is to identify which subsystems will be developed in which order. Many issues are considered, such as dependencies between the various

tasks, sequential versus parallel development, project team availability, and project urgency. In our case, the team decides that the project will proceed in a serial fashion, with the Supplier Information subsystem scheduled as the first iteration.

Planning the First Iteration

Each iteration is like a systems development mini-project. The core processes described earlier can all be applied, with the scope limited to the component that is to be developed during the iteration. The planning process for an iteration consists of these three steps:

- Identify the tasks required for the iteration.
- Organize and sequence these tasks into a schedule.
- Identify required resources (especially people) and assign people to tasks.

The first step is to identify—or attempt to identify—all the individual tasks that need to be done. As these tasks are identified, they are compiled and organized. Sometimes, this organized list of tasks is called a Work Breakdown Structure. **Figure 1-6** shows the Work Breakdown Structure for this iteration.

Part of the effort is trying to estimate how long each task will take. Because this project has a very limited scope (only six days), all the estimates will be in hours. These estimates do not include the time expended by those who are not on the team. However, of those on the team, the estimates include the time for the original work, the time for discussion, and the time for reviewing and checking the Work Breakdown Structure for accuracy and correctness.

The next step is to get these tasks organized into a schedule. Again, we can be very formal and use a sophisticated project-scheduling tool or we can just list the tasks in the order we think they need to be done. An important part of building the schedule is identifying any dependencies between the tasks. For

FIGURE 1-6
Sample handwritten Work Breakdown Structure

Work Breakdown Structure

I. Discover and understand the details of all aspects of the problem.
   1. Meet with the Purchasing Department manager. ~ 3 hours
   2. Meet with several purchasing agents. ~ 4 hours
   3. Identify and define use cases. ~ 3 hours
   4. Identify and define information requirements. ~ 2 hours
   5. Develop workflows and descriptions for the use cases. ~ 6 hours

II. Design the components of the solution to the problem.
   1. Design (lay out) input screens, output screens, and reports. ~ 8 hours
   2. Design and build database (attributes, keys, indexes). ~ 4 hours
   3. Design overall architecture. ~ 4 hours
   4. Design program details. ~ 6 hours

III. Build the components and integrate everything into the solution.
   1. Code and unit test GUI layer programs. ~ 14 hours
   2. Code and unit test Logic layer programs. ~ 8 hours

IV. Perform all system-level tests and then deploy the solution.
   1. Perform system functionality tests. ~ 5 hours
   2. Perform user acceptance test. ~ 8 hours

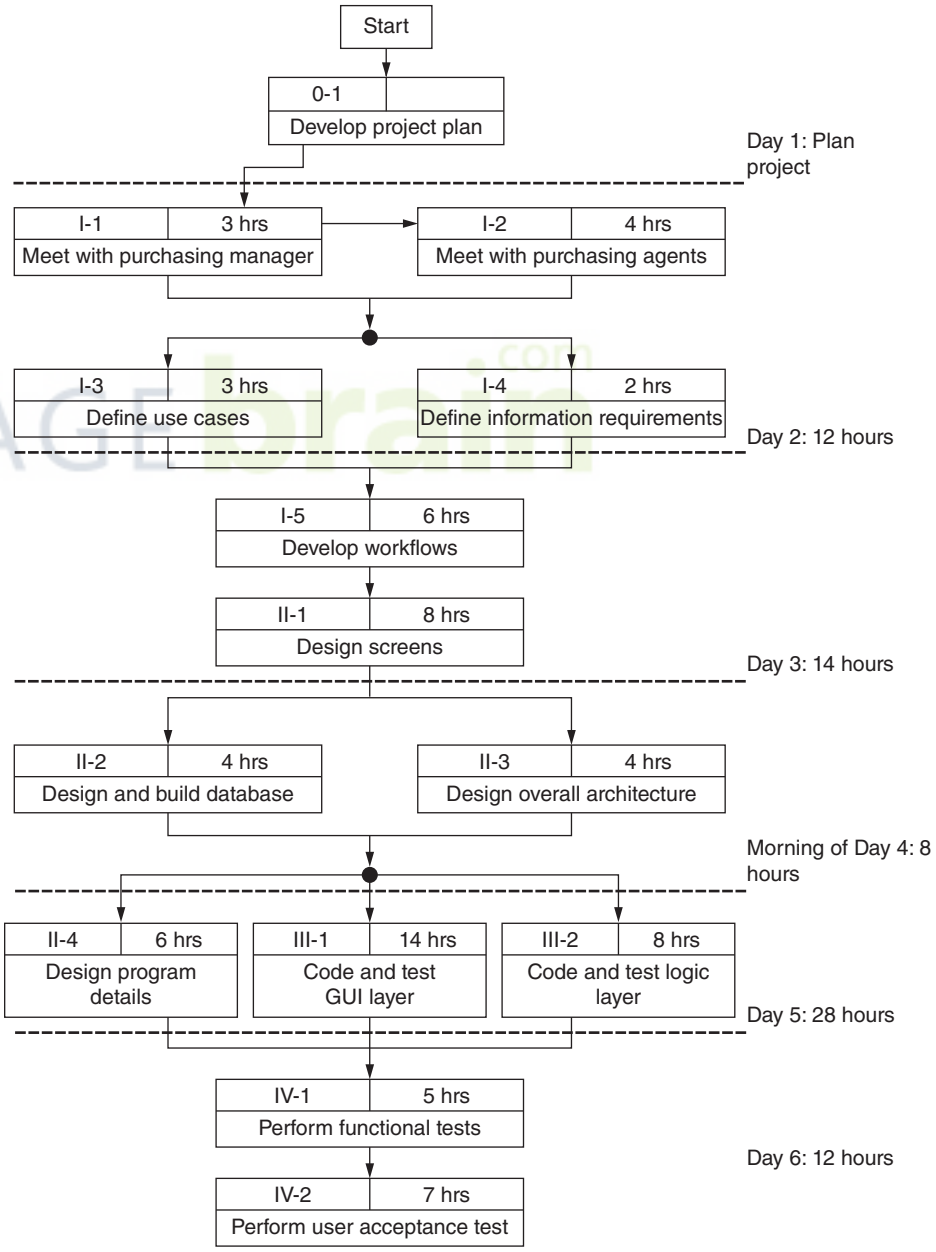
example, it does not make sense to try to design the database before we have identified the information requirements. But many tasks can be done in parallel.

Again, the great benefit of a single iteration is that we can make the schedule informal, and we will be able to adjust the work day by day to respond to specific complexities that occur.

For our project, we will not build a complete schedule. You will learn how to do that in a later chapter. However, in order to organize our six-day project, we have taken the tasks from the Work Breakdown Structure and placed them on a day-by-day sequence that we call a *work sequence draft*, as shown in **Figure 1-7**. To develop a formal schedule, the project leader will use this diagram to assign people to the tasks as well as put the tasks on a specific schedule chart with calendar dates.

You should be aware that the sequence of activities and the dependencies of those activities are represented in this diagram with only partial accuracy. For example, we show that programming does not start until design has finished. However, in reality, there may be some overlap between the two activities.

FIGURE 1-7
Work sequence draft



The benefit of a work sequence draft is threefold. First, it helps the team organize its work so there is enough time set aside to think through the critical design issues before programming begins. Second, it provides a measuring rod to see if the iteration is on schedule. For example, if meetings with the purchasing agents take all day or more than a day, the team will know early on that this iteration will take longer than expected. Third, the project leader can see that programming may require more resources if the project is going to stay on this schedule. Hence, the project leader can begin lining up resources early on to help with that part of the project. It should be obvious that even this simple dependency diagram can help a project manager plan and organize the work.

Day 2 Activities

Day 1 involved planning and organizing the project. Day 2 involves systems analysis activities that help us understand and document requirements. On Day 2, we will specify the functions in more detail. These activities are included:

- Do preliminary fact-finding tasks to understand the requirements. (Core Process 3)
- Develop a preliminary list of use cases and a use case diagram. (Core Process 3)
- Develop a preliminary list of classes and a class diagram. (Core Process 3)

Fact Finding and User Involvement

Before the project commenced, a preliminary, broad definition of functions was developed. It is now time to examine the specifics of those functions and determine exactly what the user needs the system to do. The first step is to identify the key users that will help define these details. Obviously, the manager of the purchasing department will be one of the first ones to meet with. She will probably designate one or two knowledgeable purchasing agents who can work with the team on an ongoing basis to develop the specifications and to verify that the system performs as required. All successful projects depend on heavy user involvement. In Chapter 2, you will learn more about identifying key stakeholders.

There are various techniques to ensure that the fact finding is complete and thorough. These include interviewing the key users, observing existing work processes, reviewing existing documentation and existing systems, and even researching other companies and other systems.

Identifying Use Cases

A use case documents a single user-triggered business event and the system's response to that event. For example, let us say a purchasing agent goes to a trade show and finds some new lightweight sports jackets that will work well for RMO's fall merchandise offerings. Maybe the first task the purchasing agent needs to do is find out if this supplier has worked with RMO before. Thus, the business event that requires the Tradeshow System might be "Look up a supplier." Activities leading up to the event of using the system are important, but we do not identify them as business events until the Tradeshow System is used—hence, the term *use case*—a case or situation where the system is used. One good way to help you identify use cases is to say, "The purchasing agent 'uses' the system to 'Look up a supplier.'"

There are multiple methods used to identify use cases, which you will learn about later in this book. **Figure 1-8** is a preliminary list of use cases for the entire Tradeshow System. When the project team meets with the purchasing agents in brainstorming sessions, they together identify every business event in which a purchasing agent might use the system. However, because this first iteration is focusing only on the Supplier Information subsystem, the project team will also focus its attention on only the first four use cases on the list.

FIGURE 1-8
List of use cases

Use Case	Description
Look up supplier	Using supplier name, find supplier information and contacts
Enter/update supplier information	Enter (new) or update (existing) supplier information
Look up contact	Using contact name, find contact information
Enter/update contact information	Enter (new) or update (existing) contact information
Look up product information	Using description or supplier name, look up product information
Enter/update product information	Enter (new) or update (existing) product information
Upload product image	Upload images of the merchandise product

Identifying Object Classes

Object classes identify those things in the real world that the system needs to know about and keep track of. In order to find object classes, we look for all the objects, or things, that the system uses or captures. Objects come in all types and variations, from tangible items (such as merchandise products that you can see and touch) to more abstract concepts that you cannot touch (such as an order), which, though intangible, definitely exist.

Object classes are identified in the discussions with purchasing agents by looking for the nouns that describe categories of things. For example, the agents will often talk about suppliers, merchandise products, or inventory items. More details about how to identify object classes and their attributes are provided later in this book.

Figure 1-9 illustrates which nouns have been determined to be fundamental object classes for the Tradeshow System. The attributes are descriptors that help define and describe an object class.

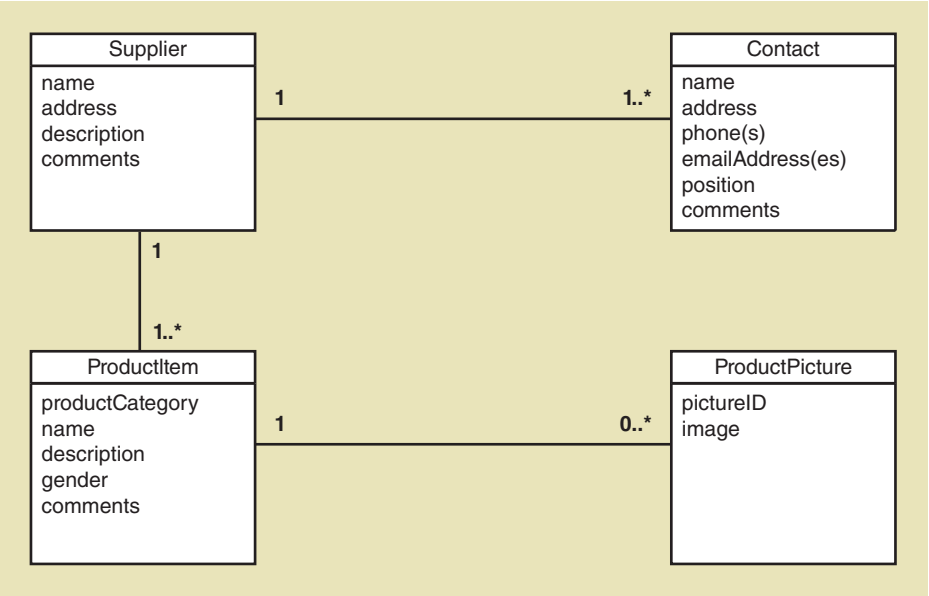
In addition to just providing a list of object classes, systems analysts often develop a visual diagram of the classes, their attributes, and their relationships to other classes. This diagram is called a class diagram. **Figure 1-10** illustrates a class diagram for the Tradeshow System.

Each box is a class and can be thought of as a particular set of objects that are important to the system. Important attributes of each class are also included in each box. These represent the detailed information about each object that will be maintained by the system. Note that some classes have lines between them. These represent relationships between the classes that need to be captured in the system. For example, a contact is a person who works for a particular supplier. A specific example might be that Bill Williams is the contact person for

FIGURE 1-9
List of object classes

Object Classes	Attributes
Supplier	supplier name, address, description, comments
Contact	name, address, phone(s), e-mail address(es), position, comments
Product	category, name, description, gender, comments
ProductPicture	ID, image

FIGURE 1-10
Preliminary class diagram for the Tradeshow System



the South Pacific Sportswear Company. Thus, the system needs to associate Bill Williams and the South Pacific Sportswear Company. The relationship line documents that requirement.

Class diagrams are a powerful and frequently used way to understand and document the information requirements of a system. The Tradeshow System is extremely simple, with only four classes identified—two of which belong to the Supplier Information subsystem. Most real-life systems are much larger and have dozens of classes.

Day 3 Activities

The purpose of Day 3 activities is to analyze in detail those use cases and classes that were selected to be implemented in this first iteration. During Day 3, we are still performing processes that are considered systems analysis. We are still trying to understand the requirements at a detailed level for the system. Included are these activities:

- Perform in-depth fact finding to understand details. (Core Process 3)
- Understand and document the detailed workflow of each use case. (Core Process 3)
- Define the user experience with screens and reports. (Core Processes 3 and 4)

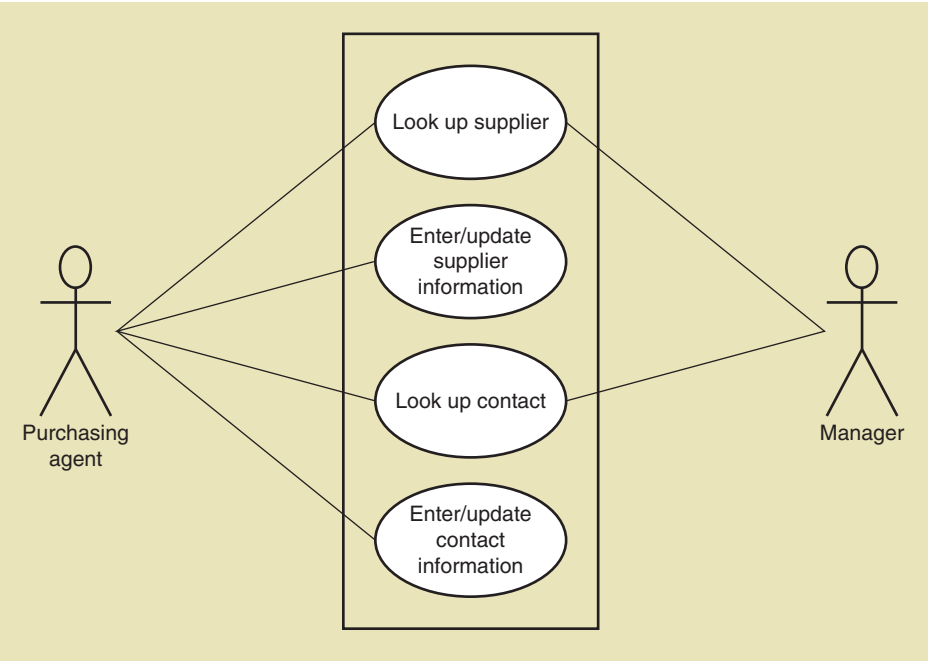
It is important to note that the use cases help the project team organize its work. These drill-down activities are done for each use case. As mentioned earlier, these use cases pertain to the Supplier Information subsystem:

- *Look up supplier.*
- *Enter/update supplier information.*
- *Look up contact information.*
- *Enter/update contact information.*

The project team will develop a workflow for each use case to better understand how it works and to identify what screens and possibly what reports will need to be developed. As the team gets more into the details, it may discover that some of the initial analysis is incomplete, if not incorrect. This is a good time to make such discoveries—much better than after the programs have been written.

Figure 1-11 illustrates a simple use case diagram. It shows the four aforementioned use cases identified and the user who will be the primary person

FIGURE 1-11
Use case diagram



performing that function. What the diagram means in detail and how to develop one will be discussed in a later chapter.

Developing Use Case Descriptions and Workflow Diagrams

There are various methods for documenting the details of a use case. One that you will learn later in this text is called a *use case description*. Another method is developing a workflow diagram, which shows all the steps within the use case. The purpose with either method is to document the interactions between the user and the system (i.e., how the user interacts and uses the system to carry out a specific work task for a single use case).

Let us develop a workflow for one use case. To develop a workflow, we will use a simple type of diagram called an activity diagram. **Figure 1-12** illustrates the workflow for the *Look up supplier* use case. The ovals in the diagram represent the tasks, the diamonds represent decision points, and the arrows represent the sequence of the flow. The columns represent who performs which tasks. Usually, workflow diagrams are quite easy to understand.

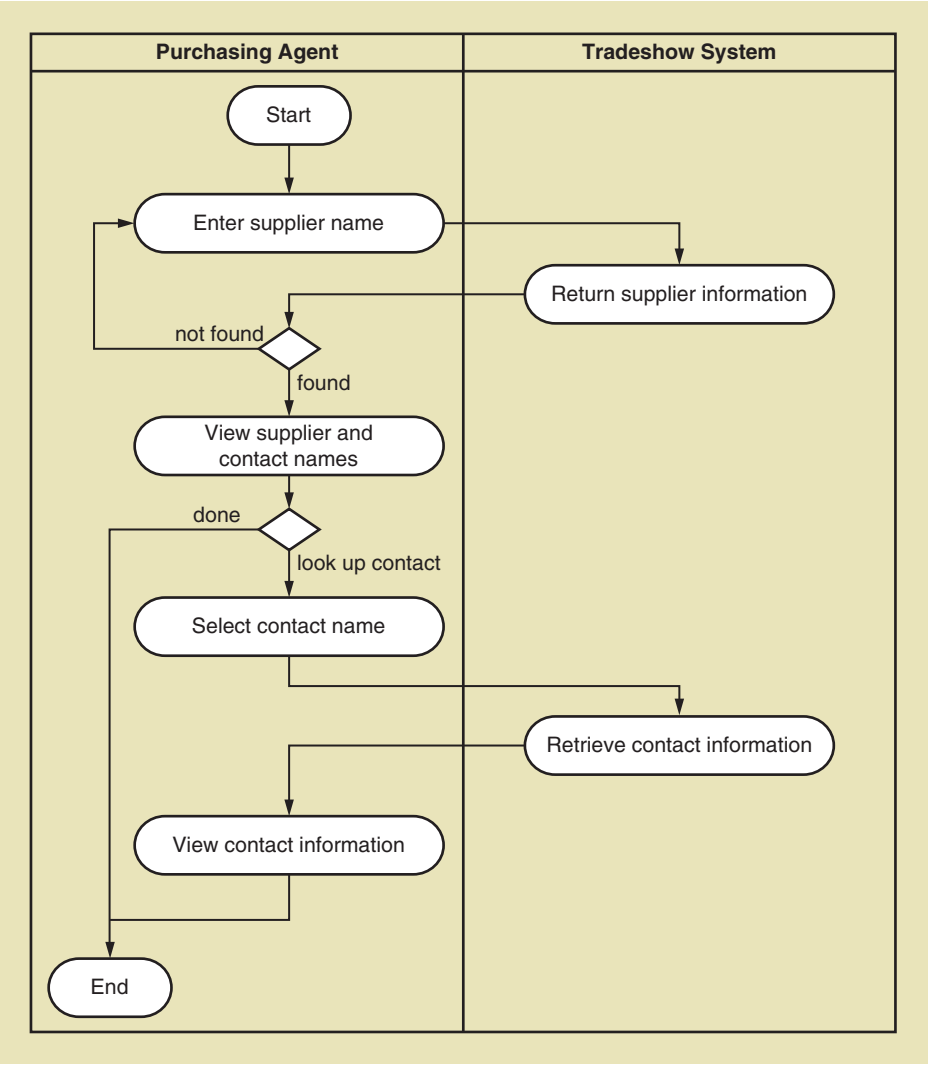
The arrows that cross the center line represent the interactions between the system and the user. These are critically important because the developers must provide a screen or Web page that either captures or displays information. The arrows that cross the center line identify the data elements that become part of the user interface.

Looking at Figure 1-12, we see that the top arrow indicates that the supplier name enters into the system. Thus, we infer that the user must have an online form in order to enter the supplier name for the initial lookup. The next arrow indicates that there must be a form that displays all the details for an individual supplier, including a list of existing contacts. The user may also want to see more details about a specific contact person for this supplier, so the user may request detailed information for a particular person. Because the user can select one of the displayed results, it appears that we must design the form so each entry on the list is either a hotlink or has some mechanism to select it.

Defining Screen Layout

User-interface design includes all those tasks that describe the look and feel of the system to the user. Because the user interface is the window that the users

FIGURE 1-12
Workflow diagram for the Look up supplier *use case*

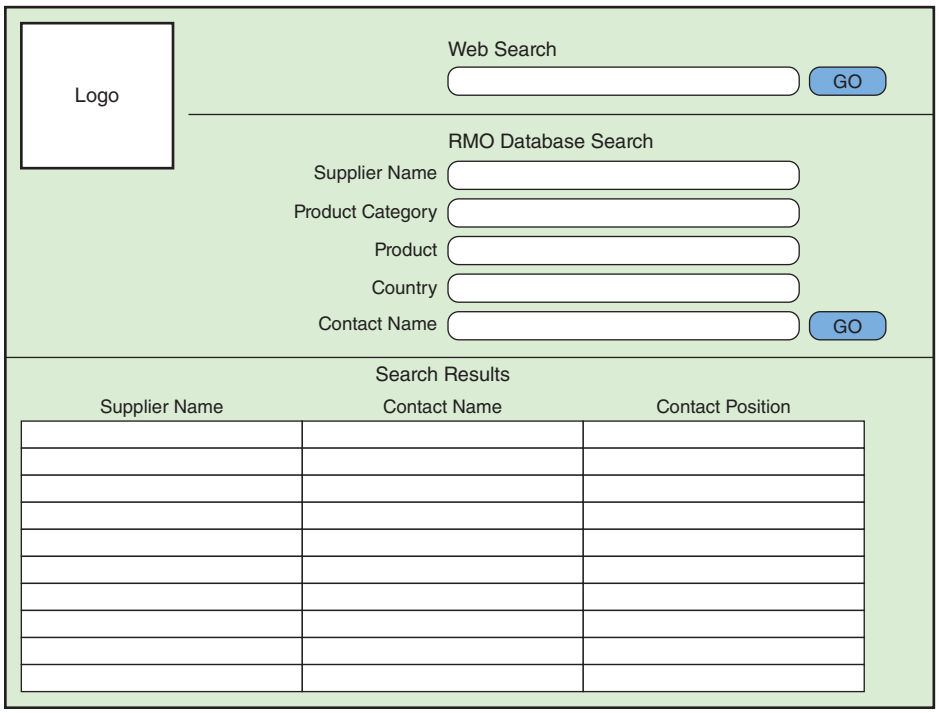


work with to utilize the functionality of the system, the user interface is essentially the system. If the interface is poorly designed, users will not be able to take full advantage of the system; they may even consider the system to be less than optimal. On the other hand, a well-designed user interface—one that is intuitive and easy to use, with a full range of features to facilitate navigation, and that provides good information—will enhance the utility of the system tremendously.

Figure 1-13 illustrates the layout of the first screen used for the workflow in the use case *Look up supplier*. The top portion of the screen provides the locations for the user to enter the supplier information, and the bottom portion of the screen shows the results. When results are provided, the search box for data entry will remain visible to allow the user to enter another search. Each entry in the results will be built as a hotlink, so the user can click on any particular supplier to retrieve more detailed information. This drill-down technique is a common method used in today's systems and will be intuitively easy for the users.

Searches are conducted on the RMO database, resulting in such RMO information as name, address, and contact information. An Internet-wide search is also possible. This allows the purchasing agents to look for and view the suppliers' own Web sites, which can be helpful, as can looking at forums and discussions about the supplier. Note that this tangential activity is not captured in the workflow of Figure 1-12. It is intended to assist in screen design, not document everything the user can or will do.

FIGURE 1-13
Draft of screen layout for the Look up supplier *use case*



Day 4 Activities

The primary focus of Day 4 activities is to design the various components of the solution system. Up to now, we have mostly been trying to understand the user requirements. On Day 4, we carry out design activities that direct programming efforts. In that sense, design activities can be considered somewhat of a bridge. During analysis activities, the project team's objective is to understand user needs. During programming, the objective is to produce the solution. Thus, design is the bridge between understanding and construction. It provides the outline for how the solution will be structured and how it is to be programmed. System design also tends to involve the technical people, with less need for user participation.

Design can be a complex process. In our small project, we will limit our design examples to only a few models and techniques. Later in this textbook, you will learn additional design techniques. Day 4 Activities include the following:

- Design the database structure (schema). (Core Process 4)
- Design the system's high-level structure. (Core Process 4)

Database design is a fairly straightforward activity that uses the class diagram as input and develops the detailed database schema that can be directly implemented by a database management system. Such elements as table design, key and index identification, attribute types, and other efficiency decisions are made during this activity.

Designing the high-level system structure and the individual programs can be an intricate and complex process. First, the overall structure of the system is designed, including identifying the subsystems and connections to other systems. Within each subsystem, decisions are made about individual programs, such as user-interface programs, business logic programs, and database access programs. Then, at the lowest level, the login within each program is defined, including what program functions are required and what variables are used.

It is not uncommon for developers to begin writing program code as they develop portions of the design. It is a good idea to complete most of the

structural design before writing code. But as the lower levels of the system are being designed, programming often begins. However, in the RMO Tradeshow System project, we will list them as separate activities.

Designing the Database

Designing the database uses the information provided by the class diagram to determine the tables, the columns in the tables, and other components. Sometimes, the database design is done for the entire system or subsystem. At other times, it is built piecemeal—use case by use case. To keep our project simple, we will just show the database design for the two classes that are required for the Supplier Information subsystem. **Figure 1-14** shows the database schema for the Supplier Information subsystem. Two tables are defined: Supplier and Contact.

Approaching High-Level Systems Design

There are fundamental design principles that will guide you through systems and program design; they will be explained in detail later in this book. For now, we will describe the general approach to design.

One of the first questions encountered in systems design is how and where to start. So far, we have three types of documents that can provide specifications to help answer that question. We have use cases, with their accompanying documentation, such as use case workflow diagrams. We have a class diagram that will help us identify some of the object-oriented classes that will be needed in the system. (In the previous section, we used the class diagram as the basis for the database design. Those same classes are important in developing object-oriented program classes.) Finally, we have screens and reports that also provide specifications for program logic and display logic.

Before we jump into design, let us briefly discuss the objective of systems design and what we expect to have as the output or result. Object-oriented programs are structured as a set of interacting classes. Therefore, in order to program, we need to know what those programming classes are, what the logic is within each class (i.e., the functions), and which programming classes must interact together. That is the final objective of systems design: to define the classes, the methods within those classes, and the interactions between classes.

FIGURE 1-14
Database schema for Supplier Information subsystem

Table Name	Attributes
Supplier	SupplierID: integer {key} Name: string {index} Address1: string Address1: string City: string State-province: string Postal-code: string Country: string SupplierWebURL: string Comments: string
Contact	ContactID: integer {key} SupplierID: integer {foreign key} Name: string {index} Title: string WorkAddress1: string WorkAddress2: string WorkCity: string WorkState: string WorkPostal-code: string WorkCountry: string WorkPhone: string MobilePhone: string EmailAddress1: string EmailAddress2: string Comments: string

We perform this design by starting at the very highest level and then drilling down to the lowest level until we have defined all the functions within each class. Detailed design is the thought process of how to program each use case. Later in the text, you will learn techniques to carry out detailed design. For Day 4, we will focus only on the overall design.

Designing the Overall Architecture

Figure 1-15 shows the overall architecture or structure of the new system. Although the figure itself appears rather simple, some important decisions have been involved in the development of this design. First, note that the decision was made to build this application as a browser-based system. A different and very popular approach would have been to build smartphone or tablet applications. Browser-based systems sometimes do not provide the same connectivity speed and control as smartphone or tablet applications, but they are more versatile in that they can be more easily deployed on different equipment, such as laptops, without modification.

These high-level design decisions will determine the detailed structure of the system. A browser-based system is structured and constructed differently than an application system that runs on a smartphone or a tablet computer.

Defining the Preliminary Design Class Diagram

Given that the Tradeshow System will be built by using object-oriented programming (OOP) techniques, an important component of the design is developing the set of object classes and functions that will be needed for the system. This process can become quite detailed, and we will not try to explain it for this project. You will learn those techniques later in this book.

Figure 1-16 is a preliminary design class diagram for the Tradeshow System. A design class diagram (DCD) identifies the OOP classes that will be needed for the system. The set of design classes includes problem domain classes, view layer classes, sometimes separate data access classes, and utility classes. In Figure 1-16, we show only the problem domain classes and the view layer classes. Problem domain classes are usually derived from those classes that were identified during analysis activities—hence, the name: problem (user need) domain classes. You will also notice that they very closely correspond to the

FIGURE 1-15
Tradeshow System architectural configuration diagram

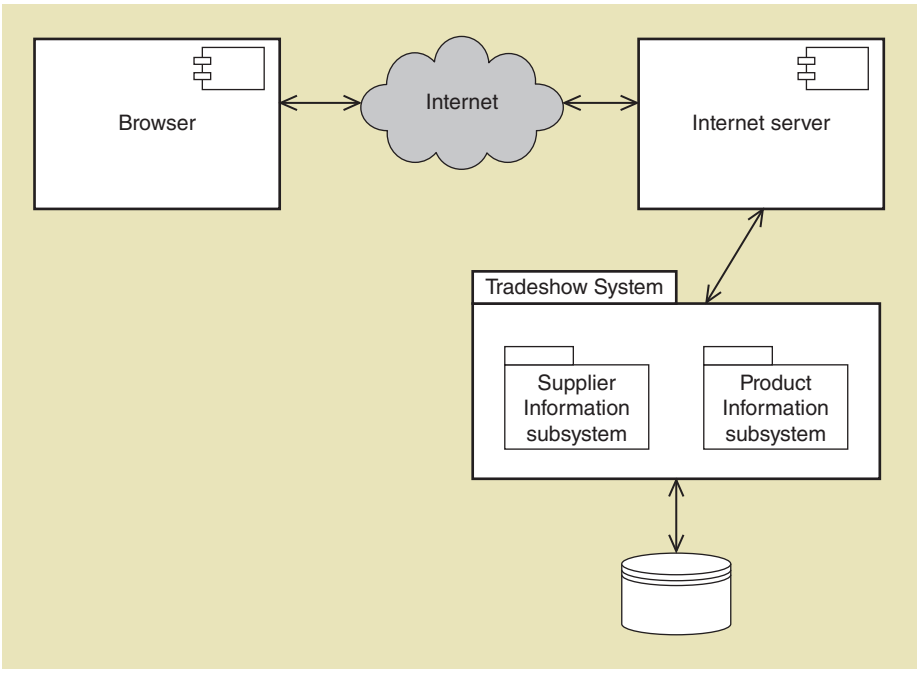
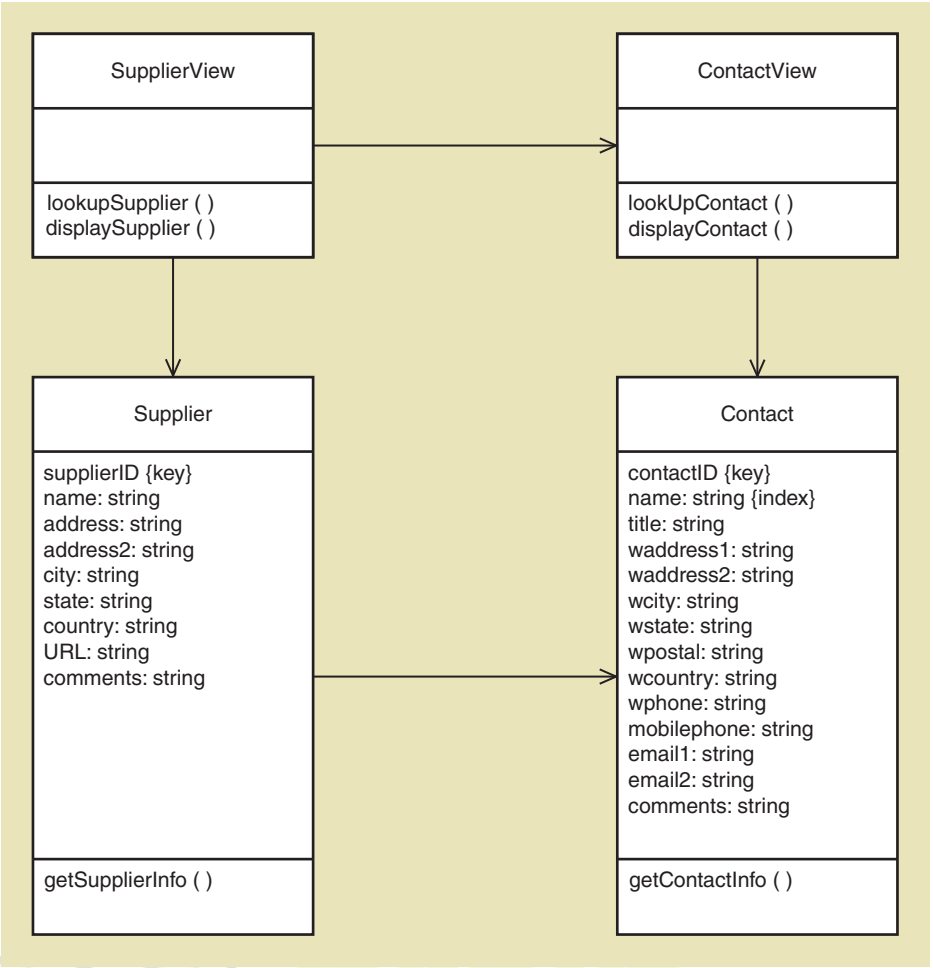


FIGURE 1-16
Preliminary design class diagram



database tables; in fact, in this simple project, they are almost exactly the same as the database tables. On more complex systems, they will be similar but not exactly the same. However, remember that programming classes are distinct from database tables.

Other classes are required for the graphical user interface (GUI). In a dynamic Web system, such as the Tradeshow System, they are the classes that receive the input from the browser and format the output HTML files to be displayed by the browser.

The design classes in Figure 1-16 include the class-level variables that are needed for the class. These classes also show method names of the important methods within each class. These methods are identified and specified during high-level design and detailed design. One final element in the design class diagram are arrows that show which classes can access the methods of which other classes.

Designing Subsystem Architecture

Once we have an overall structure and an overall approach for implementing the new system, we begin to drill down to the subsystem design. **Figure 1-17** illustrates the architectural design of the Supplier Information subsystem. Notice that this subsystem is further divided into layers: a view layer and a model layer. You will learn much more about multilevel design later in this textbook. One of the advantages of partitioning the system into layers is that the system is much easier to build and maintain with this kind of structure. For example, the system will be browser based, but different browsers require different techniques. It is better not to get these complexities mixed in with the basic program functions. Hence, they are separated out into a distinct layer.

FIGURE 1-17
Supplier subsystem architectural design diagram

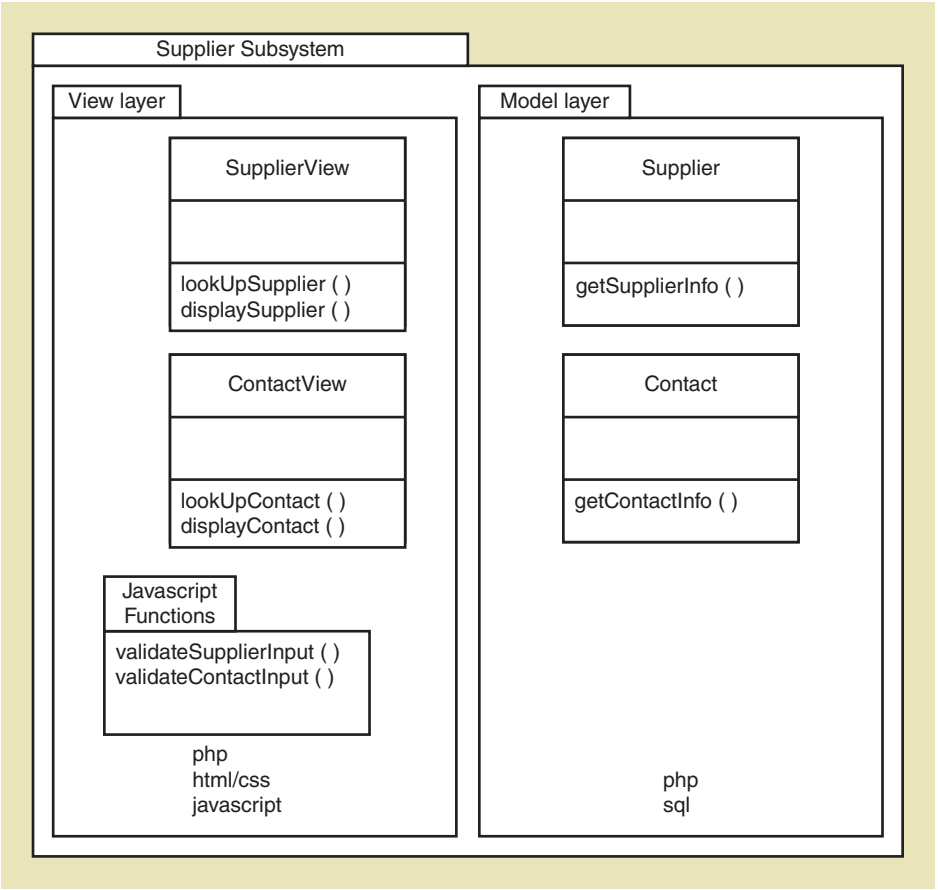


Figure 1-17 shows that the view layer has two PHP classes that process user inputs from the browser and format output HTML files. It also contains various JavaScript functions that will execute within the browser itself. The model layer classes are those classes that perform the business logic as well as access the database. Sometimes, the data layer and the business logic layer are further divided.

Managing the Project

Design is a complex activity with multiple levels—from high-level structural design to low-level detailed program design. In our project, we have separated the tasks for designing the overall system structure from detailed design of the programs themselves. However, these activities are often done concurrently. The basic high-level architectural structure is defined first, but mid-level and low-level design are often done concurrently with programming.

In Figure 1-17, we can see that detailed design and programming are quite time-consuming activities. A project manager must decide whether to extend the project or bring on additional programmers to help write the code. In our project, we have elected to insert a half-day of free time to bring in two additional programmers and train them. Of course, we could go ahead and begin Day 5's activities to ensure that we keep the project on schedule.

Day 5 Activities

Even though detailed design and programming may frequently begin earlier in the project, we have identified it as a separate day's activities. We do this for a couple of very important reasons. First, we want to emphasize that it is not a good practice to begin programming before critical information is obtained and decisions are made. Often, novice programmers will begin to program before

the users' needs are adequately understood or even before the structure of the overall system has been determined. But a much better approach is to understand, design, and build small chunks of the system at a time. Agile Development anticipates and plans for the expected changes and refinements to the problem requirements that happen during detailed design and programming.

As the programmers write the code, they also perform individual testing on the classes and functions they program. This textbook does not focus on programming activities. However, we include an example of program code so you can see how systems design relates to the final program code. **Figure 1-18** is an example of a class that receives and processes the request for supplier information.

Day 6 Activities

The focus of Day 6 activities is to do the final testing that is required before the system is ready to be deployed. There are many types of testing that are required. In this example, we mention only two types of testing: overall system functional testing and user acceptance testing. Functional testing is usually a system-level test of all user functions and is often done by a quality assurance team. User acceptance tests are similar in nature, but they are done by the users, who test not only the correctness of the system but its "fitness" to accomplish the business requirements.

Each of the various testing activities in Day 6 has a somewhat similar sequence of tasks to perform. The tasks themselves highly depend on the test data and on the method for testing a particular test case. In some instances, the testing may be automated. In others, individuals may need to manually conduct the tests. Many new systems are interactive systems with user activity. There are some testing tools that automate that process somewhat, but it tends to be a rather complex task.

FIGURE 1-18
Code for the SupplierView class

```
<?php
    class SupplierView
    {
        private Supplier $theSupplier;

        function __construct()
        {
            $this->theSupplier = new Supplier();
        }

        function lookupSupplier()
        {
          include('lookupSupplier.inc.html');
        }

        function displaySupplier()
        {
              include('displaySupplierTop.inc.html');
              extract($_REQUEST);   // get Form data
            //Call Supplier class to retrieve the data
            $results = $theSupplier->getSupplierInfo($supplier, $category,
                                          $product, $country, $contact);

           foreach ($results as $resultItem){
           ?>
                    <tr>
                        <td style="border:1px solid black">
                              <?php echo $resultItem->supplierName?></td>
                        <td style="border:1px solid black">
                               <?php echo $resultItem->contactName?></td>
                        <td style="border:1px solid black">
                           <?php echo $resultItem->contactPosition?></td>
                    </tr>
            <?php }
            include('displaySupplierFoot.inc.html');
        }
    }
?>
```

FIGURE 1-19
Generalized workflow of testing tasks

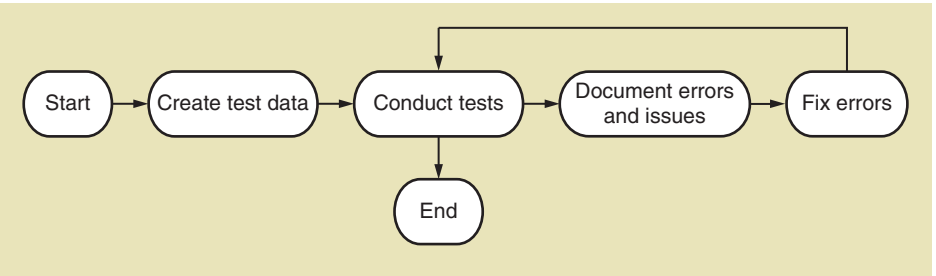


Figure 1-19 is a generalized workflow for testing the new system. In this workflow, we have shown the different testing tasks as separate steps. In reality, they all tend to be carried out together. However, any given test case will follow this flow.

First Iteration Recap

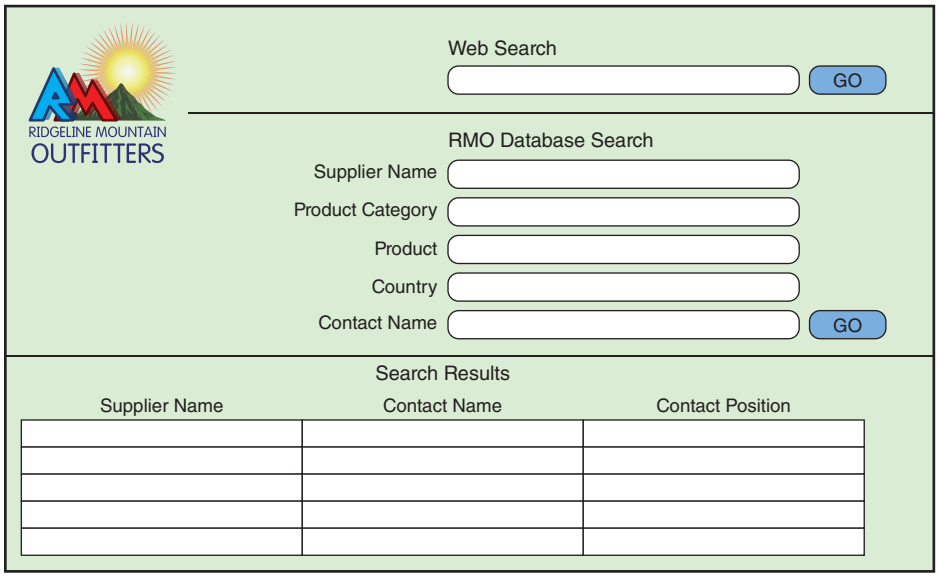
Figure 1-20 is a screenshot of the browser page that is used in the Tradeshow System to enter and view suppliers.

As stated previously, this is the first (six-day) iteration of a longer project. Using Agile techniques and iterations within an overall project allows flexibility in defining and building a new system. One of the Agile mandates is that the user should be heavily involved in the development of the new system. In this six-day project, the users have had major involvement during all days except Day 4 and Day 5.

A primary problem in developing a new system is that as the project progresses, new requirements are often identified. This happens because the users and the project team learn more about how to solve the business need. Agile, iterative projects are structured to handle these new requirements—often by adding another iteration to the overall project.

As a final step in a current iteration, or perhaps as part of the planning process for the next iteration, there should be a review of the processes and success of the current iteration. The lessons learned and issues to be carried forward create an environment of continual improvement and refinement. Iterative projects tend to improve and become more efficient during the life of the project.

FIGURE 1-20
Screen capture for Look up supplier *use case*



Where You Are Headed—The Rest of This Book

This sixth edition of *Systems Analysis and Design in a Changing World* includes the printed textbook and supporting online chapters. The current printed textbook provides a compact, streamlined, and focused presentation of those topics that are essential for information systems developers. The online chapters extend those concepts and provide a broader presentation of several topics. The online chapters may be integrated into the course or simply used as additional reading as prescribed by the instructor.

Part 1: Introduction to Systems Development

Part 1—comprising Chapter 1 and Online Chapter A—presents an overview of systems development. The introductory chapter provides a detailed, concrete example of what is required in a typical software development project. Of course, many details had to be left out to keep this chapter at a reasonable length. However, this chapter does present many processes, techniques, and diagrams. You are not expected to understand all the elements from this brief introduction. However, you should have a general idea of the approach to developing systems. You may want to refer back to this chapter from time to time to help understand the big picture.

This first chapter begins by briefly explaining the objectives of systems analysis and systems design. Many students who take a programming class think that programming is all you need to develop software and deploy a system. This chapter and the rest of this book should dispel that myth.

Online Chapter A, "The Role of the Systems Analyst," describes the many skills required of a systems analyst. It also discusses the various career options available to information systems majors. For those of you who are new to the discipline of information systems, this chapter will provide interesting and helpful knowledge about information systems careers.

There are three major subject areas discussed in this book: systems analysis, systems design, and project management. There is also one minor subject area, which is no less important but not discussed in as much depth, and that is systems implementation, testing, and deployment. In addition, we have taken an approach that is quite different from other texts. Because you already have a basic understanding of systems analysis and design from Chapter 1, we can immediately present in-depth concepts about systems analysis and design. We present project management topics later in this text. This will allow you to learn those project management concepts after understanding the elements of systems analysis and design. We think it will be more meaningful for you at that point in the course.

Part 2: Systems Analysis Tasks

Chapters 2 through 5 cover systems analysis in detail. Chapter 2 discusses techniques for gathering information about the business problem. Developing the right system solution is possible only if the problem is accurately understood. The various people who are affected by the system (the stakeholders) are also included in the development of the solution. Chapter 2 also explains how to identify and involve the stakeholders and introduces the concept of models and modeling. Chapters 3 and 4 present methods for capturing the detailed requirements for the system in a useful form. When discussing an information system, two key concepts are particularly useful: use cases, which define what the end users need the system to do, and the things, called data entities or classes, that users work with while carrying out their tasks. These two concepts—use cases and data entities/classes—are important no matter what approach to system

development is being used. Chapter 5 presents more in-depth models, such as use case descriptions, use case diagrams, and system sequence diagrams.

Online Chapter B, "The Traditional Approach to Requirements," presents the traditional, structured approach to developing systems. To those instructors and students who desire to learn about data flow diagrams, this chapter provides an in-depth presentation.

All these modeling techniques provide in-depth analysis of the user's needs and allow the analyst to develop requirements and specifications. Again, the purpose of systems analysis is to thoroughly understand and specify the user's needs and requirements.

Part 3: Essentials of Systems Design

Chapters 6 and 7 provide the fundamental concepts related to systems design and to defining and designing the user experience. Chapter 6 provides broad and comprehensive coverage of important principles of systems design. It serves not only as a broad overview of design principles but as a foundation for later chapters that explain the detailed techniques, tasks, skills, and models used to carry out design.

Chapter 7 presents additional design principles related to designing the user interface and the system interfaces. Designing the user interface is a combination of analysis and design. It is related to analysis because it requires heavy user involvement and includes specifying user activities and desires. On the other hand, it is a design activity because it is creating specific final components that are used to drive the programming effort. The screens and reports and other user interaction components must be precisely designed so they can be programmed as part of the final system. System interfaces occur when one information system communicates or interacts with another information system without human intervention. System interfaces are becoming increasingly important because of Web services and cloud computing.

Part 4: Projects and Project Management

By this point, you will have a basic understanding of all the elements of systems development.

Part 4 brings together all these concepts by explaining more about the process of organizing and managing development projects. Chapter 8 describes different approaches to systems development in today's environment, including several important System Development Life Cycle models. It is an important chapter to help you understand how projects actually get executed.

Chapter 9 extends these concepts by teaching foundation principles of project management. Every systems analyst is involved in helping organize, coordinate, and manage software development projects. In addition, almost all of you will become team leaders and project managers. The principles presented in Chapter 9 are essential to a successful career.

Online Chapter C, "Project Management Techniques," goes into more detail regarding the tools and techniques used by systems analysts and project managers to plan and monitor development projects. For those instructors and students who would like to learn specific project management skills, this is an important chapter.

Part 5: Advanced Design and Deployment Concepts

Part 5 goes into more depth with respect to systems design, database design, and other important issues related to effective and successful system development and deployment.

Chapters 10 and 11 explain in detail the models, skills, and techniques used to design software systems. As mentioned earlier in this chapter, systems design is a fairly complex activity, especially if it is done correctly. The objective of

these two chapters is to teach you the various techniques—from simple to complex—that can be used to effectively design software systems.

Chapter 12 explains how to design the database from the information gleaned during analysis and the identification of the object classes. Other related concepts, such as controls and security, are also presented in this chapter. Chapter 13 describes the final elements in systems development: final testing, deployment, maintenance, and version control. Chapter 14 concludes this book by looking toward the future of software development and assessing the current trends that may eventually enhance and improve the approaches to software development.

Chapter Summary

This chapter provided a quick overview of a particular software system development project called the Tradeshow System. The six core processes that control software development were presented. Then, the various activities that support the execution of these six core processes were explained as we went through an implementation of the Tradeshow System. The six core processes are:

1. Identify the problem or need and obtain approval to proceed.
2. Plan and monitor the project—what to do, how to do it, and who does it.
3. Discover and understand the details of the problem or the need.
4. Design the system components that solve the problem or satisfy the need.
5. Build, test, and integrate system components.
6. Complete system tests and then deploy the solution.

In order to facilitate learning and to help you remember the core processes and related activities, we divided the project into pre-project activities and then six other groupings, which we called *project days*. It should be noted that there is nothing magical or mandatory about this way of organizing a project. It was done in this manner simply to help you understand the various activities related to the core processes.

Key Terms

Agile Development 6
computer application (app) 4
information system 4
information systems development process 6
iterative development 8
project 5
subsystem 11
systems analysis 5
systems design 5
Systems Development Life Cycle (SDLC) 6

Review Questions

1. What is the difference between an information system and a computer application?
2. What is the purpose of systems analysis? Why is it important?
3. What is the difference between systems analysis and systems design?
4. What is a project?
5. What are the six core processes for software systems development?
6. What is meant by Agile Development?
7. What is the purpose of a System Vision Document?
8. What is the difference between a system and a subsystem?
9. What is the purpose of a Work Breakdown Structure?
10. What are the components of a Work Breakdown Structure? What does it show?
11. What information is provided by use cases or a use case diagram?
12. What information is provided by a class diagram?
13. How do a use case diagram and a class diagram drive the system development process?

14. What is another way to describe an activity diagram? What does it show?
15. How does an activity diagram help in user-interface design?
16. What is the purpose of architectural design?
17. What new information is provided in a design class diagram (more than a class diagram)?
18. What are the steps of system testing?
19. What is the purpose of user acceptance testing?
20. Why is it a good practice to divide a project into separate iterations?
21. What should be the objective or result of an iteration?

CHAPTER CASE

Keeping Track of Your Geocaching Outings

When Wayne Johansen turned 16, his dad bought him a new Garmin handheld GPS system. His family had always enjoyed camping and hiking, and Wayne was usually the member of the family who monitored their hikes with his dad's GPS system. He always liked to carry the GPS because he really enjoyed monitoring the routes, distances, and altitudes of their hikes. More recently, though, he had found a new hobby by using his GPS system: geocaching.

Geocaching is akin to the treasure hunts that most of us did when we were kids. The difference is that geocaching is a high-tech version of a treasure hunt that uses GPS but also calls on one's basic treasure-hunting skills.

As Wayne became more involved with his hobby, he discovered that there are many different kinds of activities for geocaching enthusiasts. The simplest ones are those that involve caches that can be found by using GPS coordinates, although even some of these can be difficult if the caches are well hidden and well camouflaged. Some of the activities involve multipoint drops in which there is a set of clues at multiple locations that must be followed in order to arrive at the final cache point. Some activities involve puzzles that must be solved in order to determine the coordinates and location of the final cache.

Before long, Wayne wanted to make his own caches and post them for people to find. He discovered that there were several Web sites, including individuals' blogs, with geocaching information, caches, and memberships. He joined one of the geocaching Web sites and used it to log his finds. It was fun to log and publish his finds and to post the geocaches that he created. But he decided he would like his own little system for keeping track of all the information he wanted to maintain about his caches. Conveniently, Wayne's older brother Nick, a college student majoring in information systems, was looking for a semester project for one of his programming classes. The two of them decided to work together and develop a system to help Wayne keep track of all his geocaching activities.

In this end-of-chapter case, you will go through the various core processes of an SDLC and perform some of the activities of a development project. Of course, this is a very small project with very limited requirements. The project and various assignments are divided into days, as was our Tradeshow project. You have not learned all the skills required to effectively produce all the documents illustrated in the chapter. Hence, the daily assignments for this case should be considered as preliminary efforts and rough drafts. The objective of these assignments is simply to help you remember the overall approach to software development. Several assignments have been listed for each day to allow your instructor to select those that best meet the objectives of the course.

Day 0: Define the Vision

The primary purpose of pre-project activities is to define a vision for the new system. Either by yourself or with another class member, brainstorm all the neat functions this geocaching system might do. Keep it at a very high level. You just want to think of the major functions that Wayne might want the system to do for him. These activities closely relate to Core Process 1: Identify the problem or need and obtain approval to proceed.

Assignment D0-1: Write a rough draft of the System Vision Document based on your brainstorming ideas. [Hint: Think of what Wayne wants the system to do and why this is a benefit to him.]

Day 1: Plan the Project

Obviously, this is a small project and does not require an elaborate project plan. Based on the scope and vision you described in the System Vision Document, divide the project into at least two separate subsystems or

(continued on page 30)

(continued from page 29)

sets of functions that can be done in two different iterations. For example, perhaps a first version can run on a laptop, with a second version that includes mobile components for a smartphone. Or perhaps the first version can maintain a history of past geocache hunts, and the second version can allow Wayne to record geocaches that he creates. Or the first version can create a simple database system, and the second version can enhance it by linking to photo albums and/or blog posts. As you can see, there are often many different ways that a project can be partitioned. These activities are related to Core Process 2: Plan and monitor the project—what to do, how to do it, and who does it.

Assignment D1-1: Divide the system into at least two separate components or subsystems, which can be supported with two iterations. Briefly describe each.

Assignment D1-2: Create a Work Breakdown Structure that lists all the steps to complete the first iteration. Put a time estimate on each step. [Hint: Use the one in this chapter as a model.]

Day 2: Define and Understand the Requirements

We often describe the activities on Day 2 and Day 3 with the word *understanding*. On Day 2, we want to get an overall view of what the system needs to do for Wayne. As you learned in this chapter, there are two primary areas we focus on to obtain this high-level understanding of the system: a list of use cases and a list of object classes. We could document this information in lists, but diagrams provide a visual representation that is often easier to remember and understand. These activities support Core Process 3: Discover and understand the details of the problem or the need.

Assignment D2-1: Identify a few use cases that apply to one subsystem. [Hint: Think of what Wayne plans to do with the system. He will use the system to "do what"?]

Assignment D2-2: Try to identify the classes that apply to the first project iteration. [Hint: Think of "information things" that Wayne wants the system to "remember."]

Assignment D2-3: Create a simple use case diagram from the list of use cases. [Hint: Drawing by hand is fine. Use the one in this chapter as a model.]

Assignment D2-4: Create a simple class diagram from the list of classes. [Hint: Drawing by hand is fine. Use the one in this chapter as a model. Think of some other pieces of information that apply to each class.]

Day 3: Define the User Experience

These activities are a continuation of what we began in Day 2. The objective here is to further understand what Wayne will need and how he will actually use the system. We will determine exactly how each use case works—what steps and options are available with the use case and even what the display and data entry screens will look like. Unfortunately, this often requires a lot of work. For this case, though, let us keep it simple. These activities primarily support Core Process 3: Discover and understand the details of the problem or the need.

Assignment D3-1: Select a single use case and then identify the individual steps required to perform the use case. [Hint: Think of what Wayne does and how the system responds.]

Assignment D3-2: Make a workflow diagram of the selected use case. [Hint: Drawing by hand is fine. Each step from D3-1 goes in an oval. Connect the ovals with arrows.]

Assignment D3-3: Sketch out one of the screens that will be required to support a use case. The screen should allow for data entry and display of information. [Hint: Don't make it elaborate. Focus only on the input and output data fields that apply to only one use case.]

Day 4: Develop the Architectural Design

The high-level architectural design of the system generally includes decisions about how the system will be built and what the database will look like. Design is a technical activity that requires experience in programming, database development, and system architecture. These activities support Core Process 4: Design the system components that solve the problem or satisfy the need.

Assignment D4-1: Design a preliminary database schema for the classes in this iteration. [Hint: Each class becomes a table. The attributes become table columns.]

Assignment D4-2: Decide whether you will build a desktop system or a browser-based system. Write a couple of paragraphs listing the pros and cons of each alternative in order to defend your decision. [Hint: Either option is valid. Think of reasons to support your decision.]

Day 5: Develop the Detailed Design and Program the System

You have learned how to do these activities in your programming classes. You probably have had many class projects where you designed a system and then

(continued on page 31)

(continued from page 30)

programmed it. These kind of activities support Core Process 5: Build, test, and integrate system components.

Assignment D5-1: Write a paragraph describing what programming language(s) you would recommend and what development environment you prefer. For this answer, draw on your previous programming and development experiences. [Hint: There are many valid solutions. Give reasons for your preference.]

Day 6: Test and Deploy the System

You may have had opportunities to perform comprehensive testing of your programming class projects, especially if you have developed systems that integrated with other systems. These activities support Core Process 6: Complete system tests and then deploy the solution. Obviously, you can only do this if you have programmed the system.

Assignment D6-1: Write a paragraph describing the difference between programmer testing and user testing. [Hint: Why is it hard to test your own work? What do the users know that you don't know?]

Assignment D6-2: Write a paragraph describing all the issues that might need to be addressed to deploy this system. [Hint: You might want to search the Internet to learn about deployment issues.]

Assignment D6-3: Look at *www.geocaching.com*, which is a commercial Web site. What other issues need to be addressed to deploy this type of Web site? [Hint: Think about all the issues related to security, robustness, financial protection, high volumes, up-time, different browsers, and so forth.]