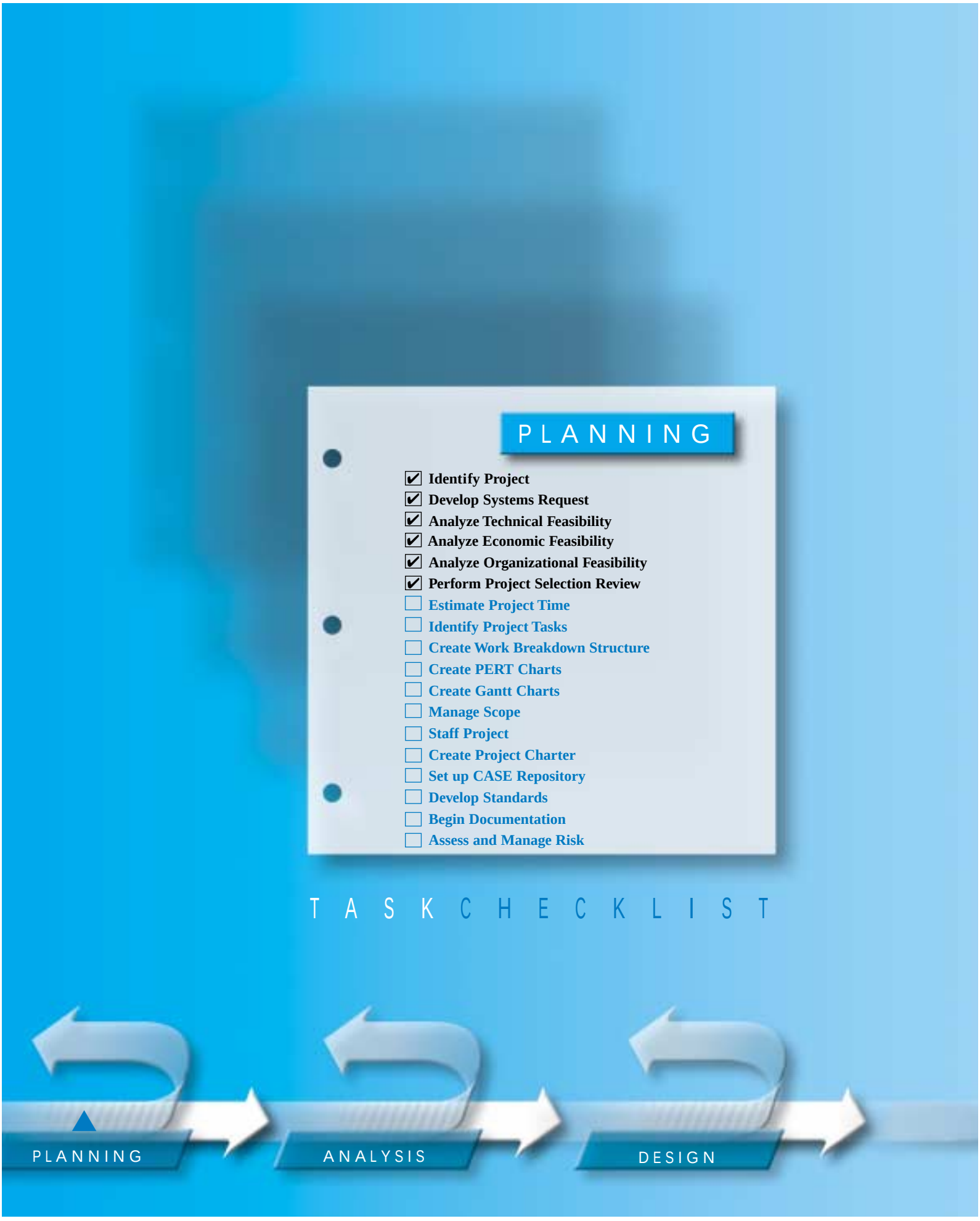

CHAPTER 3

PROJECT MANAGEMENT

This chapter describes the important steps of project management, which begins in the Planning Phase and continues throughout the systems development life cycle (SDLC). First, the project manager estimates the size of the project and identifies the tasks that need to be performed. Next, he or she staffs the project and puts several activities in place to help coordinate project activities. These steps produce important project management deliverables, including the workplan, staffing plan, and standards list.

OBJECTIVES

- Become familiar with estimation.
- Be able to create a project workplan.
- Understand why project teams use timeboxing.
- Become familiar with how to staff a project.
- Understand how computer-aided software engineering, standards, and documentation improve the efficiency of a project.
- Understand how to reduce risk on a project.

CHAPTER OUTLINE

INTRODUCTION

Think about major projects that occur in people's lives, such as throwing a big party like a wedding or graduation celebration. Months are spent in advance identifying and performing all of the tasks that need to get done, such as sending out invitations and selecting a menu, and time and money are carefully allocated among them. Along the way, decisions are recorded, problems are addressed, and changes are made. The increasing popularity of the party planner, a person whose sole job is to coordinate a party, suggests how tough this job can be. In the end, the success of any party has a lot to do with the effort that went into planning along the way. System development projects can be much more complicated than the projects we encounter in our personal lives—usually, more people are involved (e.g., the organization), the costs are higher, and more tasks need to be completed. Therefore, you should not be surprised to learn that "party planners" exist for information system projects—they are called project managers.

Project management is the process of planning and controlling the development of a system within a specified time frame at a minimum cost with the right functionality.[1] A *project manager* has the primary responsibility for managing the hundreds of tasks and roles that need to be carefully coordinated. Nowadays, project management is an actual profession, and analysts spend years working on projects prior to tackling the management of them. In a 1999 *Computerworld* survey, more than half of 103 companies polled said they now offer formal project management training for IT project teams. There also is a variety of *project management software* available like Microsoft Project, Plan View, and PMOffice that support project management activities.

Although training and software are available to help project managers, unreasonable demands set by project sponsors and business managers can make project management very difficult. Too often, the approach of the holiday season, the chance at winning a proposal with a low bid, or a funding opportunity pressures project managers to promise systems long before they are able to deliver them. These overly optimistic timetables are thought to be one of the biggest problems that projects face; instead of pushing a project forward faster, they result in delays.

Thus, a critical success factor for project management is to start with a realistic assessment of the work that needs to be accomplished and then manage the project according to that assessment. This can be achieved by carefully following the four steps that are presented in this chapter: identifying the project size, creating and managing the workplan, staffing the project, and coordinating project activities. The project manager ultimately creates a workplan, staffing plan, standards list, and risk assessment, which are used and refined throughout the entire SDLC.

IDENTIFYING PROJECT SIZE

The science (or art) of project management is in making *trade-offs* among three important concepts: the size of the system (in terms of what it does), the time to complete the project (when the project will be finished), and the cost of the project.

[1] A good book on project management is by Robert K. Wysocki and Rudd McGary, *Effective Project Management: Traditional, Adaptive, Extreme,* 3rd ed., New York: John Wiley & Sons, 2003. Also, the Project Management Institute (www.pmi.org) and the Information Systems Special Interest Group of the Project Management Institute (www.pmi-issig.org) have valuable resources on project management in information systems.

Think of these three things as interdependent levers that the project manager controls throughout the SDLC. Whenever one lever is pulled, the other two levers are affected in some way. For example, if a project manager needs to readjust a deadline to an earlier date, then the only solution is to decrease the size of the system (by eliminating some of its functions) or to increase costs by adding more people or having them work overtime. Often, a project manager will have to work with the project sponsor to change the goals of the project, such as developing a system with less functionality or extending the deadline for the final system, so that the project has reasonable goals that can be met.

Therefore, in the beginning of the project, the manager needs to estimate each of these levers and then continuously assess how to roll out the project in a way that meets the organization's needs. *Estimation*[2] is the process of assigning projected values for time and effort, and it can be performed manually or with the help of an estimation software package like Costar or Construx—there are over fifty available on the market. The estimates developed at the start of a project are usually based on a range of possible values (e.g., the design phase will take 3 to 4 months) and gradually become more specific as the project moves forward (e.g., the design phase will be completed on March 22).

The numbers used to calculate these estimates can come from several sources. They can be provided with the methodology that is used, taken from projects with similar tasks and technologies, or provided by experienced developers. Generally speaking, the numbers should be conservative. A good practice is to keep track of the actual time and effort values during the SDLC so that numbers can be refined along the way, and the next project can benefit from real data. One of the greatest

CONCEPTS IN ACTION

3-A TRADE-OFFS

I was once on a project to develop a system that should have taken a year to build. Instead, the business need demanded that the system be ready within 5 months—impossible!

On the first day of the project, the project manager drew a triangle on a white board to illustrate some trade-offs that he expected to occur over the course of the project. The corners of the triangle were labeled Functionality, Time, and Money. The manager explained, "We have too little time. We have an unlimited budget. We will not be measured by the bells and whistles that this system contains. So over the next several weeks, I want you as developers to keep this triangle in mind and do everything it takes to meet this 5-month deadline."

At the end of the 5 months, the project was delivered on time; however, the project was incredibly over budget, and the final product was "thrown away" after it was used because it was unfit for regular usage. Remarkably, the business users felt that the project was very successful because it met the very specific business needs for which it was built. They believed that the trade-offs that were made were worthwhile. *Barbara Wixom*

QUESTIONS:

1. What are the risks in stressing only one corner of the triangle?
2. How would you have managed this project? Can you think of another approach that might have been more effective?

[2] A good book for further reading on software estimation is that by Capers Jones, *Estimating Software Costs*, New York: McGraw-Hill, 1989.

strengths of systems consulting firms is the past experience that they offer to a project; they have estimates and methodologies that have been developed and honed over time and applied to hundreds of projects.

There are two basic ways to estimate the time required to build a system. The simplest method uses the amount of time spent in the planning phase to predict the time required for the entire project. The idea is that a simple project will require little planning and a complex project will require more planning, so using the amount of time spent in the planning phase is a reasonable way to estimate overall project time requirements.

With this approach, you take the time spent in (or estimated for) the planning phase and use industry standard percentages (or percentages from the organization's own experiences) to calculate estimates for the other SDLC phases. Industry standards suggest that a "typical" business application system spends 15% of its effort in the planning phase, 20% in the analysis phase, 35% in the design phase, and 30% in the implementation phase. This would suggest that if a project takes 4 months in the planning phase, then the rest of the project likely will take a total of 22.66 person-months (4 ÷ .15 = 22.66). These same industry percentages are then used to estimate the amount of time in each phase (Figure 3-1). The obvious limitation of this approach is that it can be difficult to take into account the specifics of your individual project, which may be simpler or more difficult than the "typical" project.

Function Point Approach

The second approach to estimation, sometimes called the *function point approach*,[3] uses a more complex—and, it is hoped, more reliable—three-step process (Figure 3-2). First, the project manager estimates the size of the project in terms of the number of lines of code the new system will require. This size estimate is then converted into the amount of effort required to develop the system in terms of the number of person-months. The estimated effort is then converted into an estimated schedule time in terms of the number of months from start to finish.

Step 1: Estimate System Size The first step is to estimate the size of a project using function points, a concept developed in 1979 by Allen Albrecht of IBM. A

	Planning	Analysis	Design	Implementation
Typical industry standards for business applications	15%	20%	35%	30%
Estimates based on actual figures for first stages of SDLC	Actual: 4 person-months	Estimated: 5.33 person-months	Estimated: 9.33 person-months	Estimated: 8 person-months

SDLC = systems development life cycle.

FIGURE 3-1
Estimating Project Time Using the Planning Phase Approach

[3] A set of good books that focus on function points are J. Brian Dreger, *Function Point Analysis*, Englewood Cliffs, NJ: Prentice Hall, 1989; and C. R. Symons, *Software Sizing and Estimating: MK II FPA*, New York: John Wiley & Sons, 1991. Additional information on function point analysis can be found at www.ifpug.org.

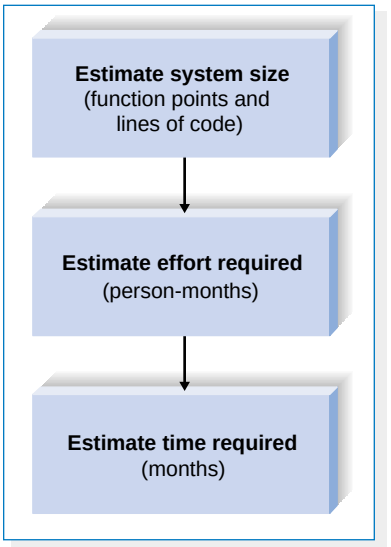

FIGURE 3-2
Estimating Project Time Using the Function Point Approach

function point is a measure of program size that is based on the system's number and complexity of inputs, outputs, queries, files, and program interfaces.

To calculate the function points for a project, components are listed on a worksheet to represent the major elements of the system. For example, data-entry screens are kinds of inputs, reports are outputs, and database queries are kinds of queries (see Figure 3-3). The project manager records the total number of each component that the system will include, and then he or she breaks down the number to show the number of components that have low, medium, and high complexity. In Figure 3.3, there are 19 outputs that need to be developed for the system, 4 of which have low complexity, 10 that have medium complexity, and 5 that are very complex. After each line is filled in, a total number of points are calculated per line by multiplying each number by a complexity index. The complexity index values are drawn from function point research and tell us, for example, that a low complexity input is "worth" three function points, while a high complexity output is "worth" seven function points. The line totals are added up to determine the *total unadjusted function points (TUFP)* for the project.

The *complexity* of the overall system is greater than the sum of its parts. Things like the familiarity of the project team with the business area and the technology that will be used to implement the project also may influence how complex a project will be. A project that is very complex for a team with little experience might have little complexity for a team with lots of experience. To create a more realistic size for the project, a number of additional system factors, such as end-user efficiency, reusability, and data communications are assessed in terms of their effect on the project's complexity (see Figure 3-3). These assessments are totaled and placed into a formula to calculate an *adjusted project complexity (APC)* factor. The APC factor has a baseline value of 0.65, and the Total Processing Complexity score (converted to hundredths) is added to the baseline amount. The TUFP value is multiplied by the APC factor to determine the ultimate size of the project in terms of *total adjusted function points (TAFP)*. This number should give the project manager a reasonable idea as to how big the project will be.

System Components:

Description		Complexity			
	Total Number	Low	Medium	High	Total
Inputs	6	3 × 3	2 × 4	1 × 6	23
Outputs	19	4 × 4	10 × 5	5 × 7	101
Queries	10	7 × 3	0 × 4	3 × 6	39
Files	15	0 × 7	15 × 10	0 × 15	150
Program Interfaces	3	1 × 5	0 × 7	2 × 10	25
Total Unadjusted Function Points (TUFP):					338

Overall System:

Data communications	3
Heavy use configuration	0
Transaction rate	0
End-user efficiency	0
Complex processing	0
Installation ease	0
Multiple sites	0
Performance	0
Distributed functions	2
Online data entry	2
Online update	0
Reusability	0
Operational ease	0
Extensibility	0
Total Processing Complexity (PC):	7

(0 = no effect on processing complexity; 3 = great effect on processing complexity)

Adjusted Project Complexity (APC):

.65 + (0.01 x 7) = .72

Total Adjusted Function Points (TAFP):

.72 (APC) x 338 (TUFP) = 243 **(TAFP)**

FIGURE 3-3
Function Point-Estimation Worksheet

CONCEPTS IN ACTION

3-B Function Points at Nielsen

Nielsen Media used function point analysis (FPA) for an upgrade to the Global Sample Management System (GSMS) for Nielsen Media/NetRatings, which keeps track of the Internet rating sample, a group of 40,000 homes nationwide that volunteer to participate in ongoing ratings.

In late fall of 1998, Nielsen Media did a FP count based on the current GSMS. (FPA is always easier and more accurate when there is an existing system.) Nielsen Media had its counters—three quality assurance staff—do their FPA, and then input their count into KnowledgePlan, a productivity modeling tool. In early 1999, seven programmers began writing code for the system, which they were expected to complete in 10 months. As November approached, the project was adding staff to try to meet the deadline. When it became evident that the deadline would not be met, a new FP count was conducted. The GSMS had grown to 900 FPs. Besides the original 500 plus 20%, there were 300 FPs attributable to features and functions that had crept into the project.

How did that happen? The way it always does: The developers and users had added a button here, a new feature there, and soon the project was much larger than it was originally. But Nielsen Media had put a stake in the ground at the beginning from which they could measure growth along the way.

The best practice is to run the FPA and productivity model at the project's launch and again when there is a full list of functional requirements. Then do another analysis anytime there is a major modification in the functional definition of the project.

Source: "Ratings Game," *CIO Magazine,* October 2000, by Bill Roberts.

Sometimes a shortcut is used to determine the complexity of the project. Instead of calculating the exact APC score for the 14 factors listed in Figure 3-3, project managers estimate an APC value that ranges from 0.65 for very simple systems to 1.00 for "normal" systems to as much as 1.35 for complex systems. This estimated APC score is then applied to the TUFP to compute the TAFP. For example, a very simple system that has 200 unadjusted function points would have a size of 130 adjusted function points ($200 \times .65 = 130$). However, if the system with 200 unadjusted function points were very complex, its function point size would be 270 ($200 \times 1.35 = 270$).

In the Planning Phase, the exact nature of the system has not yet been determined, so it is impossible to know *exactly* how many inputs, outputs, and so forth will be in the system. It is up to the project manager to make an intelligent guess. Some people feel that using function points this early in a project is not practical for this reason. We believe function points can be a useful tool for understanding a project's size at any point in the SDLC. Later in the project, once more is known about the system, the project manager will revise the estimates using this better knowledge to produce more accurate results.

Once you have estimated the number of function points, you need to convert the number of function points into the lines of code that will be required to build the system. The number of lines of code depends on the programming language you choose to use. Figure 3-4 presents a very rough conversion guide for some popular languages.

For example, the system in Figure 3-3 has 243 function points. If you were to develop the system in COBOL, it would typically require approximately 26,730 lines of code to write it. Conversely, if you were to use Visual Basic, it typically would take 7290 lines of code. If you could develop the system using a package such as Excel or Access, it would take between 2,430 and 9,720 lines of code. There

Language	Approximate Number of Lines of Code per Function Point
C	130
COBOL	110
Java	55
C++	50
Turbo Pascal	50
Visual Basic	30
PowerBuilder	15
HTML	15
Packages (e.g., Access, Excel)	10–40

Source: Capers Jones, Software Productivity Research, http://www.spr.com

FIGURE 3-4
Converting from Function Points to Lines of Code

is a great range for packages, because different packages enable you to do different things, and not all systems can be built using certain packages. Sometimes you end up writing lots of extra code to do some simple function because the package does not have the capabilities you need.

There is also a very important message from the data in this figure. Since there is a direct relationship between lines of code and the amount of effort and time required to develop a system, the choice of development language has a significant impact on the time and cost of projects.

YOUR TURN 3-1 Calculate System Size

Imagine that job hunting has been going so well that you need to develop a system to support your efforts. The system should allow you to input information about the companies with which you interview, the interviews and office visits that you have scheduled, and the offers that you receive. It should be able to produce reports, such as a company contact list, an interview schedule, and an office visit schedule, as well as produce thank-you letters to be brought into a word processor to customize. You also need the system to answer queries, such as the number of interviews by city and your average offer amount.

QUESTIONS:

1. Determine the number of inputs, outputs, interfaces, files, and queries that this system requires. For each element, determine if the complexity is low, medium, or high. Record this information on a worksheet similar to the one in Figure 3-3.
2. Calculate the total function points for each line on your worksheet by multiplying the number of each element with the appropriate complexity score.
3. Sum up the total unadjusted function points.
4. Suppose the system will be built by you using Visual Basic (VB). Given your VB skills, multiply the TUFP score by the APC score that best estimates how complex the system will be for you to develop (.65 = simple, 1 = average, 1.35 = complex), and calculate a TAFP value.
5. Using the table in Figure 3-4, determine the number of lines of code that correspond to VB. Multiply this number with the TAFP to find the total lines of code that your system will require.

YOUR TURN

3–2 Calculate Effort and Schedule Time

Refer to the project size and lines of code that you calculated in "Your Turn 3-1."

Questions:

1. Determine the effort of your project in person-months of effort by multiplying your lines of code (in thousands) by 1.4.
2. Calculate the schedule time in months for your project using the formula, $3.0 \times \text{person-months}^{1/3}$.
3. Based on your numbers, how much time will it take to complete the project if you are the developer?

Step 2: Estimate Effort Required Once an understanding is reached about the size of the system, the next step is to estimate the effort that is required to build it. *Effort* is a function of the system size combined with production rates (how much work someone can complete in a given time). Much research has been done on software production rates. One of the most popular algorithms, the COCOMO model,[4] was designed by Barry W. Boehm to convert a lines-of-code estimate into a person-month estimate. There are different versions of the *COCOMO model* that vary based on the complexity of the software, the size of the system, the experience of the developers, and the type of software you are developing (e.g., business application software such as the registration system at your university; commercial software such as Word; or system software such as Windows). For small to moderate-size business software projects (i.e., 100,000 lines of code and 10 or fewer programmers), the model is quite simple:

$$\text{effort (in person-months)} = 1.4 \times \text{thousands of lines of code}$$

For example, let's suppose that we were going to develop a business software system requiring 10,000 lines of code. This project would typically take 14 person-months of effort. If the system in Figure 3.3 were developed in COBOL (which equates to 26,730 lines of code), it would require about 37.42 person-months of effort.

Step 3: Estimate Time Required Once the effort is understood, the optimal schedule for the project can be estimated. Historical data or estimation software can be used as aids, or one rule of thumb is to determine schedule using the following equation:

$$\text{schedule time (months)} = 3.0 \times \text{person-months}^{1/3}$$

This equation is widely used, although the specific numbers vary (e.g., some estimators may use 3.5 or 2.5 instead of 3.0). The equation suggests that a project

[4] The original COCOMO model is presented by Barry W. Boehm in *Software Engineering Economics*, Englewood Cliffs, NJ: Prentice Hall, 1981. Since then, much additional research has been done. The latest version of COCOMO, COCOMO II, is described in B.W. Boehm, C. Abts, A.W. Brown, S. Chulani, B.K. Clark, E. Horowitz, R. Madachy, D. Reifer, and B. Steece, *Software Cost Estimation with COCOMO II*, Upper Saddle River, NJ: Prentice Hall PTR, 2000. For the latest updates, see http://sunset.usc.edu/COCOMOII/cocomo.html.

that has an effort of 14 person-months should be scheduled to take a little more than 7 months to complete. Continuing the Figure 3.3 example, the 37.42 person-months would require a little over 10 months. It is important to note that this estimate is for the analysis, design, and implementation phases; it does not include the planning phase.

CREATING AND MANAGING THE WORKPLAN

Once a project manager has a general idea of the size and approximate schedule for the project, he or she creates a *workplan*, which is a dynamic schedule that records and keeps track of all of the tasks that need to be accomplished over the course of the project. The workplan lists each task, along with important information about it, such as when it needs to be completed, the person assigned to do the work, and any deliverables that will result (Figure 3-5). The level of detail and the amount of information captured by the workplan depend on the needs of the project (and the detail usually increases as the project progresses). Usually, the workplan is the main component of the project management software that we mentioned earlier.

To create a workplan, the project manager first identifies the tasks that need to be accomplished and determines how long they will take. Then the tasks are organized within a work breakdown structure and presented graphically using Gantt and PERT charts. All of these techniques help a project manager understand and manage the project's progress over time, and they are described in detail in the next sections.

Identify Tasks

The overall objectives for the system should be listed on the system request, and it is the project manager's job to identify all of the tasks that need to be accomplished to meet those objectives. This sounds like a daunting task—how can someone know everything that needs to be done to build a system that has never been built before?

One approach for identifying tasks is to get a list of tasks that has already been developed and to modify it. There are standard lists of tasks, or methodolo-

Workplan Information	Example
Name of the task	Perform economic feasibility
Start date	Jan 05, 2005
Completion date	Jan 19, 2005
Person assigned to the task	Project sponsor; Mary Smith
Deliverable(s)	Cost–benefit analysis
Completion status	Open
Priority	High
Resources that are needed	Spreadsheet software
Estimated time	16 hours
Actual time	14.5 hours

FIGURE 3-5
Workplan Information

gies, that are available for use as a starting point. As we stated in Chapter 1, a *methodology* is a formalized approach to implementing the SDLC (i.e., it is a list of steps and deliverables). A project manager can take an existing methodology, select the steps and deliverables that apply to the current project, and add them to the workplan. If an existing methodology is not available within the organization, methodologies can be purchased from consultants or vendors, or books like this textbook can serve as guidance. Using an existing methodology is the most popular way to create a workplan because most organizations have a methodology that they use for projects.

If a project manager prefers to begin from scratch, he or she can use a structured, top-down approach whereby high-level tasks are defined first and then broken down into subtasks. For example, Figure 3-6 shows a list of high-level tasks that are needed to implement a new IT training class. Some of the main steps in the process include identifying vendors, creating and administering a survey, and building new classrooms. Each step is then broken down in turn and numbered in a hierarchical fashion. There are eight subtasks (i.e., 7.1–7.8) for creating and administering a survey, and there are three subtasks (7.2.1–7.2.3) that comprise the review initial survey task. A list of tasks hierarchically numbered in this way is called a *work breakdown structure*, and it is the backbone of the project workplan.

The work breakdown structure can be organized in one of two ways: by SDLC phase or by product. For example, if a firm decided that it needed to develop a Web

Task Number	Task Name	Duration (in weeks)	Dependency	Status
1	Identify vendors	2		Complete
2	Review training materials	6	1	Complete
3	Compare vendors	2	2	In Progress
4	Negotiate with vendors	3	3	Open
5	Develop communications information	4	1	In Progress
6	Disseminate information	2	5	Open
7	Create and administer survey	4	6	Open
7.1	Create initial survey	1		Open
7.2	Review initial survey	1	7.1	Open
7.2.1	Review by Director of IT Training	1		Open
7.2.2	Review by Project Sponsor	1		Open
7.2.3	Review by Representative Trainee	1		Open
7.3	Pilot test initial survey	1	7.1	Open
7.4	Incorporate survey changes	1	7.2, 7.3	Open
7.5	Create distribution list	.5		Open
7.6	Send survey to distribution list	.5	7.4, 7.5	Open
7.7	Send follow-up message	.5	7.6	Open
7.8	Collect completed surveys	1	7.6	Open
8	Analyze results and choose vendor	2	4, 7	Open
9	Build new classrooms	11	1	In Progress
10	Develop course options	3	8, 9	Open

FIGURE 3-6
Identifying Tasks

site, the firm could create a work breakdown structure based on the SDLC phases: planning, analysis, design, and implementation. In this case, a typical task that would take place during planning would be feasibility analysis. This task would be broken down into the different types of feasibility analysis: technical, economic, and organizational. Each of these would be further broken down into a series of subtasks. Alternatively, the firm could organize the workplan along the lines of the different products to be developed. In the case of a Web site, for example, the products could include applets, application servers, database servers, the various sets of Web pages, a site map, and so on. Each of these products could be decomposed into the different tasks associated with the phases of the SDLC. With either approach, once the overall structure is determined, tasks are identified and included in the work breakdown structure of the workplan.

The number of tasks and level of detail depend on the complexity and size of the project. The larger the project, the more important it becomes to define tasks in detail so that essential steps are not overlooked.

The Project Workplan

The project workplan is the mechanism that is used to manage the tasks that are listed in the work breakdown structure. It is the project manager's primary tool for managing the project. Using it, the project manager can tell if the project is ahead or behind schedule, how well the project was estimated, and what changes need to be made to meet the project deadline.

Basically, the workplan is a table that lists all of the tasks in the work breakdown structure along with important task information, such as the people who are assigned to perform the tasks, the actual hours that the tasks took, and the variances between estimated and actual completion times (see Figure 3-6). At a minimum, the information should include the duration of the task, the current statuses of the tasks (i.e., open, complete), and the *task dependencies,* which occur when one task cannot be performed until another task is completed. For example, Figure 3-6 shows that incorporating changes to the survey (task 7.4) takes a week to perform, but it cannot occur until after the survey is reviewed (task 7.2) and pilot tested (task 7.3). Key *milestones*, or important dates, are also identified on the workplan. Presentations to the approval committee, the start of end-user training, a company retreat, and the due date of the system prototype are the types of milestones that may be important to track.

Gantt Chart

A *Gantt chart* is a horizontal bar chart that shows the same task information as the project workplan, but in a graphical way. Sometimes a picture really is worth a thousand words, and the Gantt chart can communicate the high-level status of a project much faster and easier than the workplan. Creating a Gantt chart is simple and can be done using a spreadsheet package, graphics software (e.g., Microsoft VISIO), or a project management package.

First, tasks are listed as rows in the chart, and time is listed across the top in increments based on the needs of the projects (see Figure 3-7). A short project may be divided into hours or days; whereas, a medium-sized project may be represented using weeks or months. Horizontal bars are drawn to represent the duration of each task; the bar's beginning and end mark exactly when the task will begin and end. As

ID	Task Name	Duration	Start	Finish	Prede
1	Identify vendors	2 wks	Wed 1/1/05	Tue 1/14/05	
2	Review training materials	6 wks	Wed 1/1/05	Tue 2/11/05	
3	Compare vendors	2 wks	Wed 2/12/05	Tue 2/25/05	2
4	Negotiate with vendors	3 wks	Wed 2/26/05	Tue 3/18/05	3
5	Develop communications information	4 wks	Wed 1/15/05	Tue 2/11/05	1
6	Disseminate information	2 wks	Wed 2/12/05	Tue 2/25/05	5
7	Create and administer survey	4 wks	Wed 2/26/05	Tue 3/25/05	6
8	Analyze results and choose	2 wks	Wed 3/26/05	Tue 4/8/05	4, 7
9	Build new classroom	11 wks	Wed 1/15/05	Tue 4/1/05	1
10	Develop course options	3 wks	Wed 4/9/05	Tue 4/29/05	8, 9
11	Budget Meeting	1 day	Wed 1/15/05	Wed 1/15/05	
12	Software Installation	1 day	Tue 4/1/05	Tue 4/1/05	

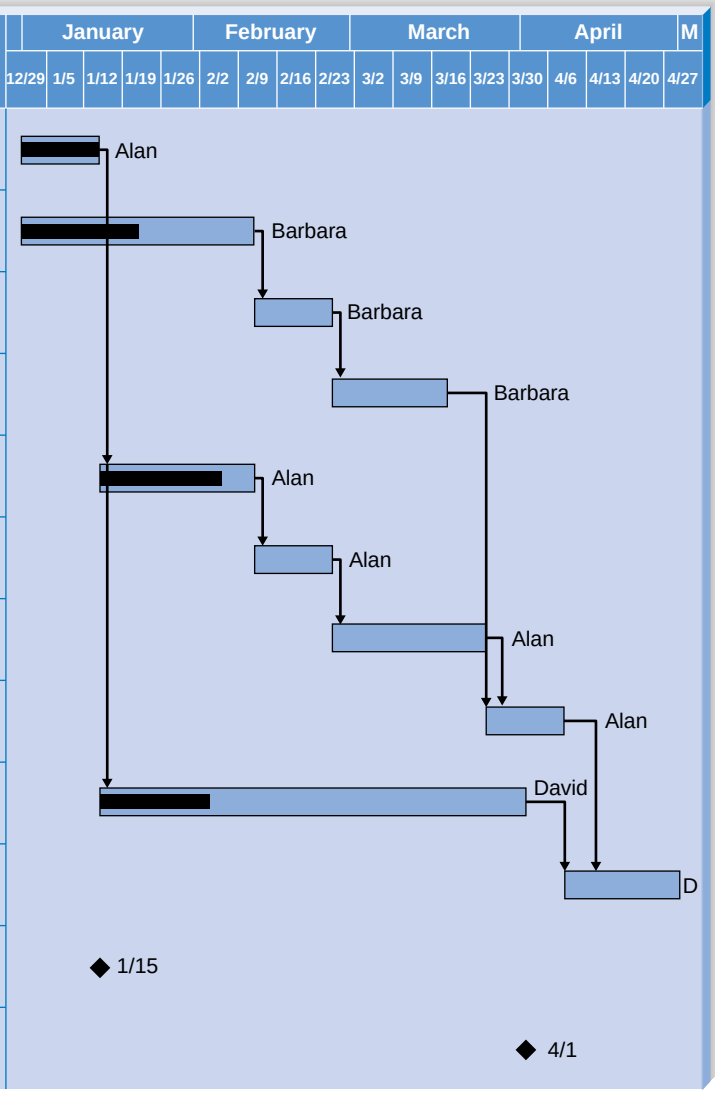

FIGURE 3-7
Gantt Chart

people work on tasks, the appropriate bars are filled in proportionately to how much of the task is finished. Too many tasks on a Gantt chart can become confusing, so it's best to limit the number of tasks to around 20 to 30. If there are more tasks, break them down into subtasks and create Gantt charts for each level of detail.

There are many things a project manager can see by looking quickly at a Gantt chart. In addition to seeing how long tasks are and how far along they are, the project manager also can tell which tasks are sequential, which tasks occur at the same time, and which tasks overlap in some way. He or she can get a quick view of tasks that are ahead of schedule and behind schedule by drawing a vertical line on today's date. If a bar is not filled in and is to the left of the line, that task is behind schedule.

There are a few special notations that can be placed on a Gantt chart. Project milestones are shown using upside-down triangles or diamonds. Arrows are drawn between the task bars to show task dependencies. Sometimes, the names of people assigned to each task are listed next to the task bars to show what human resources have been allocated to each task.

PERT Chart

A second graphical way to look at the project workplan information is the *PERT chart*, which displays the project tasks in a flowchart (see Figure 3-8). *PERT (Program Evaluation and Review Technique)* is a network analysis technique that can be used when the individual task time estimates are fairly uncertain. Instead of assigning a specific value as the duration estimate, PERT uses three time estimates: optimistic, most likely, and pessimistic. It then combines the three estimates into a single weighted average estimate using the following formula:

$$\text{PERT weighted average} = \frac{\text{optimistic value} + (4 * \text{most likely value}) + \text{pessimistic value}}{6}$$

The PERT chart is drawn graphically with boxes (called *nodes*) representing each task and lines (called *arcs*) showing the dependency between tasks. The time estimates are shown in the nodes. Usually the partially completed tasks are displayed with a diagonal line through the node, and completed tasks contain crossed lines.

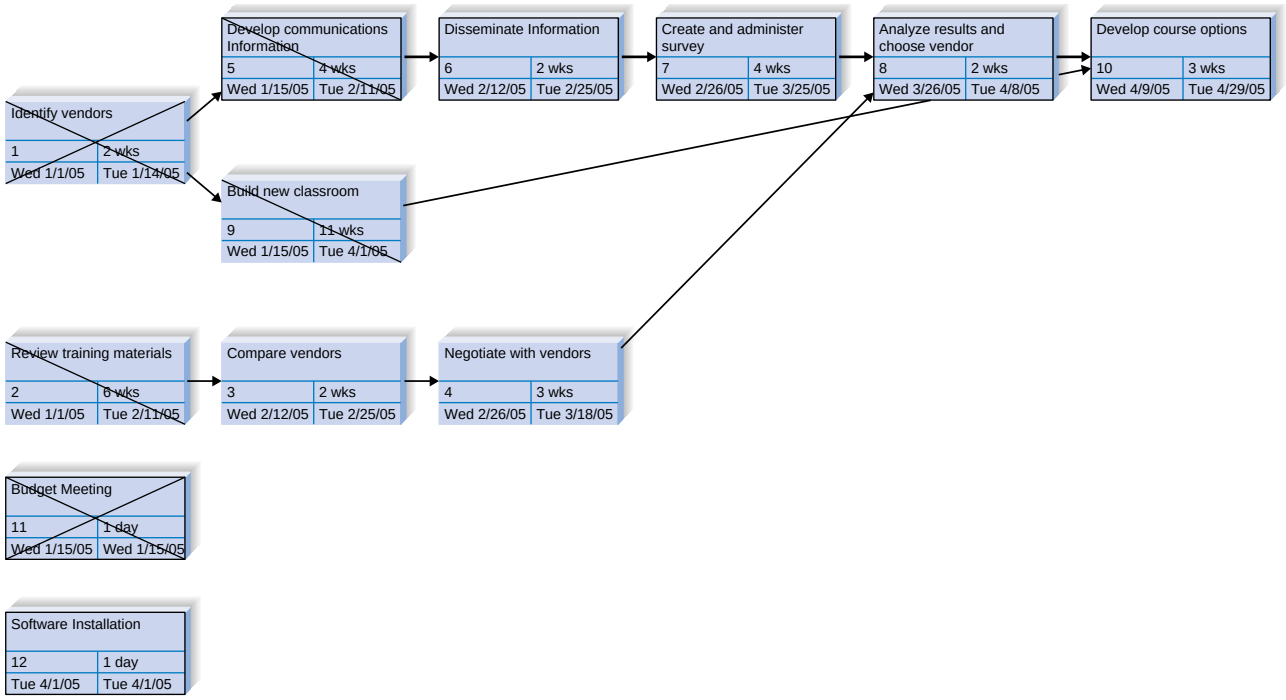

FIGURE 3-8
PERT Chart

PERT charts are the best way to communicate task dependencies because they lay out the tasks in the order in which they need to be completed. The *Critical Path Method (CPM)* allows the identification of the *critical path* in the network, the longest path from the project inception to completion. The critical path shows all of the tasks that must be completed on schedule for the project as a whole to finish on schedule. If any of the tasks on the critical path (called *critical tasks*) takes longer than expected, the entire project will fall behind. CPM can be used with or without PERT.

Project management software packages like Microsoft Project enable the project manager to input the workplan once and then can display the information in many different formats. You can toggle between the workplan, a Gantt chart, and a PERT chart depending on your project management needs.

Refining Estimates

The estimates that are produced during the planning phase will need to be refined as the project progresses. This does not necessarily mean that estimates were poorly done at the start of the project; it is virtually impossible to develop an exact assessment of the project's schedule before the analysis and design phases are conducted. A project manager should expect to be satisfied with broad ranges of estimates that become more and more specific as the project's product becomes better defined.

In many respects, estimating what an IS development project will cost, how long it will take, and what the final system will actually do follows a *hurricane model*. When storms and hurricanes first appear in the Atlantic or Pacific, forecasters watch their behavior and, on the basis of minimal information about them (but armed with lots of data on previous storms), attempt to predict the hurricane's arrival location, time, and strength. As storms move closer to North America, forecasters refine their estimates and develop better predictions. The predictions become more and more accurate as the storms approach a coast and finally arrive.

In the Planning Phase when a system is first requested, the project sponsor and project manager attempt to predict how long the SDLC will take, how much it will cost, and what it will ultimately do when it is delivered (i.e., its functionality). However, the estimates are based on very little knowledge of the system. As the project moves into the Analysis Phase, more information is gathered, the system concept is developed, and the estimates become even more accurate and precise. As the system moves closer to completion, the accuracy and precision increase until the final system is delivered (Figure 3-9).

According to one of the leading experts in software development,[5] a well-done project plan (prepared at the end of the planning phase) has a 100% margin of error for project cost and a 25% margin of error for schedule time. In other words, if a carefully done project plan estimates that a project will cost $100,000 and take 20 weeks, the project will actually cost between $0 and $200,000 and take between 15 and 25 weeks. Figure 3-10 presents typical margins of error for other stages in the project. It is important to note that these margins of error apply only to well-done plans; a plan developed without much care has a much greater margin of error.

[5] Barry W. Boehm and colleagues, "Cost Models for Future Software Life Cycle Processes: COCOMO 2.0," in J. D. Arthur and S. M. Henry (editors), *Annals of Software Engineering: Special Volume on Software Process and Product Measurement*, Amsterdam: J. C. Baltzer AG Science Publishers, 1995.

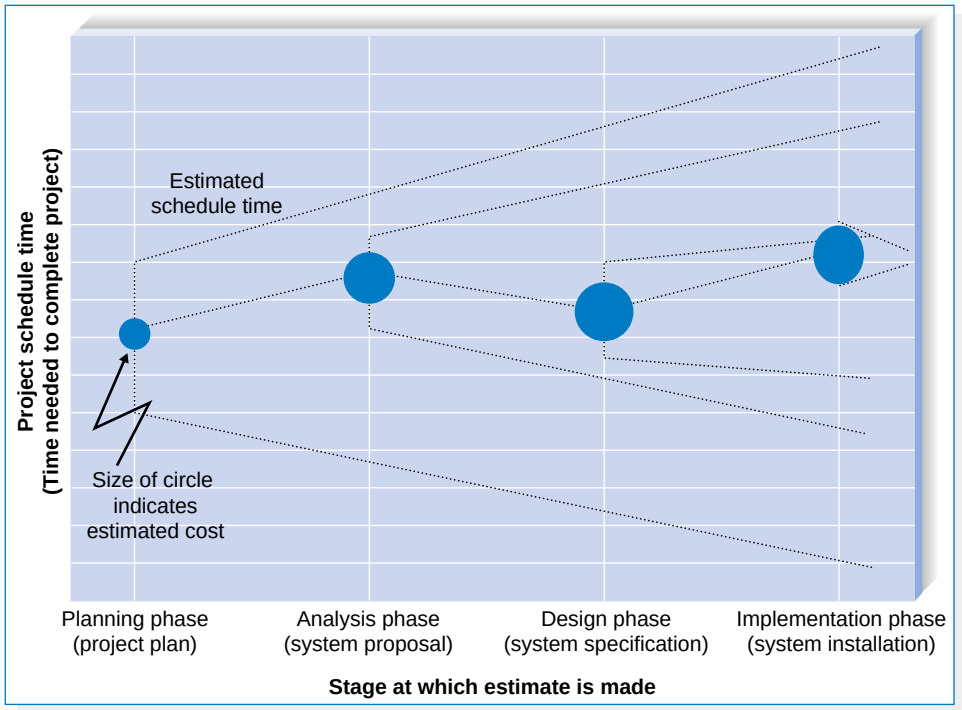

FIGURE 3-9
Hurricane Model

What happens if you overshoot an estimate (e.g., the Analysis Phase ends up lasting two weeks longer than expected)? There are a number of ways to adjust future estimates. If the project team finishes a step ahead of schedule, most project managers shift the deadlines sooner by the same amount but do not adjust the promised completion date. The challenge, however, occurs when the project team is late in meeting a scheduled date. Three possible responses to missed schedule dates are presented in Figure 3-11. We recommend that if an estimate proves too optimistic early in the project, do not expect to make up for lost time—very few projects end up doing this. Instead, change your future estimates to include an increase similar to the one that was experienced. For example, if the first phase was completed 10% over schedule, increase the rest of your estimates by 10%.

		Typical Margins of Error for Well-Done Estimates	
Phase	Deliverable	Cost (%)	Schedule Time (%)
Planning phase	System request	400	60
	Project plan	100	25
Analysis phase	System proposal	50	15
Design phase	System specifications	25	10

Source: Barry W. Boehm and colleagues, "Cost Models for Future Software Life Cycle Processes: COCOMO 2.0," in J. D. Arthur and S. M. Henry (editors) *Annals of Software Engineering Special Volume on Software Process and Product Measurement,* Amsterdam: J. C. Baltzer AG Science Publishers, 1995.

FIGURE 3-10
Margins of Error in Cost and Time Estimates

Assumptions	Actions	Level of Risk
If you assume the rest of the project is simpler than the part that was late and is also simpler than believed when the original schedule estimates were made, you can make up lost time	Do not change schedule.	High risk
If you assume the rest of the project is simpler than the part that was late and is no more complex than the original estimate assumed, you can't make up the lost time, but you will not lose time on the rest of the project.	Increase the entire schedule by the total amount of time that you are behind (e.g., if you missed the scheduled date by two weeks, move the rest of the schedule dates to two weeks later). If you included padded time at the end of the project in the original schedule, you may not have to change the promised system delivery date; you'll just use up the padded time.	Moderate risk
If you assume that the rest of the project is as complex as the part that was late (your original estimates too optimistic), then all the scheduled dates in the future underestimate the real time required by the same percentage as the part that was late.	Increase the entire schedule by the percentage of weeks that you are behind (e.g., if you are two weeks late on part of the project that was supposed to take eight weeks, you need to increase all remaining time estimates by 25%). If this moves the new delivery date beyond what is acceptable to the project sponsor, the scope of the project must be reduced.	Low risk

FIGURE 3-11
Possible Actions When a Schedule Date Is Missed

Scope Management

You may assume that your project will be safe from scheduling problems because you carefully estimated and planned your project up front. However, the most common reason for schedule and cost overruns occurs after the project is underway—*scope creep*.

Scope creep happens when new requirements are added to the project after the original project scope was defined and "frozen." It can happen for many reasons: users may suddenly understand the potential of the new system and realize new functionality that would be useful; developers may discover interesting capabilities to which they become very attached; a senior manager may decide to let this system support a new strategy that was developed at a recent board meeting.

Unfortunately, after the project begins, it becomes increasingly difficult to address changing requirements. The ramifications of change become more extensive, the focus is removed from original goals, and there is at least some impact on cost and schedule. Therefore, the project manager must actively work to keep the project tight and focused.

The keys are to identify the requirements as well as possible in the beginning of the project and to apply analysis techniques effectively. For example, if needs are fuzzy at the project's onset, a combination of intensive meetings with the users and prototyping could be used so that users "experience" the requirements and better visualize how the system could support their needs. In fact, the use of meetings and prototyping has been found to reduce scope creep to less than 5% on a typical project.

Of course, some requirements may be missed no matter what precautions you take, but several practices can be helpful to control additions to the task list. First, the project manager should allow only absolutely necessary requirements to be added after the project begins. Even at that point, members of the project team should carefully assess the ramifications of the addition and present the assessment back to the users. For example, it may require two more person-months of work to create a newly defined report, which would throw off the entire project deadline by several weeks. Any change that is implemented should be carefully tracked so that an audit trail exists to measure the change's impact.

Sometimes changes cannot be incorporated into the present system even though they truly would be beneficial. In this case, these additions to scope should be recorded as future enhancements to the system. The project manager can offer to provide functionality in future releases of the system, thus getting around telling someone no.

Timeboxing

Another approach to scope management is a technique called *timeboxing.* Up until now, we have described projects that are task oriented. In other words, we have described projects that have a schedule that is driven by the tasks that need to be accomplished, so the greater number of tasks and requirements, the longer the project will take. Some companies have little patience for development projects that take a long time, and these companies take a time-oriented approach that places meeting a deadline above delivering functionality.

Think about your use of word processing software. For 80% of the time, you probably use only 20% of the features, such as the spelling checker, boldfacing, and cutting and pasting. Other features, such as document merging and creation of mailing labels, may be nice to have, but they are not a part of your day-to-day needs. The same goes for other software applications; most users rely on only a small subset of their capabilities. Ironically, most developers agree that typically 75% of a system can be provided relatively quickly, with the remaining 25% of the functionality demanding most of the time.

To resolve this incongruency, a technique called timeboxing has become quite popular, especially when using rapid application development (RAD) methodologies. This technique sets a fixed deadline for a project and delivers the system by that deadline no matter what, even if functionality needs to be reduced. Timeboxing ensures that project teams don't get hung up on the final "finishing touches" that can drag out indefinitely, and it satisfies the business by providing a product within a relatively fast time frame.

There are several steps to implement timeboxing on a project (Figure 3-12). First, set the date of delivery for the proposed goals. The deadline should not be impossible to meet, so it is best to let the project team determine a realistic due date. Next, build the core of the system to be delivered; you will find that timeboxing helps create a sense of urgency and helps keep the focus on the most important features. Because the schedule is absolutely fixed, functionality that cannot be completed needs to be postponed. It helps if the team prioritizes a list of features beforehand to keep track of what functionality the users absolutely need. Quality cannot be compromised, regardless of other constraints, so it is important that the time allocated to activities is not shortened unless the requirements are changed (e.g., don't reduce the time allocated to testing without reducing features). At the end of

1. Set the date for system delivery.
2. Prioritize the functionality that needs to be included in the system.
3. Build the core of the system (the functionality ranked as most important).
4. Postpone functionality that cannot be provided within the time frame.
5. Deliver the system with core functionality.
6. Repeat steps 3 through 5, to add refinements and enhancements.

FIGURE 3-12
Steps for Timeboxing

the time period, a high-quality system is delivered; likely, future iterations will be needed to make changes and enhancements, and the timeboxing approach can be used once again.

STAFFING THE PROJECT

Staffing the project includes determining how many people should be assigned to the project, matching people's skills with the needs of the project, motivating them to meet the project's objectives, and minimizing project team conflict that will occur over time. The deliverable for this part of project management is a staffing plan, which describes the number and kinds of people who will work on the project, the overall reporting structure, and the project charter, which describes the project's objectives and rules.

Staffing Plan

The first step to staffing is determining the average number of staff needed for the project. To calculate this figure, divide the total person-months of effort by the

CONCEPTS IN ACTION

3-C TIMEBOXING

DuPont was one of the first companies to use timeboxing. The practice originated in the company's fibers division, which was moving to a highly automated manufacturing environment. It was necessary to create complex application software quickly, and DuPont recognized that it is better to get a basic version of the system working, learn from the experience of operating with it, and then design an enhanced version than it is to wait for a comprehensive system at a later date.

DuPont's experience implies that:

- The first version must be built quickly.
- The application must be built so that it can be changed and added to quickly.

DuPont stresses that the timebox methodology works well for the company and is highly practical. It has resulted in automation being introduced more rapidly and effectively. DuPont quotes large cost savings from the methodology, and variations of timebox techniques have since been used in many other corporations.

QUESTIONS:

1. Why do you think DuPont saves money by using this technique?
2. Are there situations in which timeboxing would not be appropriate?

Source: "Within the Timebox, Development Deadlines Really Work," *PC Week,* March 12, 1990, by James Martin.

optimal schedule. So to complete a 40 person-month project in 10 months, a team should have an average of four full-time staff members, although this may change over time as different specialists enter and leave the team (e.g., business analysts, programmers, technical writers).

Many times, the temptation is to assign more staff to a project to shorten the project's length, but this is not a wise move. Adding staff resources does not translate into increased productivity; staff size and productivity share a disproportionate relationship, mainly because a large number of staff members is more difficult to coordinate. The more a team grows, the more difficult it becomes to manage. Imagine how easy it is to work on a two-person project team: the team members share a single line of communication. But adding two people increases the number of communication lines to six, and greater increases lead to more dramatic gains in communication complexity. Figure 3-13 illustrates the impact of adding team members to a project team.

One way to reduce efficiency losses on teams is to understand the complexity that is created in numbers and to build in a *reporting structure* that tempers its

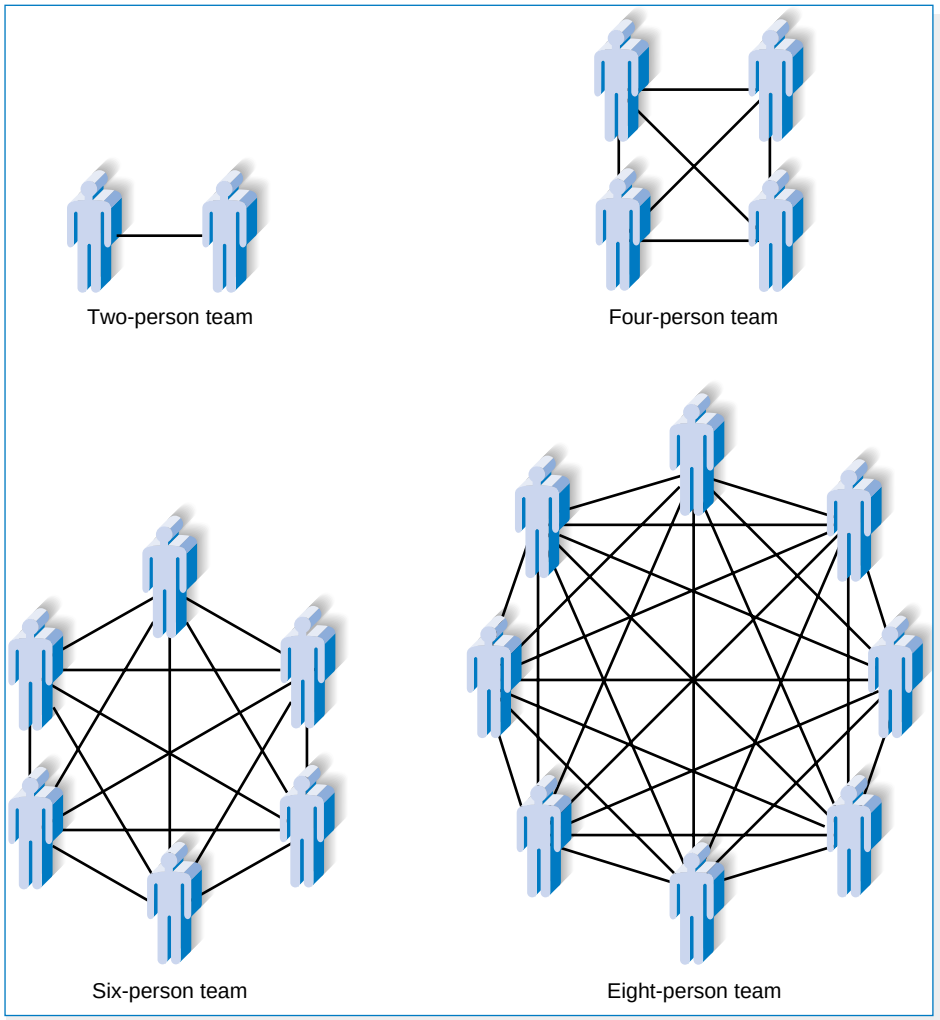

FIGURE 3-13
Increasing Complexity with Larger Teams

effects. The rule of thumb is to keep team sizes under 8 to 10 people; therefore, if more people are needed, create subteams. In this way, the project manager can keep the communication effective within small teams, which in turn communicate to a contact at a higher level in the project.

After the project manager understands how many people are needed for the project, he or she creates a *staffing plan* that lists the roles that are required for the project and the proposed reporting structure for the project. Typically, a project will have one project manager who oversees the overall progress of the development effort, with the core of the team composed of the various types of analysts described in Chapter 1. A *functional lead* usually is assigned to manage a group of analysts, and a *technical lead* oversees the progress of a group of programmers and more technical staff members.

There are many structures for project teams; Figure 3-14 illustrates one possible configuration of a project team. After the roles are defined and the structure is in place, the project manager needs to think about which people can fill each role. Often, one person fills more than one role on a project team.

When you make assignments, remember that people have *technical skills* and *interpersonal skills*, and both are important on a project. Technical skills are useful when working with technical tasks (e.g., programming in Java) and in trying to understand the various roles that technology plays in the particular project (e.g., how a Web server should be configured on the basis of a projected number of hits from customers).

Interpersonal skills, on the other hand, include interpersonal and communication abilities that are used when dealing with business users, senior management executives, and other members of the project team. They are particularly critical when performing the requirements-gathering activities and when addressing organizational feasibility issues. Each project will require unique technical and interpersonal skills. For example, a Web-based project may require Internet experience or Java programming knowledge, or a highly controversial project may need analysts who are particularly adept at managing political or volatile situations.

Ideally, project roles are filled with people who have the right skills for the job; however, the people who fit the roles best may not be available; they may be working on other projects, or they may not exist in the company. Therefore, assigning project team members really is a combination of finding people with the appropriate

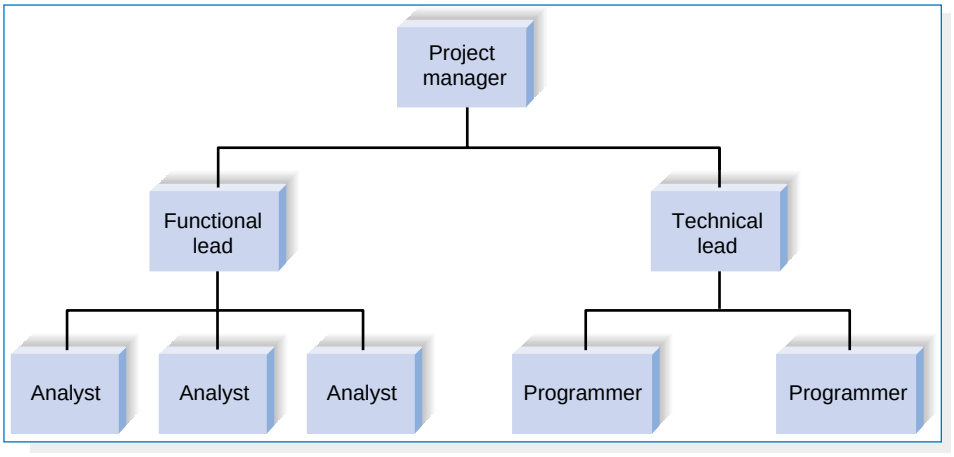

FIGURE 3-14
Possible Reporting Structure

YOUR TURN

3-3 Staffing Plan

Now it is time to staff the project that was described in "Your Turn 3-1." On the basis of the effort required for the project ("Your Turn 3-2"), how many people will be needed on the project? Given this number, select classmates who will work with you on your project.

Questions:

1. What roles will be needed to develop the project? List them and write short descriptions for each of these roles, almost as if you had to advertise the positions in a newspaper.
2. Which roles will each classmate perform? Will some people perform multiple roles?
3. What will the reporting structure be for the project?

skill sets and finding people who are available. When the skills of the available project team members do not match what is actually required by the project, the project manager has several options to improve the situation. First, people can be pulled off other projects, and resources can be shuffled around. This is the most disruptive approach from the organization's perspective. Another approach is to use outside help—such as a consultant or contractor—to train team members and start them off on the right foot. Training classes are usually available for both technical and interpersonal instruction, if time is available. Mentoring may also be an option; a project team member can be sent to work on another similar project so that he or she can return with skills to apply to the current job.

Motivation

Assigning people to tasks isn't enough; project managers need to motivate the people to make the project a success. *Motivation* has been found to be the number-one influence on people's performance,[6] but determining how to motivate the team can be quite difficult. You may think that good project managers motivate their staff by rewarding them with money and bonuses, but most project managers agree that this is the last thing that should be done. The more often you reward team members with money, the more they expect it—and most times monetary motivation won't work.

Assuming that team members are paid a fair salary, technical employees on project teams are much more motivated by recognition, achievement, the work itself, responsibility, advancement, and the chance to learn new skills.[7] If you feel like you need to give some kind of reward for motivational purposes, try a pizza or free dinner, or even a kind letter or award. They often have much more effective results. Figure 3-15 lists some other motivational don'ts that you should avoid to ensure that motivation on the project is as high as possible.

[6] Barry W. Boehm, *Software Engineering Economics*, Englewood Cliffs, NJ: Prentice Hall, 1981. One of the best books on managing project teams is by Tom DeMarco and Timothy Lister, *Peopleware: Productive Projects and Teams*, New York: Dorset House, 1987.

[7] F. H. Hertzberg, "One More Time: How Do You Motivate Employees?" *Harvard Business Review*, 1968, January–February.

Don'ts	Reasons
Assign unrealistic deadlines	Few people will work hard if they realize that a deadline is impossible to meet.
Ignore good efforts	People will work harder if they feel like their work is appreciated. Often, all it takes is public praise for a job well done.
Create a low-quality product	Few people can be proud of working on a project that is of low quality.
Give everyone on the project a raise	If everyone is given the same reward, then high-quality people will believe that mediocrity is rewarded—and they will resent it.
Make an important decision without the team's input	Buy-in is very important. If the project manager needs to make a decision that greatly affects the members of her team, she should involve them in the decision-making process.
Maintain poor working conditions	A project team needs a good working environment or motivation will go down the tubes. This includes lighting, desk space, technology, privacy from interruptions, and reference resources.

Source: Adapted from *Rapid Development,* Redmond, WA: Microsoft Press, 1996, by Steve McConnell.

FIGURE 3-15
Motivational Don'ts

Handling Conflict

The third component of staffing is organizing the project to minimize conflict among group members. *Group cohesiveness* (the attraction that members feel to the group and to other members) contributes more to productivity than do project members' individual capabilities or experiences.[8] Clearly defining the roles on the

CONCEPTS IN ACTION

3-D Haste Slows Microsoft

In 1984, Microsoft planned for the development of Microsoft Word to take one year. At the time, this was two months less than the most optimistic estimated deadline for a project of its size. In reality, it took Microsoft five years to complete Word. Ultimately, the overly aggressive schedule for Word slowed its development for a number of reasons. The project experienced high turnover due to developer burn-out from unreasonable pressure and work hours. Code was "finalized" prematurely, and the software spent much longer in "stabilization" (i.e., fixing bugs) than was originally expected (i.e., 12 months versus 3 months). And, the aggressive scheduling resulted in poor planning (the delivery date consistently was off by more than 60% for the first four years of the project).

QUESTION:

Suppose you take over as project manager in 1986 after the previous project manager has been fired. Word is now 1 year overdue. Describe three things you would do on your first day on the job to improve the project.

Source: Microsoft Corporation: Office Business Unit, *Harvard Business School Case Study 9-691-033,* revised May 31, 1994, Boston: Harvard Business School, 1994, by Marco Iansiti.

[8] B. Lakhanpal, "Understanding the Factors Influencing the Performance of Software Development Groups: An Exploratory Group-Level Analysis," *Information and Software Technology,* 1993, 35(8):468–473.

- Clearly define plans for the project.
- Make sure the team understands how the project is important to the organization.
- Develop detailed operating procedures and communicate these to the team members.
- Develop a project charter.
- Develop schedule commitments ahead of time.
- Forecast other priorities and their possible impact on project.

Source: H. J. Thamhain and D. L. Wilemon, "Conflict Management in Project Life Cycles," *Sloan Management Review,* Spring 1975.

FIGURE 3-16
Conflict Avoidance Strategies

project and holding team members accountable for their tasks is a good way to begin mitigating potential conflict on a project. Some project managers develop a *project charter* that lists the project's norms and ground rules. For example, the charter may describe when the project team should be at work, when staff meetings will be held, how the group will communicate with each other, and the procedures for updating the workplan as tasks are completed. Figure 3-16 lists additional techniques that can be used at the start of a project to keep conflict to a minimum.

COORDINATING PROJECT ACTIVITIES

Like all project management responsibilities, the act of coordinating project activities continues throughout the entire project until a system is delivered to the project sponsor and end users. This step includes putting efficient development practices in place and mitigating risk. These activities occur over the course of the entire SDLC, but it is at this point in the project when the project manager needs to put them in place. Ultimately, these activities ensure that the project stays on track and that the chance of failure is kept at a minimum. The rest of this section will describe each of these activities in more detail.

CASE Tools

Computer-aided software engineering (CASE) is a category of software that automates all or part of the development process. Some CASE software packages are primarily used during the analysis phase to create integrated diagrams of the system

YOUR TURN

3-4 Project Charter

Get together with several of your classmates and pretend that you are all staffed on the project described in "Your Turn 3-1" Discuss what would most motivate each of you to perform well on the project. List three potential sources of conflict that could surface as you work together.

Question:
Develop a project charter that lists five rules that all team members will need to follow. How might these rules help avoid potential team conflict?

and to store information regarding the system components (often called *upper CASE*), whereas others are design-phase tools that create the diagrams and then generate code for database tables and system functionality (often called *lower CASE*). *Integrated CASE*, or I-CASE, contains functionality found in both upper-CASE and lower-CASE tools in that it supports tasks that happen throughout the SDLC. CASE comes in a wide assortment of flavors in terms of complexity and functionality, and there are many good programs available in the marketplace, such as the Visible Analyst Workbench, Oracle Designer/2000, Rational Rose, and the Logic Works suite.

The benefits to using CASE are numerous. With CASE tools, tasks are much faster to complete and alter, development information is centralized, and information is illustrated through diagrams, which typically are easier to understand. Potentially, CASE can reduce maintenance costs, improve software quality, and enforce discipline, and some project teams even use CASE to assess the magnitude of changes to the project.

Of course, like anything else, CASE should not be considered a silver bullet for project development. The advanced CASE tools are complex applications that require significant training and experience to achieve real benefits. Often, CASE serves only as a glorified diagramming tool that supports the practices described in Chapter 6 (process modeling) and Chapter 7 (data modeling). Our experience has shown that CASE is a helpful way to support the communication and sharing of project diagrams and technical specifications—as long as it is used by trained developers who have applied CASE on past projects.

The central component of any CASE tool is the *CASE repository*, otherwise known as the information repository or data dictionary. The CASE repository stores the diagrams and other project information, such as screen and report designs, and it keeps track of how the diagrams fit together. For example, most CASE tools will warn you if you place a field on a screen design that doesn't exist in your data model. As the project evolves, project team members perform their tasks using CASE. As you read through the textbook, we will indicate when and how the CASE tool can be used so that you can see how CASE supports the project tasks.

Standards

Members of a project team need to work together, and most project management software and CASE tools provide access privileges to everyone working on the system. When people work together, however, things can get pretty confusing. To make matters worse, people sometimes get reassigned in the middle of a project. It is important that their project knowledge does not leave with them and that their replacements can get up to speed quickly.

YOUR TURN

3-5 Computer-Aided Software Engineering Tool Analysis

Select a computer-aided software engineering (CASE) tool—either one that you will use for class, a program that you own, or a tool that you can examine over the Web. Create a list of the capabilities that are offered by the CASE tool.

Question:
Would you classify the CASE as upper CASE, lower CASE, or integrated CASE (I-CASE)? Why?

Standards are created to ensure that team members are performing tasks in the same way and following the same procedures. Standards can range from formal rules for naming files to forms that must be completed when goals are reached to programming guidelines. See Figure 3-17 for some examples of the types of standards that a project may create. When a team forms standards and then follows them, the project can be completed faster because task coordination becomes less complex.

Standards work best when they are created at the beginning of each major phase of the project and well communicated to the entire project team. As the team moves forward, new standards are added when necessary. Some standards (e.g., file-naming conventions, status reporting) are applied to the entire SDLC, whereas others (e.g., programming guidelines) are only appropriate for certain tasks.

Documentation

Another technique that project teams put in place during the planning phase is good *documentation*, which includes detailed information about the tasks of the SDLC. Often, the documentation is stored in *project binder(s)* that contain all the deliverables and all the internal communication that takes place—the history of the project.

Types of Standards	Examples
Documentation standards	The date and project name should appear as a header on all documentation.
	All margins should be set to 1 inch.
	All deliverables should be added to the project binder and recorded in its table of contents.
Coding standards	All modules of code should include a header that lists the programmer, last date of update, and a short description of the purpose of the code.
	Indentation should be used to indicate loops, if-then-else statements, and case statements.
	On average, every program should include one line of comments for every five lines of code.
Procedural standards	Record actual task progress in the work plan every Monday morning by 10 A.M.
	Report to project update meeting on Fridays at 3:30 P.M.
	All changes to a requirements document must be approved by the project manager.
Specification requirement standards	Name of program to be created
	Description of the program's purpose
	Special calculations that need to be computed
	Business rules that must be incorporated into the program
	Pseudocode
	Due date
User interface design standards	Labels will appear in boldface text, left-justified, and followed by a colon.
	The tab order of the screen will move from top left to bottom right.
	Accelerator keys will be provided for all updatable fields.

FIGURE 3-17
A Sampling of Project Standards

A poor project management practice is permitting the project team to wait until the last minute to create documentation. This typically leads to an undocumented system that no one understands. In fact, many problems that companies had updating their systems to handle the year 2000 crisis were the result of the lack of documentation. Good project teams learn to document the system's history as it evolves while the details are still fresh in their memory.

A simple way to set up your documentation is to get some binders and use dividers to separate content according to the major phases of the project. An additional divider should contain internal communication, such as the minutes from status meetings, written standards, letters to and from the business users, and a dictionary of relevant business terms. Then, as the project moves forward, place the deliverables from each task into the project binder with descriptions so that someone outside of the project will be able to understand it, and keep a table of contents up to date with the content that is added. Documentation takes time up front, but it is a good investment that will pay off in the long run.

Managing Risk

One final facet of project management is *risk management*, the process of assessing and addressing the risks that are associated with developing a project. Many things can cause risks: weak personnel, scope creep, poor design, and overly optimistic estimates. The project team must be aware of potential risks so that problems can be avoided or controlled well ahead of time.

Typically, project teams create a *risk assessment*, or a document that tracks potential risks along with an evaluation of the likelihood of the risk and its potential impact on the project (Figure 3-18). A paragraph or two is also included that explains potential ways that the risk can be addressed. There are many options: risks could be publicized, avoided, or even eliminated by dealing with its root cause. For example, imagine that a project team plans to use new technology but its members have identified a risk in the fact that its members do not have the right technical skills. They believe that tasks may take much longer to perform because of a high

CONCEPTS IN ACTION

3-E Poor Naming Standards

I once started on a small project (four people) in which the original members of the project team had not set up any standards for naming electronic files. Two weeks into the project, I was asked to write a piece of code that would be referenced by other files that had already been written. When I finished my piece, I had to go back to the other files and make changes to reflect my new work. The only problem was that the lead programmer decided to name the files using his initials (e.g., GG1.prg, GG2.prg, GG3.prg)—and there were over 200 files! I spent two days opening every one of those files because there was no way to tell what their contents were.

Needless to say, from then on, the team created a code for file names that provided basic information regarding the file's contents and they kept a log that recorded the file name, its purpose, the date of last update, and programmer for every file on the project.

Barbara Wixom

QUESTION:

Think about a program that you have written in the past. Would another programmer be able to make changes to it easily? Why or why not?

RISK ASSESSMENT

RISK #1:	The development of this system likely will be slowed considerably because project team members have not programmed in Java prior to this project.
Likelihood of risk:	High probability of risk
Potential impact on the project:	This risk likely will increase the time to complete programming tasks by 50%.

Ways to address this risk:
It is very important that time and resources are allocated to up-front training in Java for the programmers who are used for this project. Adequate training will reduce the initial learning curve for Java when programming begins. Additionally, outside Java expertise should be brought in for at least some part of the early programming tasks. This person should be used to provide experiential knowledge to the project team so that JAVA-related issues (of which novice Java programmers would be unaware) are overcome.

RISK #2: …

FIGURE 3-18
Sample Risk Assessment

learning curve. One plan of attack could be to eliminate the root cause of the risk—the lack of technical experience by team members—by finding time and resources that are needed to provide proper training to the team.

Most project managers keep abreast of potential risks, even prioritizing them according to their magnitude and importance. Over time, the list of risks will change as some items are removed and others surface. The best project managers, however, work hard to keep risks from having an impact on the schedule and costs associated with the project.

CONCEPTS IN ACTION

3-F The Real Names of the Systems Development Life Cycle (SDLC) Phases

Dawn Adams, Senior Manager with Asymetrix Consulting, has renamed the SDLC phases:

1. Pudding (Planning)
2. Silly Putty (Analysis)
3. Concrete (Design).
4. Touch-this-and-you're-dead-sucker (Implementation)

Adams also uses icons, such as a skull and crossbones for the implementation phase. The funny labels lend a new depth of interest to a set of abstract concepts. But her names have had another benefit. "I had one participant who adopted the names wholeheartedly," she says, "including my icons. He posted an icon on his office door for the duration of each of the phases, and he found it much easier to deal with requests for changes from the client, who could see the increasing difficulty of the changes right there on the door."

QUESTION:
What would you do if your project sponsor demanded that an important change be made during the "touch-this-and-you're-dead-sucker" phase?

Source: Learning Technology Shorttakes (Wednesday, August 26, 1998, 1(2).

APPLYING THE CONCEPTS AT CD SELECTIONS

Alec Adams was very excited about managing the Internet Order System project at CD Selections, but he realized that his project team would have very little time to deliver at least some parts of the system because the company wanted the application developed in time for the holiday season. Therefore, he decided that the project should follow a RAD phased-development methodology, combined with the timeboxing technique. In this way, he could be sure that some version of the product would be in the hands of the users within several months, even if the completed system would be delivered at a later date.

As project manager, Alec had to estimate the project's size, effort, and schedule—some of his least favorite jobs because of how tough it is to do at the very beginning of the project. But he knew that the users would expect at least general ranges for a product delivery date. He began by attempting to estimate the number of inputs, outputs, queries, files, and program interfaces in the new system. For the Web part of the system to be used by customers, he could think of four main queries (searching by artist, by CD title, by song title, and by ad hoc criteria), three input screens (selecting a CD, entering information to put a CD on hold, and a special order screen), four output screens (the home page with general information,

PRACTICAL TIP

3-1 AVOIDING CLASSIC PLANNING MISTAKES

As Seattle University's David Umphress has pointed out, watching most organizations develop systems is like watching reruns of *Gilligan's Island.* At the beginning of each episode, someone comes up with a cockamamie scheme to get off the island that seems to work for a while, but something goes wrong and the castaways find themselves right back where they started—stuck on the island. Similarly, most companies start new projects with grand ideas that seem to work, only to make a classic mistake and deliver the project behind schedule, over budget, or both. Here we summarize four classic mistakes in the planning and project management aspects of the project and discuss how to avoid them:

1. Overly optimistic schedule: Wishful thinking can lead to an overly optimistic schedule that causes analysis and design to be cut short (missing key requirements) and puts intense pressure on the programmers, who produce poor code (full of bugs).
   Solution: Don't inflate time estimates; instead, explicitly schedule slack time at the end of each phase to account for the variability in estimates, using the margins of error from Figure 3-10.
2. Failing to monitor the schedule: If the team does not regularly report progress, no one knows if the project is on schedule.
   Solution: Require team members to honestly report progress (or the lack or progress) every week. There is no penalty for reporting a lack of progress, but there are immediate sanctions for a misleading report.
3. Failing to update the schedule: When a part of the schedule falls behind (e.g., information gathering uses all of the slack in item 1 above plus 2 weeks), a project team often thinks it can make up the time later by working faster. It can't. This is an early warning that the entire schedule is too optimistic.
   Solution: Immediately revise the schedule and inform the project sponsor of the new end date or use timeboxing to reduce functionality or move it into future versions.
4. Adding people to a late project: When a project misses a schedule, the temptation is to add more people to speed it up. This makes the project take longer because it increases coordination problems and requires staff to take time to explain what has already been done.
   Solution: Revise the schedule, use timeboxing, throw away bug-filled code, and add people only to work on an isolated part of the project

Source: Adapted from *Rapid Development*, Redmond, WA: Microsoft Press, 1996, pp. 29–50, by Steve McConnell.

information about CDs, information about the customer's special order, and the hold status), three files (CD information, inventory information, and customer orders), and two program interfaces (one to the company's special order system and one that communicates hold information to the retail store systems). For the part of the system to be used by CD Selections staff (to maintain the marketing materials), he identified three additional inputs, three outputs, four queries, one file, and one program interface. He believed most of these to be of medium complexity. He entered these numbers in a worksheet (Figure 3-19).

Rather than attempt to assess the complexity of the system in detail, Alec chose to use a value of 1.20 for APC. He reasoned that the system was of medium complexity and that the project team has little to moderate experience with Internet-based systems. This produced a TAFP of about 190.

Converting function points into lines of code was challenging. The project would use a combination of C (for most programs) and HTML for the Web screens. Alec decided to assume that about 75% of the function points would be C and 25% would be HTML, which produced a total of about 19,200 lines of code [(.75 × 190 × 130) + (.25 × 190 × 15)].

Using the COCOMO formula, he found that this translated into about 27 person-months of effort (1.4 × 19.2). This in turn suggested a schedule time of about 9 months ($3.0 \times 27^{1/3}$). After much consideration, Alec decided to pad the estimate by 10% (by adding 1 extra month).

Once the estimation was underway, Alec began to identify the tasks that would be needed to complete the system. He used a RAD methodology that CD Selections had in-house, and he borrowed its high-level phases (e.g., analysis) and the major tasks associated with them (e.g., create requirements definition, gather requirements, analyze current system). These were recorded in a workplan using Microsoft Project. Alec expected to define the steps in much more detail at the beginning of each phase (Figure 3-20).

System Components:

Description		Complexity			
	Total Number	Low	Medium	High	Total
Inputs	6	0 × 3	4 × 4	2 × 6	28
Outputs	7	2 × 4	4 × 5	1 × 7	35
Quieries	8	3 × 3	4 × 4	1 × 6	31
Files	4	0 × 7	4 × 10	0 × 15	40
Program Interfaces	3	0 × 5	2 × 7	1 × 10	24
Total Unadjusted Function Points (TUFP):					158

Adjusted Project Complexity (APC): 1.2

Total Adjusted Function Points (TAFP):

1.2 (APC) × 158 (TUFP) = 190 **(TAFP)**

FIGURE 3-19
Function Points for the Internet Order System

ID	WBS	Task Name	Duration	Depend.
1	**1**	**Analysis Phase**	**35 days**	
2	**1.1**	**Create Requirements Definition**	**8 days**	
3	1.1.1	Determine requirements to track	1 day	
4	1.1.2	Compile requirements as they are gathered	5 days	3
5	1.1.3	Review requirements with sponsor	2 days	4
6	**1.1**	**Gather requirements**	**5 days**	**3**
7	1.1.1	Perform document analysis	1 wk	
8	**1.1.2**	**Conduct interviews**	**4.5 days**	
9	1.1.2.1	Interview project sponsor	0.5 days	
10	1.1.2.2	Interview inventory system contact	0.5 days	9
11	1.1.2.3	Interview special order system contact	0.5 days	10
12	1.1.2.4	Interview ISP contact	0.5 days	11
13	1.1.2.5	Interview CD Selections Web contact	0.5 days	12
14	1.1.2.6	Interview other	2 days	13
15	1.1.3	Observe retail store processes	2 days	
16	**1.2**	**Analyze current system**	**6 days**	**6**
17	1.2.1	Draw as-is context diagram	1 day	
18	1.2.2	Create use cases	2 days	17
19	1.2.3	Draw as-is process models	3 days	18
20	1.2.4	Draw as-is data model	3 days	18
21	**1.3**	**Identity Improvements**	**5 days**	**16**
22	1.3.1	Conduct 3 JAD sessions	1 wk	
23	**1.4**	**Develop concept for the new system**	**18 days**	**21**
24	1.5.1	Draw to-be context diagram	3 days	
25	1.5.2	Create use cases	2 wks	24
26	1.5.3	Draw to- be process model	1wk	25
27	1.5.4	Draw to-be data model	1wk	25
28	**2**	**Design Phase**	**0 days**	
29	**3**	**Implementation Phase**	**0 days**	
30	**4**	**Post-Implementation**	**0 days**	

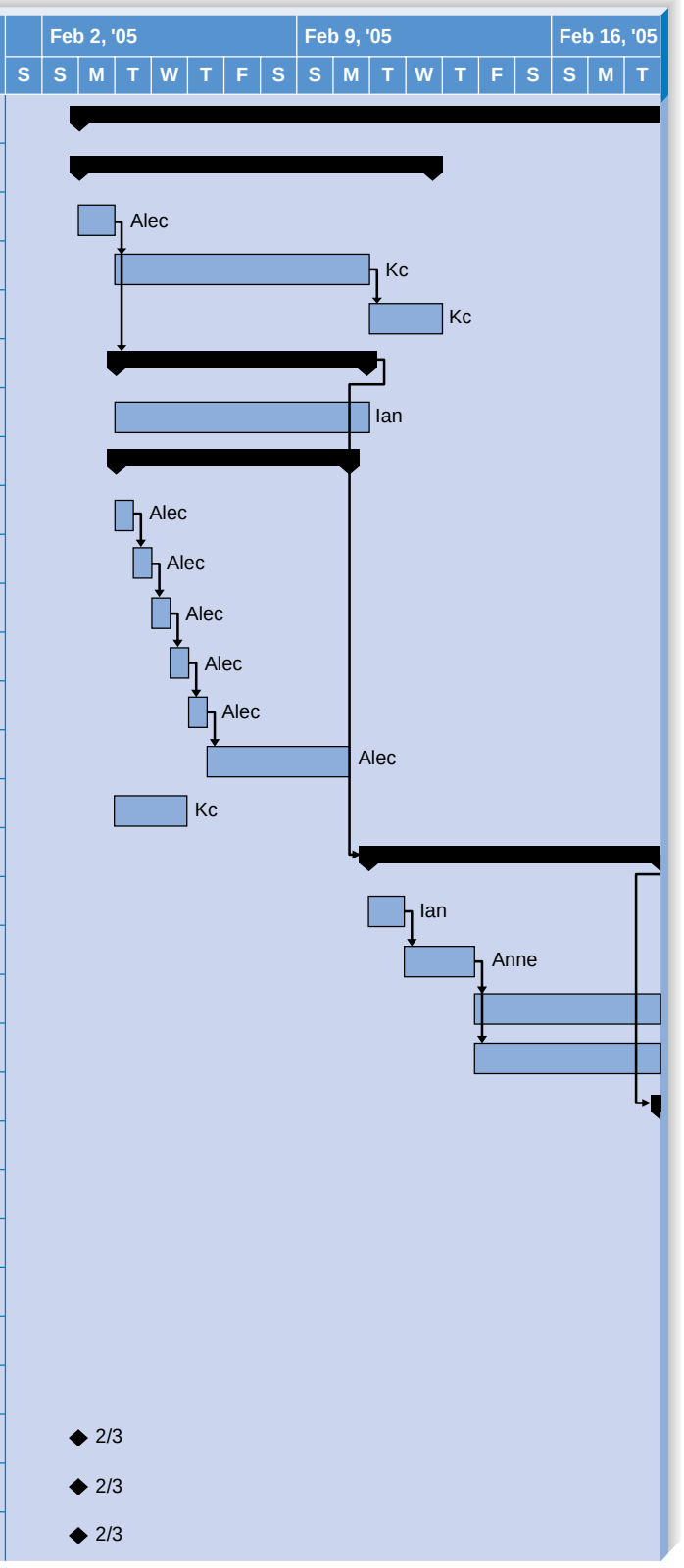

FIGURE 3-20
Gantt Chart

Staffing the Project

Alec next turned to the task of how to staff his project. On the basis of his earlier estimates, it appeared that about three people would be needed to deliver the system by the holidays (27 person-months over 10 months of calendar time means three people, rounded up).

First, he created a list of the various roles that he needed to fill. He thought he would need several analysts to work with the analysis and design of the system as well as an infrastructure analyst to manage the integration of the Internet order system with CD Selections' existing technical environment. Alec also needed people who had good programmer skills and who could be responsible for ultimately implementing the system. Ian, Anne, and K.C. are three analysts with strong technical and interpersonal skills (although Ian is less balanced, having greater technical than interpersonal abilities), and Alec believed that they were available to bring onto this project. He wasn't certain if they had experience with the actual Web technology that would be used on the project, but he decided to rely on vendor training or an external consultant to build those skills later when they were needed. Because the project was so small, Alec envisioned all of the team members reporting to him because he would be serving as the project's manager.

Alec created a staffing plan that captured this information, and he included a special incentive structure in the plan (Figure 3-21). Meeting the holiday deadline was very important to the project's success, so he decided to offer a day off to the team members who contributed to meeting that date. He hoped that this incentive would motivate the team to work very hard. Alec also planned to budget money for pizza and sodas for times when the team worked long hours.

Before he left for the day, Alec drafted a project charter, to be fine-tuned after the team got together for its *kick-off meeting* (i.e., the first time the project team gets together). The charter listed several norms that Alec wanted to put in place from the

Role	Description	Assigned To
Project manager	Oversees the project to ensure that it meets its objectives in time and within budget	Alec
Infrastructure analyst	Ensures the system conforms to infrastructure standards at CD Selections; ensures that the CD Selections infrastructure can support the new system	Ian
Systems analyst	Designs the information system—with a focus on interfaces with the distribution system	Ian
Systems analyst	Designs the information system—with a focus on the process models and interface design	K.C.
Systems analyst	Designs the information system—with a focus on the data models and system performance	Anne
Programmer	Codes system	K.C.
Programmer	Codes system	Anne

Reporting structure: All project team members will report to Alec.

Special incentives: If the deadline for the project is met, all team members who contributed to this goal will receive a free day off, to be taken over the holiday season.

FIGURE 3-21
Staffing plan for the Internet Order System

Project objective: The Internet Order System project team will create a working Web-based system to sell CDs to CD Selections' customers in time for the holiday season.

The Internet Order System team members will

1. Attend a staff meeting each Friday at 2 P.M. to report on the status of assigned tasks.
2. Update the workplan with actual data each Friday by 5 P.M.
3. Discuss all problems with Alec as soon as they are detected.
4. Agree to support each other when help is needed, especially for tasks that could hold back the progress of the project.
5. Post important changes to the project on the team bulletin board as they are made.

FIGURE 3-22
Project Charter for the Internet Order System

start to eliminate any misunderstanding or problems that could come up otherwise (Figure 3-22).

Coordinating Project Activities

Alec wanted the Internet Order System project to be well coordinated, so he immediately put several practices in place to support his responsibilities. First, he acquired the CASE tool used at CD Selections and set up the product so that it could be used for the analysis-phase tasks (e.g., drawing the data flow diagrams). The team members would likely start creating diagrams and defining components of the system fairly early on. He pulled out some standards that he uses on all development projects and made a note to review them with his project team at the kick-off meeting for the system. He also had his assistant set up binders for the project deliverables that would start rolling in. Already he was able to include the system request, feasibility analysis, initial workplan, staffing plan, project charter, standards list, and risk assessment.

SUMMARY

Project Management

Project management is the second major component of the planning phase of the systems development life cycle (SDLC), and it includes four steps: identifying the project size, creating and managing the workplan, staffing the project, and coordinating project activities. Project management is important in ensuring that a system is delivered on time, within budget, and with the desired functionality.

Identifying Project Size

The project manager estimates the amount of time and effort that will be needed to complete the project. First, the size is estimated by relying on past experiences or industry standards or by calculating the function points, a measure of program size based on the number and complexity of inputs, outputs, queries, files, and program interfaces. Next, the project manager calculates the effort for the project, which is a function of size and production rates. Algorithms like the COCOMO model can be used to determine the effort value. Third, the optimal schedule for the project is estimated.

Creating and Managing the Workplan

Once a project manager has a general idea of the size and approximate schedule for the project, he or she creates a workplan, which is a dynamic schedule that records and keeps track of all of the tasks that need to be accomplished over the course of the project. To create a workplan, the project manager first identifies the work breakdown structure, or the tasks that need to be accomplished, and then he or she determines how long the tasks will take. Important information about each task is entered into a workplan.

The workplan information can be presented graphically using Gantt and PERT charts. In the Gantt chart, horizontal bars are drawn to represent the duration of each task, and as people work on tasks, the appropriate bars are filled in proportionately to how much of the task is finished. PERT charts are the best way to communicate task dependencies because they lay out the tasks as a flowchart in the order in which they need to be completed. The longest path from the project inception to completion is referred to as the critical path.

Estimating what an IS development project will cost, how long it will take, and what the final system will actually do follows a hurricane model. The estimates become more accurate as the project progresses. One threat to the reliability of estimates is scope creep, which occurs when new requirements are added to the project after the original project scope was defined and "frozen." If the final schedule will not deliver the system in a timely fashion, timeboxing can be used. Timeboxing sets a fixed deadline for a project and delivers the system by that deadline no matter what, even if functionality must be reduced.

Staffing the Project

Staffing involves determining how many people should be assigned to the project, assigning project roles to team members, developing a reporting structure for the team, and matching people's skills with the needs of the project. Staffing also includes motivating the team to meet the project's objectives and minimizing conflict among team members. Both motivation and cohesiveness have been found to greatly influence performance of team members in project situations. Team members are motivated most by such nonmonetary things as recognition, achievement, and the work itself. Conflict can be minimized by clearly defining the roles on a project and holding team members accountable for their tasks. Some managers create a project charter that lists the project's norms and ground rules.

Coordinating Project Activities

Coordinating project activities includes putting efficient development practices in place and mitigating risk, and these activities occur over the course of the entire SDLC. Three techniques are available to help coordinate activities on a project: computer-aided software engineering (CASE), standards, and documentation. CASE is a category of software that automates all or part of the development process; standards are formal rules or guidelines that project teams must follow during the project; and documentation includes detailed information about the tasks of the SDLC. Often, documentation is stored in project binder(s) that contain all the deliverables and all the internal communication that takes place—the history of the project. A risk assessment is used to mitigate risk because it identifies potential risks and evaluates the likelihood of risk and its potential impact on the project.

KEY TERMS

Adjusted project complexity (APC)
Arc
Complexity
Computer-aided software engineering (CASE)
CASE repository
COCOMO model
Critical path
Critical path method (CPM)
Critical task
Documentation
Effort
Estimation
Function point
Function point approach
Functional lead
Gantt chart
Group cohesiveness
Hurricane model
Integrated CASE
Interpersonal skills
Kick-off meeting
Lower CASE
Methodology
Milestone
Motivation
Node
PERT chart
Project binder
Project charter
Project management
Project management software
Project manager
Reporting structure
Risk assessment
Risk management
Scope creep
Staffing plan
Standards
Task dependency
Technical lead
Technical skills
Timeboxing
Total adjusted function points (TAFP)
Total unadjusted function points (TUFP)
Trade-offs
Upper CASE
Work breakdown structure
Workplan

QUESTIONS

1. Why do many projects end up having unreasonable deadlines? How should a project manager react to unreasonable demands?
2. What are the trade-offs that project managers must manage?
3. What are two basic ways to estimate the size of a project?
4. What is a function point, and how is it used?
5. Describe the three steps of the function point approach.
6. What is the formula for calculating the effort for a project?
7. Name two ways to identify the tasks that need to be accomplished over the course of a project.
8. What is the difference between a methodology and a workplan? How are the two terms related?
9. Compare and contrast the Gantt chart and the PERT chart.
10. Describe the hurricane model.
11. What is scope creep, and how can it be managed?
12. What is timeboxing, and why is it used?
13. Describe the differences between a technical lead and a functional lead. How are they similar?
14. Describe three technical skills and three interpersonal skills that would be very important to have on any project.
15. What are the best ways to motivate a team? What are the worst ways?
16. List three techniques to reduce conflict.
17. What is the difference between upper CASE (computer-aided software engineering) and lower CASE?
18. Describe three types of standards, and provide examples of each.
19. What belongs in the project binder? How is the project binder organized?
20. Create a list of potential risks that could affect the outcome of a project.
21. Some companies hire consulting firms to develop the initial project plans and manage the project, but use their own analysts and programmers to develop the system. Why do you think some companies do this?

EXERCISES

A. Visit a project management Web site, such as the Project Management Institute (www.pmi.org). Most have links to project management software products, white papers, and research. Examine some of the links for project management to better understand a variety of Internet sites that contain information related to this chapter.

B. Select a specific project management topic like computer-aided software engineering (CASE), project management software, or timeboxing and search for information on that topic using the Web. The URL listed in question A or any search engine (e.g., Yahoo!, Alta Vista, Excite, InfoSeek) can provide a starting point for your efforts.

C. Pretend that the Career Services office at your university wants to develop a system that collects student résumés and makes them available to students and recruiters over the Web. Students should be able to input their résumé information into a standard résumé template. The information then is presented in a résumé format, and it also is placed in a database that can be queried using an online search form. You have been placed in charge of the project. Develop a plan for estimating the project. How long do you think it would take for you and three other students to complete the project? Provide support for the schedule that you propose.

D. Refer to the situation in question C. You have been told that recruiting season begins a month from today and that the new system must be used. How would you approach this situation? Describe what you can do as the project manager to make sure that your team does not burn out from unreasonable deadlines and commitments.

E. Consider the system described in question C. Create a workplan that lists the tasks that will need to be completed to meet the project's objectives. Create a Gantt chart and a PERT chart in a project management tool (e.g., Microsoft Project) or using a spreadsheet package to graphically show the high level tasks of the project.

F. Suppose that you are in charge of the project that is described in question C, and the project will be staffed by members of your class. Do your classmates have all of the right skills to implement such a project? If not, how will you go about making sure that the proper skills are available to get the job done?

G. Consider the application that is used at your school to register for classes. Complete a function point worksheet to determine the size of such an application. You will need to make some assumptions about the application's interfaces and the various factors that affect its complexity.

H. Read "Your Turn 3-1" near the beginning of this chapter. Create a risk assessment that lists the potential risks associated with performing the project, along with ways to address the risks.

I. Pretend that your instructor has asked you and two friends to create a Web page to describe the course to potential students and provide current class information (e.g., syllabus, assignments, readings) to current students. You have been assigned the role of leader, so you will need to coordinate your activities and those of your classmates until the project is completed. Describe how you would apply the project management techniques that you have learned in this chapter in this situation. Include descriptions of how you would create a workplan, staff the project, and coordinate all activities—yours and those of your classmates.

J. Select two project management software packages and research them using the Web or trade magazines. Describe the features of the two packages. If you were a project manager, which one would you use to help support your job? Why?

K. Select two estimation software packages and research them using the Web or trade magazines. Describe the features of the two packages. If you were a project manager, which one would you use to help support your job? Why?

L. In 1997, Oxford Health Plans had a computer problem that caused the company to overestimate revenue and underestimate medical costs. Problems were caused by the migration of its claims processing system from the Pick operating system to a UNIX-based system that uses Oracle database software and hardware from Pyramid Technology. As a result, Oxford's stock price plummeted, and fixing the system became the number-one priority for the company. Pretend that you have been placed in charge of managing the repair of the claims processing system. Obviously, the project team will not be in good spirits. How will you motivate team members to meet the project's objectives?

MINICASES

1. Emily Pemberton is an IS project manager facing a difficult situation. Emily works for the First Trust Bank. which has recently acquired the City National Bank. Prior to the acquisition, First Trust and City National were bitter rivals, fiercely competing for market share in the region. Following the acrimonious takeover, numerous staff were laid off in many banking areas, including IS. Key individuals were retained from both banks' IS areas, however, and were assigned to a new consolidated IS department. Emily has been made project manager for the first significant IS project since the takeover, and she faces the task of integrating staffers from both banks on her team. The project they are undertaking will be highly visible within the organization, and the time frame for the project is somewhat demanding. Emily believes that the team can meet the project goals successfully, but success will require that the team become cohesive quickly and that potential conflicts are avoided. What strategies do you suggest that Emily implement in order to help ensure a successfully functioning project team?

2. Tom, Jan, and Julie are IS majors at Great State University. These students have been assigned a class project by one of their professors, requiring them to develop a new Web-based system to collect and update information on the IS program's alumni. This system will be used by the IS graduates to enter job and address information as they graduate, and then make changes to that information as they change jobs and/or addresses. Their professor also has a number of queries that she is interested in being able to implement. Based on their preliminary discussions with their professor, the students have developed this list of system elements:

Inputs: 1 low complexity, 2 medium complexity, 1 high complexity
Outputs: 4 medium complexity
Queries: 1 low complexity, 4 medium complexity, 2 high complexity
Files: 3 medium complexity
Program Interfaces: 2 medium complexity

Assume that an adjusted project complexity factor of 1.2 is appropriate for this project. Calculate the total adjusted function points for this project.