2 The Origins of Software

Learning Objectives

After studying this chapter, you should be able to

2.1 explain outsourcing;

2.2 describe six different sources of software;

2.3 discuss how to evaluate off-the-shelf software; and

2.4 explain reuse and its role in software development.

Introduction

As you learned in Chapter 1, there was a time, not too long ago, when no systems analysts and no symbolic computer programming languages existed. Yet people still wrote and programmed applications for computers. Even though today's systems analyst has dozens of programming languages and development tools to work with, you could easily argue that systems development is even more difficult now than it was years ago. Then, as well as even more recently, certain issues were decided for you: If you wanted to write application software, you did it in-house, and you wrote the software from scratch. Today there are many different sources of software, and many of you reading this book will end up working for firms that produce software, rather than in the information systems department of a corporation. But for those of you who do go on to work in a corporate information systems department, the focus is no longer exclusively on in-house development. Instead, the focus will be on where to obtain the many pieces and components that you will combine into the application system you have been asked to create. You and your peers will still write code, mainly to make all the different pieces work together, but more and more of your application software will be written by someone else. Even though you will not write the code, you will still use the basic structure and processes of the systems analysis and design life cycle to build the application systems your organization demands. The organizational process of systems development remains the focus for the rest of the book, but first you need to know more about where software originates in today's development environment.

In this chapter, you will learn about the various sources of software for organizations. The first source considered is outsourcing, in which all or part of an organization's information systems, their development, and their maintenance are given over to another organization. You will then read about six different sources of software: (1) information technology services firms, (2) packaged software providers, (3) vendors of enterprise-wide solution software, (4) cloud computing, (5) open-source software, and (6) the organization itself when it develops software in-house. You will learn about criteria to evaluate software from these different sources. The chapter closes with a discussion of reuse and its impact on software development.

SYSTEMS ACQUISITION

Although there will always be some debate about when and where the first administrative information system was developed, there is general agreement that the first such system in the United Kingdom was developed at J. Lyons & Sons. In the United States, the first administrative information system was General Electric's (GE) payroll system, which was developed in 1954 (Computer History Museum, 2003). At that time, and for many years afterward, obtaining an information system meant one thing only: in-house development. The software industry did not even come into existence until a decade after GE's payroll system was implemented.

Since GE's payroll system was built, in-house development has become a progressively smaller piece of all the systems development work that takes place in and for organizations. Internal corporate information systems departments now spend a smaller and smaller proportion of their time and effort on developing systems from scratch.

Companies continue to spend relatively little time and money on traditional software development and maintenance. Instead, they invest in packaged software, open-source software, and outsourced services. Organizations today have many choices when seeking an information system. We will start with a discussion of outsourcing development and operation and then move on to a presentation on the sources of software.

Outsourcing

Outsourcing
The practice of turning over responsibility for some or all of an organization's information systems applications and operations to an outside firm.

If one organization develops or runs a computer application for another organization, that practice is called **outsourcing**. Outsourcing includes a spectrum of working arrangements. At one extreme is having a firm develop and run your application on its computers—all you do is supply input and take output. A common example of such an arrangement is a company that runs payroll applications for clients so that clients do not have to develop an independent in-house payroll system. Instead, they simply provide employee payroll information to the company, and, for a fee, the company returns completed paychecks, payroll accounting reports, and tax and other statements for employees. For many organizations, payroll is a very cost-effective operation when outsourced in this way. Another example of outsourcing would be if you hired a company to run your applications at your site on your computers. In some cases, an organization employing such an arrangement will dissolve some or all of its information systems (IS) unit and fire all of its IS employees. Often the company brought in to run the organization's computing will rehire many of the organization's original IS unit employees.

Outsourcing is big business. Some organizations outsource the information technology (IT) development of many of their IT functions at a cost of billions of dollars. Most organizations outsource at least some aspect of their information systems activities. Global outsourcing revenues in 2017 were estimated at $88.9 billion USD, with $64.3 billion USD due to IT outsourcing (Statistica, 2018). Individual outsourcing vendors, such as HPE, IBM, and Accenture, typically sign large contracts for their services. These vendors have multiple outsourcing contracts in place with many different firms all over the world.

Why would an organization outsource its information systems operations? As we saw in the payroll example, outsourcing may be cost-effective. If a company specializes in running payroll for other companies, it can leverage the economies of scale it achieves from running one stable computer application for many organizations into very low prices. Outsourcing also provides a way for firms to leapfrog their current position in information systems and to turn over development and operations to outside staff who possess knowledge and skills not found internally (Ketler & Willems, 1999). Other reasons for outsourcing include

- freeing up internal resources,
- increasing the revenue potential of the organization,
- reducing time to market,
- increasing process efficiencies, and
- outsourcing noncore activities.

An organization may move to outsourcing and dissolve its entire information processing unit for political reasons as well, such as overcoming operating problems the organization faces in its information systems unit. For example, the City of Grand Rapids, Michigan, hired an outside firm to run its computing center 50 years ago in order to better manage its computing center employees. Union contracts and civil service constraints then in force made it difficult to fire people, so the City brought in a facilities management organization to run its computing operations, and it was able to get rid of problem employees at the same time. As mentioned earlier, another reason for total outsourcing is that an organization's management may feel its core mission does not involve managing an information systems unit and that it might achieve more effective computing by turning over all its operations to a more experienced, computer-oriented company.

Although you have most likely heard about outsourcing in terms of IT jobs from all over the world going to India, it turns out that the global outsourcing marketplace is much more complicated. According to a 2017 report by ATKearney (2017), India is the number one outsourcing nation, while China is close behind, and Malaysia is third. Despite much turmoil in the overall outsourcing market over the years, the top three rankings have not changed since ATKearney's first report on outsourcing in 2003. Not all of the 2017 top 10 outsourcing countries are located in Asia. Although seven are in Asia, three are in Latin America (Brazil, Chile, and Colombia). Even the United States is an outsourcing nation, number 22 on the ATKearney list. In fact, Indian outsourcing firms, such as Wipro, Infosys, and Tata Consulting, operate offices in the United States. As Indian firms have become so successful at offshoring, and as currencies have fluctuated, it has become more expensive for firms to contract with Indian companies, so many firms have started to look elsewhere. Many U.S. firms have turned to what is called *nearshoring*, or contracting with companies in Latin American countries. Many of these countries are no more than one time zone away, and they maintain some of the labor cost advantages that are eroding in India (King, 2008a). Mexico, number 13 on the list, is increasingly seen as a complement to India and other offshore locations. It is also becoming more common for firms to distribute their outsourcing work across vendors in several countries at the same time.

Analysts need to be aware of outsourcing as an alternative. When generating alternative system development strategies for a system, as an analyst you should consult organizations in your area that provide outsourcing services. It may well be that at least one such organization has already developed and is running an application very close to what your users are asking for. Perhaps outsourcing the replacement system should be one of your alternatives. Knowing what your system requirements are before you consider outsourcing means that you can carefully assess how well the suppliers of outsourcing services can respond to your needs. However, should you decide not to consider outsourcing, you need to determine whether some software components of your replacement system should be purchased and not built in-house.

Sources of Software

We can group the sources of software into six major categories: information technology services firms, packaged software producers, enterprise-wide solutions, cloud computing vendors, open-source software, and in-house developers (Figure 2-1). These various sources represent points along a continuum of options, with many hybrid combinations along the way.

Information Technology Services Firms If a company needs an information system but does not have the expertise or the personnel to develop the system in-house, and a suitable off-the-shelf system is not available, the company will likely consult an information technology services firm. IT services firms help companies develop custom information systems for internal use, or they develop, host, and run applications for customers, or they provide other services. Note in Table 2-1 that many of the leading software companies in the world specialize in IT services, which includes custom systems development. These firms employ people with expertise in the development of information systems. Their consultants may also have expertise in a given business area. For example, consultants who work with banks understand financial institutions as well as information systems. Consultants use many of the same methodologies, techniques, and tools that companies use to develop systems in-house.

It may surprise you to see IBM listed as a top global software producer; some people still think of it as primarily a hardware company. Yet IBM has been moving away from a reliance on hardware development for many years. IBM long ago solidified its move into services and consulting. IBM is also well known for its development of Web server and middleware software. Other leading IT services firms include traditional consulting firms, such as Computer Sciences Corp., Accenture, and HPE

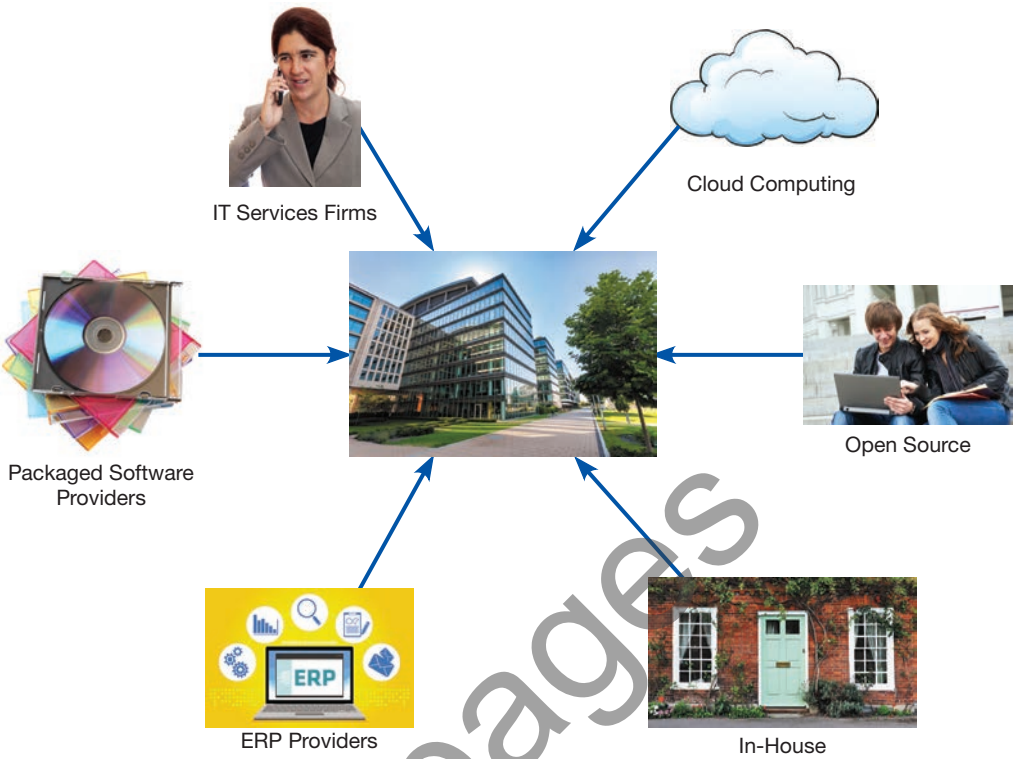

FIGURE 2-1
Sources of application software
(*Sources:* Middle: Pixachi/Shutterstock, Clockwise starting with upper left: Kamira/Shutterstock; Amit John/Pearson India Education Services Pvt. Ltd; Dmitry Kalinovsky/Shutterstock; 1000 Words/Shutterstock; Aa Amie/Shutterstock; Le Do/Shutterstock)

TABLE 2-1 Leading Software Firms and Their Development Specializations

Specialization	Example Firms or Websites
IT Services	Accenture
	Computer Sciences Corporation (CSC)
	IBM
	HPE
Packaged Software Providers	Intuit
	Microsoft
	Oracle
	SAP AG
	Symantec
Enterprise Software Solutions	Oracle
	SAP AG
Cloud Computing	Amazon.com
	Google
	IBM
	Microsoft
	Salesforce.com
Open Source	SourceForge.net

(Hewlett-Packard Enterprise). HPE, another company formerly focused on hardware, has also made the transition to an IT services firm.

Packaged Software Producers The growth of the software industry has been phenomenal since its beginnings in the mid-1960s. Some of the largest computer companies in the world are companies that produce software exclusively. A good example is Microsoft, probably the best-known software company in the world. The majority of Microsoft's revenue comes from its software sales, mostly of its Windows operating

systems and its personal productivity software, the Microsoft Office Suite. Also listed in Table 2-1, Oracle is exclusively a software company known primarily for its database software, but Oracle also makes enterprise systems. Another company on the list, SAP, is also a software-focused company that develops enterprise-wide system solutions. You will read more about Oracle and SAP shortly, in the section on enterprise systems.

Software companies develop what are sometimes called *prepackaged* or *off-the-shelf systems*. Microsoft's Word (Figure 2-2) and Intuit's Quicken, QuickBooks, and TurboTax are popular examples of such software. The packaged software development industry serves many market segments. Their software offerings range from general, broad-based packages, such as productivity tools, to very narrow, niche packages, such as software to help manage a day care center. Software companies develop software to run on many different computer platforms, from microcomputers to large mainframes. The companies range in size from just a few people to thousands of employees.

Software companies consult with system users after the initial software design has been completed and an early version of the system has been built. The systems are then tested in actual organizations to determine whether there are any problems or if any improvements can be made. Until testing is completed, the system is not offered for sale to the public.

Some off-the-shelf software systems cannot be modified to meet the specific, individual needs of a particular organization. Such application systems are sometimes

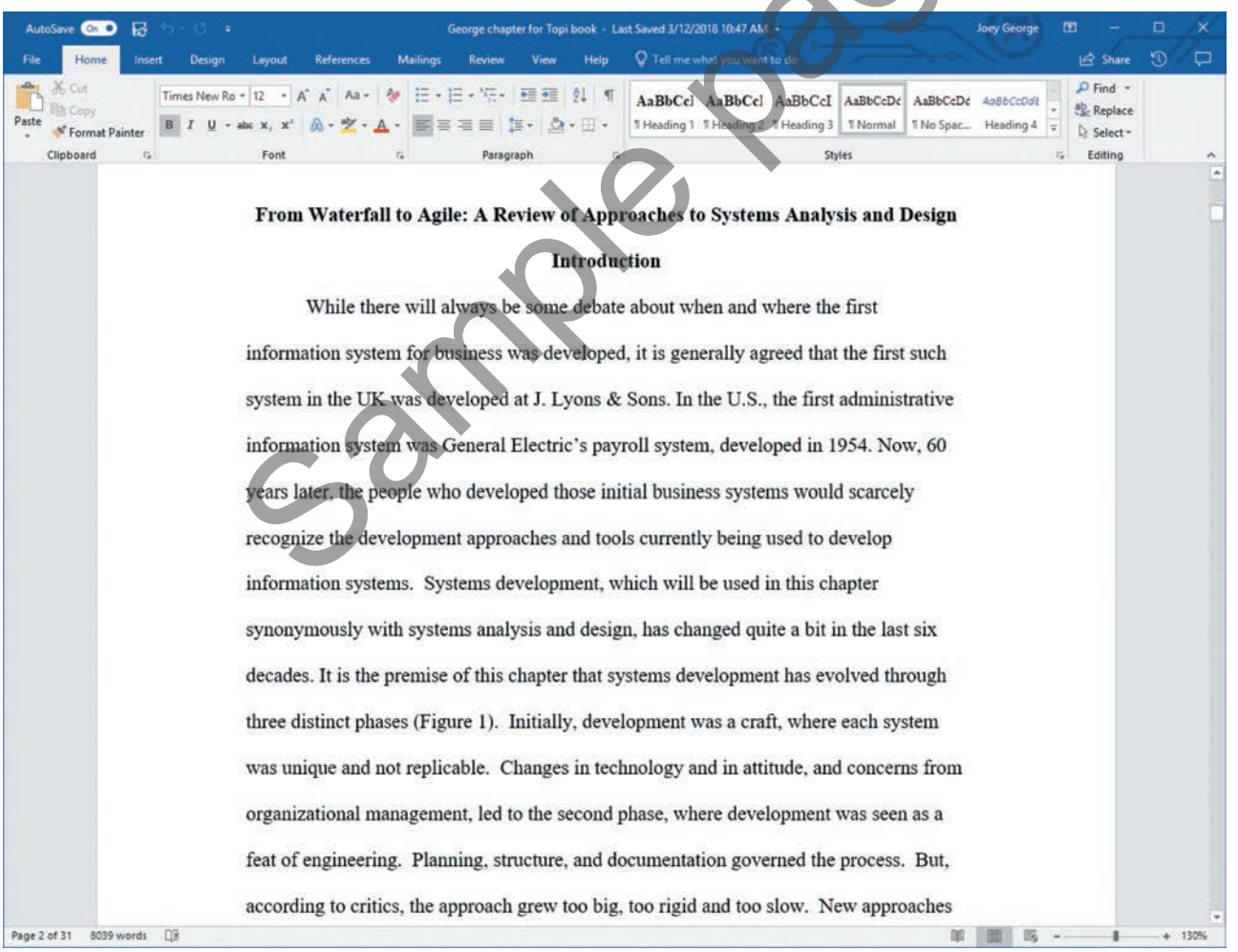

FIGURE 2-2
A document created in Microsoft's Word
(*Source:* Microsoft Corporation.)

called *turnkey systems*. The producer of a turnkey system will make changes to the software only when a substantial number of users ask for a specific change. However, other off-the-shelf application software can be modified or extended, by the producer or by the user, to more closely fit the needs of the organization. Even though many organizations perform similar functions, no two organizations do the same thing in quite the same way. A turnkey system may be good enough for a certain level of performance, but it will never perfectly match the way a given organization does business. A reasonable estimate is that off-the-shelf software can at best meet 70 percent of an organization's needs. Thus, even in the best case, 30 percent of the software system does not match the organization's specifications.

Enterprise resource planning (ERP) systems
A system that integrates individual traditional business functions into a series of modules so that a single transaction occurs seamlessly within a single information system rather than several separate systems.

Enterprise Solutions Software As mentioned in Chapter 1, many firms have chosen complete software solutions, called *enterprise solutions* or **enterprise resource planning (ERP) systems**, to support their operations and business processes. These ERP software solutions consist of a series of integrated modules. Each module supports an individual, traditional business function, such as accounting, distribution, manufacturing, or human resources. The difference between the modules and traditional approaches is that the modules are integrated to focus on business processes rather than on business functional areas. For example, a series of modules will support the entire order entry process, from receiving an order, to adjusting inventory, to shipping, to billing, to after-the-sale service. The traditional approach would use different systems in different functional areas of the business, such as a billing system in accounting and an inventory system in the warehouse. Using enterprise software solutions, a firm can integrate all parts of a business process in a unified information system. All aspects of a single transaction occur seamlessly within a single information system, rather than as a series of disjointed, separate systems focused on business functional areas.

The benefits of the enterprise solutions approach include a single repository of data for all aspects of a business process and the flexibility of the modules. A single repository ensures more consistent and accurate data, as well as less maintenance. The modules are flexible because additional modules can be added as needed once the basic system is in place. Added modules are immediately integrated into the existing system. However, there are disadvantages to enterprise solutions software. The systems are very complex, so implementation can take a long time to complete. Organizations typically do not have the necessary expertise in-house to implement the systems, so they must rely on consultants or employees of the software vendor, which can be very expensive. In some cases, organizations must change how they do business in order to benefit from a migration to enterprise solutions.

Several major vendors provide enterprise solution software. The best known is probably SAP AG, the German firm mentioned earlier, known for its flagship product R/3. SAP AG was founded in 1972, but most of its growth has occurred since 1992. Since 2010, SAP has been one of the largest suppliers of software in the world. Another major vendor of enterprise solutions is Oracle Corp., a U.S.-based firm, perhaps better known for its database software. Oracle captured a large share of the ERP market through its own financial systems and through the acquisition of other ERP vendors. At the end of 2004, Oracle acquired PeopleSoft, Inc., a U.S. firm founded in 1987. PeopleSoft began with enterprise solutions that focused on human resources management and expanded to cover financials, materials management, distribution, and manufacturing before Oracle acquired them. Global revenues for ERP software was $82.2 billion USD in 2016, with the top 10 firms accounting for 28.5 percent of the market (Pang, 2017). SAP controlled the largest single slice of the market, at 7 percent. ERP revenues are expected to hit $84.7 billion USD by 2021.

Cloud Computing Another method for organizations to obtain applications is to rent them or license them from third-party providers who run the applications at remote sites. Users have access to the applications through the Internet or through virtual private networks. The application provider buys, installs, maintains, and upgrades

the applications. Users pay on a per-use basis or they license the software, typically month to month. Although this practice has been known by many different names over the years, today it is part of **cloud computing**. Cloud computing refers to the provision of applications or related services over the Internet, where customers do not have to invest in the hardware and software resources needed to run and maintain the applications. You may have seen the Internet referred to as a cloud in other contexts, which comes from how the Internet is depicted on computer network diagrams. A well-known example of cloud computing is Google Docs, Sheets, and Slides, where users can share and create documents, spreadsheets, and presentations, respectively (Figure 2-3). Another well-known example is Salesforce.com, which provides customer relationship management software online. Cloud computing encompasses many areas of technology, including software as a service (often referred to as *SaaS*), which includes Salesforce.com, and hardware as a service, which includes Amazon Web Services and allows companies to order server capacity and storage on demand.

Cloud computing
The provision of computing resources, including applications, over the Internet, so customers do not have to invest in the computing infrastructure needed to run and maintain the resources.

The global market for public cloud computing was estimated at $209.2 billion USD in 2016, and it is projected to grow to $383.3 billion USD by 2020 (Gartner, 2017). The software as a service sector accounts for about 20 percent of the total. The companies most likely to profit immediately from the growth in cloud computing are those that can quickly adjust their product lines to meet the needs of cloud computing. These include such well-known names as IBM, which has built multiple cloud computing centers worldwide; Microsoft, whose Azure platform supports the development and operation of business applications and consumer services on its own servers; and Amazon.com, which provides storage and capacity from its own servers to customers.

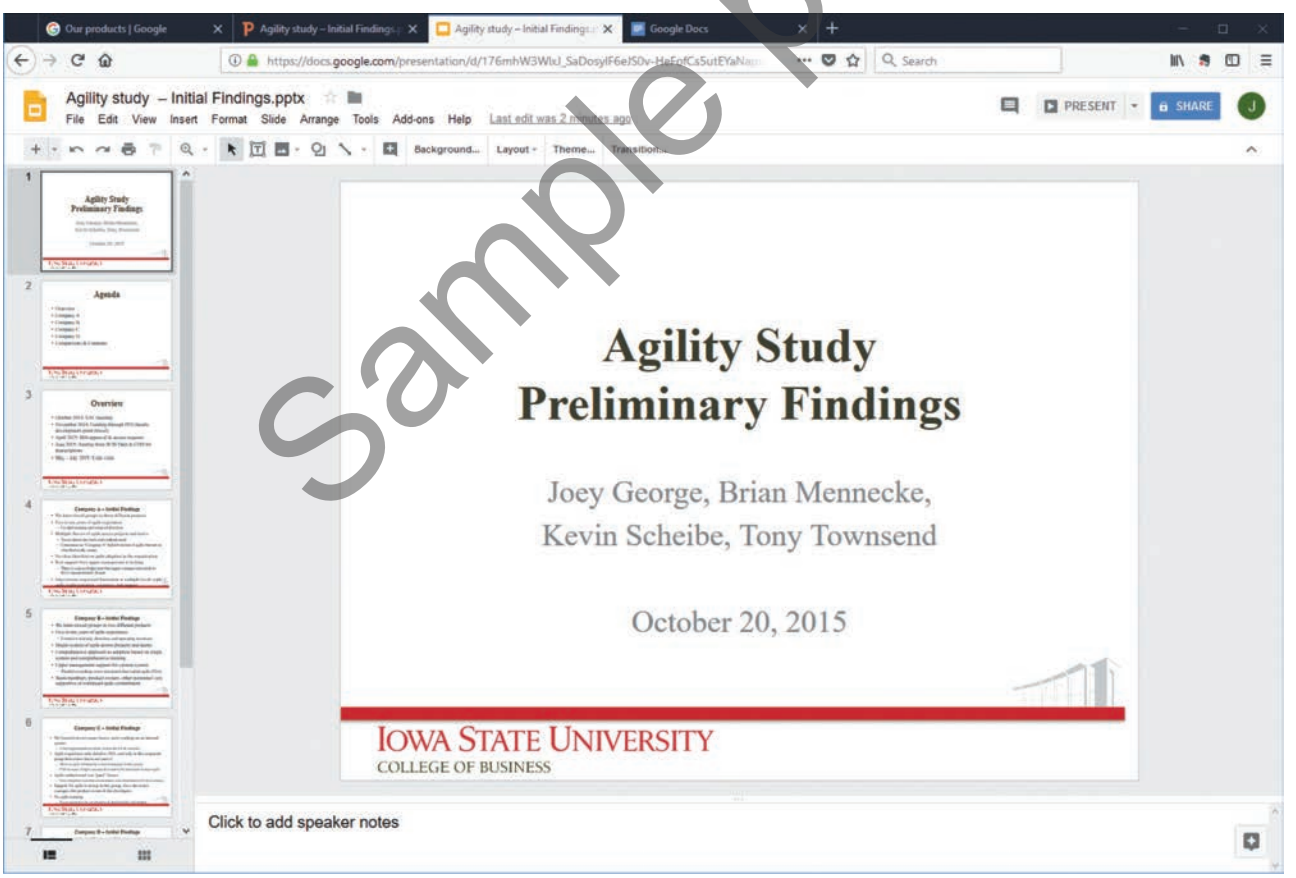

FIGURE 2-3
A presentation edited in Google Slides
(*Source:* Reprinted by permission from Joey F. George.)

Taking the cloud computing route has its advantages. The top three reasons for choosing to go with cloud computing, all of which result in benefits for the company, are (1) freeing internal IT staff, (2) gaining access to applications faster than via internal development, and (3) achieving lower-cost access to corporate-quality applications. Especially appealing is the ability to gain access to large and complex systems without having to go through the expensive and time-consuming process of implementing the systems themselves in-house. Getting your computing through a cloud also makes it easier to walk away from an unsatisfactory systems solution. Other reasons include cost effectiveness, speed to market, and better performance (Moyle & Kelley, 2011).

IT managers do have some concerns about cloud computing, however. The primary concern is over security. Concerns over security are based on storing company data on machines that one does not own and that others can access. In fact, the top two reasons for not using cloud services are concerns about unauthorized access to proprietary information and unauthorized access to customer information (Moyle & Kelley, 2011). Another concern is reliability. Some warn that the cloud is actually a network of networks, and as such, it is vulnerable to unexpected risks due to its complexity (kfc, 2012). Still another concern is compliance with government regulations, such as the Sarbanes-Oxley Act. Experts recommend a three-step process for secure migration to the cloud (Moyle & Kelley, 2011). First, have the company's security experts involved early in the migration process, so that a vendor who understands the company's security and regulatory requirements can be selected. Second, set realistic security requirements. Make sure the requirements are clearly spelled out as part of the bidding process. Third, do an honest risk assessment. Determine which data will be migrated and pay attention to how it will be managed by the cloud vendor. Once migration has occurred, it is important for companies to continue to monitor their data and systems and actively work with the vendor to maintain security.

Open-Source Software Open-source software is unlike the other types of software you have read about so far. Open-source software is different because it is freely available, not just the final product but the source code itself. It is also different because it is developed by a community of interested people instead of by employees of a particular company. Open-source software performs the same functions as commercial software, such as operating systems, e-mail, database systems, Web browsers, and so on. Some of the most well-known and popular open-source software names are Linux, an operating system; mySQL, a database system; and Firefox, a Web browser. Open source also applies to software components and objects. Open source is developed and maintained by communities of people, and sometimes these communities can be very large. Developers often use common Web resources, such as SourceForge.net, to organize their activities. As of March 2018, SourceForge.net hosted over 500,000 projects and 33 million monthly users. There is no question that the open-source movement would not be having the success it enjoys without the availability of the Internet for providing access and organizing development activities.

If the software is free, you might wonder how anybody makes any money by developing open-source software. Companies and individuals can make money with open source in two primary ways: (1) by providing maintenance and other services or (2) by providing one version of the software free and selling a more fully featured version. Some open-source solutions have more of an impact on the software industry than others. Linux, for example, has been very successful in the server software market, where it is estimated to have as much as 30 percent of the market share (Netcraft, 2017).

In-House Development We have talked about several different types of external organizations that serve as sources of software, but in-house development remains an option. In-house development has become a progressively smaller piece of all systems development work that takes place in and for organizations. As you read earlier in this chapter, internal corporate information systems departments now spend a smaller and smaller proportion of their time and effort on developing systems from scratch.

In-house development can lead to a larger maintenance burden than other development methods, such as packaged applications. A study by Banker, Davis, and Slaughter (1998) found that using a code generator as the basis for in-house development was related to an increase in maintenance hours, whereas using packaged applications was associated with a decrease in maintenance effort.

Of course, in-house development need not entail development of all of the software that will constitute the total system. Hybrid solutions involving some purchased and some in-house software components are common. If you choose to acquire software from outside sources, this choice is made at the end of the analysis phase. The choice between a package and an external supplier will be determined by your needs, not by what the supplier has to sell. As we will discuss, the results of your analysis study will define the type of product you want to buy and will make working with an external supplier much easier, more productive, and worthwhile. Table 2-2 compares the six different software sources discussed in this section.

Choosing Off-the-Shelf Software

Once you have decided to purchase off-the-shelf software rather than write some or all of the software for your new system, how do you decide what to buy? There are several criteria to consider, and special criteria may arise with each potential software purchase. For each criterion, an explicit comparison should be made between the software package and the process of developing the same application in-house. The most common criteria include the following:

- Cost
- Functionality
- Vendor support
- Viability of vendor
- Flexibility
- Documentation
- Response time
- Ease of installation

These criteria are presented in no particular order. The relative importance of the criteria will vary from project to project and from organization to organization. If you

TABLE 2-2 Comparison of Six Different Sources of Software Components

Producers	When to Go to This Type of Organization for Software	Internal Staffing Requirements
IT services firms	When task requires custom support and system can't be built internally or system needs to be sourced	Internal staff may be needed, depending on application
Packaged software producers	When supported task is generic	Some IS and user staff to define requirements and evaluate packages
Enterprise-wide solutions vendors	For complete systems that cross functional boundaries	Some internal staff necessary but mostly need consultants
Cloud computing	For instant access to an application; when supported task is generic	Few; frees up staff for other IT work
Open-source software	When supported task is generic but cost is an issue	Some IS and user staff to define requirements and evaluate packages
In-house developers	When resources and staff are available and system must be built from scratch	Internal staff necessary though staff size may vary

had to choose two criteria that would always be among the most important, those two would probably be vendor viability and vendor support. You do not want to get involved with a vendor that might not be in business tomorrow. Similarly, you do not want to license software from a vendor with a reputation for poor support. How you rank the importance of the remaining criteria will very much depend on the specific situation in which you find yourself.

Cost involves comparing the cost of developing the same system in-house with the cost of purchasing or licensing the software package. You should include a comparison of the cost of purchasing vendor upgrades or annual license fees with the costs you would incur to maintain your own software. Costs for purchasing and developing in-house can be compared based on economic feasibility measures (e.g., a present value can be calculated for the cash flow associated with each alternative).

Functionality refers to the tasks the software can perform and the mandatory, essential, and desired system features. Can the software package perform all or just some of the tasks your users need? If only some, can it perform the necessary core tasks? Note that meeting user requirements occurs at the end of the analysis phase because you cannot evaluate packaged software until user requirements have been gathered and structured. Purchasing application software is not a substitute for conducting the systems analysis phase; rather, purchasing software is part of one design strategy for acquiring the system identified during analysis.

As we said earlier, *vendor support* refers to whether vendor can provide support, and how much it can provide. Support occurs in the form of assistance with installing the software, training user and systems staff on the software, and providing help as problems arise after installation. Recently, many software companies have significantly reduced the amount of free support they will provide customers, so the cost to use telephone, on-site, fax, or computer bulletin-board support facilities should be considered. Related to support is the vendor's viability. You do not want to get stuck with software developed by a vendor that might go out of business soon. This latter point should not be minimized. The software industry is quite dynamic, and innovative application software is created by entrepreneurs working from home offices—the classic cottage industry. Such organizations, even with outstanding software, often do not have the resources or business management ability to stay in business very long. Further, competitive moves by major software firms can render the products of smaller firms outdated or incompatible with operating systems. One software firm we talked to while developing this book was struggling to survive, just trying to make its software work on any supposedly Windows PC (given the infinite combination of video boards, monitors, BIOS chips, and other components). Keeping up with hardware and system software changes may be more than a small firm can handle, and good off-the-shelf application software can be lost.

Flexibility refers to how easy it is for you, or the vendor, to customize the software. If the software is not very flexible, your users may have to adapt the way they work to fit the software. Are they likely to adapt in this manner? Purchased software can be modified in several ways. Sometimes the vendor will be willing to make custom changes for you, if you are willing to pay for the redesign and programming. Some vendors design the software for customization. For example, the software may include several different ways of processing data and, at installation time, the customer chooses which to initiate. Also, displays and reports may be easily redesigned if these modules are written in a fourth-generation language. Reports, forms, and displays may be easily customized using a process whereby your company name and chosen titles for reports, displays, forms, column headings, and so forth are selected from a table of parameters you provide. You may want to employ some of these same customization techniques for systems developed in-house so that the software can be easily adapted for different business units, product lines, or departments.

Documentation includes the user's manual as well as technical documentation. How understandable and up-to-date is the documentation? What is the cost for multiple copies, if required? *Response time* refers to how long it takes the software package

to respond to the user's requests in an interactive session. Another measure of time would be how long it takes the software to complete running a job. Finally, ease of installation is a measure of the difficulty of loading the software and making it operational.

Of course, the criteria for software acquisition will vary with the type of system you are acquiring. For example, if you are thinking about licensing an ERP system, you will certainly take all of the prior criteria into account, but you will also want to investigate criteria that are specific to ERP systems. Verville, Bernadas, and Halingten (2005) studied organizations that had acquired ERP systems to discover what the critical factors were for success. They found 10 success factors, with 5 related to the acquisition process and 5 related to the people in the process. They found the acquisition process had to be highly planned and structured, and it had to be rigorous. For the process to be successful, nothing could be overlooked during planning. It was important that two of the five success factors related to the process were completed before ERP vendors were contacted. These two factors were determining all of the system requirements and establishing the selection and evaluation criteria. They helped the organizations compose clear descriptions of their needs and evaluate bids from vendors. The fifth process-related criterion was obtaining accurate information. Information sources needed to be verified and cross-validated.

The other five success factors dealt with the people involved in the acquisition process. The first factor was clear and unambiguous authority. The person in charge of the process needed to be objective and a strong leader. Second, the composition of the acquisition team was important. The team needed to be diverse, with each member having a particular skill set that was complementary to those of the other team members. Third, it was considered important to approach the relationship with the vendor as a partnership, as opposed to an adversarial or neutral relationship. Given the complexity and cost of ERP systems, members of the acquiring organization would be working with the vendors for several years, so a comfortable working relationship was essential. Fourth, future users of the ERP system were active participants in the acquisition process. Lastly, the fifth success factor related to people in the process was user buy-in. In the companies studied, user buy-in often translated into user acceptance of the system and even enthusiasm and excitement about it.

Validating Purchased Software Information

One way to get all of the information you want about a software package is to collect it from the vendor. Some of this information may be contained in the software documentation and technical marketing literature. Other information can be provided upon request. For example, you can send prospective vendors a questionnaire, asking specific questions about their packages. This may be part of a **request for proposal (RFP)** or a request for quote (RFQ) process your organization requires when major purchases are made. Space does not permit us to discuss the topic of RFPs and RFQs here; you may wish to refer to purchasing and marketing texts if you are unfamiliar with such processes (additional references about RFPs and RFQs are found at the end of this chapter).

Request for proposal (RFP)
A document provided to vendors that asks them to propose hardware and system software that will meet the requirements of a new system.

There is, of course, no replacement for actually using the software yourself and running it through a series of tests based on your software selection criteria. Remember to test not only the software, but also the documentation, training materials, and even the technical support facilities. One requirement you can place on prospective software vendors as part of the bidding process is that they install (free or at an agreed-upon cost) their software for a limited amount of time on your computers. This way you can determine how their software works in your environment, not in some optimized environment they have for demonstration purposes.

One of the most reliable and insightful sources is other users of the software. Vendors will usually provide a list of customers (remember, they will naturally tell you about satisfied customers, so you may have to probe for a cross section of customers)

and people who are willing to be contacted by prospective customers. And here is where your personal network of contacts, developed through professional groups, college friends, trade associations, or local business clubs, can be a resource; do not hesitate to find some contacts on your own. Such current or former customers can provide a depth of insight on the use of a package at their organizations.

To gain a range of opinions about possible packages, you can use independent software testing and abstracting services that periodically evaluate software and collect user opinions. Such surveys are available for a fee either as subscription services or on demand (two popular services are Auerbach Publishers and DataPro); occasionally, unbiased surveys appear in trade publications. Often, however, articles in trade publications, even software reviews, are actually seeded by the software manufacturer and are not unbiased.

If you are comparing several software packages, you can assign scores for each package on each criterion and compare the scores using the quantitative method we demonstrate in Chapter 4 for comparing alternative system design strategies.

REUSE

Reuse
The use of previously written software resources, especially objects and components, in new applications.

Reuse is the use of previously written software resources in new applications. Because so many bits and pieces of applications are relatively generic across applications, it seems intuitive that great savings can be achieved in many areas if those generic bits and pieces do not have to be written anew each time they are needed. Reuse should increase programmer productivity because being able to use existing software for some functions means they can perform more work in the same amount of time. Reuse should also decrease development time, minimizing schedule overruns. Because existing pieces of software have already been tested, reusing them should also result in higher-quality software with lower defect rates, decreasing maintenance costs.

Although reuse can conceivably apply to many different aspects of software, typically it is most commonly applied to two different development technologies: object-oriented and component-based development. You were briefly introduced to object-oriented development in Chapter 1. Following is a brief introduction to object-oriented development. For example, consider an object class created to model an employee. The Employee object class would contain both the data about employees and the instructions necessary for calculating payroll for a variety of job types. The object class could be used in any application that dealt with employees, but if changes had to be made in calculating payroll for different types of employees, the changes would have to be made only to the object class and not to the various applications that used it. By definition, using the Employee object class in more than one application constitutes reuse.

Component-based development is similar to object-oriented development in that the focus is on creating general-purpose pieces of software that can be used interchangeably in many different programs. Components can be as small as objects or as large as pieces of software that handle single business functions, such as currency conversion. The idea behind component-based development is the assembly of an application from many different components at many different levels of complexity and size. Many vendors are working on developing libraries of components that can be retrieved and assembled, as needed, into desired applications.

Reuse has become a standard part of corporate systems development. However, for reuse to work in an organizational setting, many different issues must be addressed. The most important factors related to successful reuse include knowledge of the domain in which reuse is to occur, customer support, the commitment and understanding of senior management, sound organizational processes, and skilled and experienced developers (Bombonatti, Goulao, & Moreira, 2017). These same factors can be associated with reuse failure, if developers lack the appropriate domain knowledge, if senior management does not promote reuse, if there are no organization wide reuse

standards, and if developers are resistant to reuse or lack the skills to make it successful. Royce (1998) argues that, due to the considerable costs of developing a reusable component, most organizations cannot compete economically with established commercial organizations that focus on selling components as their main line of business. Success depends on being able to leverage the cost of components across a large user and project base (Figure 2-4).

When an organization's management decides to pursue reuse as a strategy, it is important for the organization to match its approach to reuse with its strategic business goals (Griss, 2003). The benefits of reuse grow as more corporate experience is gained from it, but so do the costs and the amount of resources necessary for reuse to work well. Software reuse has three basic steps: abstraction, storage, and recontextualization (Grinter, 2001). Abstraction involves the design of a reusable piece of software, starting from existing software assets or from scratch. Storage involves making software assets available for others to use. Although it sounds like a simple problem, storage can actually be very challenging. The problem is not simply putting software assets on a shelf; the problem is correctly labeling and cataloging assets so that others can find the ones they want to use. Once an asset has been found, recontextualization becomes important. This involves making the reusable asset understandable to developers who want to use it in their systems. Software is complex, and a software asset developed for a particular system under system-specific circumstances may not at all be the asset it appears to be. What appears to be a generic asset called "customer" may actually be something quite different, depending on the context in which it was developed. It may often appear to be easier to simply build your own assets rather than invest the time and energy it takes to establish a good understanding of software someone else has developed. A key part of a reuse strategy, as mentioned previously, is establishing rewards, incentives, and organizational support for reuse to help make it more worthwhile than developing your own assets.

According to Griss (2003), an organization can take one of four approaches to reuse (Table 2-3). The ad hoc approach to reuse is not really an approach at all, at least from an official organizational perspective. With this approach, individuals are free to find or develop reusable assets on their own, and there are few, if any, organizational rewards for reusing assets. Storage is not an issue, because individuals keep track of and distribute their own software assets. For such an ad hoc, individually driven approach, it is difficult to measure any potential benefits to the company.

Another approach to reuse is *facilitated* reuse. With this approach, developers are not required to practice reuse, but they are encouraged to do so. The organization makes available some tools and techniques that enable the development and sharing

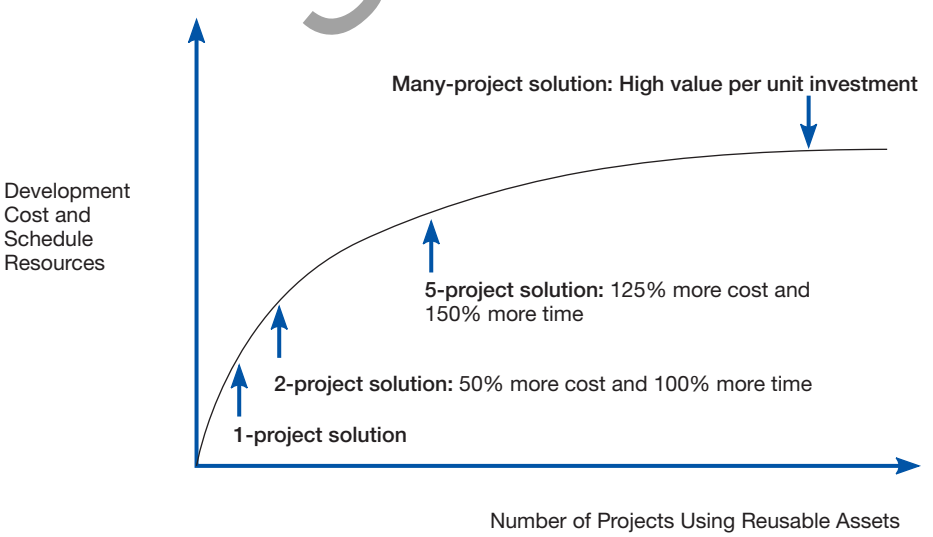

FIGURE 2-4
Investments necessary to achieve reusable components

(*Source:* Royce, Walker, Software Project Management: A Unified Framework, 1st ed., ©1998. Reprinted and Electronically reproduced by permission of Pearson Education, Inc. Upper Saddle River, New Jersey.)

TABLE 2-3 Four Approaches to Reuse

Approach	Reuse Level	Cost	Policies and Procedures
Ad hoc	None to low	Low	None.
Facilitated	Low	Low	Developers are encouraged to reuse but are not required to do so.
Managed	Moderate	Moderate	Development, sharing, and adoption of reusable assets are mandated; organizational policies are established for documentation, packaging, and certification.
Designed	High	High	Reuse is mandated; policies are put in place so that reuse effectiveness can be measured; code must be designed for reuse during initial development, regardless of the application it is originally designed for; there may be a corporate office for reuse.

(*Source:* Based on Griss, 2003.)

of reusable assets, and one or more employees may be assigned the role of evangelist to publicize and promote the program. Very little is done to track the quality and use of reusable assets, however, and the overall corporate investment is small.

Managed reuse is a more structured, and more expensive, mode of managing software reuse. With managed reuse, the development, sharing, and adoption of reusable assets is mandated. The organization establishes processes and policies for ensuring that reuse is practiced and that the results are measured. The organization also establishes policies and procedures for ensuring the quality of its reusable assets. The focus is on identifying existing assets that can be potentially reused from various sources, including from utility asset libraries that come with operating systems, from companies that sell assets, from the open-source community, from internal repositories, from scouring existing legacy code, and so on.

The most expensive and extensive approach to reuse is *designed reuse*. In addition to mandating reuse and measuring its effectiveness, the designed reuse approach takes the extra step of mandating that assets be designed for reuse as they are being designed for specific applications. The focus is more on developing reusable assets than on finding existing assets that might be candidates for reuse. A corporate reuse office may be established to monitor and manage the overall methodology. Under such an approach, as much as 90 percent of software assets may be reused across different applications.

Each approach to reuse has its advantages and disadvantages. No single approach is a silver bullet that will solve the reuse puzzle for all organizations and for all situations. Successful reuse requires an understanding of how reuse fits within larger organizational goals and strategies, as well as an understanding of the social and technical world into which the reusable assets must fit.

SUMMARY

As a systems analyst, you must be aware of where you can obtain software that meets some or all of an organization's needs. You can obtain application (and system) software from information technology services firms, packaged software providers, vendors of enterprise-wide solution software, cloud computing vendors, and open-source software providers, as well as from internal systems development resources, including the reuse of existing software components. You can even hire an organization to handle all of your systems development needs, which is called *outsourcing*. You must also know the criteria to use when choosing among off-the-shelf software products. These criteria include cost, functionality, vendor support, vendor viability, flexibility, documentation, response time, and ease of installation. Requests for proposals are one way you can collect more information about system software, its performance, and its costs.

KEY TERMS

2.1 Cloud computing
2.2 Enterprise resource planning (ERP) systems
2.3 Outsourcing
2.4 Request for proposal (RFP)
2.5 Reuse

Match each of the key terms above with the definition that best fits it.

____The practice of turning over responsibility for some or all of an organization's information systems applications and operations to an outside firm.

____A system that integrates individual traditional business functions into a series of modules so that a single transaction occurs seamlessly within a single information system, rather than several separate systems.

____A document that is provided to vendors to ask them to propose hardware and system software that will meet the requirements of your new system.

____The use of previously written software resources, especially objects and components, in new applications.

____The provision of computing resources, including applications, over the Internet so customers do not have to invest in the computing infrastructure needed to run and maintain computing resources.

REVIEW QUESTIONS

2.6 What is outsourcing? Identify two outsourcing arrangements. Identify two reasons for outsourcing.

2.7 List the six sources of software used by organizations.

2.8 Why would an analyst need to have an awareness of outsourcing as an alternative to handling IT services in-house?

2.9 Why would an organization outsource information systems operations?

2.10 Briefly explain why systems development is more difficult now than it was in the past.

2.11 What are enterprise resource planning systems? How do they differ from traditional approaches? Identify three enterprise resource planning system vendors.

2.12 List the various development specializations and provide an example from the leading software firms for each.

PROBLEMS AND EXERCISES

2.13 Research how cloud computing can be used as a PaaS.

2.14 Search the Internet for reasons for outsourcing other than those listed in this chapter. Explain each of these reasons.

2.15 An organization that deals with a large amount of data wants to implement a database management system (DBMS) to properly store and manage it. This organization has shortlisted Oracle, MySQL, and Microsoft SQL Server as the top DBMS tools available in the market. List the features provided by these DBMS tools.

2.16 Suppose you are an IT manager at your institute. List any two open-source software packages besides Microsoft Office that can be used for routine work in your office. Download both and test them out to decide which one is better suited to your work requirements.

FIELD EXERCISES

2.17 Interview businesspeople who participate in the purchase of off-the-shelf software in their organizations. Review with them the criteria for selecting off-the-shelf software presented in this chapter. Have them prioritize the list of criteria as they are used in their organization and provide an explanation of the rationale for the ranking of each criterion. Ask them to list and describe any other criteria that are used in their organization.

2.18 Obtain copies of actual RFPs used for information systems developments and/or purchases. If possible, obtain RFPs from public and private organizations. Find out how they are used. What are the major components of these proposals? Do these proposals seem to be useful? Why or why not? How and why do RFPs from public and private organizations differ?

2.19 Contact an organization that has implemented or is implementing an integrated ERP application. Why did it choose this design strategy? How has it managed this development project differently from prior large projects? What organizational changes have occurred due to this design strategy? How long did the implementation last, and why?

REFERENCES

ATKearney. (2017). The widening impact of automation. Retrieved March 26, 2018 from http://www.atkearney.com/documents/20152/793366/The+Widening+Impact+of+Automation.pdf/.

Banker, R. D., Davis, G. B., & Slaughter, S. A. (1998). Software development practices, software complexity, and software maintenance performance: A field study. *Management Science,* 44(4), 433–50.

Bombonatti, D., Goulao, M., & Moreira, A. (2017). Synergies and tradeoffs in software reuse–A systematic mapping study. *Software: Practice and Experience,* 47, 943–957.

Computer History Museum. (2003). Timeline of computer history. Retrieved February 14, 2009 from http://www.computerhistory.org.

Gartner. (2017). Gartner says worldwide public cloud service market to grow 18 percent in 2017. Accessed March 26, 2018 from http://www.gartner.com.

Grinter, R. E. (2001). From local to global coordination: Lessons from software reuse. In C. A. Ellis (Ed.), *Group'01: Proceedings of the 2001 International ACM SIGGROUP Conference on Supporting Group Work* (pp. 144–53). Boulder, CO: Association for Computing Machinery.

Griss, M. (2003). *Reuse comes in several flavors* [Flashline white paper]. Retrieved February 10, 2009 from http://www.flashline.com.

Ketler, K., & Willems, J. R. (1999). A study of the outsourcing decision: Preliminary results. In *Proceedings of the 1999 ACM SIGCPR Conference on Computer Personnel Research* (pp. 182–89). Boulder, CO: Association for Computing Machinery.

kfc. (2012). The hidden risk of a meltdown in the cloud." *Technology Review*. Retrieved April 17, 2012 from http://www.technologyreview.com/printer_friendly_blog.aspx?id=27642.

King, R. (2008a, April 7). The new economics of outsourcing. *BusinessWeek*. Retrieved August 25, 2012 from http://www.businessweek.com/stories/2008-04-07/the-new-economics-ofoutsourcingbusinessweek-business-news-stock-market-and-financial-advice.

Moyle, E., & Kelley, D. (2011). Cloud security: Understand the risks before you make the move. *InformationWeek Analytics*. Retrieved April 14, 2012 from http://analytics.informationweek.com.

Netcraft. (2017). September 2017 Web Server Survey. Retrieved March 30, 2018 from http://news.netcraft.com/archives/2017/09/11/september-2017-web-server-survey.html.

Pang, A. (2017). Top 10 ERP software vendors and market forecast 2016-2021. Retrieved March 26, 2018 from http://www.appsruntheworld.com.

Royce, W. (1998). *Software project management: A unified framework*. Boston: Addison-Wesley.

Statistica. (2018). Global outsourcing industry revenue from 2010 to 2017, by service type (in billion U.S. dollars). Retrieved March 26, 2018 from http://www.statista.com/statistics/189800/global-outsourcing-industry-revenue-by-service-type/.

Verville, J., Bernadas, C., & Halingten, A. (2005). So you're thinking of buying an ERP? Ten critical factors for successful acquisition. *Journal of Enterprise Information Management*, 18(6), 665–77.

PETRIE ELECTRONICS

Chapter 2: The Origins of Software

Jim Watanabe looked around his new office. He couldn't believe that he was the assistant director of information technology at Petrie Electronics, his favorite consumer electronics retail store. He always bought his new DVDs and video games for his Xbox 360 at Petrie. In fact, he bought his Blu-ray player and his Xbox 360 at Petrie, along with his surround sound system and his 40-inch flat-screen HD LED TV. And now he worked there, too. The employee discount was a nice perk[1] of his new job, but he was also glad that his technical and people skills were finally recognized by the people at Petrie. He worked for five years at Broadway Entertainment Company as a senior systems analyst, and it was clear that he was not going to be promoted there. He was really glad he posted his résumé on Monster.com and that now he had a bigger salary and a great job with more responsibility at Petrie.

Petrie Electronics started as a single electronics store in 1984 in San Diego, California. The store was started by Jacob Rosenstein in a strip mall. It was named after Rob Petrie, the TV writer played by Dick Van Dyke in the TV show of the same name. Rosenstein always liked that show. When he had grown the store to a chain of 13 locations in Southern California, the business became too much for Rosenstein to handle. He sold out in 1992, for a handsome profit, to the Matsutoya Corporation, a huge Japanese conglomerate that saw the chain of stores as a place to sell its many consumer electronics goods in the United States.

Matsutoya aggressively expanded the chain to 218 stores nationwide by the time they sold it in 2002, for a handsome profit, to Sam and Harry's, a maker and seller of ice cream. Sam and Harry's was looking for a way to diversify and invest the considerable cash they made creating and selling ice cream, with flavors named after actors and actresses, like their best-selling Lime Neeson and Jim Carrey-mel. Sam and Harry's brought in professional management to run the chain, and since they bought it, they had added 15 more stores, including 1 in Mexico and 3 in Canada. Even though they originally wanted to move the headquarters to their home-base state of Delaware, Sam and Harry decided to keep Petrie headquartered in San Diego.

The company had made some smart moves and had done well, Jim knew, but he also knew that competition was fierce. Petrie competitors included big electronics retail chains like BestBuy. In California, Fry's was a ferocious competitor. Other major players in the arena included the electronics departments of huge chains like Walmart and Target and online vendors like Amazon.com. Jim knew that part of his job in IT was to help the company grow and prosper and beat the competition—or at least survive.

[1]perquisite

Just then, as Jim was trying to decide if he needed a bigger TV, Ella Whinston, the CEO at Petrie, walked into his office. "How's it going, Jim? Joe keeping you busy?" Joe was Joe Swanson, Jim's boss, the director of IT. Joe was away for the week, at a meeting in Tucson, Arizona. Jim quickly pulled his feet off his desk.

"Hi, Ella. Oh, yeah, Joe keeps me busy. I've got to get through the entire corporate strategic IT plan before he gets back—he's going to quiz me—and then there's the new help desk training we're going to start next week."

"I didn't know we had a strategic IT plan," Ella teased. "Anyway, what I came in here for is to give you some good news. I want you to be the project manager for a project that's crucial to our corporate survival."

"Me?" Jim said, "But I just got here."

"Who better than you? You have a different perspective, new ideas. You aren't chained down by the past and by the Petrie way of doing things, like the rest of us. Not that it matters, since you don't have a choice. Joe and I both agree that you're the best person for the job."

"So," Jim asked, "what's the project about?"

"Well," Ella began, "the executive team has decided that the number one priority we have right now is to not only survive but to thrive and prosper, and the way to do that is to develop closer relationships with our customers. The person on the executive team who's even more excited about this than me is John Smith, head of marketing. We want to attract new customers, like all of our competitors. But also like our competitors, we want to keep our customers for life, kind of like a frequent-flier program, but better. Better for us and for our loyal customers. And we want to reward most the customers who spend the most. We're calling the project 'No Customer Escapes.'"

"I hope that's only an internal name," Jim joked. "Seriously, I can see how something like this would be good for Petrie, and I can see how IT would play an important, no, crucial role in making something like this happen. So, what's the next step in getting the project approved?"

Case Questions

2.20 How do information systems projects get started in organizations?

2.21 How are organizational information systems related to company strategy? How does strategy affect the information systems a company develops and uses?

2.22 Research customer loyalty programs in retail firms. How common are they? What are their primary features?

2.23 What do you think Jim's next step would be? Why?

2.24 Why would a systems analyst new to a company be a good choice to lead an important systems development effort?