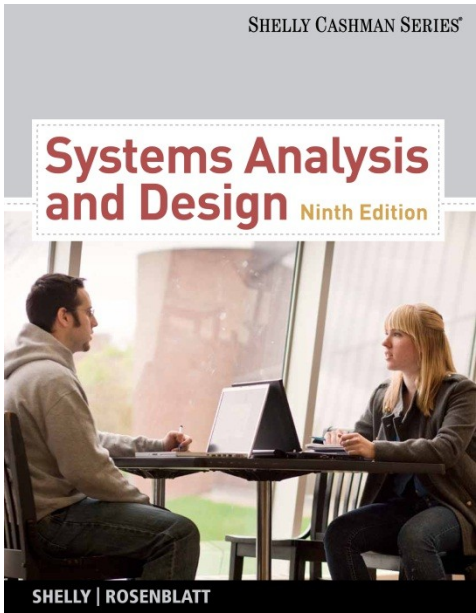
COURSE TECHNOLOGY
CENGAGE Learning

Systems Analysis and Design 9th Edition

Chapter 7

Development Strategies



Chapter Objectives

- Describe the concept of Software as a Service
- Define Web 2.0 and cloud computing
- Explain software acquisition alternatives, including traditional and Web-based software development strategies
- Describe software outsourcing options, including offshore outsourcing and the role of service providers

Chapter Objectives

- Explain advantages and disadvantages of in-house software development
- Explain cost-benefit analysis and financial analysis tools
- Explain the differences between a request for proposal (RFP) and a request for quotation (RFQ)
- Describe the system requirements document

Chapter Objectives

- Explain the transition from systems analysis to systems design, and the importance of prototyping
- Discuss guidelines for systems design
- Describe software development trends

Introduction

- Chapter 7 describes the remaining activities in the systems analysis phase
- The chapter also describes the transition to systems design, prototyping, and systems design guidelines
- The chapter concludes with a discussion of trends in software development

Development Strategies Overview

- Selecting the best development path is an important decision that requires companies to consider three key topics
  - The impact of the Internet
  - Software outsourcing options
  - In-house software development alternatives

The Impact of the Internet

- The internet has transformed how companies develop, deploy, and manage software.
- It enables new development methodologies, collaboration tools, and deployment options. Companies need to consider how internet-based technologies can enhance their development processes and product offerings.
- ***Examples*: Cloud-Based Development Platforms**: Tools like GitHub, GitLab, and Bitbucket provide version control and collaborative features that allow teams to work together from anywhere.

Software Outsourcing Options

- Outsourcing software development involves contracting external vendors or teams to build software solutions.
- This strategy can **reduce costs**, access **specialized skills**, and **accelerate project timelines**.
- ***Examples***: **Offshore Outsourcing**: Hiring development teams in countries with lower labor costs, such as India or the Philippines, to handle specific projects or tasks

In-House Software Development Alternatives

- Developing software internally involves building and maintaining a dedicated team within the company.
- This approach **offers greater control** over the development process, intellectual property, and alignment with the company's vision and culture.
- *Examples*: **Dedicated In-House Teams**: Companies like Google and Apple maintain large in-house teams to develop core products, ensuring tight integration and alignment with company goals

Software as a Service

Software as a Service (SaaS) is a cloud computing model where applications are hosted by a service provider and made available to customers over the internet.

Unlike traditional software, which requires installation and maintenance on individual devices, SaaS applications are accessed via a web browser.

This model offers flexibility, scalability, and cost savings, as it eliminates the need for companies to manage the underlying infrastructure or perform extensive software updates

- **Salesforce**: A widely-used customer relationship management (CRM) platform that helps businesses manage their sales, customer service, and marketing operations.
- **Microsoft 365**: An online suite of productivity tools that includes Word, Excel, PowerPoint, and Outlook, available via subscription for business and personal use.
- **Dropbox**: A cloud storage solution that allows users to store, share, and manage files securely over the internet.
- **Shopify**: An e-commerce platform that allows businesses to create online stores, manage inventory, and process payments

Traditional Systems Development

- This approach to system development was prominent before the rise of web-based applications and primarily focuses on applications that run within an organization's internal network, often involving compatibility with existing hardware and software infrastructures.
- **Characteristics:**
  - **Compatibility-Focused**: are designed to work within a predefined set of hardware and software configurations to ensure compatibility and stability.
  - **Local and Wide-Area Networks (LAN/WAN)**: These systems are typically built to operate over a company's local (LAN) or wide-area network (WAN), rather than relying on external/cloud-based services.

Example of Traditional Development:

- A Human Resources Management System (HRMS) designed for internal use within a company might be installed on the organization's servers.
- Users access the system through the company's internal network, and its primary function is to support core HR functions like payroll and employee records management.
- It might allow for occasional web-based access to specific data (e.g., employee login for payslips), but core functionalities rely on the local infrastructure.

Web-Based Systems Development

- web-based systems are designed primarily to be accessed via the Internet.
- **Characteristics:**

➢ **Internet-Centric Design**: be accessed through web browsers, often hosted on cloud servers to allow scalability and remote access.

➢ **Cross-Platform Compatibility**: these systems are more platform-agnostic, easily accessible from various devices and operating systems.

➢ **Enhanced Accessibility and Integration**: inherently better suited for integrating with other online services, databases, and applications, making them ideal for distributed teams

Example of Web-Based Development: A Customer Relationship Management (CRM) tool like Salesforce, which is fully web-based, allows sales and customer support teams to access customer data from anywhere with an Internet connection. The system is inherently designed to be accessed online, with frequent updates, cross-device compatibility, and integrations with other web-based tools such as email marketing platforms and social media

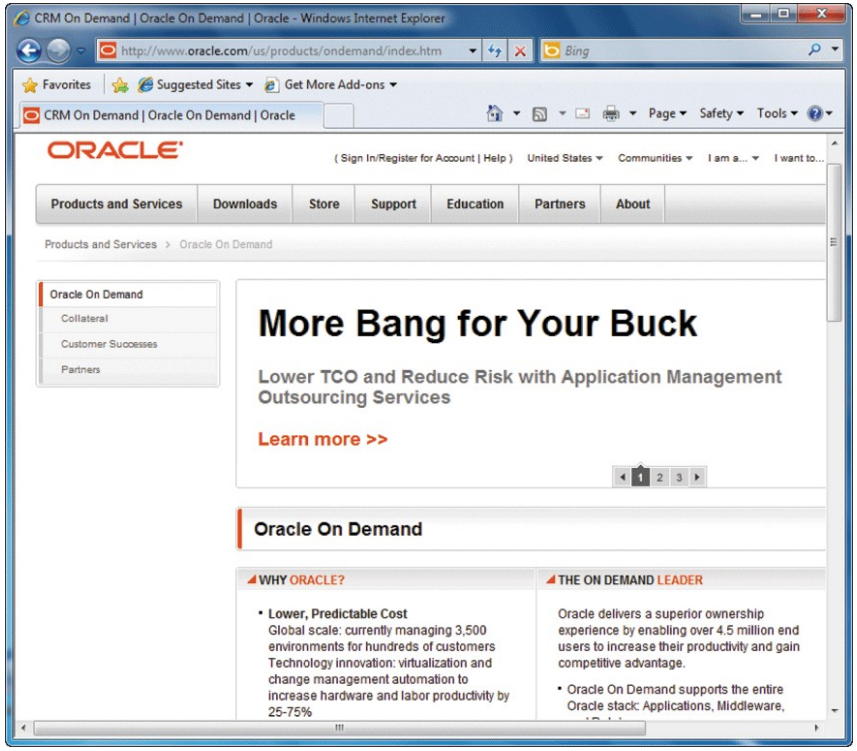
Web 2.0 and Cloud Computing

- **Web 2.0** refers to the second generation of web development, focusing on user interaction, collaboration, and content sharing.
- **Key Characteristics:**
- **User-Generated Content**: Platforms encourage users to create and share content, fostering a dynamic and interactive environment.
- **Enhanced Interactivity**: Features like commenting, liking, and real-time updates allow users to engage actively with content and with one another.

- **Examples of Web 2.0:**
- **Wikipedia**: Wikipedia exemplifies the collaborative nature of Web 2.0, allowing users to add and edit information, thereby creating a vast, community-driven knowledge base.
- **Blogs**: Blogging platforms like WordPress and Blogger let users easily publish content, allowing readers to interact through comments and shares, creating a dialogue around topics.
- **Social Media Sites**: Platforms like Facebook, Twitter, and Instagram rely on user-generated content, where people can post, share, and engage with multimedia content, making social networking a core part of the Web 2.0 experience

Outsourcing

- The Growth of Outsourcing
  - A firm that offers outsourcing solutions is called a service provider
  - Application service providers (ASP)
  - Internet business services (IBS)
    - Also called managed hosting



Outsourcing

- Outsourcing Fees
  - A fixed fee model uses a set fee based on a specified level of service and user support
  - A subscription model has a variable fee based on the number of users or workstations that have access to the application
  - A usage model or transaction model charges a variable fee based on the volume of transactions or operations performed by the application

Outsourcing

- Outsourcing Issues and Concerns
  - Mission-critical IT systems should be outsourced only if the result is a cost-attractive, reliable, business solution that fits the company's long-term business strategy
  - Outsourcing also can affect day-to-day company operations and can raise some concerns

Outsourcing

- Offshore Outsourcing
  - Offshore outsourcing – global outsourcing
  - Many firms are sending IT work overseas at an increasing rate
  - The main reason for offshore outsourcing is the same as domestic outsourcing: lower bottom-line costs
  - Offshore outsourcing, however, involves some unique risks and concerns

In-House Software Development Options

- Make or Buy Decision
  - The choice between developing versus purchasing software often is called a make or buy, or build or buy decision
  - The company's IT department makes, builds, and develops in-house software
  - A software package is obtained from a vendor or application service provider.

In-House Software Development Options

- Developing Software In-House
  - Satisfy unique business requirements
  - Minimize changes in business procedures and policies
  - Meet constraints of existing systems
  - Meet constraints of existing technology
  - Develop internal resources and capabilities

In-House Software Development Options

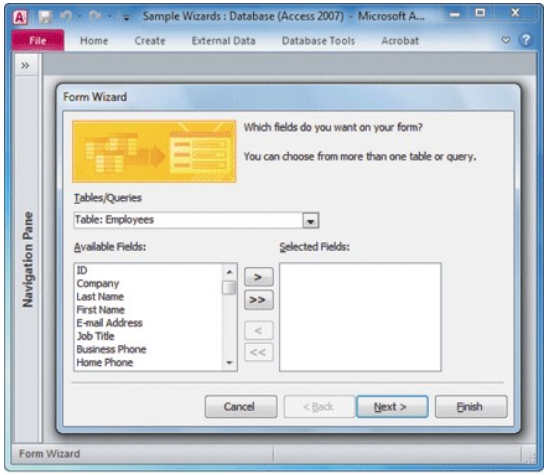
- Purchasing a Software Package
  - Lower costs
  - Requires less time to implement
  - Proven reliability and performance benchmarks
  - Requires less technical development staff
  - Future upgrades provided by the vendor
  - Input from other companies

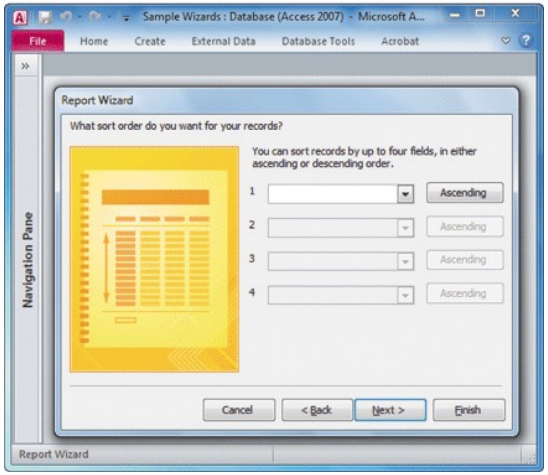
In-House Software Development Options

- Customizing a Software Package
  1. You can purchase a basic package that vendors will customize to suit your needs
  2. You can negotiate directly with the software vendor to make enhancements to meet your needs by paying for the changes
  3. You can purchase the package and make your own modifications, if this is permissible under the terms of the software license

In-House Software Development Options

- Creating User Applications
  - User application
  - User interface
  - Help desk or information center (IC)
  - Screen generators
  - Report generators
  - Read-only properties

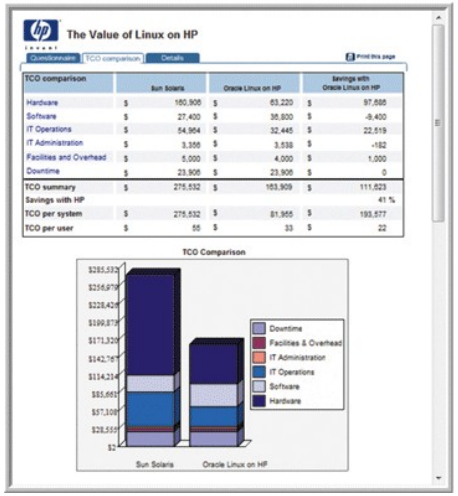




Role of the Systems Analyst

- When selecting hardware and software, systems analysts often work as an evaluation and selection team
- The primary objective of the evaluation and selection team is to eliminate system alternatives that will not meet requirements, rank the system alternatives that are feasible, and present the viable alternatives to management for a final decision

Analyzing Cost and Benefits



- Financial Analysis Tools
  - Payback Analysis
  - Return on investment (ROI)
  - Net present value (NPV)

Analyzing Cost and Benefits

- Cost-Benefit Analysis Checklist
  - List each development strategy being considered
  - Identify all costs and benefits for each alternative. Be sure to indicate when costs will be incurred and benefits realized
  - Consider future growth and the need for scalability
  - Include support costs for hardware and software

Identify all costs and benefits for each alternative

This step involves detailing every cost and benefit associated with each strategy.

Costs can be **direct** (e.g., initial setup, licensing fees) or indirect (e.g., training, maintenance).

- Benefits might include improved efficiency, time savings, or increased revenue.
- Also, specify the timing of these costs and benefits to understand cash flow better.
- **Example:** For a cloud-based CRM system, costs could include subscription fees and data migration costs, while benefits might include enhanced customer engagement and reduced downtime

Consider future growth and the need for scalability

- Analyzing the scalability of each option is essential, as a system that can't scale might become obsolete with company growth, **leading to further costs down the line**.
- This consideration ensures the chosen strategy can accommodate **increasing demands** or new functionality requirements.
- **Example**: A small e-commerce company might initially need a lightweight inventory management system, but as it grows, it may require a more comprehensive, scalable solution to handle a larger product range and increased order volume

Analyzing Cost and Benefits

- Cost-Benefit Analysis Checklist
  - Analyze various software licensing options, including fixed fees and formulas based on the number of users or transactions
  - Apply the financial analysis tools to each alternative
  - Study the results and prepare a report to management

List each development strategy being considered

- Before diving into costs and benefits, identify all the potential approaches to develop the system. This could include in-house development, outsourcing, using off-the-shelf software, or employing a hybrid approach.
- Each strategy has its unique cost implications and potential benefits, so listing them clearly allows for a more structured comparison.
- **Example:** For a new payroll system, the strategies might include developing a custom solution, purchasing a third-party software package, or contracting an external development team

The Software Acquisition Process

- Step 1: Evaluate the Information System Requirements
  - Identify key features
  - Consider network and web-related issues
  - Estimate volume and future growth
  - Specify hardware, software, or personnel constraints
  - Prepare a request for proposal or quotation

The Software Acquisition Process

- Step 2: Identify Potential Vendors or Outsourcing Options
  - The Internet is a primary marketplace
  - Another approach is to work with a consulting firm
  - Another valuable resource is the Internet bulletin board system that contains thousands of forums, called newsgroups

The Software Acquisition Process

- Step 3: Evaluate the Alternatives
  - Existing users
  - Application testing
  - Benchmarking - benchmark
  - Match each package against the RFP features and rank the choices

Existing Users

- Checking with current users of the software can offer valuable insights into its real-world performance, usability, and any issues that may not be apparent in the initial specifications. This feedback can highlight practical advantages or limitations.
- **Example:** A healthcare organization considering a new electronic medical records (EMR) system may reach out to other hospitals using the system to ask about its reliability, ease of use for staff, and support services

Application Testing

- Testing the software in a controlled setting within the organization allows you to verify if it meets your specific requirements and integrates well with existing systems.
- This step can help identify issues like compatibility, performance, and functionality.
- **Example:** A company looking to implement a Customer Relationship Management (CRM) tool might perform a trial run of the software with a small team to test if it supports their sales process and customer data tracking needs effectively

Benchmarking

- Benchmarking involves comparing each software option's performance, reliability, and efficiency against standard criteria or other alternatives.
- This objective evaluation can reveal how each product measures up in terms of speed, functionality, or other key metrics.
- **Example:** A logistics company evaluating routing software could benchmark various products to see which one calculates routes the fastest and has the most accurate delivery estimates.

The Software Acquisition Process

- Step 4: Perform Cost-Benefit Analysis
  - Identify and calculate TCO for each option you are considering
  - When you purchase software, what you are buying is a software license
  - If you purchase a software package, consider a supplemental maintenance agreement

The Software Acquisition Process

- Step 5: Prepare a Recommendation
  - You should prepare a recommendation that evaluates and describes the alternatives, together with the costs, benefits, advantages, and disadvantages of each option
  - At this point, you may be required to submit a formal system requirements document and deliver a presentation

The Software Acquisition Process

- Step 6: Implement the Solution
  - Implementation tasks will depend on the solution selected
  - Before the new software becomes operational, you must complete all implementation steps, including loading, configuring, and testing the software; training users; and converting data files to the new system's format

Completion of Systems Analysis Tasks

- System Requirements Document
  - The system requirements document, or software requirements specification, contains the requirements for the new system, describes the alternatives that were considered, and makes a specific recommendation to management
  - Like a contract
  - Format and organize it so it is easy to read and use

Contains Requirements for the New System

- This document details the functional and non-functional requirements of the new system. Functional requirements describe specific behaviors or functions (like processing transactions or generating reports), while non-functional requirements cover system performance, usability, and reliability standards.
- **Example:** For an online banking system, functional requirements might include capabilities like transferring funds, checking balances, and paying bills. Non-functional requirements might specify that transactions should process in under two seconds and that the system must be available 99.9% of the time

Describes the Alternatives that Were Considered

- The document should also include a summary of the various approaches considered for meeting the requirements, such as in-house development, outsourcing, or purchasing off-the-shelf software.
- Each alternative should be evaluated based on factors like cost, scalability, and alignment with business goals.
- **Example:** In the development of a customer support system, alternatives might include customizing an existing CRM, building a new platform from scratch, or adopting a cloud-based solution. Each option's pros and cons should be discussed in the document.

Acts Like a Contract

- The System Requirements Document is akin to a contract between the stakeholders and the development team. It defines the scope, expectations, and deliverables, helping to manage stakeholder expectations and minimize misunderstandings.
- **Example:** For a new HR management system, the document might specify that it will automate tasks like payroll and leave management but won't initially include performance tracking. This clarity sets boundaries for the project's scope

Completion of Systems Analysis Tasks

- Presentation to Management
  - Summarize the primary viable alternatives
  - Explain why the evaluation and selection team chose the recommended alternative
  - Allow time for discussion and for questions and answers
  - Obtain a final decision from management or agree on a timetable for the next step in the process

Completion of Systems Analysis Tasks

- Presentation to Management
  - Depending on their decision, your next task as a systems analyst will be one of the following
    1. Implement an outsourcing alternative
    2. Develop an in-house system
    3. Purchase or customize a software package
    4. Perform additional systems analysis work
    5. Stop all further work

The Transition to Systems Design

- Preparing for Systems Design Tasks
  - It is essential to have an accurate and understandable system requirements document
- Logical and Physical Design
  - The logical design defines the functions and features of the system and the relationships among its components
  - The physical design of an information system is a plan for the actual implementation of the system

Systems Design Guidelines

- Overview
  - **A system is effective if it supports business requirements and meets user needs**
  - **A system is reliable** if it handles input errors, processing errors, hardware failures, or human mistakes
  - **A system is maintainable** if it is flexible, scalable, and easily modified

Systems Design Guidelines

- Overview
  - User Considerations
    - Carefully consider any point where users receive output from, or provide input
    - Anticipate future needs - **Y2K Issue**
    - Provide flexibility
    - Parameter, default

Systems Design Guidelines

- Overview
  - Data Considerations
    - **Enter data as soon as possible**
    - **Verify data as it is entered**
    - Use automated methods of data entry whenever possible

Systems Design Guidelines

- Overview
  - Data Considerations
    - Control data entry access and report all entries or changes to critical values – audit trail
    - **Log every instance of data entry** and changes
    - **Enter data once**
    - Avoid data duplication

Systems Design Guidelines

- Overview
  - Architecture considerations
    - Use a modular design
    - Design modules that perform a single function are easier to understand, implement, and maintain

Architecture Considerations

- The system's architecture should be thoughtfully designed to meet the project's requirements, including scalability, flexibility, and performance. Architecture decisions include selecting the type of architecture (e.g., client-server, microservices, monolithic) based on how the system needs to function and evolve over time.
- Example: For an e-commerce platform, a microservices architecture might be chosen because it allows separate services for product management, payment processing, and order handling. This structure enables individual services to be scaled independently as the platform grows

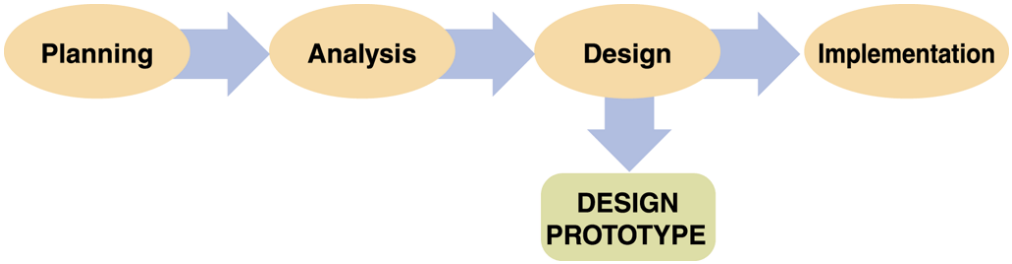
Design Modules that Perform a Single Function

- When each module is focused on performing a single function, the system becomes easier to understand and maintain. Such modules have a clear purpose and are simpler to implement, debug, and test. Single-function modules also follow the principle of "separation of concerns," making the system more cohesive.
- **Example**: In an online payment system, there could be separate modules for user authentication, transaction processing, and payment history. Each module has a specific task, reducing complexity and making it easier to identify issues if they arise

Systems Design Guidelines

- Design Trade-Offs
  - Design goals often **conflict with each other**
  - Most design trade-off decisions that you will face come down to the basic conflict of quality versus cost
  - Avoid decisions that achieve short-term savings but might mean higher costs later

Prototyping



- Prototyping Methods
  - System prototyping
  - Design prototyping
  - Throwaway prototyping
  - Prototyping offers many benefits
  - Consider potential problems

Prototyping

- Prototyping Tools
  - CASE tools
  - Application generators
  - Report generators
  - Screen generators
  - Fourth-generation language (4GL)
  - Fourth-generation environment

Prototyping

- Limitations of Prototypes
  - A prototype is a functioning system, but it is less efficient than a fully developed system
  - Systems developers can upgrade the prototype into the final information system by adding the necessary capability
  - Otherwise, the prototype is discarded

Software Development Trends

- Views from the IT Community
  - Software quality will be more important than ever
  - Project management will be a major focus of IT managers

Software Development Trends

- Views from the IT Community
  - Service-oriented architecture (SOA)
    - Loose coupling
  - Growth in open-source software
  - Developers will use more Web services
  - Programmers will continue to use dynamic languages

Chapter Summary

- This chapter describes system development strategies, the preparation and presentation of the system requirements document, and the transition to the systems design phase of the SDLC
- An important trend that views software as a service, rather than a product, has created new software acquisition options
- Systems analysts must consider Web-based development environments

Chapter Summary

- The systems analyst's role in the software development process depends on the specific development strategy
- The most important factor in choosing a development strategy is total cost of ownership (TCO)
- The process of acquiring software involves a series of steps
- A prototype is a working model of the proposed system

Chapter Summary

- Chapter 7 complete