The design phase of the SDLC uses the requirements that were gathered during analysis to create a blueprint for the future system. A successful design builds on what was learned in earlier phases and leads to a smooth implementation by creating a clear, accurate plan of what needs to be done. This chapter describes the initial transition from analysis to design and presents three ways to accomplish the design for the new system.

OBJECTIVES

■ Explain the initial transition from analysis to design.
■ Create a system specification.
■ Describe three ways to acquire a system: custom, packaged, and outsourced alternatives.
■ Create an alternative matrix.

CHAPTER OUTLINE

Introduction
Transition from Requirements to Design
System Acquisition Strategies
   Custom Development
   Packaged Software
   Outsourcing
Influences on the Acquisition Strategy
   Business Need
   In-House Experience

   Project Skills
   Project Management
   Time Frame
   Selecting an Acquisition Strategy
   Alternative Matrix
   Applying the Concepts at Tune Source
   Summary
INTRODUCTION

The design phase decides how the new system will operate. Many activities will be involved as the development team develops the system requirements. This chapter provides an outline of those design phase activities, which culminates in the creation of the system specification. We also describe three alternative strategies for acquiring the system that are available to the development team.

TRANSITION FROM REQUIREMENTS TO DESIGN

The purpose of the analysis phase is to figure out what the business needs. The purpose of the design phase is to decide how to build it. System design is the determination of the overall system architecture—consisting of a set of physical processing components, hardware, software, people, and the communication among them—that will satisfy the system’s essential requirements.¹

During the initial part of design, the project team converts the business requirements for the system into system requirements that describe the technical details for building the system. Unlike business requirements, which are listed in the requirements definition and communicated through use cases and logical process and data models, system requirements are communicated through a collection of design documents and physical process and data models. Together, the design documents and physical models make up the blueprint for the new system.

We should note here that our focus is on the design of the technical system blueprint that will satisfy the system’s requirements. An important element of the final, complete information system, however, will be redesigned work flows and procedures that users will follow when using the new system. Business analysts often turn their attention to the design of these components at this stage of the project, while systems analysts focus on more technical design elements. Ultimately, the redesigned business processes and procedures will be communicated in user documentation and training materials, which we discuss in Chapter 12.

The design phase has a number of activities that lead to the system blueprint. (See Figure 7-1.) An important initial part of the design phase is the examination of several system acquisition strategies to decide which will be used to meet the requirements of the system. Systems can be built from scratch, purchased and customized, or outsourced to others, and the project team needs to investigate the viability of each alternative. The decision to make, to buy, or to outsource influences the design tasks that are performed throughout the rest of the phase.

The project team carefully considers the nonfunctional business requirements that were identified during analysis. The nonfunctional business requirements influence the system requirements that drive the design of the system’s architecture. Major considerations of the “how” of a system are operational, performance, security, cultural, and political in nature. For example, the project team needs to plan for the new system’s performance: how fast the system will operate, what its capacity should be, and its availability and reliability. The team needs to create a secure system by specifying access restrictions and by identifying the need for encryption, authentication,

and virus control. The nonfunctional requirements are converted into system requirements that are described in the *architecture design* document (Chapter 8).

At the same time, architecture decisions are made regarding the hardware and software that will be purchased to support the new system (Chapter 8). These decisions are documented in the *hardware and software specification*, which is a document that describes what hardware and software are needed to support the new application. The actual acquisition of hardware and software is sometimes the responsibility of the purchasing department or the area in the organization that handles capital procurement; however, the project team uses the hardware and software specification to communicate the hardware and software needs to the appropriate people.

The user’s interactions with the system also must be designed. The system inputs and outputs will be designed along with a plan or roadmap of the way the system’s features will be navigated. Chapter 9 describes these activities in detail, along with techniques, such as storyboarding and prototyping, that help the project team design a system that meets the needs of its users and is satisfying to use. Design decisions made regarding the interface are communicated through the design document called the interface design.

The processes described in the logical process model provide the foundation for the system’s functionality. Chapter 10 describes how these logical DFDs are converted into physical DFDs that document physical design decisions about how the system will be built. CASE repository entries are updated to reflect specific technology decisions as they are made. Program specifications are prepared to provide the final design details and ensure that programmers have sufficient information to build the right system efficiently. The program design document contains all of the information about new system’s programs.
The data component of the system, described in the logical data model, also must be designed prior to implementation. Chapter 11 discusses the development of the physical data model, updates to the CASE repository, and describes how the CRUD matrix should be updated to verify the consistency between the process and data models. Design decisions regarding data storage are written up in the data storage design document.

Although a textbook such as this must present information sequentially, the many activities of the design phase are highly interrelated. As with the steps in the analysis phase, analysts often go back and forth between them. For example, prototyping in the interface design step often uncovers additional information that is needed in the system, leading to a revision of the physical DFDs or ERDs. Alternatively, a system that is being designed for an organization with centralized systems may require substantial hardware and software investments if the project team decides to change to a system in which all the processing is distributed.

At the end of the design phase, the project team creates the final deliverable for the phase called the system specification. This document contains all of the design documents just described: physical process models, physical data model, architecture design, hardware and software specification, interface design, data storage design, and program design. Collectively, the system specification conveys exactly what system the project team will implement during the implementation phase of the SDLC. See Figure 7-2 for an outline of the system specification content.

**SYSTEM ACQUISITION STRATEGIES**

In our chapters devoted to the analysis phase of the SDLC, we have carefully avoided committing ourselves to a specific way of obtaining the new system. We have stressed that the team should focus on determining the system’s logical requirements during the analysis phase, and postpone the issue of how the system should be acquired until the design phase.
Until now, we have implicitly assumed that the system will be designed, developed, and implemented by the project team. This is not an entirely realistic assumption. In many projects, the team may recognize that some parts or even all of the new system’s software will be acquired from some outside provider. Some organizations have established acquisition policies strongly favoring purchased software. We explain in this chapter that there are many good reasons supporting this decision. Does this mean that all of the work described in Chapters 3 through 6 can be skipped? Our position on this issue is that the work performed in the analysis phase is still essential to the project’s success, especially the tools and techniques that are used to determine, define, and clarify the business and user requirements. It is essential to know what we need before seeking a product that provides the best fit. Otherwise, we run the risk of letting a software vendor tell us what we need and we obtain software that does not fulfill our real business and user requirements.

There are, however, actually three primary ways to approach the creation of a new system: (1) develop a custom application in-house; (2) buy a packaged system and (possibly) customize it; and (3) rely on an external vendor, developer, or service provider to build or provide the system. Each of these choices has its strengths and weaknesses, and each is more appropriate in different situations. There may be obvious characteristics of the project that suggest the preferred acquisition strategy. The following sections describe each acquisition choice in turn, and then we present criteria you can use to select one of the three approaches for your project.

### 7-1 Avoiding Classic Design Mistakes

In Chapters 2 and 3, we discussed several classic mistakes and how to avoid them. Here, we summarize four classic mistakes in the design phase and discuss how to avoid them:

1. **Reducing design time**: If time is short, there is a temptation to reduce the time spent in such “unproductive” activities as design so that the team can jump into “productive” programming. This results in missing important details that have to be investigated later at a much higher time cost (usually, at least 10 times longer).
   **Solution**: If time pressure is intense, use rapid application development (RAD) techniques and timeboxing to eliminate functionality or move it into future versions.

2. **Feature creep**: Even if you are successful at avoiding scope creep, about 25% of system requirements will still change. Changes—big and small—can significantly increase time and cost.
   **Solution**: Ensure that all changes are vital and that the users are aware of the impact on cost and time. Try to move proposed changes into future versions.

3. **Silver bullet syndrome**: Analysts sometimes believe the marketing claims that some design tools solve all problems and magically reduce time and costs. No one tool or technique can eliminate overall time or costs by more than 25% (although some can reduce individual steps by this much).
   **Solution**: If a design tool has claims that appear too good to be true, just say no.

4. **Switching tools in midproject**: Sometimes, analysts switch to what appears to be a better tool during design in the hopes of saving time or costs. Usually, any benefits are outweighed by the need to learn the new tool. This also applies to even “minor” upgrades to current tools.
   **Solution**: Don’t switch or upgrade unless there is a compelling need for specific features in the new tool, and then explicitly increase the schedule to include learning time.

Custom Development

Many project teams assume that custom development, or building a new system from scratch, is the best way to create a system. For one, teams have complete control over the way the system looks and functions. Let’s consider the purchasing process for Tune Source. If the company wants a Web-based feature that links tightly with its existing CD sales system, the project may involve a complex, highly specialized program. Alternatively, Tune Source might have a technical environment in which all information systems are built from standard technology and interface designs so that they are consistent and easier to update and support. In both cases, it could be very effective to create a new system from scratch that meets these highly specialized requirements.

In some situations, the challenges being addressed with the new system are so significant and demanding that serious systems engineering is required to solve them. In these cases, the developers really cannot find a packaged solution that is capable of meeting the project requirements, and a custom development project is the only real viable choice. (See Concepts in Action 7-A.)

Custom development also allows developers to be flexible and creative in the way they solve business problems. Tune Source may envision the Web interface that takes customer digital music purchases as an important strategic enabler. The company may want to use the information from the system to better understand its customers who buy digital music over the Web, and it may want the flexibility to evolve the system to incorporate technology such as data-mining software and geographic information systems to perform marketing research. A custom application would be easier to change to include components that take advantage of current technologies that can support such strategic efforts.

Building a system in-house also builds technical skills and functional knowledge within the company. As developers work with business users, their understanding of the business grows and they become better able to align information systems with strategies and needs. These same developers climb the technology learning curve so that future projects applying similar technology become much easier.

Custom application development, however, requires a dedicated effort that includes long hours and hard work. Many companies have a development staff that is already overcommitted. Facing huge backlogs of systems requests, the staff just does not have time for another project. Also, a variety of skills—technical, interpersonal, functional, project management, modeling—all have to be in place for the...
project to move ahead smoothly. IS professionals, especially highly skilled individuals, are quite difficult to hire and retain.

The risks associated with building a system from the ground up can be quite high, and there is no guarantee that the project will succeed. Developers could be pulled away to work on other projects, technical obstacles could cause unexpected delays, and the business users could become impatient with a growing timeline.

Packaged Software

Many business needs are not unique, and because it makes little sense to reinvent the wheel, many organizations buy packaged software that has already been written, rather than developing their own custom solution. In fact, there are thousands of commercially available software programs that have already been written to serve a multitude of purposes. Think about your own need for a word processor—did you ever consider writing your own word processing software? That would be very silly, considering the number of good software packages available for a relatively inexpensive cost.

Similarly, most companies have needs, such as payroll or accounts receivable, that can be met quite well by packaged software. It can be much more efficient to buy programs that have already been created, tested, and proven, and a packaged system can be bought and installed quickly compared with a custom system. Plus, packaged systems incorporate the expertise and experience of the vendor who created the software.

Let’s think about the needs that Tune Source will have in its Digital Music Download system. One requirement is to have a simple, fast, and flexible process in place to deliver the purchased tunes over the Internet to the purchaser. Server-side download management software programs are available that are designed to optimize the delivery of file downloads. Some of these products are available for free, and in some products, these tools are incorporated into an overall shopping-cart capability as well. Tune Source will certainly need to consider this type of option as it considers alternatives for the Digital Music Download system.
Packaged software can range from small single-function tools, such as the server-side download manager, to huge all-encompassing systems, such as enterprise resource planning (ERP) applications that are installed to automate an entire business. Implementing ERP systems is a popular practice in which large organizations spend millions of dollars installing packages by such companies as SAP, Oracle, and Infor and then change their businesses accordingly. Installing ERP software is much more difficult than installing small application packages, because benefits can be harder to realize and problems are much more serious.

One problem is that companies utilizing packaged systems must accept the functionality that is provided by the system, and rarely is there a perfect fit. If the packaged system is large in scope, its implementation could mean a substantial change in the way the company does business. Letting technology drive the business can be a dangerous way to go.

Most packaged applications allow for some customization or for the manipulation of system parameters to change the way certain features work. For example, the package might have a way to accept information about your company or the company logo that would then appear on input screens. An accounting software package could offer a choice of various ways to handle cash flow or inventory control so that it could support the accounting practices in different organizations. If the amount of customization is not enough and the software package has a few features that don’t quite work the way the company needs them to work, the project team can create a workaround. A workaround is a custom-built add-on program that interfaces with the packaged application to handle special needs. It can be a nice way to create needed functionality that does not exist in the software package. However, workarounds should be a last resort, for several reasons. First, workarounds are not supported by the vendor who supplied the packaged software, so when upgrades are made to the main system, they may make the workaround ineffective. Also, if problems arise, vendors have a tendency to blame the workaround as the culprit and refuse to provide support.

Although choosing a packaged software system is simpler than going with custom development, it also can benefit from following a formal methodology, just as if you were building a custom application. The search for a software package should be based on the detailed requirements identified during analysis.

Systems integration refers to the process of building new systems by combining packaged software, existing legacy systems, and new software written to integrate these. Many consulting firms specialize in systems integration, so it is not uncommon for companies to select the packaged software option and then outsource the integration of a variety of packages to a consulting firm. (Outsourcing is discussed in the next section.)

The key challenge in systems integration is finding ways to integrate the data produced by the different packages and legacy systems. Integration often hinges on taking data produced by one package or system and reformatting it for use in another package or system. The project team starts by examining the data produced by and needed by the different packages and systems and identifying the transformations that must occur to move the data from one to the other. In many cases, this involves fooling the different packages or systems into thinking that the data were produced by an existing program module that the package or system expects to produce the data, rather than by the new package or system that is being integrated.
For example, Tune Source might want to integrate its new Digital Music Download system with its existing Web-based CD sales system. The CD sales system enables customers to purchase CDs over the Web, and it interfaces with Tune Source’s accounting and inventory management systems. The new Digital Music Download system will not require integration with the inventory management system, but it will need to interface with the accounting system and perhaps could share customer data with the CD sales system. The Digital Music Download project team will need to consider these areas of system integration as it evaluates its development options.

**Outsourcing**

The acquisition choice that requires the least in-house resources is *outsourcing*, which means hiring an external vendor, developer, or service provider to create or supply the system. Outsourcing has become quite popular in recent years, with both U.S. and non-U.S. (offshore) service providers available.

The term *outsourcing* has come to include a variety of ways to obtain IT services and products. Outsourcing firms called *application service providers* (ASPs) supply software applications and/or software-related services over wide area networks or the Internet. In this approach to obtaining software, the ASP hosts and manages a software application, and owns, operates, and maintains the servers that run the application. The ASP also employs the people needed to maintain the application.

Organizations wishing to use a software application contract with the ASP, who makes it available to the customer via a wide area network or the Internet, either installed on client computers or through a browser. The customer is billed by the ASP for the application either on a per-use basis or on a monthly or annual fee basis.

*Software as a Service (SaaS)* is a popular term that is essentially an extension of the ASP model. This term is commonly used to describe situations in which SaaS vendors develop and manage their own software rather than managing and hosting a third-party independent software vendor’s software (the more traditional ASP model). Software vendor Salesforce.com was an early provider of a SaaS version of its customer relationship management (CRM) software and helped to popularize this approach to providing software solutions that are web-based and require only a browser to use.

**CONCEPTS 7-C Finding Just the Right Blend**

*Welch Foods, Inc. recognized that the new ERP system being implemented did not have the same reporting capabilities as the systems that were being replaced. Key transportation operations and cost data was going to be lost. Welch’s turned to a Software as a Service (SaaS) business intelligence solution to ensure continued access to old and new data. The SaaS solution was ideal because the company could not realistically manage another project or add an additional burden on its employees at the time, especially in light of the ERP implementation. The SaaS solution provided a variety of business intelligence reporting capabilities to Welch’s, enabling cost savings and overall transportation operational efficiencies.*

*Source: Christina Torode, “SaaS BI helps boost Welch’s efficiency, data retention,” SearchCIO.com, January 13, 2010.*
There is an array of application service providers. Some deliver high-end business applications that can serve the entire enterprise. Some are focused more on serving a small- to medium-sized business clientele. Some ASPs specialize in specific business needs (such as CRM, for example), while some specialize in specific industries (e.g., healthcare).

Obtaining access to a software package through an application service provider has many advantages. There is a low cost of entry and, in most cases, an extremely short setup time. The pay-as-you-go model is often significantly less expensive for all but the most frequent users of the service. Investments in IT staff can be reduced, and investments in specialized IT infrastructure often can be avoided.

Outsourcing firms are also available that will develop a custom system on behalf of the customer. There can be great benefit to having others develop your system. They may be more experienced in the technology or have more resources, such as experienced programmers. Many companies embark on outsourcing deals to reduce costs, whereas others see it as an opportunity to add value to the business. For example, instead of creating a program that handles the purchasing process or buying a preexisting package, Tune Source may decide to let a Web service provider provide commercial services for them.

For whatever reason, outsourcing can be a good alternative for a new system; however, it does not come without costs. If you decide to leave the creation of a new system in the hands of someone else, you could compromise confidential information or lose control over future development. In-house professionals are not benefiting from the skills that could be learned from the project; instead, the expertise is

7-D BUILDING A CUSTOM SYSTEM — WITH SOME HELP

I worked with a large financial institution in the southeast that suffered serious financial losses several years ago. A new chief executive officer was brought in to change the strategy of the organization to being more customer-focused. The new direction was quite innovative, and it was determined that custom systems, including a data warehouse, would have to be built to support the new strategic efforts. The problem was that the company did not have the in-house skills for these kinds of custom projects.

The company now has one of the most successful data warehouse implementations because of its willingness to use outside skills and its focus on project management. To supplement skills within the company, eight sets of external consultants, including hardware vendors, system integrators, and business strategists, were hired to take part and transfer critical skills to internal employees. An in-house project manager coordinated the data warehouse implementation full time, and her primary goals were to clearly set expectations, define responsibilities, and communicate the interdependencies that existed among the team members.

This company showed that successful custom development can be achieved even when the company may not start off with the right skills in-house. However, this kind of project is not easy to pull off—it takes a talented project manager to keep the project moving along and to transition the skills to the right people over time.

Barbara Wixom

Questions:
1. What are the risks in building a custom system without having the right technical skills available within the organization?
2. Why did the company select a project manager from within the organization?
3. Would it have been better to hire an external professional project manager to coordinate the project? Why or why not?
transferred to the outside organization. Ultimately, important skills can walk right out the door at the end of the contract.

Most risks can be addressed if you decide to outsource, but two are particularly important. First, assess the requirements for the project thoroughly—you should never outsource what you don’t understand. If you have conducted rigorous planning and analysis, then you should be well aware of your needs. Second, carefully choose a vendor, developer, or service with a proven track record with the type of system and technology that your system needs.

There are three primary types of contracts that can be drawn to control the outsourcing deal. A time and arrangements deal is very flexible because you agree to pay for whatever time and expenses are needed to get the job done. Of course, this agreement could result in a large bill that exceeds initial estimates. This arrangement works best when you and the outsourcer are unclear about what it is going to take to finish the job.

You will pay no more than expected with a fixed-price contract because if the outsourcer exceeds the agreed-on price, he or she will have to absorb the costs. Outsourcers are very careful about clearly defining requirements up front, and there is little flexibility for change.

The type of contract gaining in popularity is the value-added contract, whereby the outsourcer reaps some percentage of the completed system’s benefits. You have very little risk in this case, but expect to share the wealth once the system is in place.

Creating fair contracts is an art because you need to carefully balance flexibility with clearly defined terms. Needs often change over time, so you don’t want the contract to be so specific and rigid that alterations can’t be made. Think about how quickly technology like the World Wide Web changes. It is difficult to foresee how a project may evolve over a long period. Short-term contracts leave room for reassessment if needs change or if relationships are not working out the way both parties expected. In all cases, the relationship with the outsourcer should be viewed as a partnership in which both parties benefit and communicate openly.

Managing the outsourcing relationship is a full-time job. Thus, someone needs to be assigned full time to manage the outsourcer, and the level of that person should be appropriate for the size of the job. (A multimillion-dollar outsourcing engagement should be handled by a high-level executive.) Throughout the relationship, progress should be tracked and measured against predetermined goals. If you do embark on an outsourcing design strategy, be sure to get more information. Many books have been written that provide much more detailed information on the topic. Figure 7-3 summarizes some guidelines for outsourcing.

---

**FIGURE 7-3**

*Outsourcing Guidelines*

- Keep the lines of communication open between you and your outsourcer.
- Define and stabilize requirements before signing a contract.
- View the outsourcing relationship as a partnership.
- Select the vendor, developer, or service provider carefully.
- Assign a person to manage the relationship.
- Don’t outsource what you don’t understand.
- Emphasize flexible requirements, long-term relationships, and short-term contracts.

---

2 For more information on outsourcing, we recommend M. Lacity and J. Rottman, *Offshore Outsourcing of IT Work: Client and Supplier Perspectives*, Palgrave Macmillan, 2008.
INFLUENCES ON THE ACQUISITION STRATEGY

Each of the system acquisition strategies just discussed has its strengths and weaknesses, and no one strategy is inherently better than the others. Thus, it is important to understand the strengths and weaknesses of each strategy and when to use each. Figure 7-4 summarizes the project characteristics that influence the choice of acquisition strategy.

Business Need

If the business need for the system is common and technical solutions already exist in the marketplace that can fulfill the system requirements, it is usually appropriate to select a packaged software solution. Packaged systems are good alternatives for common business needs. The widespread availability and usefulness of packaged software has caused many larger companies to develop a recommended list of packaged solutions for use throughout the organization. By limiting the selection of software packages from the list of standard options, the organization is able to ensure consistency across the organizational units, streamline decision making, and ultimately reduce costs.

Packaged software is not suitable for every situation, however. A custom solution should be explored when the business need is unique, when there are especially difficult or demanding requirements that cannot be addressed successfully with a package, or when the organization is unable to change enough to adapt to the way of doing business that is embodied in a software package.

Outsourcing can be used to assist a company with custom development projects and to acquire software packages. The specialization and expertise of an outsourcing firm can be very valuable. Because outsourcing brings an outside third party into the development process, it is usually used in situations where the business need is not a critical element of company strategy. If the business need is central to the company strategy, then it is usually better for the company to retain exclusive control over the project if possible.

<table>
<thead>
<tr>
<th>When to Use Custom Development</th>
<th>When to Use a Packaged System</th>
<th>When to Use Outsourcing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Business need</td>
<td>The business need is unique.</td>
<td>The business need is common.</td>
</tr>
<tr>
<td>In-house experience</td>
<td>In-house functional and technical experience exists.</td>
<td>In-house functional experience exists.</td>
</tr>
<tr>
<td>Project skills</td>
<td>There is a desire to build in-house skills.</td>
<td>The skills are not strategic.</td>
</tr>
<tr>
<td>Project management</td>
<td>The project has a highly skilled project manager and a proven methodology.</td>
<td>The project has a project manager who can coordinate vendor’s efforts.</td>
</tr>
<tr>
<td>Time frame</td>
<td>The time frame is flexible.</td>
<td>The time frame is short.</td>
</tr>
</tbody>
</table>

FIGURE 7-4
Selecting a System Acquisition Strategy
Many organizations are using or are considering using offshore outsourcing as a way of “exporting” IT-related work to countries that have lower labor costs. Two-thirds of companies on the InformationWeek 500 list of business technology innovators say they engage in offshore IT outsourcing. Good quality IT skills are available in a number of countries, but companies considering this option in order to save money need to carefully manage the risks of this way of obtaining IT services.³

**In-House Experience**

If in-house experience exists for all the functional and technical needs of the system, it will be easier to build a custom application than if these skills do not exist. A packaged system may be a better alternative for companies that do not have the technical skills to build the desired system. For example, a project team that does not have Web commerce technology skills may want to acquire a Web commerce package that can be installed without many changes. Outsourcing is a good way to bring in outside experience that is missing in-house so that skilled people are in charge of building the system.

**Project Skills**

The skills that are applied during projects are either technical (e.g., Java, Structured Query Language [SQL]) or functional (e.g., electronic commerce), and different design alternatives are more viable, depending on how important the skills are to the company’s strategy. For example, if certain functional and technical expertise that relates to Internet sales applications and Web commerce application development is important to the organization because the company expects the Internet to

play an important role in sales over time, then it makes sense for the company to develop Web commerce applications in-house, using company employees so that the skills can be developed and improved. On the other hand, some skills, such as network security, may be either beyond the technical expertise of employees or not of interest to the company’s strategists—it is just an operational issue that needs to be handled. In this case, packaged systems or outsourcing should be considered so that internal employees can focus on other business-critical applications and skills.

**Project Management**

Custom applications require excellent project management and a proven methodology. There are so many things that can push a project off track, such as funding obstacles, staffing holdups, and overly demanding business users. Therefore, the project team should choose to develop a custom application only if it is certain that the underlying coordination and control mechanisms will be in place. Packaged and outsourcing alternatives also must be managed; however, they are more shielded from internal obstacles because the external parties have their own objectives and priorities (e.g., it may be easier for an outside contractor to say no to a user than for a person within the company to do so). The latter alternatives typically have their own methodologies, which can benefit companies that do not have an appropriate methodology to use.

**Time Frame**

When time is a factor, the project team should probably start looking for a system that is already built and tested. In this way, the company will have a good idea of how long the package will take to put in place and what the final result will contain. Of course, this assumes that the package can be installed as-is and does not need many workarounds to integrate it into the existing business processes and technical environment. The time frame for custom applications is hard to pin down, especially when you consider how many projects end up missing important deadlines. If you must choose the custom development alternative and the time frame is very short, consider using techniques like timeboxing to manage this problem. The time to produce a system through outsourcing really depends on the system and the outsourcer’s resources. If a service provider has services in place that can be used to support the company’s needs, then a business need could be met quickly. Otherwise, an outsourcing solution could take as long as a custom development initiative.

**SELECTING AN ACQUISITION STRATEGY**

Once the project team has a good understanding of how well each acquisition strategy fits with the project’s needs, it must begin to understand exactly how to implement these strategies. For example, what tools and technology would be used if a custom alternative were selected? What vendors make packaged systems that address the project needs? What service providers would be able to build this system if the application was outsourced? This information can be obtained by talking to people working in the IS Department and getting recommendations from business users by contacting other companies with similar needs and investigating the types of systems that they have put in place. Vendors and consultants are usually willing to provide information about various tools and solutions in the form of brochures, product demonstrations, and information seminars.
Project teams employ several approaches to gather additional information that is needed. One helpful tool is the request for proposal (RFP), a document that solicits a formal proposal from a potential vendor, developer, or service provider. RFPs describe in detail the system or service that is needed, and vendors respond by describing in detail how they could supply those needs.

Although there is no standard way of writing an RFP, it should include certain key facts that the vendor requires, such as a detailed description of needs, any special technical needs or circumstances, evaluation criteria, procedures to follow, and timetable. In a large project, the RFP can be hundreds of pages long, since it is essential that all required project details are included.

The RFP is not just a way to gather information. Rather, it results in a vendor proposal that is a binding offer to accomplish the tasks described in the RFP. The vendor proposal includes a schedule and a price for which the work is to be performed. Once the winning vendor proposal is chosen, a contract for the work is entered into.

For smaller projects with smaller budgets, the request for information (RFI) may be sufficient. An RFI is a shorter, less detailed request that is sent to potential vendors to obtain general information about their products and services. Sometimes, the RFI is used to determine which vendors have the capability to perform a service. It is often then followed up with an RFP to the qualified vendors.

When a list of equipment is so complete that the vendor need only provide a price, without any analysis or description of what is needed, the request for quote (RFQ) may be used. For example, if 20 long-range RFID tag readers are needed from the manufacturer on a certain date at a certain location, the RFQ can be used. If an item is described, but a specific manufacturer’s product is not named, then extensive testing will be required to verify fulfillment of the specifications.

After evaluating the acquisition strategy options and seeking additional information, the design team will likely have several viable choices to use to obtain the system. For example, the project team may find three vendors who make packaged systems that could meet the project’s needs; or the team may be debating over whether to develop a system by using Visual Basic as a development tool and the database management system from Sybase; or the team may think it worthwhile to outsource the development effort to a consulting firm like Accenture or American Management Systems. Each alternative will have pros and cons associated with it that must be considered, and only one solution can be selected in the end.

**Suppose that your university were interested in creating a new course registration system that could support Web-based registration.**

**Question:** What should the university consider when determining whether to invest in a custom, packaged, or outsourced system solution?
Alternative Matrix

An alternative matrix can be used to organize the pros and cons of the design alternatives so that the best solution will be chosen in the end. (See Figure 7-5.) This matrix is created by the same steps as the feasibility analysis, which was presented in Chapter 1. The only difference is that the alternative matrix combines several feasibility analyses into one matrix so that the alternatives can be easily compared. The alternative matrix is a grid that contains the technical, economical, and organizational feasibilities for each system candidate, pros and cons associated with adopting each solution, and other information that is helpful when making comparisons. Sometimes, weights are provided for different parts of the matrix to show when some criteria are more important to the final decision.

To create the alternative matrix, draw a grid with the alternatives across the top and different criteria (e.g., feasibilities, pros, cons, and other miscellaneous criteria) along the side. Next, fill in the grid with detailed descriptions about each alternative. This becomes a useful document for discussion because it clearly presents the alternatives being reviewed and comparable characteristics for each one.

Sometimes, weights and scores are added to the alternative matrix to create a weighted alternative matrix that communicates the project’s most important criteria and the alternatives that best address them. A scorecard is built by adding a column labeled “weight” that includes a number depicting how much each criterion matters to the final decision. Typically, analysts take 100 points and spread them out across the criteria appropriately. If five criteria were used and all mattered equally, each criterion would receive a weight of 20. However, if cost were the most important criterion for choosing an alternative, it may receive 60 points, and the other four criteria may get only 10 points each.

Then, the analysts add to the matrix a column called “Score” that communicates how well each alternative meets the criteria. Usually, number ranges like 1 to 5

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Technical Issues:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Criterion 1</td>
<td>20</td>
<td>5</td>
<td>100</td>
<td>3</td>
<td>60</td>
<td>3</td>
<td>60</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Criterion 2</td>
<td>10</td>
<td>3</td>
<td>30</td>
<td>3</td>
<td>30</td>
<td>5</td>
<td>50</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Criterion 3</td>
<td>10</td>
<td>2</td>
<td>20</td>
<td>1</td>
<td>10</td>
<td>3</td>
<td>30</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Economic Issues:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Criterion 4</td>
<td>25</td>
<td>Supporting</td>
<td>3</td>
<td>75</td>
<td>Supporting</td>
<td>3</td>
<td>75</td>
<td>Supporting</td>
<td>5</td>
<td>125</td>
</tr>
<tr>
<td>Criterion 5</td>
<td>10</td>
<td>Information</td>
<td>3</td>
<td>30</td>
<td>Information</td>
<td>1</td>
<td>10</td>
<td>Information</td>
<td>5</td>
<td>50</td>
</tr>
<tr>
<td>Organizational Issues</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Criterion 6</td>
<td>10</td>
<td>5</td>
<td>50</td>
<td>5</td>
<td>50</td>
<td>3</td>
<td>30</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Criterion 7</td>
<td>10</td>
<td>3</td>
<td>30</td>
<td>3</td>
<td>30</td>
<td>1</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Criterion 8</td>
<td>5</td>
<td>3</td>
<td>15</td>
<td>1</td>
<td>5</td>
<td>1</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TOTAL</td>
<td>100</td>
<td></td>
<td>350</td>
<td>270</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>360</td>
</tr>
</tbody>
</table>

* This denotes how well the alternative meets the criteria. 1 = poor fit; 5 = perfect fit.

**Figure 7-5**
Sample Alternative Matrix Using Weights
or 1 to 10 are used to rate the appropriateness of the alternatives by the criteria. So, for the cost criterion, the least expensive alternative may receive a 5 on a 1-to-5 scale, whereas a costly alternative would receive a 1. Weighted scores are computed with each criterion’s weight multiplied by the score it was given for each alternative. Then, the weighted scores are totaled for each alternative. The highest weighted score achieves the best match for our criteria. When numbers are used in the alternative matrix, project teams can make decisions quantitatively and on the basis of hard numbers.

It should be pointed out, however, that the score assigned to the criteria for each alternative is nothing more than a subjective assignment. Consequently, it is entirely possible for an analyst to skew the analysis according to his or her own biases. In other words, the weighted alternative matrix can be made to support whichever alternative you prefer and yet retains the appearance of an objective, rational analysis. To avoid the problem of a biased analysis, each analyst on the team could develop ratings independently; then, the ratings could be compared and discrepancies resolved in an open team discussion.

The final step, of course, is to decide which solution to design and implement. The approval committee should make the decision after the issues involved with the different alternatives are well understood. Remember that the line between the analysis and design is quite fuzzy. Sometimes alternatives are described and selected at the end of analysis, and sometimes this is done at the beginning of design. The bottom line is that at some point before moving into the heart of the design phase, the project team and the approval committee must understand all of the feasible ways in which the system can be created, and they must select the way that makes the most sense for the organization. The acquisition strategy selection that is made will then drive many of the remaining activities in the design phase.

Applying the Concepts at Tune Source

Jason Wells, senior systems analyst and project manager for Tune Source’s Digital Music Download system, had three different approaches that he could take with the new system: He could develop the entire system, using development resources from Tune Source; he could buy a packaged software program (or a set of different packages and integrate them); or he could hire a consulting firm or service provider to create the system. Immediately, Jason ruled out the third option. Building Internet applications, especially e-commerce systems, was becoming increasingly important to the Tune Source business strategy. By outsourcing the Internet system, Tune Source would not develop Internet application development skills and business skills within the organization.
Instead, Jason decided that a custom development project using the company’s standard Web development tools would be the best choice for Tune Source. In this way, the company would be developing critical technical and business skills in-house, and the project team would be able to have a high level of flexibility and control over the final product. Also, Jason wanted the new music download system to interface with the existing Internet-based CD sales system, and there was a chance that a packaged solution would not integrate as well into the Tune Source environment. Finally, Jason knew that additional features were planned for subsequent versions of this system, so he knew that having control over each version was important.

There was one part of the project that might be handled by packaged software: the purchasing portion of the application. Jason realized that a multitude of programs have been written and are available (at low prices) to handle customer transactions over the Web. These programs, called shopping-cart programs, usually allow customers to select items for an order form, input basic information, and finalize the purchase transaction. Jason believed that the project team should at least consider some of these packaged alternatives so that less time had to be spent writing a program that handled basic Web tasks and more time could be devoted to innovative marketing ideas and custom interfaces with the CD sales system.

To help better understand some of the shopping cart programs that were available in the market and how their adoption could benefit the project, Jason created a weighted alternative matrix that compared three different shopping-cart programs against one another (Figure 7-6). Although all three alternatives had positive points,
Jason saw alternative 2 (WebShop) as the best alternative for handling the shopping cart functionality for the new music download system. WebShop was written in Java, the tool that Tune Source selected as its standard Web development language; the expense was reasonable, with no hidden or recurring costs; and there was an in-house person who had some positive experience with the program. Jason made a note to look into acquiring WebShop as the shopping-cart program for the Digital Music Download system.

SUMMARY

Transition from Requirements to Design
The design phase is the phase of the SDLC in which the blueprint for the new system is developed, and it contains many steps that guide the project team through planning exactly how the system needs to be constructed. The requirements that were identified in the analysis phase serve as the primary inputs for design activities. The main deliverable from the design phase is the system specification, which includes the physical process and data models, architecture design, hardware and software specifications, interface design, data storage design, and program design.

System Acquisition Strategies
During the design phase, the project team also needs to consider three approaches to creating the new system, including developing a custom application in-house; buying a packaged system and customizing it; and relying on an external vendor, developer, or system provider to build and/or support the system. Custom development allows developers to be flexible and creative in the way they solve business problems, and it builds technical and functional knowledge within the organization. Many companies have a development staff that is already over-committed to filling huge backlogs of systems requests, however, and they just don’t have time to devote to a project for which a system is built from scratch. It can be much more efficient to buy programs that have been created, tested, and proven; and a packaged system can be bought and installed in a relatively short period of time, when compared with a custom solution. Workarounds can be used to meet the needs that are not addressed by the packaged application. The third design strategy is to outsource the project and pay an external vendor, developer, or service provider to create the system. It can be a good alternative for how to approach the new system; however, it does not come without costs. If a company does decide to leave the creation of a new system in the hands of someone else, the organization could compromise confidential information or lose control over future development.

Influences on Acquisition Strategy
Each of the acquisition strategies just discussed has its strengths and weaknesses, and no one strategy is inherently better than the others. Thus, it is important to consider such issues as the uniqueness of business need for the system, the amount of in-house experience that is available to build the system, and the importance of the project skills to the company. Also, the existence of good project management and the amount of time available to develop the application play a role in the selection process.
Selecting an Acquisition Strategy

Ultimately, the decision must be made regarding the specific acquisition strategy for the system. An alternative matrix can help the design team make this decision by presenting feasibility information for several candidate solutions in a way in which they can be compared easily. The request for proposal, request for information, and request for quote are ways to gather accurate details regarding the alternatives. At this point, the team decides exactly who will perform each part of the design phase and what packages will be used.

KEY TERMS

<table>
<thead>
<tr>
<th>Alternative matrix</th>
<th>Packaged software</th>
<th>Systems integration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application service provider (ASP)</td>
<td>Request for information (RFI)</td>
<td>Time and arrangements deal</td>
</tr>
<tr>
<td>Custom development</td>
<td>Request for proposal (RFP)</td>
<td>Value-added contract</td>
</tr>
<tr>
<td>Design phase</td>
<td>Request for quote (RFQ)</td>
<td>Weighted alternative matrix</td>
</tr>
<tr>
<td>Enterprise resource planning (ERP)</td>
<td>Software as a Service (SaaS)</td>
<td>Workaround</td>
</tr>
<tr>
<td>Fixed-price contract</td>
<td>System requirement</td>
<td></td>
</tr>
<tr>
<td>Outsourcing</td>
<td>System specification</td>
<td></td>
</tr>
</tbody>
</table>

QUESTIONS

1. Summarize the distinctions between the analysis phase and the design phase of the SDLC.
2. Describe the primary activities of the design phase of the SDLC.
3. List and describe the contents of the system specification.
4. Describe the three primary strategies that are available to obtain a new system.
5. What circumstances favor the custom design strategy?
6. What circumstances favor the use of packaged software?
7. What circumstances favor using outsourcing to obtain the new system?
8. What are some problems associated with using packaged software? How can these problems be minimized?
9. What is meant by customizing a software package?
10. What is meant by creating a workaround for a software package? What are the disadvantages of workarounds (if any)?
11. What is involved with systems integration? When is it necessary?
12. Describe the role of application service providers (ASPs) in obtaining new systems. What are their advantages and disadvantages?
13. Distinguish between a traditional ASP and a provider of software as a service. What are the pros and CMS of each solution approach?
14. Explain the distinctions between time and arrangements, fixed-price, and value-added outsourcing contracts. What are the pros and cons of each?
15. What is the purpose of a request for proposal (RFP)? How does it differ from the RFI?
16. What information is typically conveyed in an RFP?
17. What is the purpose of the weighted alternative matrix? Describe its typical content.
18. Should the analysis phase be eliminated or reduced when we intend to use a software package instead of custom development or outsourcing?

EXERCISES

A. Assume that you are developing a new system for a local real estate agency. The agency wants to keep a database of its own property listings and also wants to have access to the citywide multiple listings service used by all real estate agents. Which design strategy would you recommend for the construction of this system? Why?
B. Assume that you are developing a new system for a multistate chain of video stores. Each store will run a fairly standardized set of video store processes
(cataloging video inventory, customer registration, video rentals, video returns, overdue fees, etc.). In addition, each store’s system will be networked to the corporate offices for sales and expense reporting. Which design strategy would you recommend for the construction of this system? Why?

C. Assume that you are part of a development team that is working on a new warehouse management system. You have the task of investigating software packages that are available through application service providers. Using the World Wide Web, identify at least two potential sources of such software. What are the pros and cons of this approach to obtaining a software package?

D. Assume that you are leading a project that will implement a new course registration system for your university. You are thinking about using either a packaged course registration application or outsourcing the job to an external consultant. Create a request for proposal (RFP) to which interested vendors and consultants could respond.

E. Assume that you and your friends are starting a small business painting houses in the summertime. You need to buy a software package that handles the financial transactions of the business. Create an alternative matrix that compares three packaged systems (e.g., Quicken, Microsoft Money, QuickBooks). Which alternative appears to be the best choice?

MINICASES

1. Susan, president of MOTO, Inc., a human resources management firm, is reflecting on the client management software system her organization purchased four years ago. At that time, the firm had just gone through a major growth spurt, and the mixture of automated and manual procedures that had been used to manage client accounts became unwieldy. Susan and Nancy, her IS department head, researched and selected the package that is currently used. Susan had heard about the software at a professional conference she attended, and at least initially, it worked fairly well for the firm. Some of their procedures had to change to fit the package, but they expected that and were prepared for it.

Since that time, MOTO, Inc. has continued to grow, not only through an expansion of the client base, but through the acquisition of several smaller employment-related businesses. MOTO, Inc. is a much different business than it was four years ago. Along with expanding to offer more diversified human resource management services, the firm’s support staff has also expanded. Susan and Nancy are particularly proud of the IS department they have built up over the years. Using strong ties with a local university, an attractive compensation package, and a good working environment, the IS department is well staffed with competent, innovative people, plus a steady stream of college interns keeps the department fresh and lively. One of the IS teams pioneered the use of the Internet to offer MOTO’s services to a whole new market segment, an experiment that has proven very successful.

It seems clear that a major change is needed in the client management software, and Susan has already begun to plan financially to undertake such a project. This software is a central part of MOTO’s operations, and Susan wants to be sure that a quality system is obtained this time. She knows that the vendor of their current system has made some revisions and additions to its product line. There are also a number of other software vendors who offer products that may be suitable. Some of these vendors did not exist when the purchase was made four years ago. Susan is also considering Nancy’s suggestion that the IS department develop a custom software application.

a. Outline the issues that Susan should consider which would support the development of a custom software application in-house.

b. Outline the issues that Susan should consider which would support the purchase of a software package.

c. Within the context of a systems development project, when should the decision of “make-versus-buy” be made? How should Susan proceed? Explain your answer.