Solution Manual for Modern Systems Analysis and Design 8th Edition Valacich George 0134204921 9780134204925
Full link download:
Test Bank:
https://testbankpack.com/p/test-bank-for-modern-systems-analysis-and-design-8th-edition-valacich-george-0134204921-9780134204925/

Solution Manual:
https://testbankpack.com/p/solution-manual-for-modern-systems-analysis-and-design-8th-edition-valacich-george-0134204921-9780134204925/

Chapter 2
The Origins of Software

Chapter Overview

The primary purposes of Chapter 2 are to show students that not all of the software associated with a systems development project is developed in-house and to emphasize that analysts should consider several design strategies based on the organization's resources before choosing one to pursue for further development in design. The secondary purpose is to emphasize that the consideration of a packaged software solution should be done after the analysis efforts are complete, not as a substitute for analysis.

The chapter discusses six sources for software: Information Technology Services Firms, packaged software producers, enterprise-wide solutions (ERP), cloud computing, open-source and in-house development. In addition, the reasoning that should be followed when choosing among the many options available to an analysis team for developing design strategies. The point is that the "make-versus-buy" decision is not a choice of one or the other, but is in reality a spectrum of choices ranging from make at one end and buy at the other. Just as important, more choices these days are made toward the buy end of the scale. The Request for Proposal (RFP) is shown as an important element to understanding how analysis and design requirements must be translated into a document that external source organizations can review for bid. The chapter includes a discussion of outsourcing, an option for systems development and management that may not occur to many students in their first systems development course. The chapter explains reuse and the four different approaches to reuse.

Instructional Objectives

Specific student learning objectives are included at the beginning of the chapter. *From an instructor's point of view, the objectives of this chapter are to:*

1. Show students that in-house developed systems are not the only source for software.

2. Six sources for software need to be understood: Information Technology Services Firms, packaged software producers, enterprise-wide solutions (ERP), Cloud Computing Services, open-source, and in-house developers. An understanding of the advantages and disadvantages of each must be shown.

3. Cloud Computing is likely to be new to students and even though they may have heard the term, it is important to ensure that they understand what it is and its advantages. Emphasize that it enables customers (firms) to use hardware and software that is not installed on their computers but rather to access services over the Internet of a Virtual Private Network (VPN) on a pay-for-use basis. Emphasize the three key advantages of cloud computing: (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. Also mention that cost savings are achieved from elastic leasing of pooled resources dynamically resulting in lower costs by paying only for the resources actually used. This is referred to as scalability. Be sure to mention the security concerns associated with cloud computing.

4. Show students how to evaluate off-the-shelf software and why it is important to do thorough analysis first. Review the criteria to consider when purchasing off-the-shelf software (p.35). Also explain that

the claims made by software and hardware vendors need to be validated by someone outside the vendor organization, such as current users and independent software testing centers. Emphasize that vendor information may be biased and even trade publication articles may describe software in a more than realistic positive light. When in doubt, check the information out.

5. Discuss the importance of software reuse and how object-oriented and component-based development are the two most common reuse methods. Also emphasize that reuse must be aligned with the organization's overall strategic goals. Explain the four levels of adoption currently being seen in the industry: *ad hoc*, facilitated, managed, and designed (Table 2-3 lists the four approaches).

Classroom Ideas

1. Use Table 2-1 and an updated version of the same information from the most recent Software Magazine survey to begin a discussion of the many, varied sources of software in the marketplace.

2. Use Table 2-2 to summarize the six alternative sources for software and how to choose among them for specific software needs. This table can serve as the basis for a discussion of the make- versus-buy decision and can be expanded to include the "not invented here" syndrome. Additionally, invite a guest speaker who is currently responsible for software procurement and have them discuss the advantages and disadvantages of the sources of software.

3. Have students research the proper format and contents for Requests for Proposal and have them create and/or present an RFP (see Problem and Exercise 1). RFP preparation should include discussion of the hardware, software, and organizational issues presented in this chapter.

4. Find a local guest speaker (from your alumni base or a recruiter coming to your campus) from an organization that employs object-oriented design to come in and discuss the level of reuse and the real-world issues involved in their organization with promoting the concept, given some of the startup costs and constraints.

Answers to Key Terms

Suggested answers are provided below. These answers are presented top-down, left-to-right.

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

Answers to Review Questions

2.6. Six sources of software are identified in the text. These include: 1) Information Technology Services Firms, 2) packaged software producers, 3) enterprise-wide solution software, 4) cloud computing, 5) open-source, and 6) in-house development software.

Information Technology Services firms are used by companies who do not have expertise or personnel to develop IS systems. These firms have experts in the development, hosting, and running of applications and other services to fit a customer's specifications.

Packaged software producers develop a vast number of applications for different markets that fits a large market segment. Prepackaged solutions may range from general, broad-based to narrow, niche packages that can run on a variety of platforms.

Enterprise solutions (ERP) consist of a series of integrated modules; these modules are integrated to focus on business processes rather than on business functional areas allowing companies to store data in only one area without duplication. This allows the use of a single repository ensuring more accurate and consistent data with less maintenance.

Cloud Computing is the provision of applications over the Internet or a virtual private network (VPN) such that customers do not need to invest in hardware and software infrastructure and can pay on a per-use basis. A key advantage is that server and storage capacity can be ordered on demand as needed. Information security remains a concern when considering cloud computing because of 3rd party control of the applications.

Open-source software has risen in popularity because of the free availability of not only the product but the source code as well. This software is developed and maintained by a community of like-minded people dedicated to improving source code access, with Linux, MySQL, and Firefox being the most prevalent examples.

In-house development requires the resources, especially trained staff, to develop software targeted to an organization's own specific needs. Fewer companies are going this route today because of the expertise needed and the high costs of development.

Table 2-2 compares the six sources of software components.

2.7. When deciding what off-the-shelf software to buy, you should compare products and vendors. Additional criteria include (among others that are more situation-specific): cost, functionality, vendor support, vendor viability, flexibility, documentation, response time, and ease of installation. Vendor viability and vendor support are probably the two most important.

2.8. A Request for Proposal (RFP) is a formal document that provides detailed specifications about a target information system and asks vendors for information on how they would develop the system. Analysts use RFPs as a way to get vendors to perform the research to determine what application design will meet user requirements and the hardware and systems software vendors believe are necessary for developing the new system.

2.9. To verify vendor claims about a software package, an analyst can ask for a software demonstration, use the software (and its documentation and training materials) personally, talk with other users of the software, and consult independent software testing and abstracting services (surveys available for a fee). It is important to make sure that the system fits your organization's needs.

2.10. Enterprise resource planning systems consist of a series of integrated modules; these modules pertain to specific, traditional business functions. However, these modules are integrated to focus on business processes rather than on business functional areas. Enterprise resource planning systems' advantages include a single repository of data for all aspects of a business process, flexible modules,

less maintenance, more consistent and accurate data, and ease of adding and integrating new modules into the existing system. Possible disadvantages of this approach include complexity, lengthy implementation time, lack of in-house expertise, expense, and changing how the

organization conducts its business. These projects when accomplished successfully are most often approached as an institutional change project, not simply an IT project.

2.11. Reuse is the use of previously written software resources that can be reused in new applications. It most often is applied to object-oriented and component-based development. Creating, storing, and maintaining objects and components that can be drawn on again and again for new applications is the objective. Reuse should in theory increase programmer productivity, decrease development time, minimize errors, and schedule overruns. Ultimately it should produce higher quality work with fewer defects and thus reduce overall implementation and maintenance time. In current practice, due to high initial startup costs, lack of good quality methods for labeling, storage, combined with lack of senior management commitment, reuse is not practiced as often as it could be. Additionally, lack of incentives and rewards to design for and apply reuse concepts and the overall difficulty in measuring economic gain from its application all conspire against reuse in the real world. Note Figure 2-5 on the high initial startup costs when a high level of reuse is planned. As more organizations achieve success and as more componentization takes place in the *for purchase* arena more organizations will have incentive to integrate reuse into their business strategy.

2.12. In comparing and contrasting the four approaches to reuse, the student should note the advantages and disadvantages listed in Table 2-3. Note also that no one type yields the best possible solution. Successful reuse requires an understanding of how reuse fits within larger organizational goals and strategies.

Answers to Problems and Exercises

2.13. An organization uses the Request for Proposal (RFP) to solicit proposals from several competing vendors. Usually, RFPs first provide background information on the company and the business units involved in the request, an explanation of the information systems needs, a description of what is wanted from the vendors (i.e., what information they must provide or other actions they must take), and an explanation of any rules or procedures for the RFP and system development process. The bulk of the document then describes the mandatory, essential, and desirable requirements in the areas of need (e.g., functionality, hardware, software, and service). Students' RFP outlines should include these key features.

2.14. In addition to cost, functionality, vendor support, vendor viability, flexibility, documentation, response time, and ease of installation, a number of other "real-world" criteria might be included. People often choose application packages, such as word processors and spreadsheets, based solely on their familiarity with the packages and/or their bias toward one hardware platform or operating system over another. To a certain extent this is functional. On the other hand, this can be a disadvantage; for example, it is useful to consider the current staff's familiarity with the new application software and the resulting need for retraining but if a company does not choose new software because of the employees' lack of familiarity with the software, they run the risk of being left behind using antiquated technology. Additional criteria include compatibility with currently used application software (so, for example, data can be shared), compatibility with existing hardware and system software, ability to support a range from novice to experienced (or power) users, and appeal of the user interface (ease of use).

2.15. The list for evaluating alternative custom software developers is similar to that for selecting off-the-shelf application software or for computer hardware and system software. In addition to cost,

functionality, vendor support, vendor viability, flexibility, documentation, response time, and ease of installation, you might include the current staff's familiarity with the software, need for retraining, compatibility and connectivity with current systems, and the track record of the vendor in

successfully implementing similar software in other organizations. Such vendors should have an established track record for developing similar software in other organizations. Their references should be checked thoroughly, including visits to other sites. Additionally, factors such as the vendor's employee turnover rate and history should be considered to ensure the same level of skill and talent is available. Vendor capability may change over time. If the developer's role ends after the application is accepted, then the vendor's reputation for handling this transition from external development to internal maintenance is important. From a legal point of view, you may want to select a custom developer based on the willingness to sign a non-disclosure agreement, so that he is not allowed to develop a similar system for one of your competitors, at least for a certain amount of time.

2.16. The project team can use the advantages of the enterprise resource planning design as part of its strategy for selling this system. The team can stress that this solution consists of a series of integrated modules; these modules are integrated to focus on business processes, and the firm can integrate all parts of a business process. This approach includes a single repository of data, thus providing more consistent and accurate data and less maintenance. These modules are flexible and new modules are easily integrated into an existing system.

Also, an enterprise resource planning (ERP) system might be justified on the following grounds: (1) it is a complete enterprise-wide solution that models all aspects of each transaction, supposedly seamlessly and within a single system; (2) an ERP system is based on a single repository of all corporate data, which implies consistency, accuracy, and flexibility of the data; and (3) adding new modules should be relatively painless as all modules are specifically designed to work together. On the other hand, some might counter that ERP systems are complex and require expensive outside expertise for implementation, and that the organization itself has to adjust to fit the software's model of how organizations should operate rather than the reverse.

In both cases, the team may also employ testimonials and reports of results from other organizations that have benefited from the ERP strategy. Other users can be both a reliable and insightful source of information.

Guidelines for Using the Field Exercises

2.17. Business people are likely to use this list of criteria in some formal or informal way. Have your students present their findings to the class so that they can learn about a variety of companies. It is useful for students to see how companies actually use these criteria and methods in purchasing off-the-shelf software. Encourage students to understand why their interviewees prioritized the criterion list in such a manner as different companies will have different lists of priorities.

2.18. Obtaining RFPs may be difficult for your students. It may be necessary for you to obtain copies of RFPs either from business contacts or from the university. You may have to file a Freedom of Information Act (FOIA) request in order to obtain RFPs from public organizations. It is very useful for students to see real RFPs. Students are amazed at how lengthy and detailed these are for larger, more complex systems and at how complicated RFPs can become for governmental agencies. Help students understand that government agencies are subject to federal laws that do not always apply to private organizations.

2.19. Start by contacting larger organizations (as they are more likely to employ ERP) where alumni from your university or college are now employed and help your students "break the ice." Also, most if not all universities now have comprehensive ERP instillations that the IT folks would be willing to talk about. Chances are any organization that a student contacts about its ERP

implementation will have a lot to say, provided the IT staff is willing to share the information. Typically, ERP implementations take several years and cost quite a bit in terms of consultant fees. There are many reasons to move to an ERP system, just as there are many reasons not to move to an ERP system. The organization was probably attracted to the promise of uniformity and consistency made by ERP vendors, although the exact reasons differ from firm to firm. Chances are good that the organization has made some internal changes, such as realigning departments internally to take advantage of the opportunities ERP systems offer, as well as to meet the demands ERP systems make in order to operate effectively. The chances are also good that most of the implementation work has been done by outside consultants, so for firms not used to managing large numbers of contractors, an ERP implementation is a new and different experience. The implementation is likely still going on at whatever firm a student happens to talk with, but it has probably been going on for many months or years, as each ERP implementation is a learning experience for the consultants and the adopting firm.

Petrie Electronics

2.20. Typically, executives develop a set of organizational goals. These goals are then translated into strategic initiatives. These initiatives are broken down into projects, which require a combination of resources and processes to execute. IS projects are typically developed in this top-down method, although some organizations allow IS projects, to be developed by front-line employees (e.g., see Google's 20% rule).

In this case, Petrie Electronics uses the top-down approach to select projects that will help meet goals. The head office has set "number-1 priority" to develop closer relationships with their customers. In doing so they selected a customer loyalty IT project as part as this goal. There may be other projects that will also address this particular organization's goal.

2.21. IS and IS projects are directly related to company strategy in that they typically are part of a program of project that are directed toward addressing a particular organizational goal. IS cuts across all organizational boundaries (e.g., accounting, finance, marketing, and so on) to enable the organization to offer services and products. Without IS, organizations could not function. For this reason, most, if not all, organizational goals involve IS in some sort of fashion.

Further, it is critical that IS are developed (or bought) with an understanding and alignment to corporate strategy. If IS does not perform within the needs of corporate strategy, then this strategy will simply fail.

2.22. Most consumer-facing organizations have some sort of loyalty programs. There are hundreds, if not thousands, to choose from in every sort of industry (e.g., online retail, home improvement stores, car dealerships, and so on). Most programs provide a card that allows the organizations to track the activities of their customers. The organizations can then tailor marketing efforts based on customer buying behaviors. The most common loyalty programs are those of grocers. Most grocery stores in North America and Europe all provide significant discounts for customers that use loyalty programs. They then use the buying data to provide coupons and ads that will be relevant to the customer. Also, customers who receive discounts are more likely to be "loyal" to a certain store or brand.

2.23. Jim's next step is to start the first phases of the project management process. This includes six steps to project initiation that are outlined in Chapter 3.

2.24. As noted by Ella, the executives selected a new employee to lead this project because they wanted a fresh perspective for this very important project. This fresh perspective will allow for the possibility of a creative solution to be developed. Also, Jim is NOT invested in seeing other current systems succeed as he has not developed or invested time in them. His opinion, therefore, should be objective.

Modern Systems Analysis and Design

Eighth Edition

Joseph S. Valacich
Joey F. George

Chapter 2

The Origins of Software

Learning Objectives

- ✓ Explain outsourcing.
- ✓ Describe six different sources of software.
- ✓ Discuss how to evaluate off-the-shelf software.
- ✓ Explain reuse and its role in software development.

Introduction

- Historically, software development for a corporate information systems department was done primarily in-house.
- Now it involves use of components from external sources.
- Much in-house application coding involves making the components work together.

Introduction (cont.)

- Six sources of software:
  - Information technology service firms
  - Packaged software providers
  - Vendors of enterprise-wide solution software
  - Cloud computing
  - Open-source software
  - In-house development

Introduction (cont.)

- There are ways to evaluate software from sources

Sources of Software

- Information technology services firm
- Packaged software producers
- Enterprise solutions software
  - Enterprise Resource Planning (ERP)
- Cloud computing
- Open source software
- In-house developers

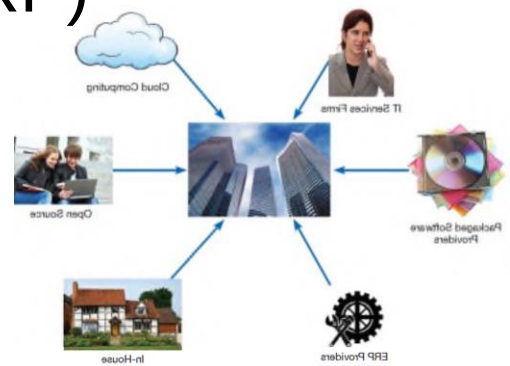

FIGURE 2-1
Sources of Application Software

- There are ways to evaluate software from sources

Systems Acquisition: Outsourcing

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

Systems Acquisition: Outsourcing (Cont.)

- Outsourcing Example
  - Shell Oil outsource spending: $3.2 billion (2008)
  - Shell's outsourcing vendors (2008-2011): EDS, T-Systems, AT&T, IBM, Logica, Wipro, Accenture

Outsourcing (Cont.)

- Reasons to outsource
  - Cost-effectiveness
  - Take advantage of economies of scale
  - Make up for lack of in-house knowledge
  - Free up internal resources
  - Reduce time to market
  - Increase process efficiencies
  - System development is a non-core activity for the organization
  - Political reasons (e.g. labor disputes)

Sources of Software (Cont.) Global Outsourcing

- Top outsourcing countries: India, China, Malaysia (A.T. Kearny report 2014)
- Top 10 are in Asia, Latin America, Europe, and Africa
- Some U.S. firms are switching to nearshoring (same time zone, low labor costs)

Sources of Software (Cont.)

TABLE 2-1 Leading Software Firms and Their Development Specializations

Specialization	Example Firms or Websites
IT Services	Accenture Computer Sciences Corporation (CSC) IBM HP
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

Information Technology (IT) Packaged Software Producers Services Firms

- Help companies develop custom information systems for internal use
- Develop, host, and run applications for customers
- Provide other services (management, accounting, auditing, financial)

Packaged Software Producers

- Serve many market segments
- Provide software ranging from broad-based packages (i.e. general ledger) to niche packages (i.e. day care management)
- Pre-packaged, off-the-shelf software

Packaged Software Producers (Cont.)

- Software runs on all size computers, from microcomputers to large mainframes.
- Prepackaged software is off-the-shelf, turnkey software (i.e. not customizable).
- Off-the-shelf software, at best, meets 70% of organizations' needs.

Prepackaged Software

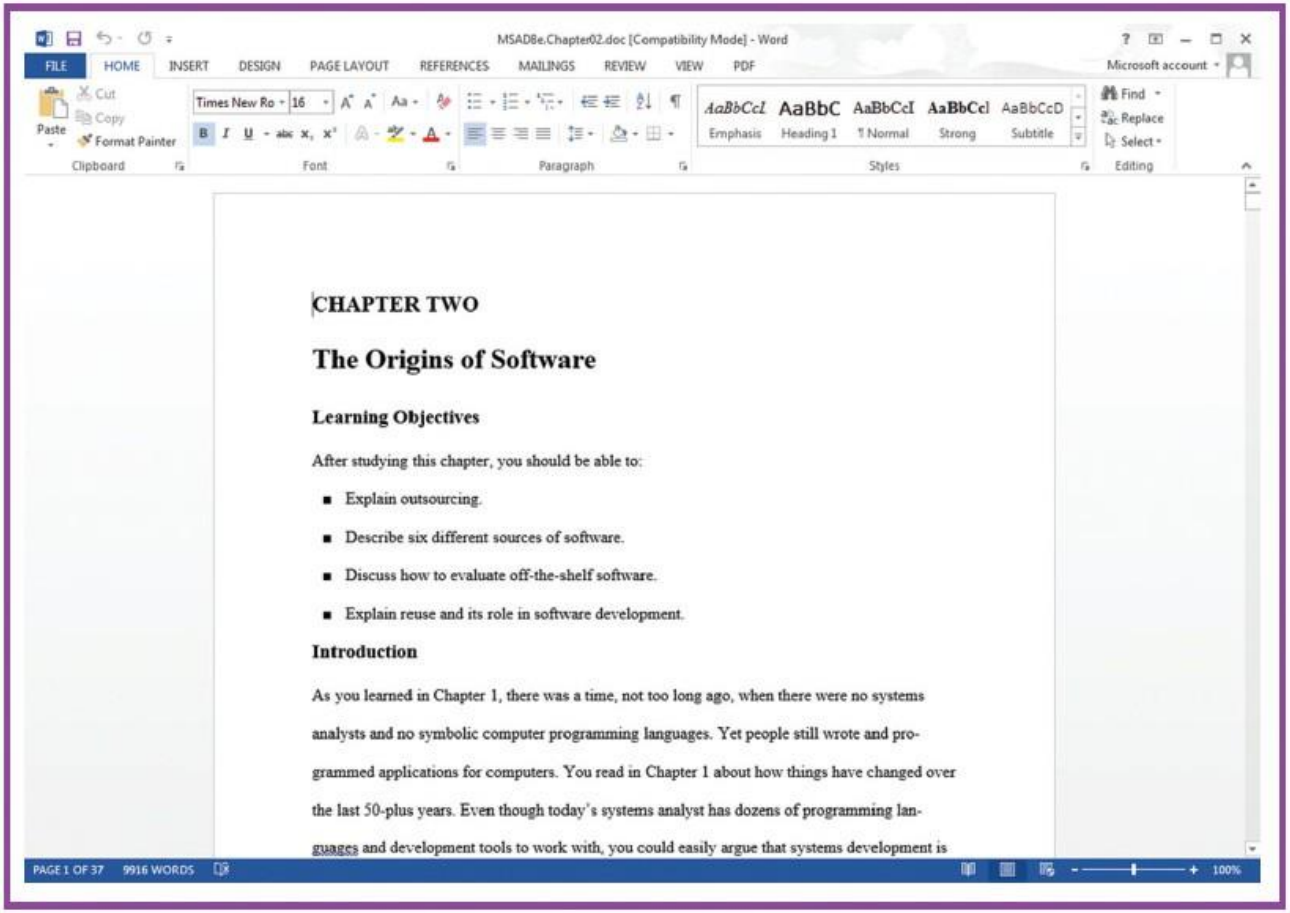

Figure 2-2 A document created in Microsoft's Word

(*Source:* Microsoft Corporation.)

Enterprise Solutions Software

- ***Enterprise Resource Planning (ERP)*** systems integrate individual traditional business functions into modules enabling a single seamless transaction to cut across functional boundaries.
- SAP AG is the leading vendor of ERP systems.

Enterprise Solutions Software (Cont.)

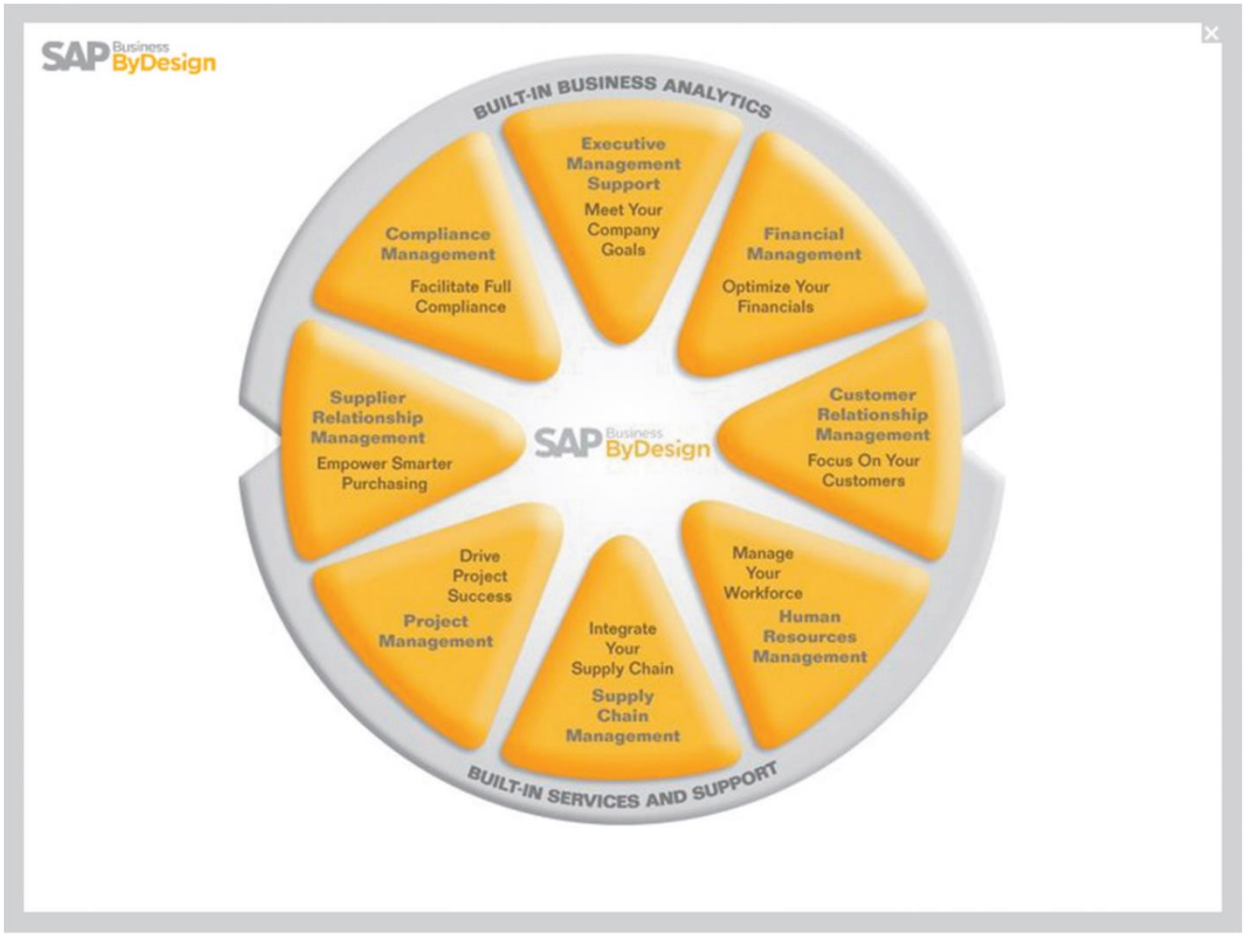

Figure 2-3 SAP's Business ByDesign, a product designed for medium sized companies. (*Source:* www.sap.com/usa/solutions/Sme/Businessbydesign/Flash/bsm/A1S.html. © Copyright SAP AG. All rights reserved.)

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

- Pay-per-use or monthly/yearly licenses

Cloud Computing (Cont.)

- Examples:
  - Google Apps– for sharing documents, spreadsheets, and presentations
  - Salesforce.com – online customer relationship management (CRM) software
    - An example of software as a service (SaaS)
  - Microsoft Azure platform
  - Amazon.com cloud infrastructure and services

- An example of hardware as a service (HaaS)

Cloud Computing (Cont.)

- Heavy growth predicted
- Benefits:
  - Frees company of internal IT staff requirements
  - Faster access to application than via internal development
  - Lower cost than internal development
- Concerns
  - Security
  - Reliability

- Regulation compliance

Open Source Software

- Freely available including source code
- Developed by a community of interested people
- Performs the same functions as commercial software
- Examples: Linux, mySQL, Firefox
- How to make money?
  - Provide maintenance/services

- Sell a more featured version of the free software

In-House Development

- If sufficient system development expertise with the chosen platform exists in-house, then some or all of the system can be developed by the organization's own staff.
  - In-house development usually leads to more maintenance burden than other approaches
- Hybrid solutions involving some purchased and some in-house components are common.

Sources of Software Components

Selection Off-the-Shelf Software

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

Selecting Off-the-Shelf Software (Cont.)

- **Cost**: comparing the cost of developing the same system in-house with the cost of purchasing or licensing the software package
- **Functionality**: the tasks that the software can perform and the mandatory, essential, and desired system features

Selecting Off-the-Shelf Software (Cont.)

- **Vendor support**: whether and how much support the vendor can provide and at what cost
- **Viability of vendor**: can vendor continue to adapt/update software

Selecting Off-the-Shelf Software (Cont.)

Selecting Off-the-Shelf Software (Cont.)

to changes in systems software and hardware

Selecting Off-the-Shelf Software (Cont.)

- **Flexibility**: the ease with which software is customized
- **Documentation**: understandable and up-to-date user's manual and technical documentation

Selecting Off-the-Shelf Software (Cont.)

- **Response time**: how long it takes the software package to respond to the user's requests in an interactive session
- **Ease of installation**: a measure of the difficulty of loading the software and making it operational

Validating Purchased Software Information

- Send a **request for proposal (RFP)** to vendors.
  - RFP – a document provided to vendors to ask them to propose hardware and system software that will meet the requirements of a new system
- Use a variety of information sources:
  - Collect information from vendor
  - Software documentation
  - Technical marketing literature

Request For Proposal (RFP)

- Sometimes called a **Request For Quote** (**RFQ**)
- Analyst selects best candidates based on:
  - vendor bids
  - a variety of information sources

Information Sources For RFP

- Vendor’s proposal
- Running software through a series of tests
- Feedback from other users of the vendor’s product
- Independent software testing services
- Customer surveys

- Articles in trade publications are sometimes biased (seeded by manufacturer)

Reuse

- The use of previously written software resources, especially objects and components, in new applications
- Commonly applied to two different development technologies:
  - Object-oriented development
  - Component-based development

Reuse (Cont.)

- **Object-oriented development**
  - Object class encapsulates data and behavior of common organizational entities (e.g. employees)
- **Component-based development**
  - Components can be as small as objects or as large as pieces of software that handle single business functions

Reuse (Cont.)

- Can be effective (increased productivity, less defects, reduced rework)
- Technical issues – lack of methodology for component library (creating and labeling reusable components)
- Organizational issues – lack of commitment, training, and organizational support; hard to measure economic benefits; legal and contractual issues

Costs and Benefits of Reuse

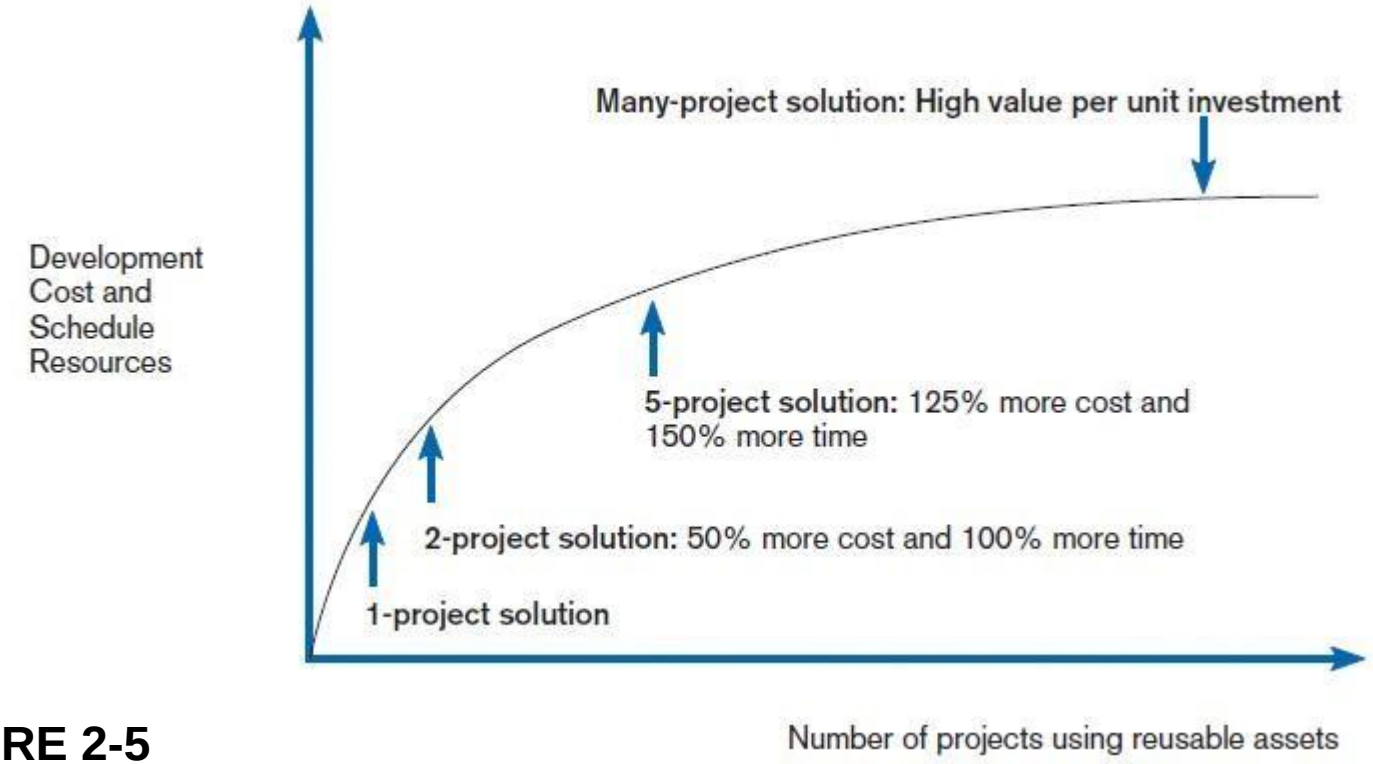

FIGURE 2-5
Investments necessary to achieve reusable components
(*Source:* Royce, Walker, *Software Project Management: A Unified Framework,* 1st ed., ©1998. Reprinted and Electronically reproduced by permission of Pearson Education, Inc. Upper Saddle

River, New Jersey.)

3 Steps of Software Reuse

- Abstraction – design of reusable piece of software
- Storage – making software assets available for others
- Recontextualization – making the software understandable to developers

3 Steps of Software Reuse
(Grinter, 2001)

Approaches to Software Reuse

- **Ad-hoc**: individuals are free to find or develop reusable assets on their own
- **Facilitated**: developers are encouraged to practice reuse
- **Managed**: the development, sharing, and adoption of reusable assets is mandated
- **Designed**: assets mandated for reuse as they are being designed for specific applications

(Griss 2003)

3 Steps of Software Reuse

Approaches to Reuse (Cont.)

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 Flashline, Inc. and Griss, 2003.)

Summary

- In this chapter you learned how to:
  - ✓ Explain outsourcing.
  - ✓ Describe six different sources of software.
  - ✓ Discuss how to evaluate off-the-shelf software.
  - ✓ Explain reuse and its role in software development.