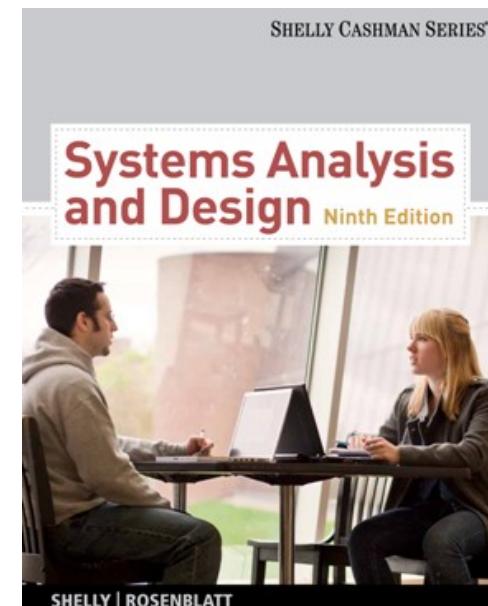


Systems Analysis and Design 9th Edition

Chapter 1

Introduction to Systems Analysis and Design



Chapter Objectives

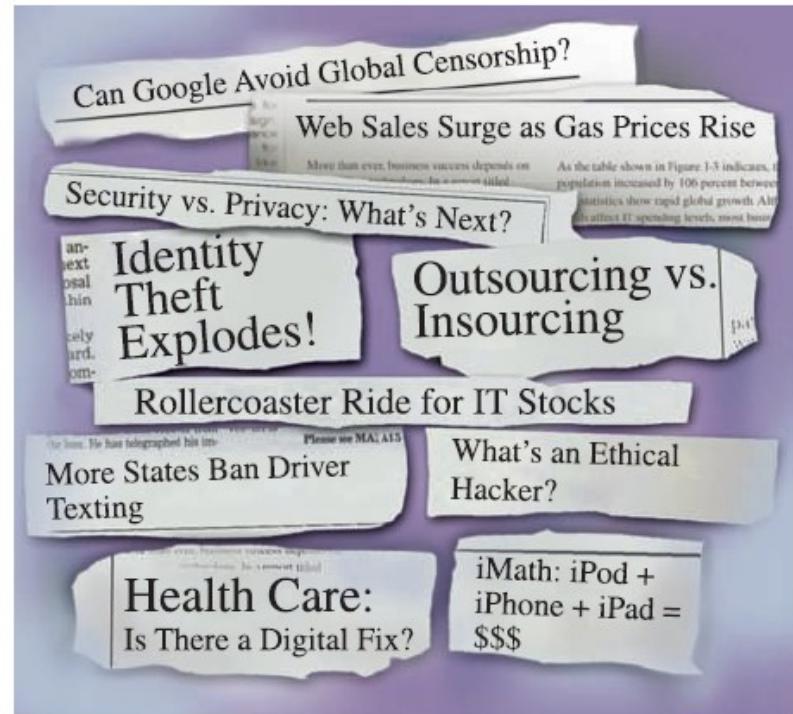
- Discuss the impact of information technology on business strategy and success
- Define an information system and describe its components
- Explain how profiles and models can represent business functions and operations
- Explain how the Internet has affected business strategies and relationships
- Identify various types of information systems and explain who uses them

Chapter Objectives

- Distinguish between structured analysis, object-oriented analysis, and agile methods
- Compare the traditional waterfall model with agile methods and models
- Apply five basic guidelines for systems development
- Discuss the role of the information technology department and the systems analysts who work there

Introduction

- Companies use information as a weapon in the battle to increase productivity, deliver quality products and services, maintain customer loyalty, and make sound decisions
- Information technology can mean the difference between success and failure



The Impact of Information Technology

- Information Technology (IT)
 - Combination of hardware and software products and services that companies use to manage, access, communicate, and share information
- The Future
 - Three issues that will shape the future
 - Changes in world
 - Changes in technology
 - Changes in client demand

The Impact of Information Technology

- Systems Development
 - Business information systems are developed by people who are technically qualified, business-oriented, and highly motivated
 - Must be good communicators with strong analytical and critical thinking skills

The Impact of Information Technology

- Systems Analysis and Design
 - Systems Analysis and Design
 - Step-by-step process for developing high-quality information systems
 - Systems Analyst
 - Plan, develop, and maintain information systems

The Impact of Information Technology

- Who develops Information Systems?
 - In-house applications
 - Software packages
 - Internet-based application services
 - Outsourcing
 - Custom solutions
 - Enterprise-wide software strategies
 - How versus What

Information System Components

- A system is a set of related components that produces specific results
- A Mission-critical system is one that is vital to a company's operations
- Data consists of basic facts that are the system's raw material
- Information is data that has been transformed into output that is valuable to users
- Information systems have five key components: hardware, software, data, processes, and people

Information System Components

- Hardware
 - Is the physical layer of the information system
 - Moore's Law
- Software
 - System software
 - Application software
 - Enterprise applications



Information System Components

- Software
 - Horizontal system
 - Vertical system
 - Legacy systems
- Data
 - Tables store data
 - Linked tables work together to supply data

Horizontal software

These software are designed to serve a broad range of industries and functions. They provide **generic solutions** that can be adapted or customized to fit various business processes across different sectors

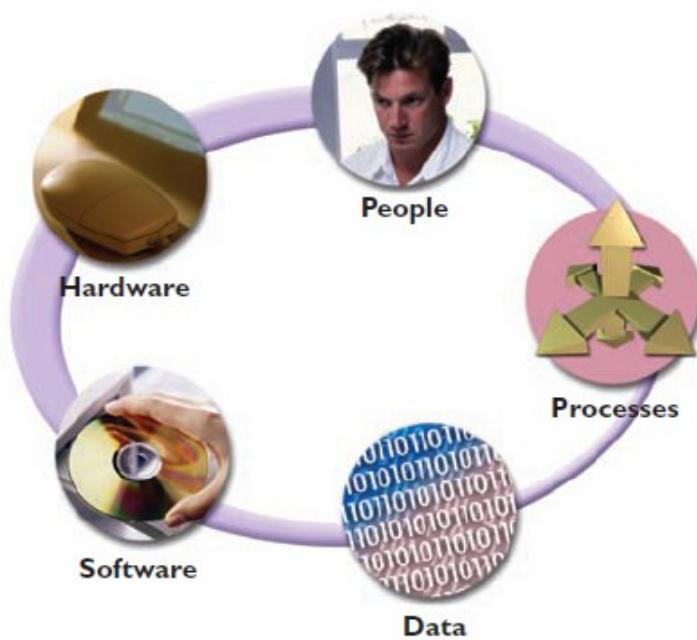
Example: Software like Microsoft Office or Google Workspace, which provides general productivity tools applicable across industries.

Vertical software

These software are specialized solutions designed **to cater to the needs of a specific industry or sector**. These systems address the unique requirements, regulations, and workflows of a particular domain.

Example: Software like Epic or Cerner, which is specifically designed for hospitals and healthcare providers, focusing on patient

Information System Components



- **Processes**
 - Describe the tasks and business functions that users, managers, and IT staff members perform to achieve specific results
- **People**
 - Stakeholders
 - Users, or end users

Understanding The Business

- **Business Process Modeling:**

It is the practice of representing the processes of an organization in a visual format. This involves mapping out the steps involved in completing a particular business activity or process, typically using flowcharts, diagrams, or models. The goal is to provide a clear and detailed representation of how business processes function, identify inefficiencies, and suggest improvements.

- **Business Profile:** It is a comprehensive overview of a company's essential information, including its mission, vision, products or services, target market, competitive advantages, and financial status. It serves as a snapshot of the organization, highlighting key aspects that define the business and its operations

Understanding The Business

- New Kinds of Companies
 - Production-oriented
 - Service-oriented
 - Internet-dependent
 - Dot-com (.com)
 - Brick-and-mortar



Impact of the Internet

- E-Commerce or I-Commerce
- B2C (Business-to-Consumer)
- B2B (Business-to-Business)
 - EDI
 - Extensible markup language (XML)
 - Supply chain management (SCM)
 - Supplier relationship management (SRM)

Business Information Systems

- In the past, IT managers divided systems into categories based on the user group the system served
 - Office systems
 - Operational systems
 - Decision support systems
 - Executive information systems

Business Information Systems

- Today, identify a system by its functions and features, rather than by its users
 - Enterprise computing systems
 - Transaction processing systems
 - Business support systems
 - Knowledge management systems
 - User productivity systems

Business Information Systems-cont.

1. Enterprise Computing Systems

- **Functions and Features:** These systems are designed to manage the entire operations of large organizations. They handle large-scale integration of core business functions like human resources, finance, manufacturing, and supply chain.
- **Key Features:** Scalability, integration with various other systems (ERP, CRM), automation of business processes, high levels of security, data management.
- **Example:** ERP (Enterprise Resource Planning) systems that integrate all departments and functions into a single IT platform.
- **Identified By:** Their ability to unify and optimize various operations across an organization

Business Information Systems-cont.

- **2. Transaction Processing Systems (TPS)**
- **Functions and Features:** These systems manage and record the day-to-day operations or transactions of a business, such as sales, payroll, orders, and payments.
- **Key Features:** Real-time processing, high availability, reliability, support for large volumes of transactions, data consistency, and security.
- **Example:** Point of Sale (POS) systems, ATM systems, or online payment gateways.
- **Identified By:** Their focus on processing, validating, and storing high volumes of transactions

Business Information Systems-cont.

3. Business Support Systems (BSS)

- **Functions and Features:** These systems provide functionalities to support business operations and decision-making processes. They typically support back-office operations and are tailored to improve efficiency in financial management, customer service, and order processing.
- **Key Features:** Analytical tools, financial management, CRM (Customer Relationship Management), inventory management.
- **Example:** Billing systems or customer relationship management platforms.
- **Identified By:** Their role in supporting business-critical tasks and decision-making processes.

Business Information Systems-cont.

4. Knowledge Management Systems (KMS)

- **Functions and Features:** These systems facilitate the organization, sharing, and retrieval of knowledge within an organization. They are used to capture both explicit knowledge (documented information) and tacit knowledge (expertise).
- **Key Features:** Document management, collaboration tools, search functionalities, expertise management, knowledge bases.
- **Example:** Wikis, corporate intranets, or expert systems.
- **Identified By:** Their ability to manage and disseminate information and expertise across the organization to improve decision-making and innovation.

Business Information Systems-cont.

5. User Productivity Systems

- **Functions and Features:** These systems are designed to help users perform specific tasks more efficiently and effectively. They enhance individual or group productivity by providing tools for communication, collaboration, and data manipulation.
- **Key Features:** Word processing, spreadsheets, email, project management, collaboration platforms (like Slack or Microsoft Teams).
- **Example:** Microsoft Office Suite, Google Workspace.
- **Identified By:** Their emphasis on boosting user efficiency and providing tools for day-to-day productivity

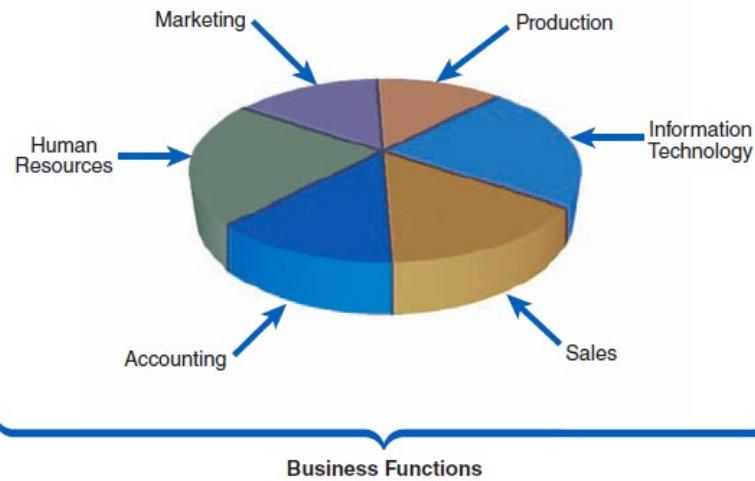
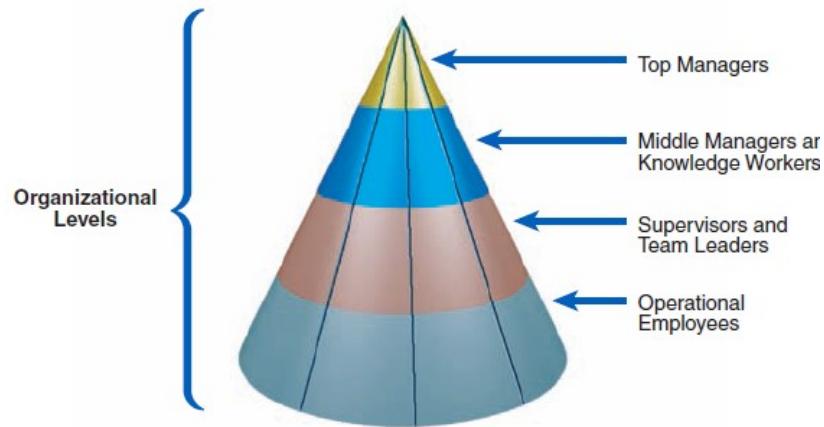
Business Information Systems

- Knowledge management systems
 - Called expert systems
 - Simulate human reasoning by combining a knowledge base and inference rules
 - Many knowledge management systems use a technique called fuzzy logic

Business Information Systems

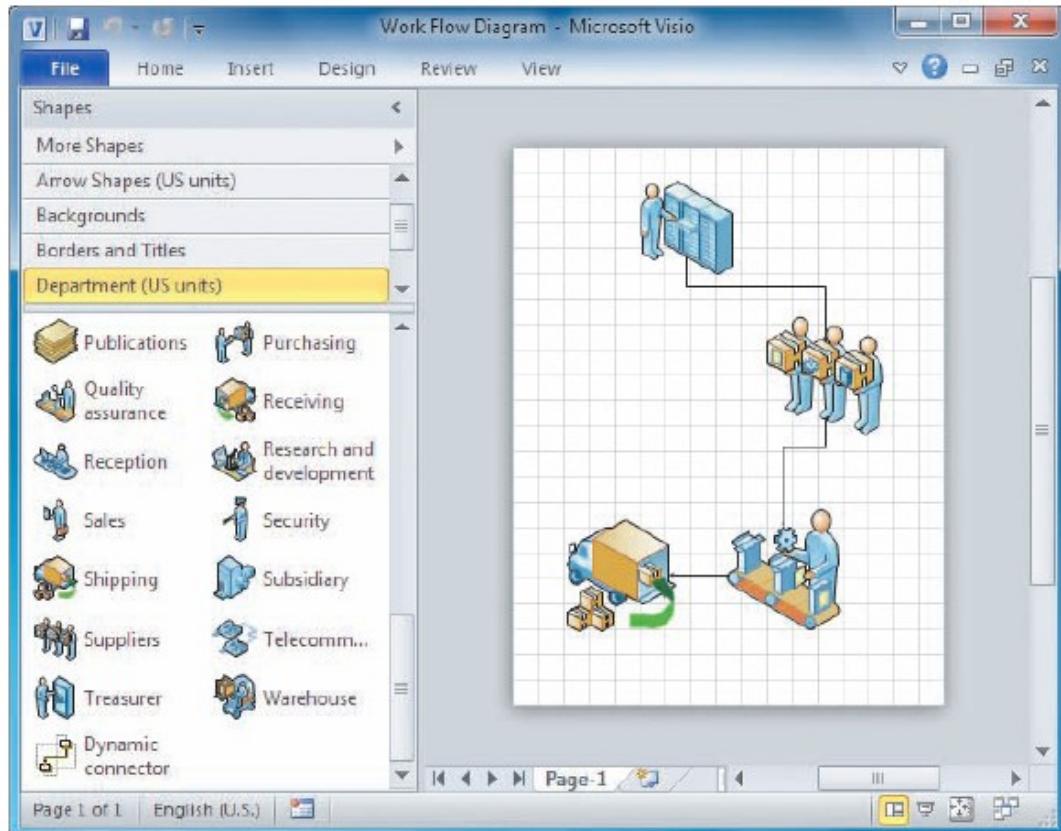
- User productivity systems
 - Technology that improves productivity
 - Groupware
- Information systems integration
 - Most large companies require systems that combine transaction processing, business support, knowledge management, and user productivity features

What Information Do Users Need?



Systems Development Tools

- Modeling
 - Business model
 - Requirements model
 - Data model
 - Object model
 - Network model
 - Process model



Systems Development Tools

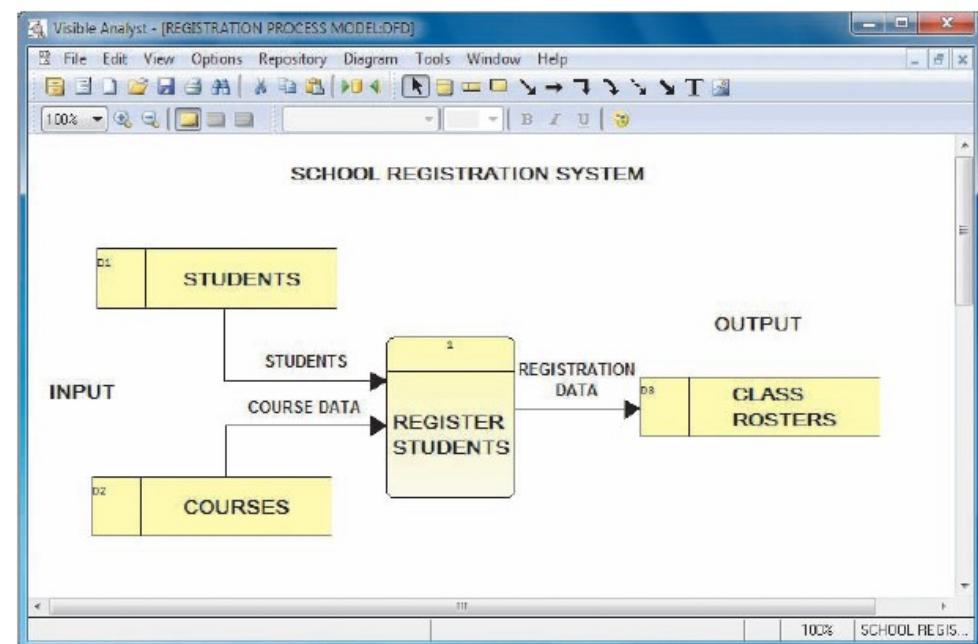
- Prototyping
 - Prototype
 - Speeds up the development process significantly
 - Important decisions might be made too early, before business or IT issues are thoroughly understood
 - Can be an extremely valuable tool

Systems Development Tools

- Computer-Aided Systems Engineering (CASE) Tools
 - Also called computer-aided software engineering
 - CASE tools
 - Can generate program code, which speeds the implementation process

Systems Development Methods

- Structured Analysis
 - Systems development life cycle (SDLC)
 - Predictive approach
 - Adaptive approach
 - Uses a set of process models to describe a system graphically
 - Process-centered technique
 - Waterfall model



Systems Development Methods

- Structured Analysis
 - Deliverable or end product
 - Disadvantage in the built-in structure of the SDLC, because the waterfall model does not emphasize interactivity among the phases
 - This criticism can be valid if the SDLC phases are followed too rigidly
 - Adjacent phases usually interact

Systems Development Methods

- Structured Analysis
 - The SDLC model usually includes five steps
 - Systems planning
 - Systems analysis
 - Systems design
 - Systems implementation
 - Systems support and security

Systems Development Methods

- Structured Analysis
 - Systems Planning
 - Systems planning phase
 - Systems request – begins the process & describes problems or desired changes
 - Purpose of this phase is to perform a preliminary investigation
 - Key part of preliminary investigation is a feasibility study

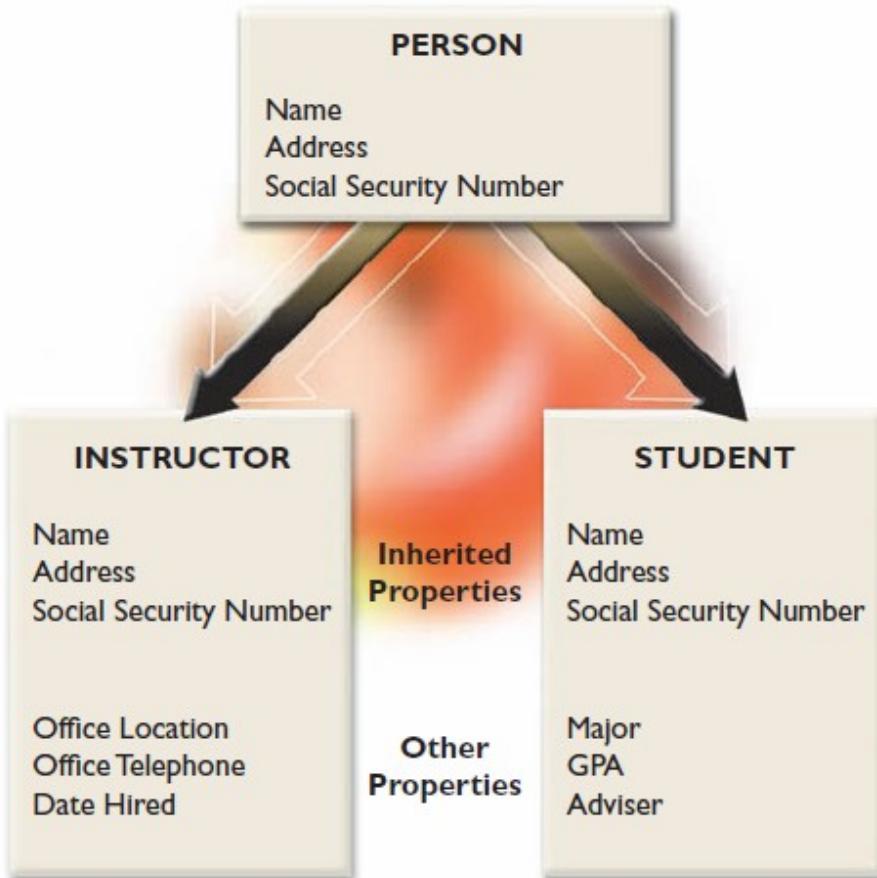
Systems Development Methods

- Structured Analysis
 - Systems Analysis
 - Deliverable is the System requirements document
 - Systems Design
 - Deliverable is system design specification
 - Management and user involvement is critical

Systems Development Methods

- Structured Analysis
 - Systems Implementation
 - New system is constructed
 - Systems Support and Security
 - A well-designed system must be secure, reliable, maintainable, and scalable
 - Most information systems need to be updated significantly or replaced after several years of operation

Systems Development Methods



Object-oriented Analysis

- Combines data & processes that act on the data into things called objects
- Object is a member of a class
- Objects possess properties
- Methods change an object's properties

Systems Development Methods

- Object-Oriented Analysis
 - A message requests specific behavior or information from another object
 - Usually follow a series of analysis and design phases that are similar to the SDLC
 - Interactive model

Systems Development Methods

- Agile Methods
 - Are the newest development
 - Emphasize continuous feedback
 - Iterative development
 - Agile community has published the Agile Manifesto
 - Spiral model

Systems Development Methods

- Agile Methods
 - Agile process determines the end result
 - Other adaptive variations and related methods exist
 - Two examples are Scrum and Extreme Programming (XP)
 - Analysts should understand the pros and cons of any approach before selecting a development method

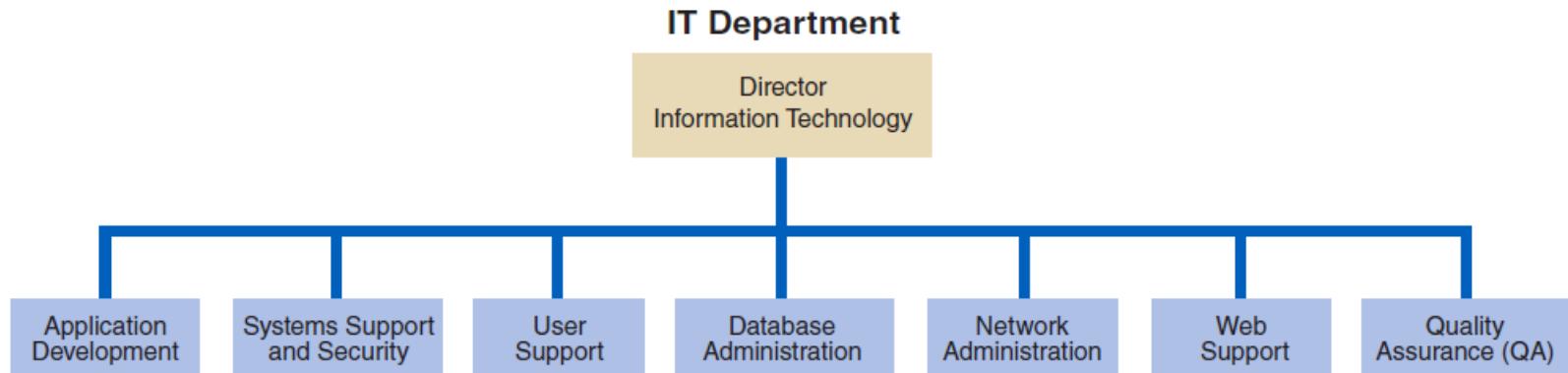
Systems Development Methods

- Other Development Methods
 - Joint application development (JAD)
 - Rapid application development (RAD)
 - Might encounter other systems development techniques
 - Rational Unified Process (RUP®)
 - Microsoft Solutions Framework (MSF)

Systems Development Guidelines

- Develop a project plan
- Involve users and listen carefully to them
- Use project management tools to identify tasks and milestones
- Develop accurate cost and benefit information
- Remain flexible

Information Technology Department



The Systems Analyst

- Responsibilities
 - Translate business requirements into IT projects
- Knowledge, Skills, and Education
 - Needs technical knowledge, strong oral and written communication skills and analytic ability, an understanding of business operations, and critical thinking skills
- Certification
 - Important credential

The Systems Analyst

- Career Opportunities
 - Job titles
 - Company organization
 - Company size
 - Corporate culture
 - Salary, location, and future growth

Chapter Summary

- IT refers to the combination of hardware and software resources that companies use to manage, access, communicate, and share information
- The essential components of an information system are hardware, software, data, processes, and people
- Successful companies offer a mix of products, technical and financial services, consulting, and customer support

Chapter Summary

- Information systems are identified as enterprise computing systems, transaction processing systems, business support systems, knowledge management systems, or user productivity systems
- Organization structure includes top managers, middle managers and knowledge workers, supervisors and team leaders

Chapter Summary

- The IT department develops, maintains and operates a company's information systems
- Systems analysts need a combination of technical and business knowledge, analytical ability, and communication skills
- Systems analysts need to consider salary, location, and future growth potential when making a career decision

Chapter Summary

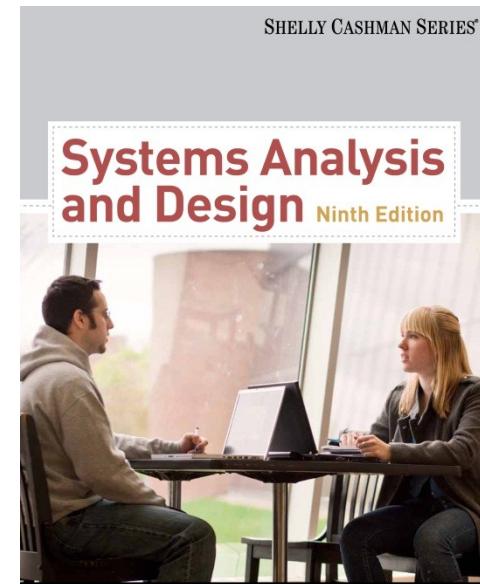
- Chapter 1 complete

Systems Analysis and Design 9th Edition

Chapter 2

Analyzing the Business Case

SHELLY CASHMAN SERIES



SHELLY | ROSENBLATT

Chapter Objectives

- Explain the concept of a business case and how a business case affects an IT project
- Describe the strategic planning process and why it is important to the IT team
- Conduct a SWOT analysis and describe the four factors involved

Chapter Objectives

- Explain the purpose of a mission statement
- Explain how the SDLC serves as a framework for systems development
- List the reasons for systems projects and factors that affect such projects

Chapter Objectives

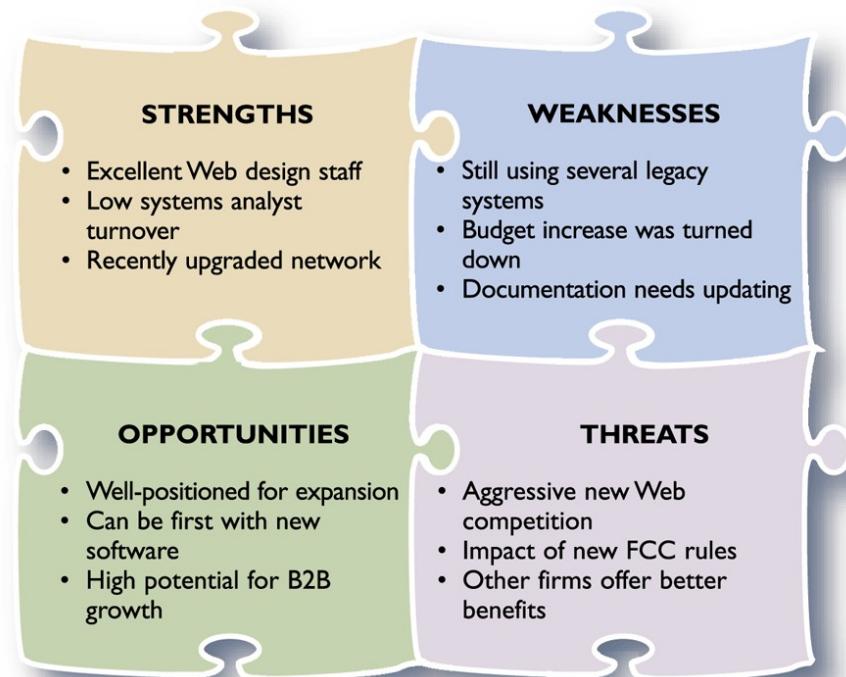
- Describe systems requests and the role of the systems review committee
- Define operational, technical, economic, and schedule feasibility
- Describe the steps and the end product of a preliminary investigation

Introduction

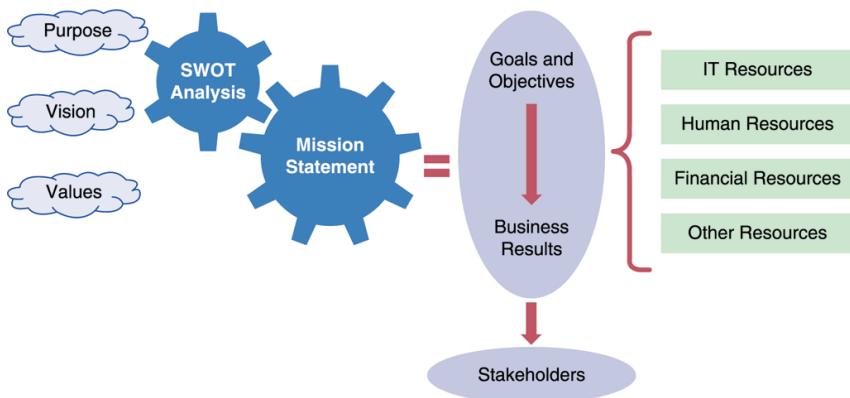
- The term business case refers to the reasons, or justification, for a proposal
- A strong business case suggests that the company should pursue the alternative, above other options, because it would be in the firm's best interest to do so
- Systems development typically starts with a systems request, followed by a preliminary investigation, which includes a feasibility study

Strategic Planning – A Framework for IT Systems Development

- Strategic Planning Overview
 - SWOT analysis



Strategic Planning – A Framework for IT Systems Development



- From Strategic Plans to Business Results
 - Mission statement
 - Stakeholders
 - Goals
 - Objectives

Strategic Planning – A Framework for IT Systems Development

- A CASE Tool Example
 - You are a systems analyst
 - You research the Visible Analyst CASE tool
 - Planning statements can include assumptions, goals, objectives, and critical success factors, and many other types of statements

Strategic Planning – A Framework for IT Systems Development

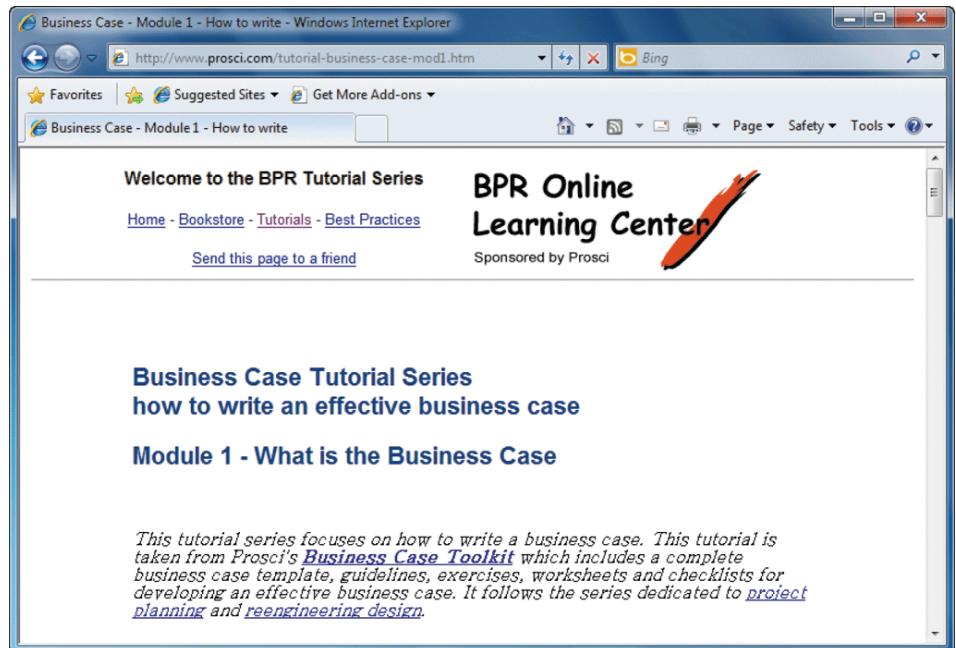
- The Role of the IT Department in Project Evaluation
 - Management leadership and information technology are linked closely, and remarkable changes have occurred in both areas
 - Today, systems development is much more team oriented
 - Although team-oriented development is the norm, some companies see the role of the IT department as a gatekeeper

Strategic Planning – A Framework for IT Systems Development

- The Future
 - If you could look into the future, here is what you might see: new industries, products, and services emerging from amazing advances in information technology, customers who expect world-class IT support, a surge in Internet-based commerce, and a global business environment that is dynamic and incredibly challenging

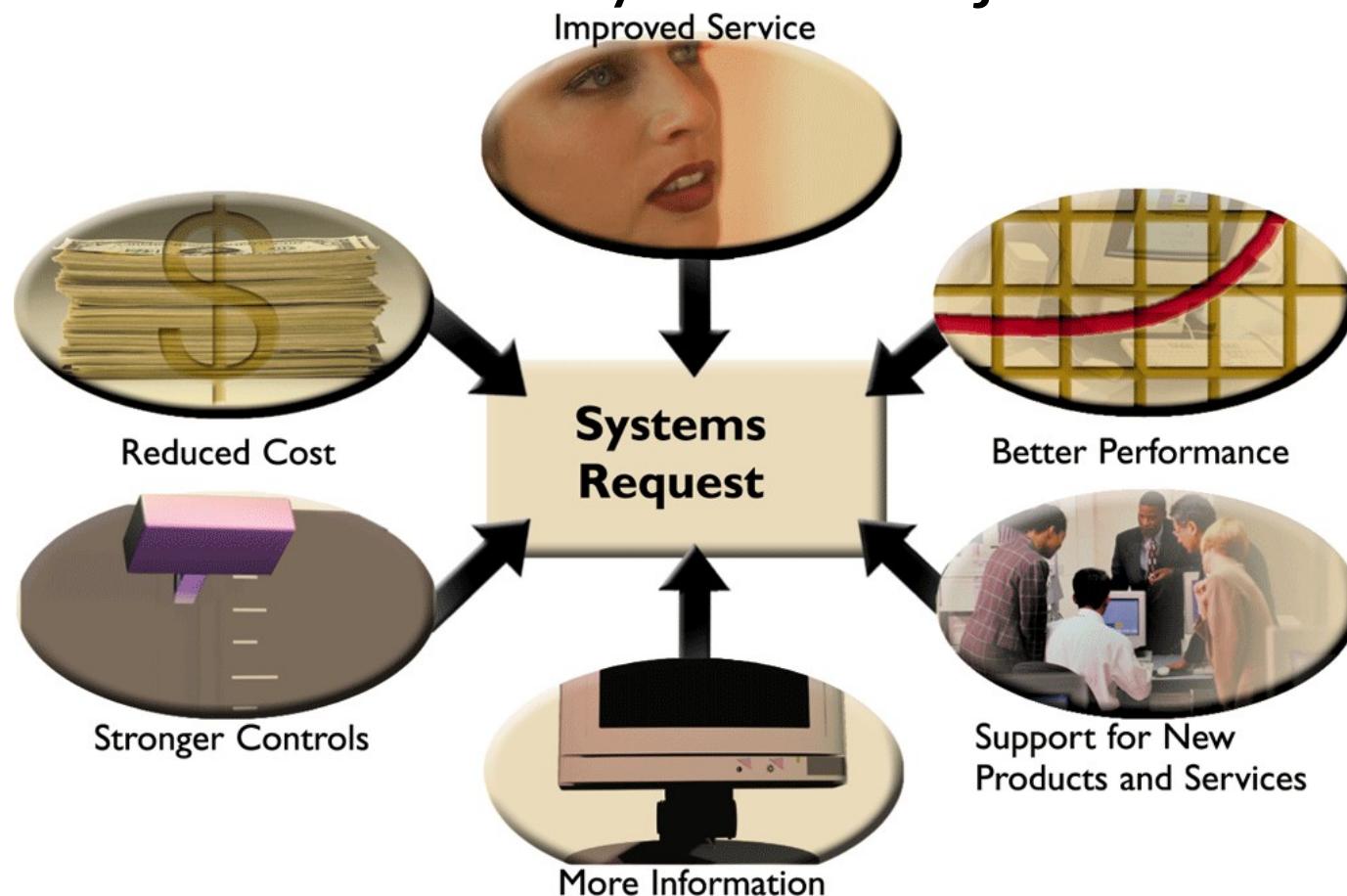
What Is a Business Case?

- Should be comprehensive, yet easy to understand
- Should describe the project clearly, provide the justification to proceed, and estimate the project's financial impact



Information Systems Projects

- Main Reasons for Systems Projects



Information Systems Projects

- Factors that Affect Systems Projects



Information Systems Projects

- Project Management
 - If the project is approved, it can be planned, scheduled, monitored and controlled, and reported upon
 - Individual analysts or IT staff members often handle small projects, but companies usually designate a project manager to coordinate the overall effort for complex projects

Evaluation of Systems Requests

- Systems review committee or a computer resources committee evaluate systems requests
- Systems Requests Forms
 - A properly designed form streamlines the request process and ensures consistency
 - Occasionally a situation will arise that requires an immediate response

Evaluation of Systems Requests

- Systems Review Committees
 - Most large companies use a systems review committee to evaluate systems requests
 - Many smaller companies rely on one person to evaluate systems requests instead of a committee
 - The goal is to evaluate the requests and set priorities

Overview of Feasibility

- A systems request must pass several tests, called a feasibility study, to see whether it is worthwhile to proceed further
- Operational Feasibility
 - Depends on several vital issues



Overview of Feasibility

- Technical Feasibility
- Economic Feasibility
 - Total cost of ownership (TCO)
 - Tangible benefits
 - Intangible benefits
- Schedule Feasibility

Operational Feasibility

- **Definition:** Operational feasibility assesses how well the proposed system will function within the organization. It evaluates if the software can be effectively integrated into the existing operational workflows, and whether the end-users are likely to accept and use the system.

- **Key Questions:**

- Will the software solve the problems it is meant to address?
- Are the users willing to adopt the new system?
- Will the system support the organization's day-to-day operations?

Technical Feasibility

- **Definition:** Technical feasibility examines whether the technology needed for the project is available, reliable, and compatible with the organization's infrastructure. It also involves evaluating the technical skills of the team.

- **Key Considerations:**

- Is the required technology available and mature?
- Do the developers and engineers have the necessary expertise?
- Are there any technical risks such as system integration issues, hardware limitations, or software incompatibility?

Economic Feasibility

- **Definition:** Economic feasibility, also known as cost-benefit analysis, evaluates whether the benefits of the project outweigh the costs. It considers whether the project is financially viable and whether the organization has the resources to support it. **Both Tangible benefits and Intangible benefits should be considered**

- **Key Questions:**

- Will the project generate more value than it costs?
- What is the return on investment (ROI)?

$$ROI = \frac{\text{Net Profit (or Gain from Investment)} - \text{Cost of Investment}}{\text{Cost of Investment}} \times 100$$

- Can the organization afford the costs, and are the resources available?

Total Cost of Ownership (TCO)

- **Definition:** TCO includes all costs associated with the software over its entire lifecycle, not just the initial development costs. This includes costs for hardware, software, training, maintenance, support, and any upgrades.

- **Key Considerations:**

- What are the upfront development costs?
- How much will ongoing maintenance, training, and upgrades cost?
- Are there additional costs, such as licensing fees or system downtime?
- **Example of TCO:** If an organization develops a custom CRM system, the **TCO** might include:
 - **Initial Costs:** \$150,000 for software development, \$50,000 for hardware, and \$20,000 for installation and configuration.
 - **Ongoing Costs:** \$10,000 annually for maintenance and updates, \$5,000 annually for user training, and \$15,000 annually for technical support and cloud services.
 - **Indirect Costs:** Estimated downtime costs due to maintenance might be \$2,000 annually.
 - Over a 5-year period, the **TCO** would be the sum of all these costs.

Evaluating Feasibility

- The first step in evaluating feasibility is to identify and weed out systems requests that are not feasible
- Even if the request is feasible, it might not be necessary
- Feasibility analysis is an ongoing task that must be performed throughout the systems development process

Setting Priorities

- Factors that Affect Priority
 - Will the proposed system reduce costs? Where? When? How? How much?
 - Will the system increase revenue for the company? Where? When? How? How much?

Setting Priorities

- Factors that Affect Priority
 - Will the systems project result in more information or produce better results? How? Are the results measurable?
 - Will the system serve customers better?
 - Will the system serve the organization better?

Setting Priorities

- Factors that Affect Priority
 - Can the project be implemented in a reasonable time period? How long will the results last?
 - Are the necessary financial, human, and technical resources available?
 - Whenever possible, the analyst should evaluate a proposed project based on tangible costs and benefits that represent actual (or approximate) dollar values

Setting Priorities

- Discretionary and Nondiscretionary Projects
 - Projects where management has a choice in implementing them are called discretionary projects
 - Projects where no choice exists are called nondiscretionary projects

Preliminary Investigation Overview

- Preliminary investigation
- Interaction with Managers and Users
 - Let people know about the investigation and explain your role
 - Employee attitudes and reactions are important and must be considered
 - Be careful in your use of the word problem
 - Question users about additional capability they would like to have

Preliminary Investigation Overview

- Planning the Preliminary Investigation
 - During a preliminary investigation, a systems analyst typically follows a series of steps
 - The exact procedure depends on the nature of the request, the size of the project, and the degree of urgency

Preliminary Investigation Overview

- Step 1: Understand the Problem or Opportunity
 - A popular technique for investigating causes and effects is called a fishbone diagram, or Ishikawa diagram

Preliminary Investigation Overview

- Step 2: Define the Project Scope and Constraints
 - Project scope
 - Project creep
 - Constraint

Preliminary Investigation Overview

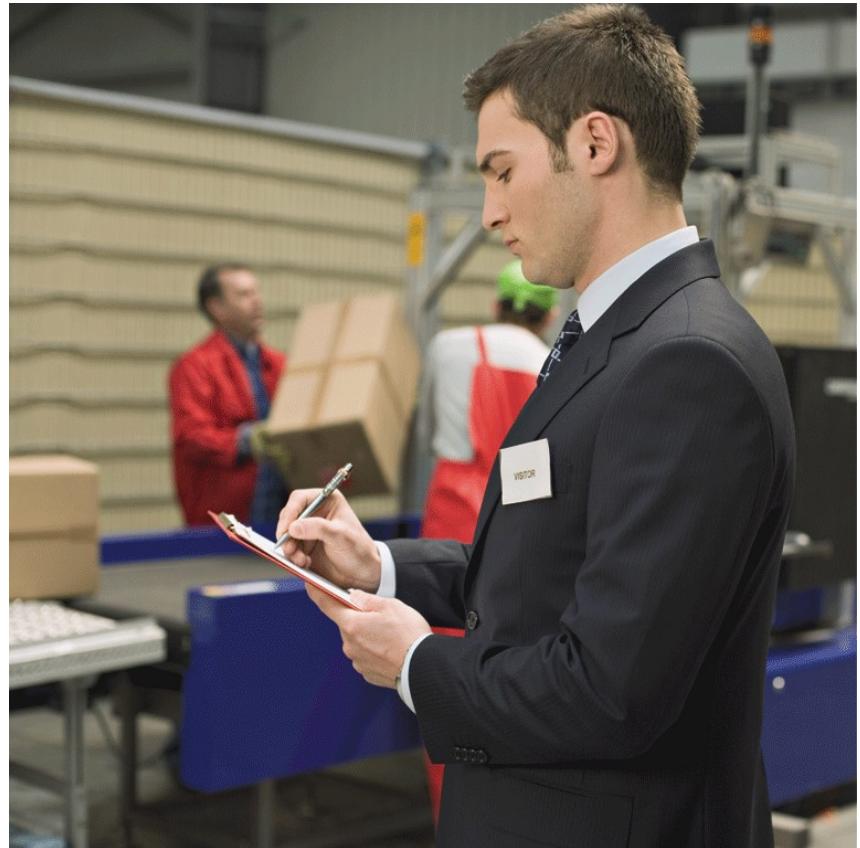
- Step 2: Define the Project Scope and Constraints
 - Present versus future
 - Internal versus external
 - Mandatory versus desirable
 - Regardless of the type, all constraints should be identified as early as possible to avoid future problems and surprises

Preliminary Investigation Overview

- Step 3: Perform Fact-Finding
 - Fact-finding involves various techniques
 - Depending on what information is needed to investigate the systems request, fact-finding might consume several hours, days, or weeks
 - Analyze Organization Charts
 - Obtain organization charts to understand how the department functions and identify individuals you might want to interview

Preliminary Investigation Overview

- Step 3: Perform Fact-Finding
 - Conduct interviews
 - Review documentation
 - Observe operations
 - Conduct a user survey



Preliminary Investigation Overview

- Step 4: Analyze Project Usability, Cost, Benefit, and Schedule Data
 - Before you can evaluate feasibility, you must analyze this data carefully
 - What information must you obtain, and how will you gather and analyze the information?
 - What sources of information will you use, and what difficulties will you encounter in obtaining information?

Preliminary Investigation Overview

- Step 4: Analyze Project Usability, Cost, Benefit, and Schedule Data
 - Will you conduct interviews? How many people will you interview, and how much time will you need to meet with the people and summarize their responses?
 - Will you conduct a survey? Who will be involved? How much time will it take people to complete it? How much time will it take to prepare it and tabulate the results?

Preliminary Investigation Overview

- Step 4: Analyze Project Usability, Cost, Benefit, and Schedule Data
 - How much will it cost to analyze the information gathered and to prepare a report with findings and recommendations?

Preliminary Investigation Overview

- Step 5: Evaluate Feasibility
 - Start by reviewing the answers to the questions you asked
 - Operational feasibility
 - Technical feasibility
 - Economic feasibility
 - Schedule feasibility

Preliminary Investigation Overview

- Step 6: Present Results and Recommendations to Management
 - The final task in the preliminary investigation is to prepare a report to management
 - The format of the preliminary investigation report varies from one company to another

Preliminary Investigation Overview

- Step 6: Present Results and Recommendations to Management
 - Introduction
 - Systems request summary
 - Findings
 - Case for action



Preliminary Investigation Overview

- Step 6: Present Results and Recommendations to Management
 - Project Roles
 - Time & cost estimates
 - Expected benefits
 - Appendix

Chapter Summary

- Strategic planning allows a company to examine its purpose, vision, and values and develops a mission statement, which leads to goals, objectives, day-to-day operations, and business results that affect company stakeholders
- Systems projects are initiated to improve performance, provide more information, reduce costs, strengthen controls, or provide better service

Chapter Summary

- Various internal and external factors affect systems projects, such as user requests, top management directives, existing systems, the IT department, software and hardware vendors, technology, customers, competitors, the economy, and government
- During the preliminary investigation, the analyst evaluates the systems request and determines whether the project is feasible from an operation, technical, economic, and schedule standpoint

Chapter Summary

- Analysts evaluate systems requests on the basis of their expected costs and benefits, both tangible and intangible
- The steps in the preliminary investigation are to understand the problem or opportunity; define the project scope and constraints; perform fact-finding; analyze project usability, cost, benefit, and schedule data; evaluate feasibility; and present results and recommendations to management

Chapter Summary

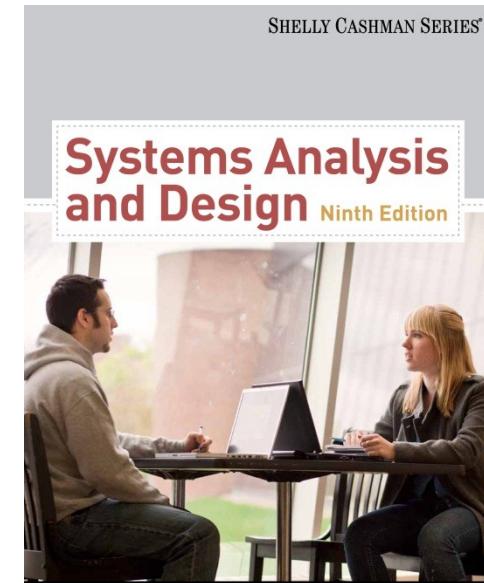
- The last task in a preliminary investigation is to prepare a report to management
- The report must include an estimate of time, staffing requirements, costs, benefits, and expected results for the next phase of the SDLC
- Chapter 2 complete

Systems Analysis and Design 9th Edition

Chapter 3

Managing Systems Projects

SHELLY CASHMAN SERIES



SHELLY | ROSENBLATT

Chapter Objectives

- Explain project planning, scheduling, monitoring, and reporting
- Describe work breakdown structures, task patterns, and critical path analysis
- Explain techniques for estimating task completion times and costs

Chapter Objectives

- Describe various scheduling tools, including Gantt charts and PERT/CPM charts
- Analyze task dependencies, durations, start dates, and end dates
- Describe project management software and how it can assist you in project planning, estimating, scheduling, monitoring, and reporting

Chapter Objectives

- Discuss the importance of project risk management
- Understand why projects sometimes fail

Introduction

- You will learn about project planning, estimating, scheduling, monitoring, reporting, and the use of project management software
- You also will learn how to control and manage project changes as they occur

Overview of Project Management

- Project Management
- A successful project must be completed on time, within budget, and deliver a quality product that satisfies users and meets requirements
- Project manager or project leader
- Project coordinator

Overview of Project Management

- What Does a Project Manager Do?
 - Project manager, project leader
 - Project planning
 - Project scheduling
 - Project monitoring and controlling
 - Project reporting

Overview of Project Management

- Project Activities and Planning Steps

	Planning	Scheduling	Monitoring	Reporting
STEP1: Create a work breakdown structure	✓			
STEP 2: Identify task patterns	✓	✓		
STEP 3: Calculate the critical path	✓	✓		
Manage the operational project				

Step 1: Create a Work Breakdown Structure

- Work breakdown structure (WBS)
- What is a Gantt Chart?
 - Task group
 - Can present an overview of the project's status, but does not provide detailed information that is necessary when managing a complex project

Step 1 – Cont.

- **Project Planning and Requirements Gathering:** This includes defining the scope, objectives, resources, and timelines of the project.
- **Software Development:**
 - **Module or Component Development:** Breaking down the software into smaller modules or components.
 - **Coding:** Writing the actual code for the various components.
 - **Unit Testing:** Testing individual modules or components.
- **Integration and Testing:**
 - **System Integration:** Combining different modules or components into a fully functioning system.
 - **System Testing:** Running tests to ensure the system meets all functional and non-functional requirements.
 - **Bug Fixes and Optimizations:** Addressing issues found during testing.
- **Deployment and Maintenance:**
 - **Deployment:** Installing the software into a production environment.
 - **User Training:** Providing documentation and training to users or clients.
- **Project Management Activities:**
 - **Risk Management:** Identifying and mitigating risks that could affect the project.
 - **Quality Assurance:** Ensuring that the software meets quality standards.

Step 1: Create a Work Breakdown Structure

- What is a PERT/CPM Chart?
 - The Program Evaluation Review Technique (PERT)
 - Critical Path Method (CPM)
 - The distinction between the two methods has disappeared over time

Step 1: Create a Work Breakdown Structure

- What is a PERT/CPM Chart ?

A **PERT (Program Evaluation Review Technique)** chart is used for **time estimation**, focusing on uncertain projects with variable task durations, while a **CPM (Critical Path Method)** chart identifies the **longest path of dependent tasks** critical to project completion. Both break down projects into tasks, showing dependencies and determining the fastest timeline.

Step 1: Create a Work Breakdown Structure

- Which Type of Chart is Better?
 - Although a Gantt chart offers a **valuable snapshot view of the project**, PERT charts are more useful for **scheduling, monitoring, and controlling the actual work**
 - PERT and Gantt charts are not mutually exclusive techniques, and project managers often use both methods

Step 1: Create a Work Breakdown Structure

- Identifying Tasks in a Work Breakdown Structure
 - Task or activity
 - Event or milestone
 - Break the project down into smaller tasks, creating a work breakdown structure

Step 1: Create a Work Breakdown

- Identifying Tasks in a Work Breakdown Structure

- Listing the tasks

- Can be challenging, because the tasks **might** be embedded in a document
 - Create a table with columns for task number, description, duration, and predecessor tasks

Task No.	Description	Duration (Days)	Predecessor Tasks
1	Reserve the meeting room		
2	Order the marketing materials		
3	Brief the managers		
4	Send out customer e-mails		
5	Burn sample DVDs		
6	Load the new software		
7	Do a dress rehearsal		

Task Number	Description	Duration (Days)	Predecessor Tasks
1	Project Planning	5	-
2	System Design	10	1
3	Module Development	15	2
4	Integration & Testing	7	3
5	Deployment & Maintenance	3	4

Step 1: Create a Work Breakdown

- Identifying Tasks in a Work Breakdown Structure
 - Estimating Task Duration
 - Person-days (**This measures the amount of work one person can complete in a day.**)
 - Best-case estimate (**B**) (**This is the quickest time a task might take under ideal conditions, without any delays**)
 - Probable-case estimate (**P**) (**A more realistic estimate that considers normal delays and obstacles, often used for reliable planning.**)

Step 1: Create a Work Breakdown Structure

- Identifying Tasks in a Work Breakdown Structure
 - Estimating Task Duration
 - Worst-case estimate (**W**)
 - Weight

$$\underline{(B+4P+W)}$$

6

Step 1: Create a Work Breakdown Structure

- Identifying Tasks in a Work Breakdown Structure
 - Factors Affecting Duration
 - Project size and scope
 - Human resources
 - Experience with similar project
 - constraints

Step 1: Create a Work Breakdown

- Displaying the Work Breakdown Structure
 - If you are managing a complex project with many tasks, you can use task groups, just as you would in a Gantt chart, to simplify the list. This structure allows for easier tracking and delegation, with a clear view of dependencies and the critical path. Would you like assistance in creating a specific WBS structure for your project?
- For example:
 - **Level 1:** Main project phases (e.g., Design, Development, Testing)
 - **Level 2:** Subtasks or deliverables within each phase (e.g., Module 1 Development, Integration Testing)
 - **Level 3:** Further breakdown of tasks if necessary (e.g., Coding, Debugging)

- Any Questions regarding the project?

Step 2: Identify Task Patterns

- What are Task Patterns?

Software Task Patterns refer to **reusable solutions** or best practices for commonly occurring tasks or problems in software development. **Some common task patterns include:**

- **Design Patterns:** Solutions to recurring design problems in software architecture.
- **Development Patterns:** Repeated approaches to handling tasks like error handling, logging, or resource management.
- **Testing Patterns:** Structured methods for writing and organizing tests.
- **Project Management Patterns:** Approaches to managing tasks in software projects, such as Agile task boards or Scrum sprints
- Task patterns can involve **dependent tasks**, **multiple successor tasks**, and **multiple predecessor tasks**

Step 2: Identify Task Patterns

- How do I Use Task Boxes to Create a Model?

TASK BOX FORMAT

Task Name	
Start Day/Date	Task ID
Finish Day/Date	Task Duration

Step 2: Identify Task Patterns

- What are the Main Types of Task Patterns?
 - Dependent Tasks
 - Multiple successor tasks
 - Concurrent task
 - Predecessor task
 - Successor task
 - Multiple Predecessor Tasks

Step 2: Identify Task Patterns

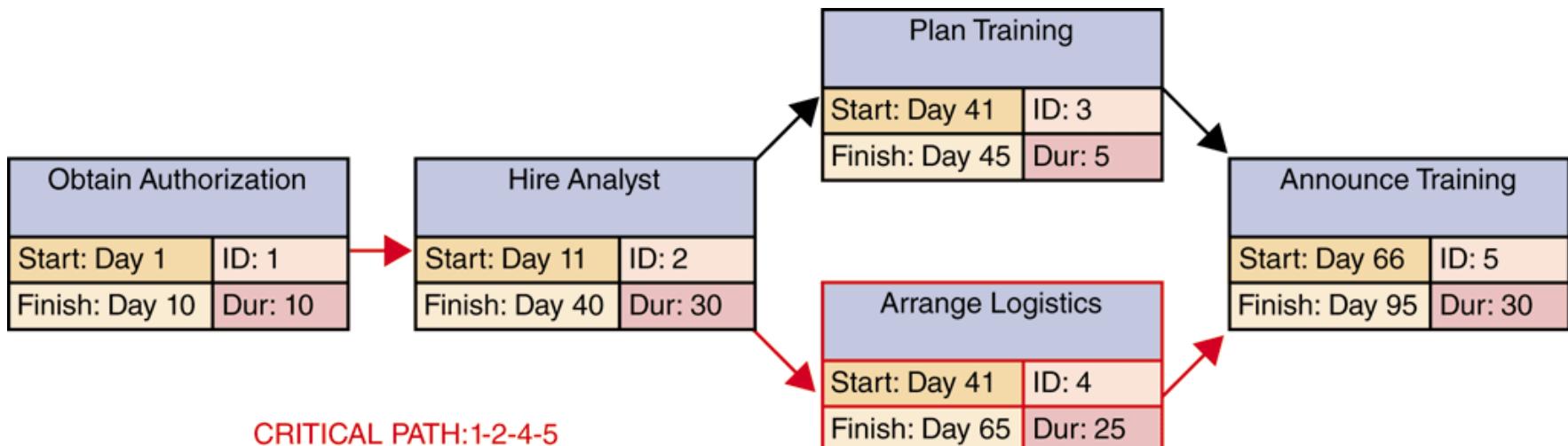
- How Do I Identify Task Patterns?
 - You can identify task patterns by looking carefully at the wording of the task statement
 - Words like *then*, *when*, or *and* are action words that signal a sequence of events
- How Do I Work With Complex Task Patterns?
 - When various task patterns combine, you must study the facts carefully in order to understand the logical sequence

Step 2: Identify Task Patterns

- How Do I Work With Complex Task Patterns?
 - Consider the following three fact statements and the task patterns they represent
 - Dependent tasks
 - Dependent tasks and multiple successor tasks
 - Dependent tasks, multiple successor tasks, and multiple predecessor tasks

Step 3: Calculate the Critical Path

- What Is a Critical Path?



Step 3: Calculate the Critical Path

- How Do I Calculate the Critical Path?
 - First, you should review the task patterns
 - The next step is to determine start and finish dates, which will determine the critical path for the project
 - Slack time

Project Monitoring and Control

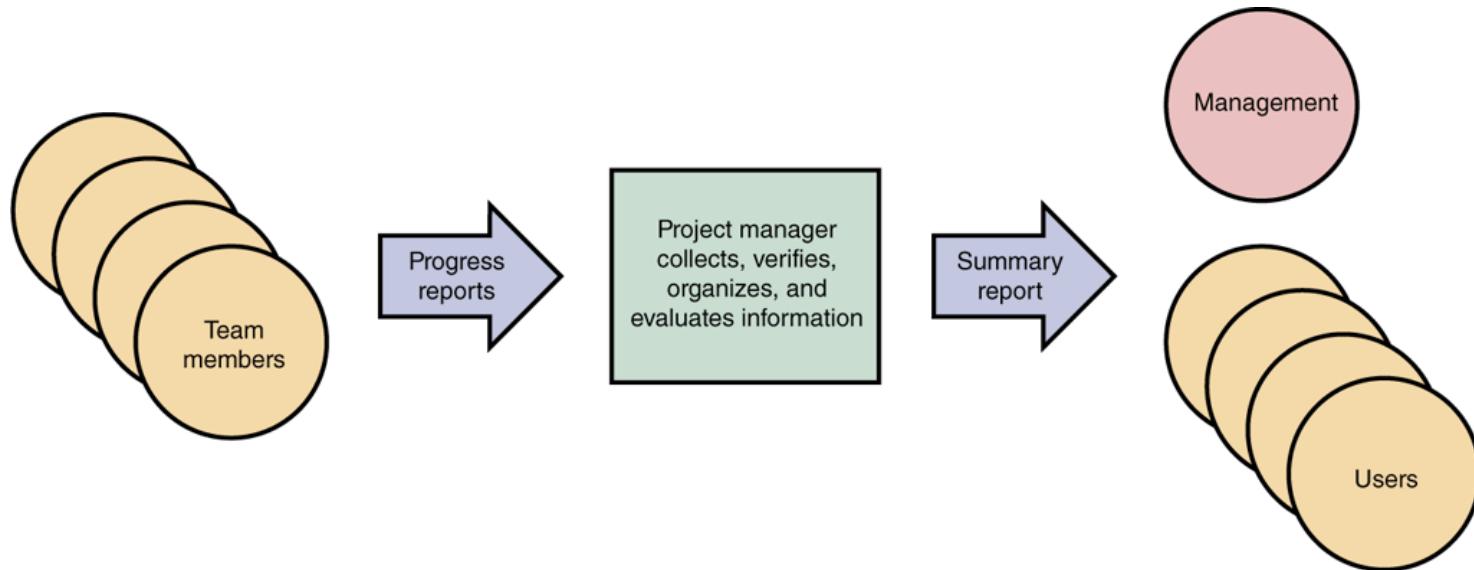
- Monitoring and Control Techniques
 - The project manager must keep track of tasks and progress of team members, compare actual progress with the project plan, verify the completion of project milestones, and set standards and ensure that they are followed
 - Structured walkthrough (**it is a formal review process in software development, where developers, testers, and stakeholders collaboratively review a software product or its components (e.g., requirements, design, code) to identify defects, verify compliance with standards, and ensure that it meets predefined criteria. The structured walkthrough is aimed at improving quality by uncovering potential issues early in the development process**)

Project Monitoring and Control

- Maintaining a Schedule
 - Maintaining a project schedule can be a challenging task
 - The better the **original plan**, the easier it will be to control the project
 - If **enough milestones** and frequent checkpoints exist, problems will be detected rapidly
 - Project managers often spend most of their time tracking the **tasks along the critical path**

Reporting

- Members of the project team regularly report their progress
- Project Status Meetings



Reporting

- Project Status Reports
 - A project manager must report regularly to his or her immediate supervisor, upper management, and users
 - Should explain what you are doing to handle and monitor the problem
 - Most managers recognize that problems do occur on most projects; it is better **to alert management sooner rather than later**

Project Management Examples

- PERT/CPM Example
 - You construct a PERT/CPM chart from this task list in a two-step process
 - Step 1: Create the work breakdown structure
 - Step 2: Enter start and finish times

TASK BOX FORMAT

Task Name	
Start Day/Date	Task ID
Finish Day/Date	Task Duration

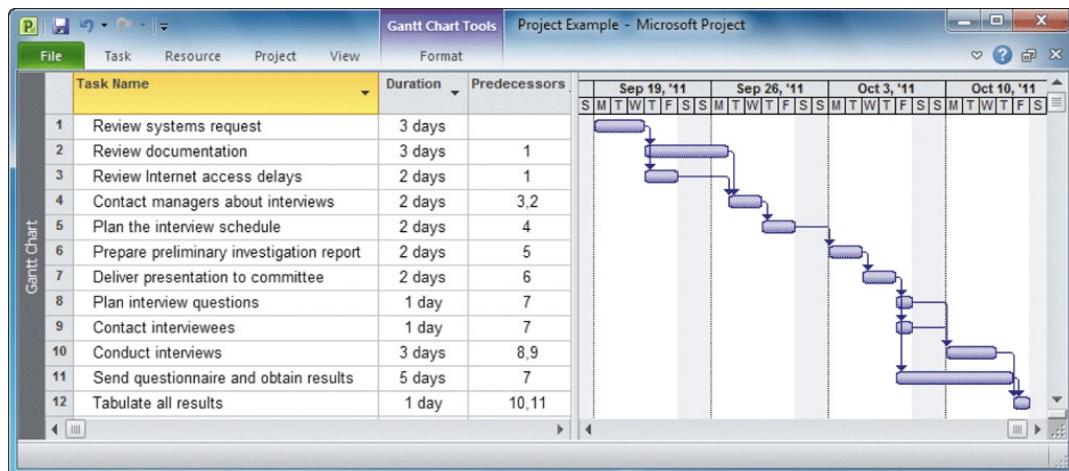
Project Management Examples

- Software-Driven Example
 - Open Workbench
 - Open-source software
 - When you use project management software, you follow the same step-by-step process to develop a WBS and create various types of charts



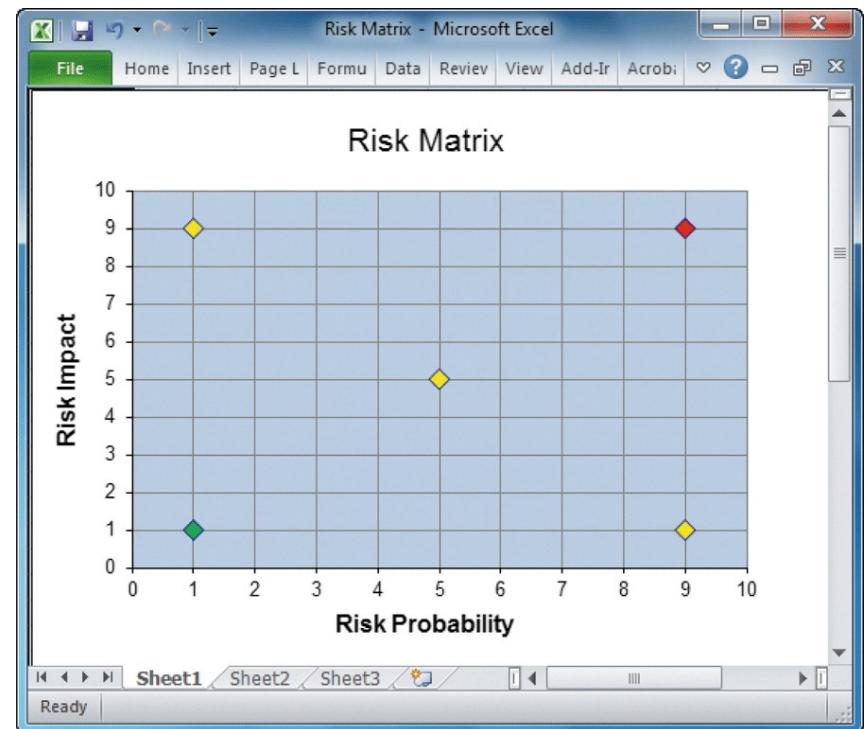
Project Management Examples

- Software-Driven Example
 - Work breakdown structure
 - Gantt chart
 - Network diagram
 - Project planning is a dynamic task and involves constant change



Risk Management

- Every IT project involves risks that systems analysts and project managers must address
- Risk management
- Steps in Risk Management
 - Develop risk management plan
 - Identify the risks
 - Risk identification



Risk Management

- Steps in Risk Management (continued)
 - Analyze the risks
 - Qualitative risk analysis
 - Quantitative risk analysis
 - Create a risk response plan
 - Monitor risks

Risk Management

- Risk Management Software
 - Most project management software includes powerful features
 - The IT team can make a recommendation regarding the risks
 - Depending on the nature and magnitude of the risk, the final decision might be made by management

Managing for Success

- Business Issues
 - The major objective of every system is to provide a solution to a business problem or opportunity
 - A system that falls short of business needs also produces problems for users and reduces employee morale and productivity
 - Project creep

Managing for Success

- Budget Issues
 - Cost overruns typically result from one or more of the following:
 - Unrealistic estimates
 - Failure to develop an accurate forecast that considers all costs over the life of the project
 - Poor monitoring of progress and slow response to early warning signs of problems

Managing for Success

- Budget Issues
 - Cost overruns typically result from one or more of the following:
 - Schedule delays due to factors that were not foreseen
 - Human resource issues, including turnover, inadequate training, and motivation

Managing for Success

- Schedule Issues
 - Problems with timetables and project milestones can indicate a failure to recognize task dependencies, confusion between effort and progress, poor monitoring and control methods, personality conflicts among team members, or turnover of project personnel

The Bottom Line

- When problems occur, the project manager's ability to handle the situation becomes the critical factor



The Bottom Line

- Sometimes, when a project experiences delays or cost overruns, the system still can be delivered on time and within budget if several less critical requirements are trimmed
- Brooks' Law

Chapter Summary

- Project management is the process of planning, scheduling, monitoring and controlling, and reporting upon the development of an information system
- Project managers are responsible for project planning, scheduling, monitoring, and reporting
- Planning, scheduling, monitoring and reporting all take place within a larger project development framework

Chapter Summary

- In project scheduling, the project manager develops a specific time for each task, based on available resources and whether or not the task is dependent on other predecessor tasks
- Every successful information system must support business requirements, stay within budget, and be available on time
- Sound project management involves the same skills as any other management

Chapter Summary

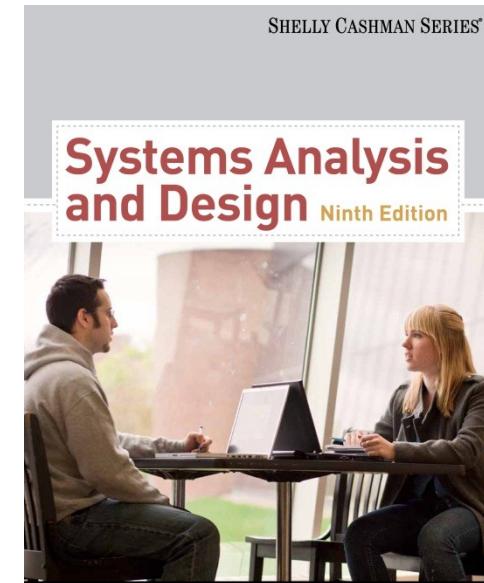
- Chapter 3 complete

Systems Analysis and Design 9th Edition

Chapter 4

Requirements Modeling

SHELLY CASHMAN SERIES



SHELLY | ROSENBLATT

Phase Description

- Systems analysis is the second of five phases in the systems development life cycle (SDLC)
- Will use requirements modeling, data and process modeling, and object modeling techniques to represent the new system
- Will consider various development strategies for the new system, and plan for the transition to systems design tasks

Chapter Objectives

- Describe systems analysis phase activities
- Explain joint application development (JAD), rapid application development (RAD), and agile methods
- Use a functional decomposition diagram (FDD) to model business functions and processes

Chapter Objectives

- Describe the Unified Modeling Language (UML) and examples of UML diagrams
- List and describe system requirements, including outputs, inputs, processes, performance, and controls
- Explain the concept of scalability

Chapter Objectives

- Use fact-finding techniques, including interviews, documentation review, observation, questionnaires, sampling, and research
- Define total cost of ownership (TCO)
- Conduct a successful interview
- Develop effective documentation methods to use during systems development

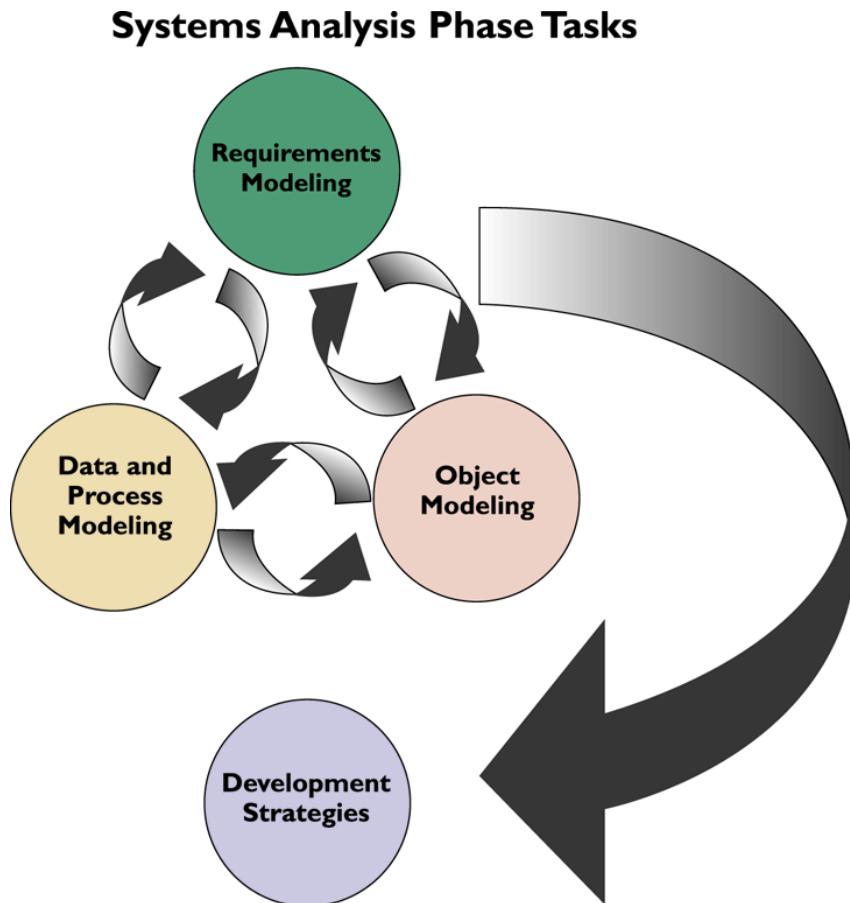
Introduction

- This chapter describes requirements modeling techniques and team-based methods that systems analysts use to visualize and document new systems
- The chapter then discusses system requirements and fact-finding techniques, which include interviewing, documentation review, observation, surveys and questionnaires, sampling, and research

Systems Analysis Phase Overview

- The overall objective of the **systems analysis** phase is to **understand the proposed project**, ensure that it will support business requirements, and build a solid foundation for system development
- You use **models** and other documentation tools **to visualize** and describe the proposed system

Systems Analysis Phase Overview



- Systems Analysis Activities
 - Requirements modeling
 - Outputs
 - Inputs
 - Processes
 - Performance
 - Security

Systems Analysis-Requirements Modeling

It is the process of defining the **functionalities**, **services**, and **constraints** of a system based on user needs.

- **Key Activities:**
 - **Gathering requirements** from users, stakeholders, and managers.
 - **Analyzing business processes** to understand the **workflow** and how the system will support it.
 - **Defining system functionalities** and expected behaviors.
- **Techniques Used:**
 - Interviews, surveys, observation, and document analysis to collect requirements.
 - **Use case modeling** and **data flow diagrams (DFDs)** to visually represent the system's operations

Systems Analysis-Inputs

- Inputs are the data or commands **entered into the system to be processed**. Inputs can come from various sources like users, other systems, sensors, or databases.
- **Examples:**
 - User input in a form (e.g., a **customer filling** out an online order form).
 - Automated inputs from IoT devices (e.g., temperature sensors sending data to a control system).

Systems Analysis -Processes

- Processes refer to the **set of activities or operations** that the system performs on inputs to generate outputs. This can involve calculations, data transformations, filtering, validation, or decision-making algorithms.
- Key Types:
 - **Business Processes:** Series of steps to achieve a business goal (e.g., order processing, payroll calculations).
 - **System Processes:** Logical operations or

Systems Analysis- Performance

- Performance defines how **well a system operates**, particularly regarding **speed, responsiveness, and efficiency**.
- **Performance Criteria:**
 - **Response time:** The time taken to respond to user input.
 - **Throughput:** The volume of data processed in a given time.
 - **Scalability:** The system's ability to handle increasing amounts of data or users.
 - **Reliability:** How consistently the system functions without failures

Systems Analysis Phase Overview

- Systems Analysis Activities
 - Data and process modeling
 - Object Modeling
 - Development Strategies
 - System requirements document

Systems Analysis Phase Overview

- Systems Analysis Skills
 - Analytical skills
 - Interpersonal skills
- Team-Oriented Methods and Techniques
 - Joint application development (JAD)
 - Rapid application development (RAD)
 - Agile methods

Joint Application Development

- User Involvement
 - Users have a vital stake in an information system and they should participate fully
 - Successful systems must be user-oriented, and users need to be involved
 - One popular strategy for user involvement is a JAD team approach

Joint Application Development

- JAD Participants and Roles

JAD PARTICIPANT	ROLE
JAD project leader	Develops an agenda, acts as a facilitator, and leads the JAD session
Top management	Provides enterprise-level authorization and support for the project
Managers	Provide department-level support for the project and understanding of how the project must support business functions and requirements
Users	Provide operational-level input on current operations, desired changes, input and output requirements, user interface issues, and how the project will support day-to-day tasks
Systems analysts and other IT staff members	Provide technical assistance and resources for JAD team members on issues such as security, backup, hardware, software, and network capability
Recorder	Documents results of JAD sessions and works with systems analysts to build system models and develop CASE tool documentation

Joint Application Development

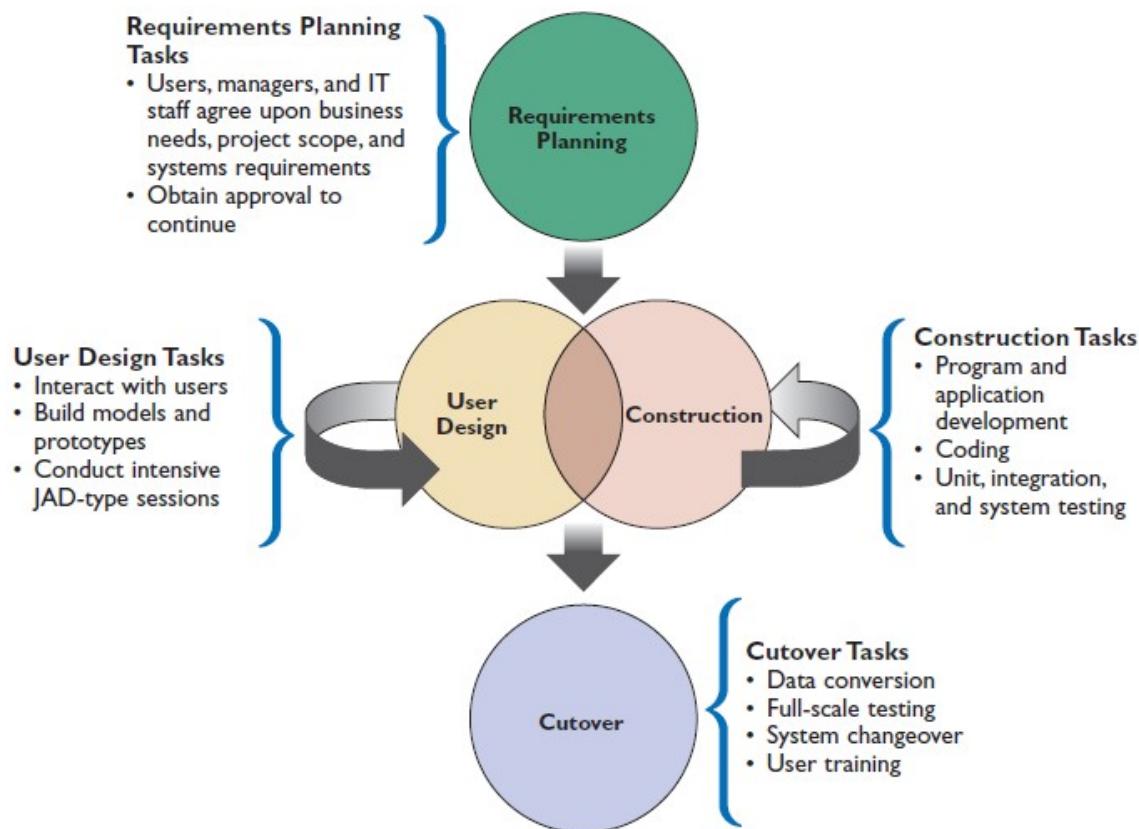
- JAD **Advantages** and **Disadvantages**
 - More **expensive** and can be cumbersome if the group is too large relative to the size of the project
 - Allows **key users to participate** effectively
 - When properly used, JAD can result in a more **accurate statement of system requirements**, a better understanding of common goals, and a stronger commitment to the success of the new system

Rapid Application Development

- Is a team-based technique that speeds up information systems development and produces a functioning information system
- Relies heavily on prototyping and user involvement
- Interactive process continues until the system is completely developed and users are satisfied

Rapid Application Development

- RAD Phases and Activities



Rapid Application Development

- RAD **Objectives**
 - To **cut development time** and expense by involving the users in every phase of systems development
 - Successful RAD team must have IT resources, skills, and **management support**
 - Helps a development team design **a system that requires a highly interactive** or complex user interface

Rapid Application Development

- RAD Advantages and Disadvantages
 - Systems can be developed more quickly with significant cost savings
 - RAD stresses the mechanics of the system itself and **does not emphasize the company's strategic business needs**
 - Might allow less time to develop quality, consistency, and design standards

Agile Methods

- Attempt to develop a system incrementally
- Agilian modeling toolset includes support for many modeling tools
- Some agile developers prefer not to use CASE tools at all, and rely instead on whiteboard displays and arrangements of movable sticky notes

Agile Methods

- Scrum is a rugby term
- Pigs include the product owner, the facilitator, and the development team; while the chickens include users, other stakeholders, and managers
- Scrum sessions have specific guidelines that emphasize time blocks, interaction, and team-based activities that result in deliverable software

Agile Methods

- Attempt to develop a system incrementally:
 - Agile development emphasizes delivering a working system in small, usable **increments** rather than a single, large release at the end. This means the system is built and improved in stages, allowing for regular feedback and adjustments.
 - Example: In a software project, **an initial release might include core features. With each increment, additional features or refinements are added**, allowing the team to adapt to changes in requirements or user feedback without overhauling the entire system.

Modeling Tools and Techniques

Agilian modeling toolset includes support for many modeling tools:

- The Agilian modeling toolset is an example of CASE (Computer-Aided Software Engineering) tools used to assist in **visualizing and designing software systems**. These tools typically support different types of modeling such as UML (Unified Modeling Language) diagrams, data flow diagrams, and process modeling, all of which are commonly used in Agile development to keep models lightweight and adaptive to changes.
- **Example:** A project team might use a tool like Visual Paradigm (Agilian) to create a UML class diagram that evolves as the system changes incrementally, instead of creating a static and rigid design document upfront.

Use CASE tools for Agile

Some agile developers prefer **NOT to use CASE tools** at all, and rely instead on **whiteboard displays** and arrangements of **movable sticky notes**:

- Agile teams often prefer **lightweight, informal methods for organizing their work and sharing ideas**. Whiteboards and sticky notes provide a simple, flexible way to visualize tasks and progress, enabling constant updates and real-time collaboration.
- **Example:** In a Scrum meeting, a team might use a **whiteboard with sticky notes representing user stories**. These can be moved across columns like "To Do," "In Progress," and "Completed," reflecting the status of tasks without the need for digital tools.

Agile Method Advantages

- Very flexible and efficient in dealing with change: Agile methods prioritize adaptability, allowing teams to respond to changes in requirements or user feedback throughout the development process.
 - Example: If a client requests a new feature mid-project, Agile teams can reprioritize their work to accommodate the change in the next sprint.
- Frequent deliverables constantly validate the project and reduce risk: Delivering small, frequent updates ensures that the project stays aligned with user needs and minimizes the risk of large-scale failures.
 - Example: Releasing a new feature every two weeks allows the client to test and provide feedback, reducing the chances of major rework later on

Agile Method Disadvantages

- Team members need a **high level of technical and interpersonal skills**: Agile teams are expected to work autonomously and collaboratively, requiring not only technical expertise but also strong communication and teamwork skills.
 - **Example:** Team members must communicate openly and adapt quickly to changes, which can be challenging for individuals who are used to more rigid project structures.
- May be subject to **significant change in scope**: Because Agile encourages flexibility and responsiveness to feedback, projects can sometimes experience "scope creep" where additional features or changes are requested, potentially leading to delays or increased costs.
 - **Example:** A project might begin with a defined scope, but over time, new features are added based on user feedback, extending the timeline and increasing the workload

Modeling Tools and Techniques

- Business Process Modeling
 - Business process model (BPM)
 - Business process modeling notation (BPMN)
 - Pool
 - Swim lanes

Business Process Modeling (BPM)

- Business Process Modeling (BPM) is the practice of **representing an organization's processes and workflows in a visual format**. The goal of BPM is to analyze and improve existing processes or design new ones to ensure efficiency and effectiveness in operations. **By mapping out the steps of a process, businesses can identify bottlenecks, inefficiencies, and opportunities for improvement.**
- **Example:** A company may model its **order** fulfillment process to visualize how orders **are received, processed, and shipped**, aiming to reduce delays and optimize resource allocation.

Pool

- In BPMN, a **pool** represents a major participant in the process, **usually a company, an organization, business unit, or system**. It is a container for a business process that shows how a specific entity interacts with other entities. Pools can be divided into multiple processes but always represent the boundary of an entity's interaction.
- **Example:** In a business-to-business process, **one pool might represent "Company A" and another pool might represent "Company B,"** each interacting with the other through the exchange of documents or data

Swim Lanes

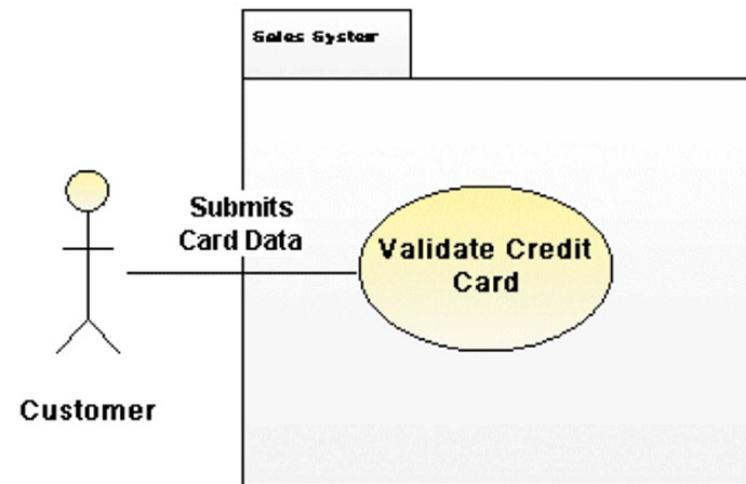
- Swim lanes are **sub-divisions within a pool** in BPMN diagrams. They represent **specific roles, departments, or systems** involved in the process. Each swim lane is responsible for **certain tasks** or activities within the process. Swim lanes help clarify which participant or group is responsible for specific steps within the process.
- **Example:** In a customer service process, a swim lane might be used to show that the "**Sales Department**" is **responsible for customer inquiries**, the "**IT Department**" is responsible for technical issues, and the "**Billing Department**" handles payment-related tasks

Modeling Tools and Techniques

- Data Flow Diagrams
 - Data flow diagram (DFD)
 - show how the system stores, processes, and transforms data
 - Additional levels of information and detail are depicted in other, related DFDs

Modeling Tools and Techniques

- Unified Modeling Language
 - Widely used method of visualizing and documenting software systems design
 - Use case diagrams
 - Actor
 - Sequence diagrams



System Requirements Checklist

- Outputs
 - The Web site must report online volume statistics every four hours, and hourly during peak periods
 - The inventory system must produce a daily report showing the part number, description, quantity on hand, quantity allocated, quantity available, and unit cost of all sorted by part number

System Requirements Checklist

- Inputs
 - Manufacturing employees must swipe their ID cards into online data collection terminals that record labor costs and calculate production efficiency
 - The department head must enter overtime hours on a separate screen

System Requirements Checklist

- Processes
 - The student records system must calculate the GPA at the end of each semester
 - As the final step in year-end processing, the payroll system must update employee salaries, bonuses, and benefits and produce tax data required by the IRS

System Requirements Checklist

- Performance
 - The system must support 25 users online simultaneously
 - Response time must not exceed four seconds

System Requirements Checklist

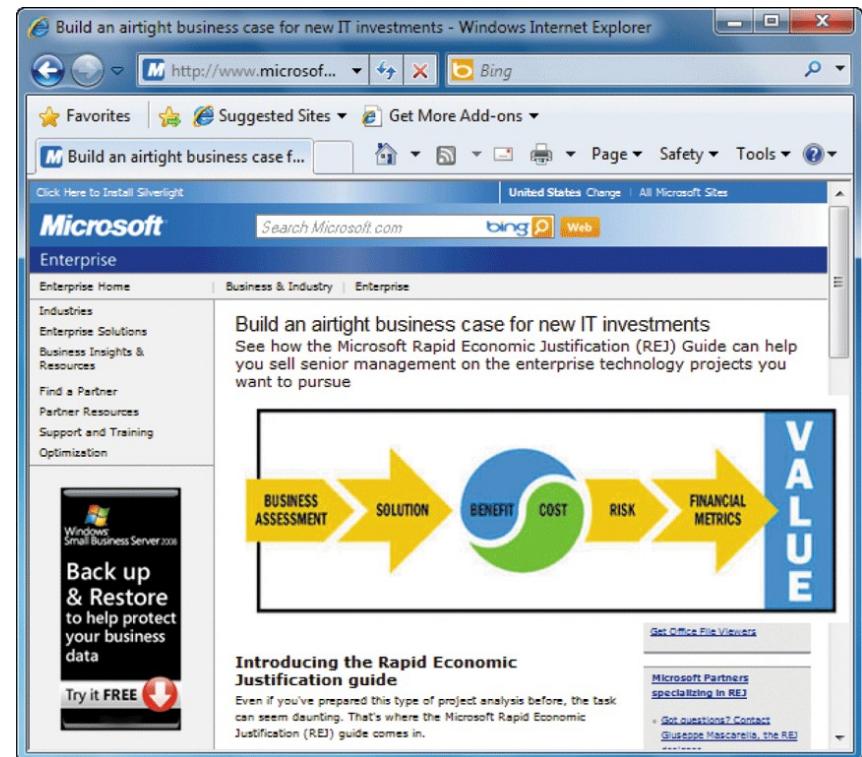
- Controls
 - The system must provide logon security at the operating system level and at the application level
 - An employee record must be added, changed, or deleted only by a member of the human resources department

Future Growth, Costs, and Benefits

- Scalability
 - A scalable system offers a better return on the initial investment
 - To evaluate scalability, you need information about projected future volume for all outputs, inputs, and processes

Future Growth, Costs, and Benefits

- Total Cost of Ownership
 - Total cost of ownership (TCO) is especially important if the development team is evaluating several alternatives
 - One problem is that cost estimates tend to underestimate indirect costs
 - Rapid Economic Justification (REJ)

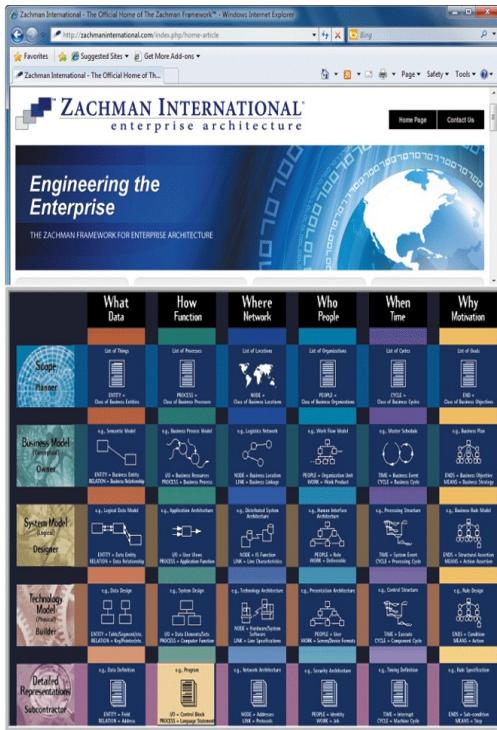


Fact-Finding

- Fact-Finding Overview
 - First, you must identify the information you need
 - Develop a fact-finding plan
- Who, What, Where, When, How, and Why?
 - Difference between asking what is being done and what could or should be done

Fact-Finding

- The Zachman Framework
- **it helps organizations manage complexity by organizing data, processes, and technology into an understandable framework**



- Zachman Framework for Enterprise Architecture
- Helps managers and users understand the model and assures that overall business goals translate into successful IT projects

Example - The Zachman Framework

Perspective	What (Data)	How (Function)	Where (Network)	Who (People)	When (Time)
Planner	Patient, Doctor Info	Appointment Booking	Web and Mobile Access	Patients, Doctors, Admin	Anytime Booking
Owner	Patient Profiles, Doctor Schedules	Search Doctors, Book Appointment	Cloud-based System	Patients, Doctors, IT Support	Notifications 24 Hours Before
Designer	Database Schema (Patients, Doctors, Appointments)	Search, Book, Sync Calendars	Hosted on Cloud, Mobile Access	User Roles: Patient, Doctor, Admin	Automated Reminders
Builder	PostgreSQL Database	Python/JavaScript for Booking	AWS Cloud, Secure API	OAuth 2.0 for Authentication	Event-Driven Notifications

Interviews

- Step 1: Determine the People to Interview
 - Informal structures
- Step 2: Establish Objectives for the Interview
 - Determine the general areas to be discussed
 - List the facts you want to gather



Interviews

- Step 3: Develop Interview Questions
 - Creating a standard list of interview questions helps to keep you on track and avoid unnecessary tangents
 - Avoid leading questions
 - Open-ended questions
 - Closed-ended questions
 - Range-of-response questions

Interviews

- Step 4: Prepare for the Interview
 - Careful preparation is essential because an interview is an important meeting and not just a casual chat
 - Limit the interview to no more than one hour
 - Send a list of topics
 - Ask the interviewee to have samples available

Interviews

- Step 5: Conduct the Interview
 - Develop a specific plan for the meeting
 - Begin by introducing yourself, describing the project, and explaining your interview objectives
 - Engaged listening
 - Allow the person enough time to think about the question
 - After an interview, you should summarize the session and seek a confirmation

Interviews

- Step 6: Document the Interview
 - Note taking should be kept to a minimum
 - After conducting the interview, you must record the information quickly
 - After the interview, send memo to the interviewee expressing your appreciation
 - Note date, time, location, purpose of the interview, and the main points you discussed so the interviewee has a written summary and can offer additions or corrections

Interviews

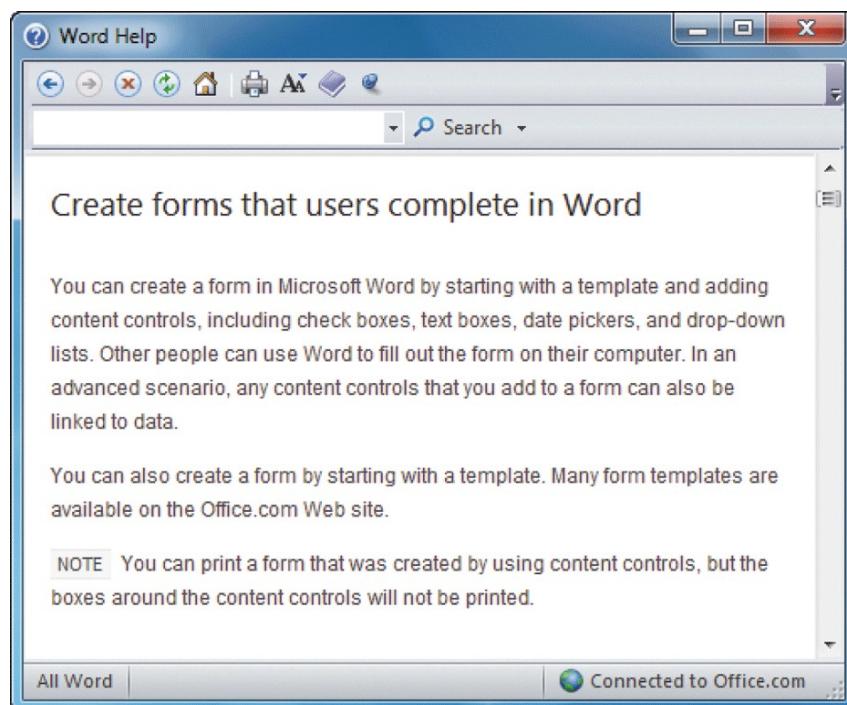
- Step 7: Evaluate the Interview
 - In addition to recording the facts obtained in an interview, try to identify any possible biases
- Unsuccessful Interviews
 - No matter how well you prepare for interviews, some are not successful

Other Fact-Finding Techniques

- Document Review
- Observation
 - Seeing the system in action gives you additional perspective and a better understanding of the system procedures
 - Plan your observations in advance
 - Hawthorne Effect



Other Fact-Finding Techniques



- Questionnaires and Surveys
 - When designing a questionnaire, the most important rule of all is to make sure that your questions collect the right data in a form that you can use to further your fact-finding
 - Fill-in form

Other Fact-Finding Techniques

- Sampling
 - Systematic sample
 - Stratified sample
 - Random sample
 - Main objective of a sample is to ensure that it represents the overall population accurately

Other Fact-Finding Techniques

- Research
 - Can include the Internet, IT magazines, and books to obtain background information, technical material, and news about industry trends and developments
 - Site visit



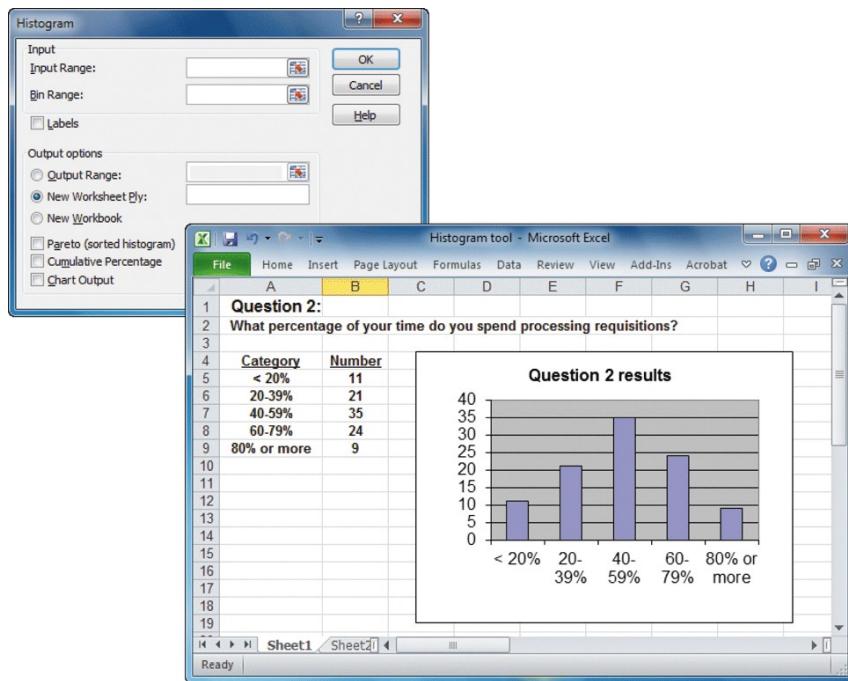
Other Fact-Finding Techniques

- Interviews versus Questionnaires
 - Interview is more familiar and personal
 - Questionnaire gives many people the opportunity to provide input and suggestions
 - Brainstorming
 - Structured brainstorming
 - Unstructured brainstorming

Documentation

- The Need for Recording the Facts
 - Record information as soon as you obtain it
 - Use the simplest recording method
 - Record your findings in such a way that they can be understood by someone else
 - Organize your documentation so related material is located easily

Documentation



- Software Tools
 - CASE Tools
 - Productivity Software
 - Word processing, spreadsheets, database management, presentation graphics, and collaborative software programs
 - Histogram

Documentation

- Software Tools
 - Graphics modeling software
 - Personal information managers
 - Wireless communication devices

Preview of Logical Modeling

- At the conclusion of requirements modeling, systems developers should have a clear understanding of business processes and system requirements
- The next step is to construct a logical model of the system
- IT professionals have differing views about systems development methodologies, and no universally accepted approach exists

Chapter Summary

- The systems analysis phase includes three activities: requirements modeling, data and process modeling, and consideration of development strategies
- The main objective is to understand the proposed project, ensure that it will support business requirements, and build a solid foundation for the systems design phase

Chapter Summary

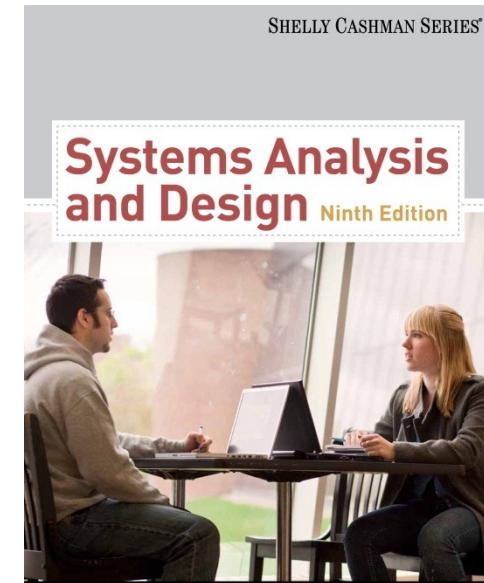
- The fact-finding process includes interviewing, document review, observation, questionnaires, sampling, and research
- Systems analysts should carefully record and document factual information as it is collected, and various software tools can help an analyst visualize and describe an information system
- Chapter 4 complete

Systems Analysis and Design 9th Edition

Chapter 5

Data and Process Modeling

SHELLY CASHMAN SERIES



SHELLY | ROSENBLATT

Chapter Objectives

- Describe data and process modeling concepts and tools, including data flow diagrams, a data dictionary, and process descriptions
- Describe the symbols used in data flow diagrams and explain the rules for their use
- Draw data flow diagrams in a sequence, from general to specific
- Explain how to level and balance a set of data flow diagrams

Chapter Objectives

- Describe how a data dictionary is used and what it contains
- Use process description tools, including structured English, decision tables, and decision trees
- Describe the relationship between logical and physical models

Introduction

- In Chapters 5 & 6, you will develop a logical model of the proposed system and document the system requirements
 - Logical model shows what the system must do
 - Physical model describes how the system will be constructed

Overview of Data and Process Modeling Tools

- Systems analysts use many graphical techniques to describe an information system
- A data flow diagram (DFD) uses various symbols to show how the system transforms input data into useful information

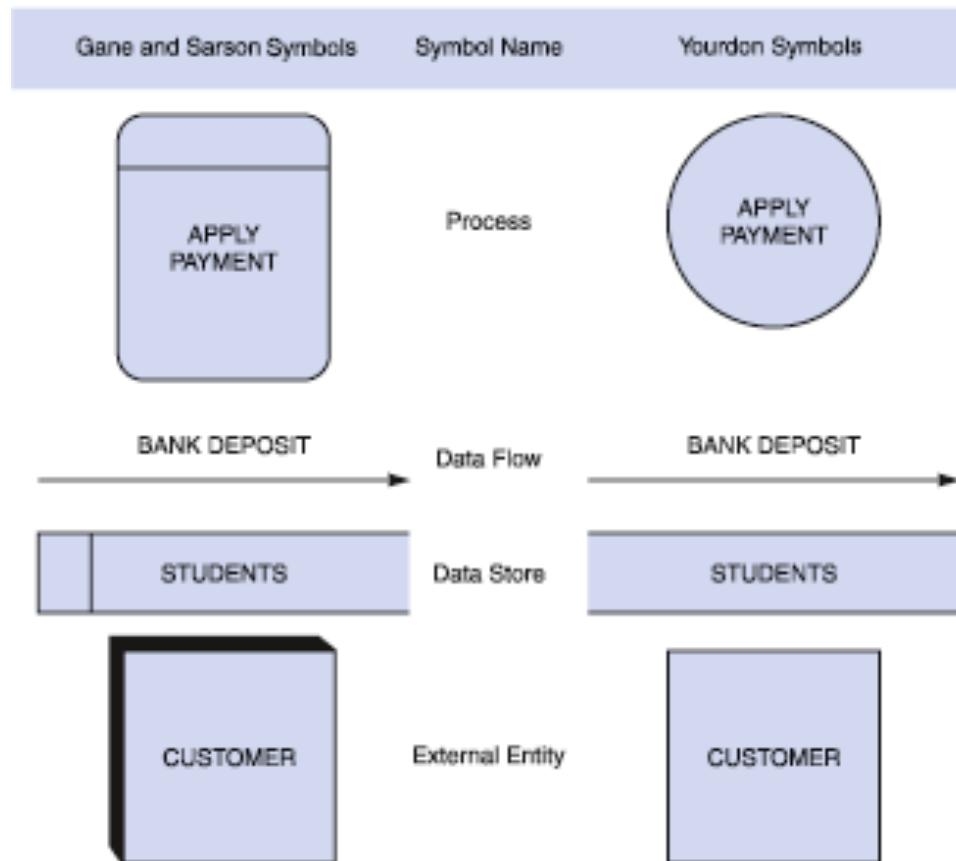
Data Flow Diagrams

- A data flow diagram (DFD) shows how data moves through an information system but does not show program logic or processing steps
- A set of DFDs provides a logical model that shows what the system does, not how it does it



Data Flow Diagrams

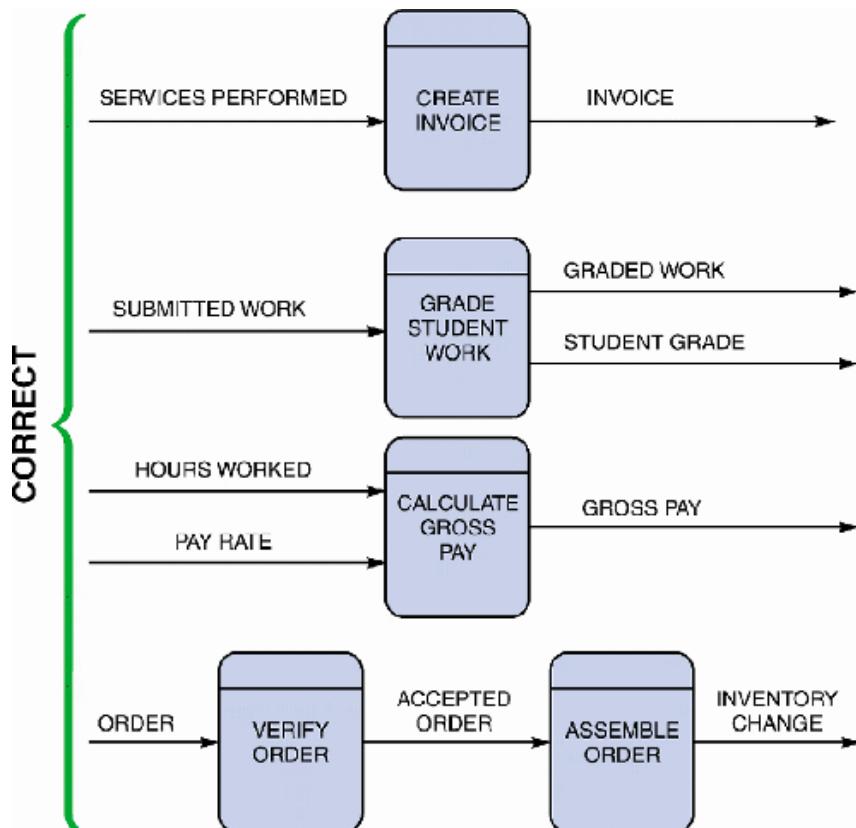
- DFD Symbols



Data Flow Diagrams

- DFD Symbols
 - Process symbol
 - Receives input data and produces output that has a different content, form, or both
 - Contain the business logic, also called business rules
 - Referred to as a black box

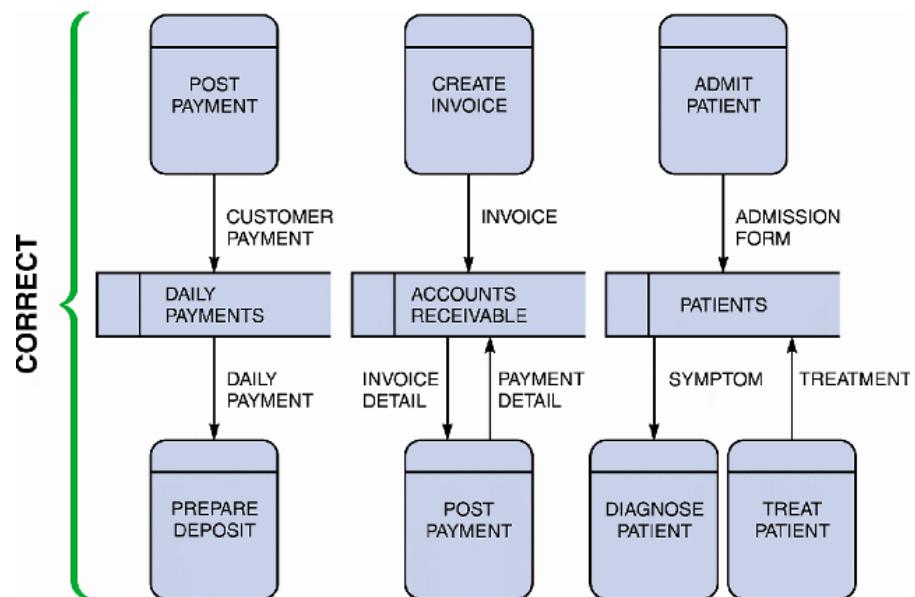
Data Flow Diagrams



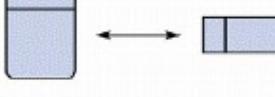
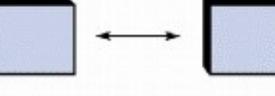
- DFD Symbols
 - Data flow symbol
 - Represents one or more data items
 - The symbol for a data flow is a line with a single or double arrowhead
 - Spontaneous generation
 - Black hole
 - Gray hole

Data Flow Diagrams

- DFD Symbols
 - Data store symbol
 - Represent data that the system stores
 - The physical characteristics of a data store are unimportant because you are concerned only with a logical model



Data Flow Diagrams

Correct and Incorrect Examples of Data Flows			
	Process to Process		
	Process to External Entity		
	Process to Data Store		
	External Entity to External Entity		
	External Entity to Data Store		
	Data Store to Data Store		

- DFD Symbols
 - Entity Symbol

- Name of the entity appears inside the symbol
- Terminators
- Source
- Sink

Creating a Set of DFDs

- Create a graphical model of the information system based on your fact-finding results
- First, you will review a set of guidelines for drawing DFDs. Then you will learn how to apply these guidelines and create a set of DFDs using a three-step process

Creating a Set of DFDs

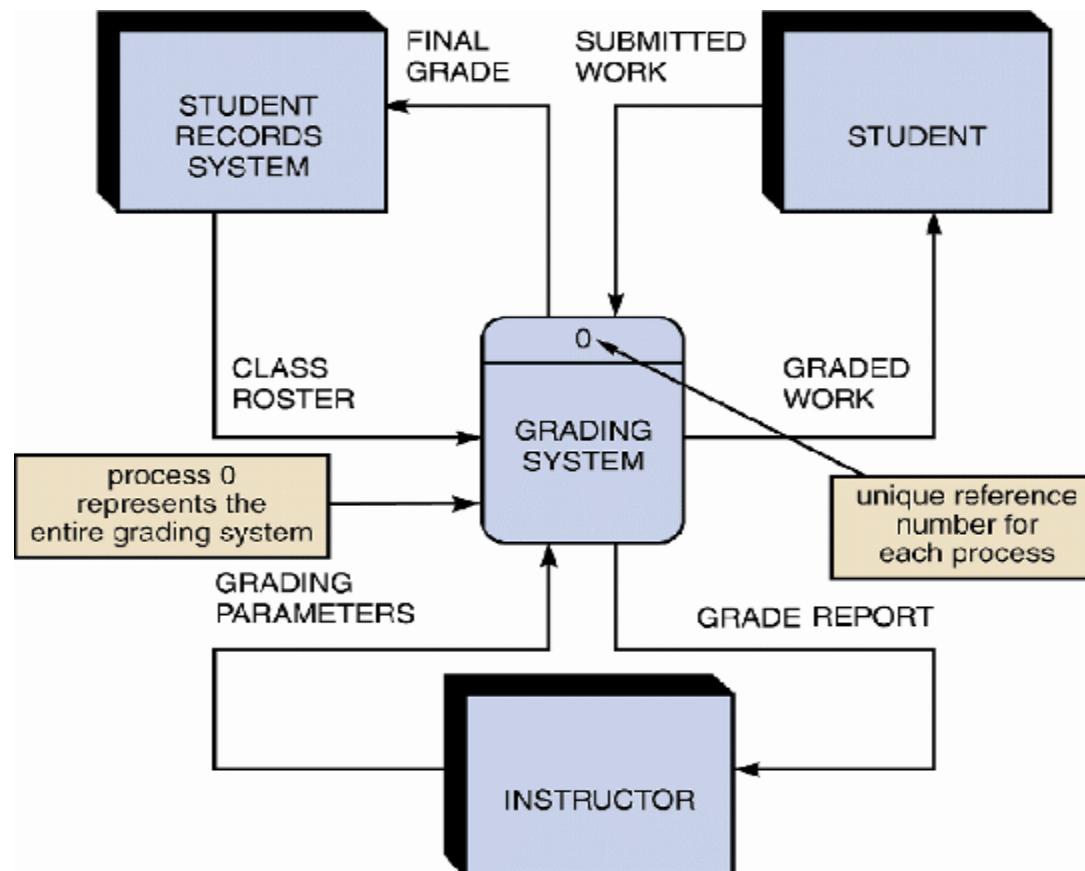
- Guidelines for Drawing DFDs
 - Draw the context diagram so that it fits on one page
 - Use the name of the information system as the process name in the context diagram
 - Use unique names within each set of symbols

Creating a Set of DFDs

- Guidelines for Drawing DFDs
 - Do not cross lines
 - Provide a unique name and reference number for each process
 - Obtain as much user input and feedback as possible

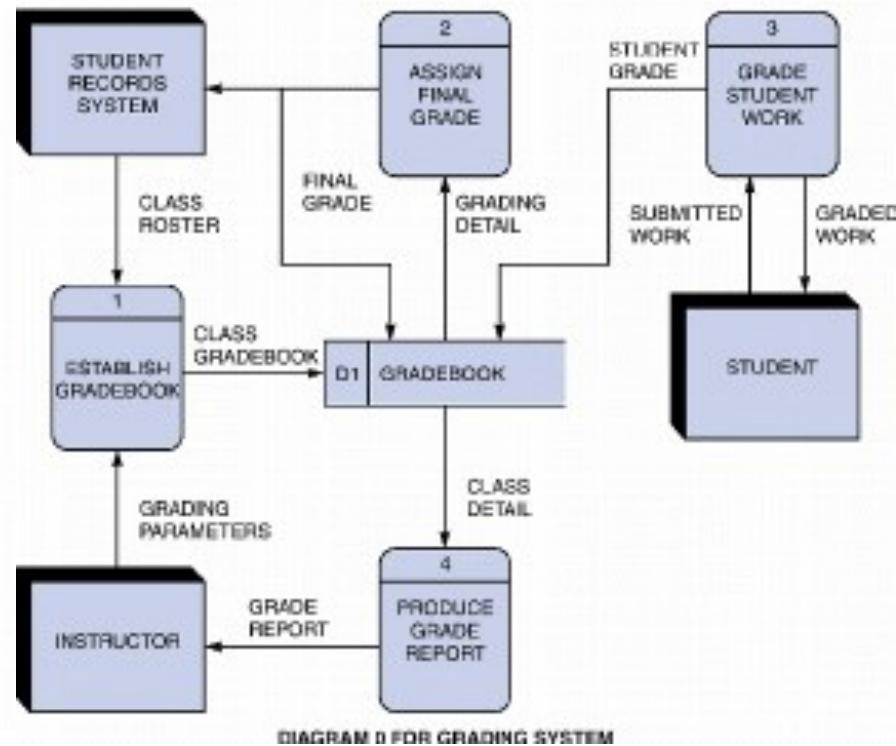
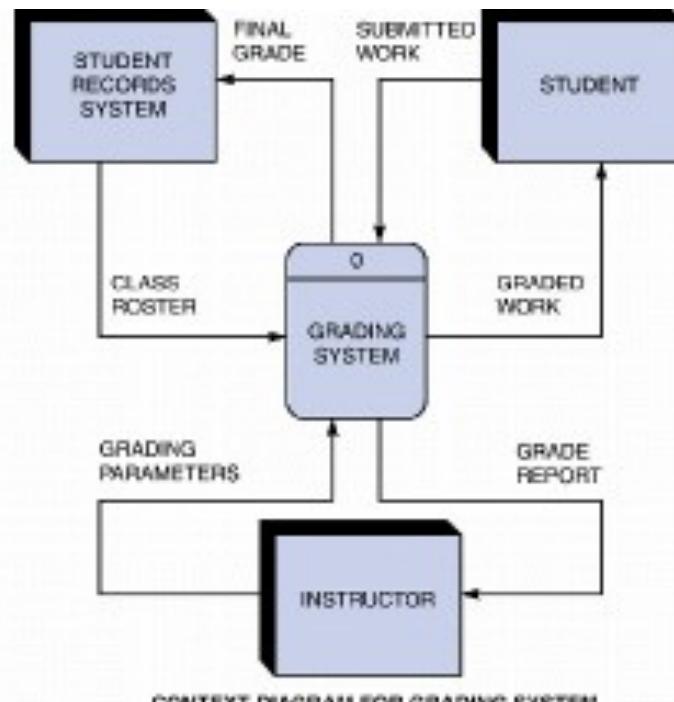
Creating a Set of DFDs

- Step 1: Draw a Context Diagram



Creating a Set of DFDs

- Step 2: Draw a Diagram 0 DFD

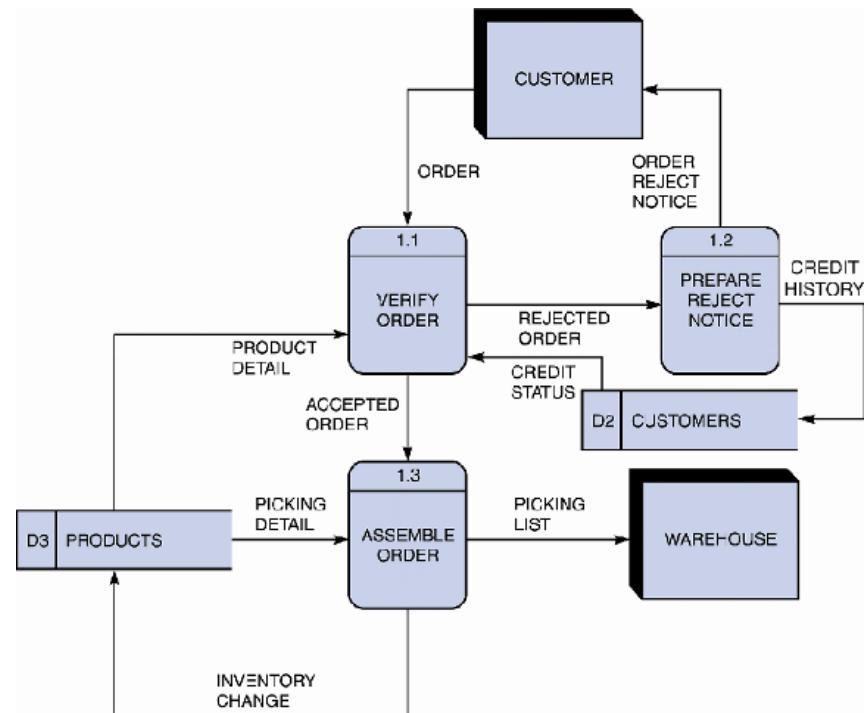


Creating a Set of DFDs

- Step 2: Draw a Diagram 0 DFD
 - If same data flows in both directions, you can use a double-headed arrow
 - Diagram 0 is an exploded view of process 0
 - Parent diagram
 - Child diagram
 - Functional primitive

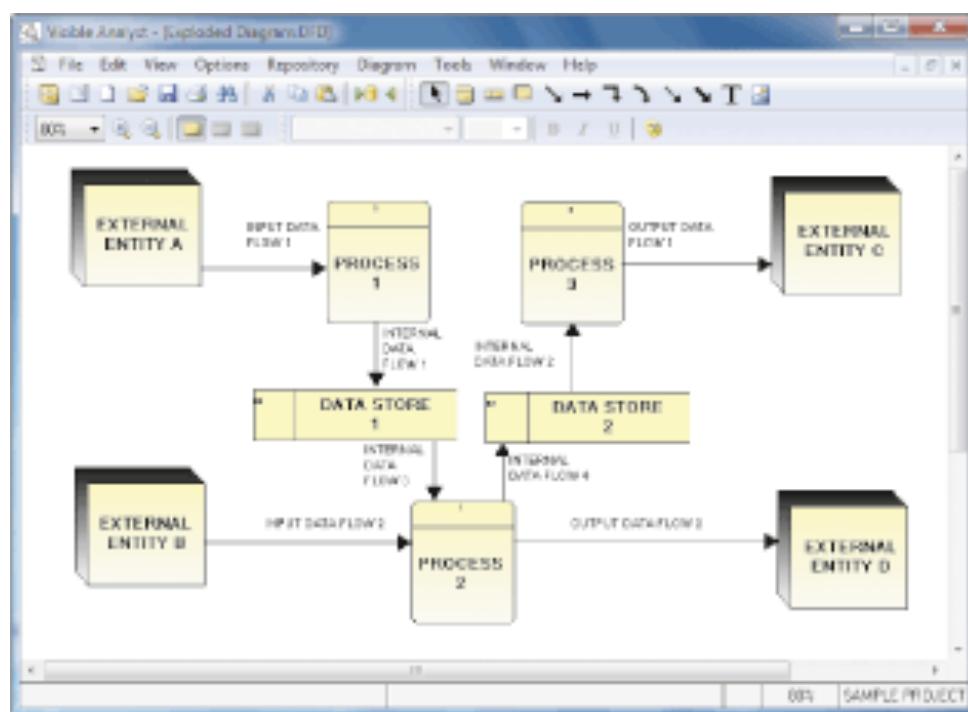
Creating a Set of DFDs

- Step 3: Draw the Lower-Level Diagrams
 - Must use leveling and balancing techniques
 - Leveling examples
 - Uses a series of increasingly detailed DFDs to describe an information system
 - Exploding, partitioning, or decomposing



Creating a Set of DFDs

- Step 3: Draw the Lower-Level Diagrams
 - Balancing Examples
 - Ensures that the input and output data flows of the parent DFD are maintained on the child DFD



Data Dictionary Overview

- A **data dictionary** is essentially a **repository** for all relevant data within a system.
- It's a comprehensive **catalog** used to **document** and **organize** system data.
- Analysts refer to it to **keep track of data specifications**, ensuring clarity and consistency across the system

- **Example:**

In a library management system, the data dictionary would include entries for **data elements** like "Book ID," "Author Name," and "Due Date."

Data Dictionary - Data Elements

- **Data elements** (also known as **data items** or fields) are the **smallest units of data** with specific meanings.
- **Records** (or **data structures**) combine multiple data elements, creating **meaningful units** stored in **data flows** or **data stores**

- **Example:**

For a university's course management system, a **data element** might be "Student ID," while a **record** could be a student profile combining "Student ID," "Name," and "Enrollment Date."

Data Dictionary - Documenting Data Elements

- Every **data element** needs thorough documentation to provide clear and comprehensive information. Typical attributes include:
- Name and label
- Alias (if applicable)
- Type and length (e.g., integer, string)
- Default value, acceptable values, and domain rules
- Source, security level, and responsible user(s)
- Description and comments

- **Example:**
A data element in a hospital system, like "**Patient ID**," might have an alias "**Patient Number**," a **length limit**, domain rules restricting it to unique identifiers, and **security settings**.

Data Dictionary - Documenting Data Flows

- **Data flows** describe the movement of data within the system.
- Key attributes include:
 - **Name/Label**
 - **Description**, **origin**, and **destination**
 - **Record volume** and **frequency**

- **Example:**

In an e-commerce system, a "Purchase Order" data flow might move from the "Cart" process to the "Order Processing" system

Data Dictionary - Documenting Data Stores

- **Data stores** hold data for later retrieval and are documented with attributes such as:
 - **Name/Label** and **description**
 - **Attributes, volume, and frequency**

- **Example:**

In a banking system, **a data store** like "**Account Balances**" would contain records of **customer balances**, updated frequently

Data Dictionary - Documenting Processes

- **Processes** represent **actions** within the system and are documented by:
 - **Name/Label, description, and process number**
 - Details about the process **functionality**
- **Example:** An ATM **withdrawal** process includes:
 - "Input: Account Information" and
 - "Output: Withdrawal Confirmation."

Data Dictionary - Documenting Entities

- **Entities** represent key objects within the system, documented by:
 - **Entity name, description, alternate names**
 - **Input and output data flows**
- **Example:**
 - In a payroll system, an entity like "**Employee**" might include **data flows** for "**Payroll Processing**" and "**Employee Benefits.**"

Data Dictionary - Data Dictionary Reports

- Reports generated from a data dictionary can provide valuable insights, such as:
 - A list of all data elements **alphabetically**
 - Reports of data elements **by department or user responsible**
 - Detailed reports of data flows, stores, and records

- **Example:**

A report listing all elements used in the "**Order Processing**" system of an **online store** helps identify **responsibilities** and **improve data management.**

Process Description Tools

- A **process description** documents the details of a **functional primitive**, which represents a specific set of **processing steps** and business logic
- It should be noted that this chapter deals with structured analysis, but the process description tools also can be used in object-oriented development, which is described in Chapter 6

Process Description Tools

- Modular Design
 - Based on combinations of three logical structures, sometimes called control structures, which serve as building blocks for the process
 - Sequence
 - Selection
 - Iteration - looping

Process Description Tools

- **Structured English**
 - Must conform to the following rules
 - Use only the **three building blocks** of sequence, selection, and iteration
 - Use indentation for readability
 - Use a limited vocabulary, including standard terms used in the data dictionary and specific words that describe the processing rules

Process Description Tools

- Structured English
 - Might look familiar to programming students because it resembles pseudocode
 - The primary purpose of structured English is to describe the underlying business logic

```
STRUCTURED ENGLISH VERSION OF THE SALES PROMOTION POLICY

    IF customer is a preferred customer, and
        IF customer orders more than $1,000 then
            Apply a 5% discount, and
            IF customer uses our charge card, then
                Apply an additional 5% discount
        ELSE
            Award a $25 bonus coupon
    ELSE
        Award a $5 bonus coupon
```

Process Description Tools

- Decision Tables
 - Shows a logical structure, with all possible combinations of conditions and resulting actions
 - It is important to consider every possible outcome to ensure that you have overlooked nothing

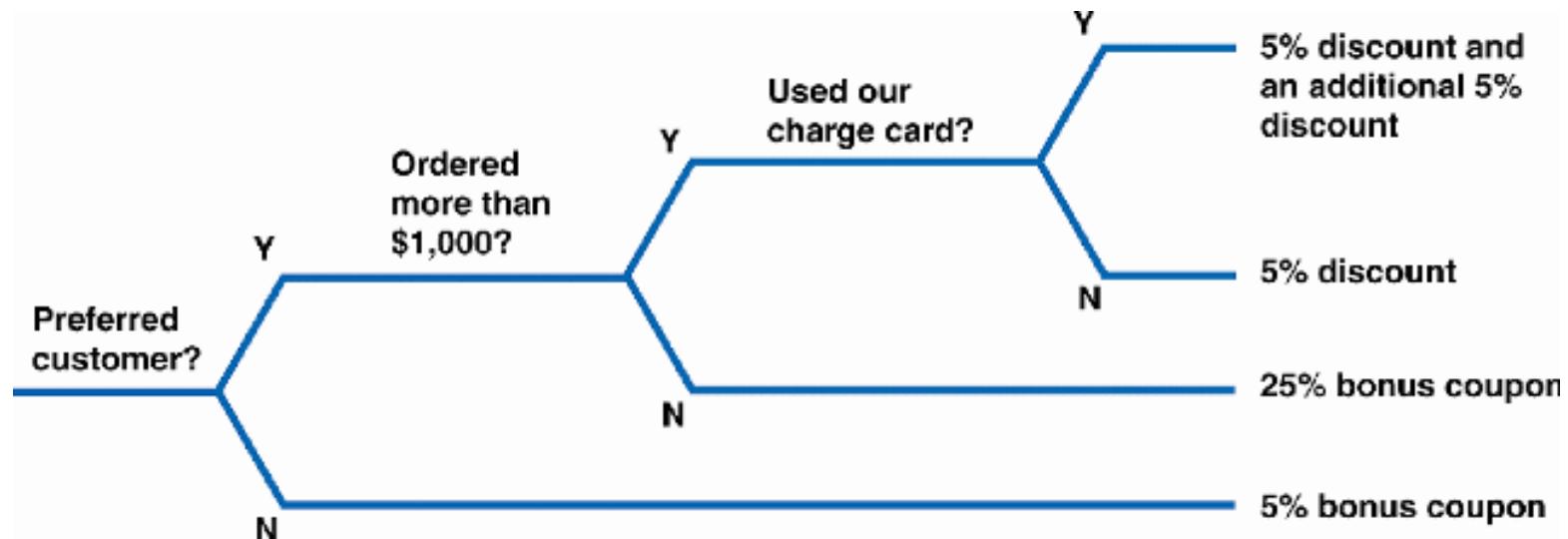
	1	2	3	4
Credit status is OK	Y	Y	N	N
Product is in stock	Y	N	Y	N
Accept order	X			
Reject order		X	X	X

Process Description Tools

- Decision Tables
 - The number of rules doubles each time you add a condition
 - Can have more than two possible outcomes
 - Often are the best way to describe a complex set of conditions

Process Description Tools

- Decision Trees



Chapter Summary

- During data and process modeling, a systems analyst develops graphical models to show how the system transforms data into useful information
- The end product of data and process modeling is a logical model that will support business operations and meet user needs
- Data and process modeling involves three main tools: data flow diagrams, a data dictionary, and process descriptions

Chapter Summary

- Data flow diagrams (DFDs) graphically show the movement and transformation of data in the information system
- DFDs use four symbols
- A set of DFDs is like a pyramid with the context diagram at the top

Chapter Summary

- The data dictionary is the central documentation tool for structured analysis
- Each functional primitive process is documented using structured English, decision tables, and decision trees
- Structured analysis tools can be used to develop a logical model during one systems analysis phase, and a physical model during the systems design phase

Chapter Summary

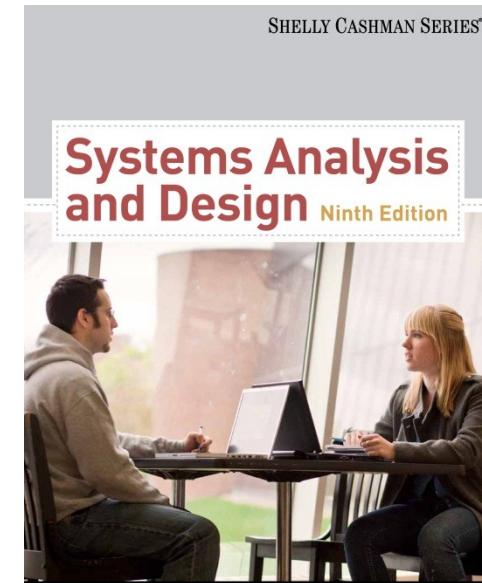
- Chapter 5 complete

Systems Analysis and Design 9th Edition

Chapter 6

Object Modeling

SHELLY CASHMAN SERIES



SHELLY | ROSENBLATT

Chapter Objectives

- Explain how object-oriented analysis can be used to describe an information system
- Define object modeling terms and concepts, including objects, attributes, methods, messages, classes, and instances
- Explain relationships among objects and the concept of inheritance
- Draw an object relationship diagram

Chapter Objectives

- Describe Unified Modeling Language (UML) tools and techniques, including use cases, use case diagrams, class diagrams, sequence diagrams, state transition diagrams, and activity diagrams
- Explain the advantages of using CASE tools in developing the object model
- Explain how to organize an object model

Introduction

- You learn about object-oriented analysis, which is another way to view and model system requirements
- You use object-oriented methods to document, analyze, and model the information system

Overview of Object-Oriented Analysis

- Object-oriented (O-O) analysis
- Object
- Object-oriented analysis is a popular approach that sees a system from the **viewpoint of the objects themselves** as they function and interact
- Object model

Example

- In a **library management system**, OOA would identify:
 - **objects** such as Book, Member, and Librarian, and
 - **analyze their properties** (e.g., title, author, memberID) and
 - **interactions** (e.g., borrowing, returning)

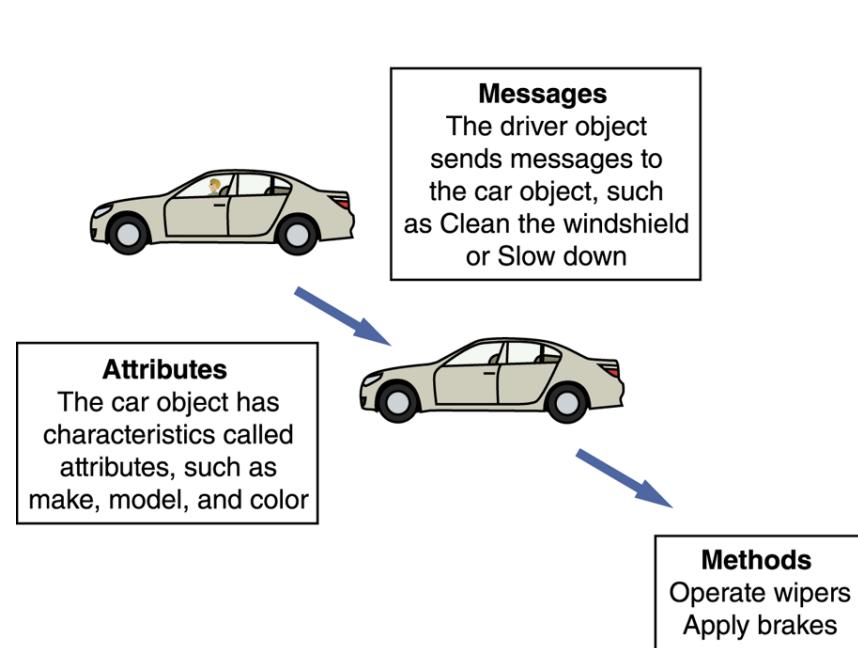
Overview of Object-Oriented Analysis (OOA)

- Object-Oriented Analysis (OOA) is a method of analyzing a problem **by looking at the system as a collection of interacting objects**.
- Each object represents a real-world entity or concept and is **characterized by its attributes and behaviors**.
- The purpose of OOA is to better understand a system by **breaking it down into manageable pieces that mimic real-world entities and their interactions**

Overview of Object-Oriented Analysis

- Object-Oriented Terms and Concepts
 - Unified Modeling Language (UML)
 - Attributes
 - Methods
 - Message
 - Class
 - Instance

Examples of Interaction Between Objects



Attributes in OOA

- Attributes are **properties or characteristics** that describe an object and define its state.
- If **objects** are analogous to **nouns** (e.g., "Car," "Employee," "Book"), then **attributes** are like **adjectives** that describe these nouns (e.g., color, age, title). Attributes give more information about an object's current state and can vary depending on the object.
- **Examples of Objects and Attributes: Car**
Attributes: color (e.g., red), make (e.g., Toyota), model (e.g., Corolla), year (e.g., 2022), engine type (e.g., hybrid).

Messages

- **Messages** are the means by which objects communicate and interact with each other.
- An object sends a message to another object to invoke a method or perform an action.
- This helps achieve interaction and coordination among various components of the system.
- **Example:** In a Car and Driver objects, the Driver object may send a startEngine() message to the Car object. The Car object then processes this message and executes the startEngine method

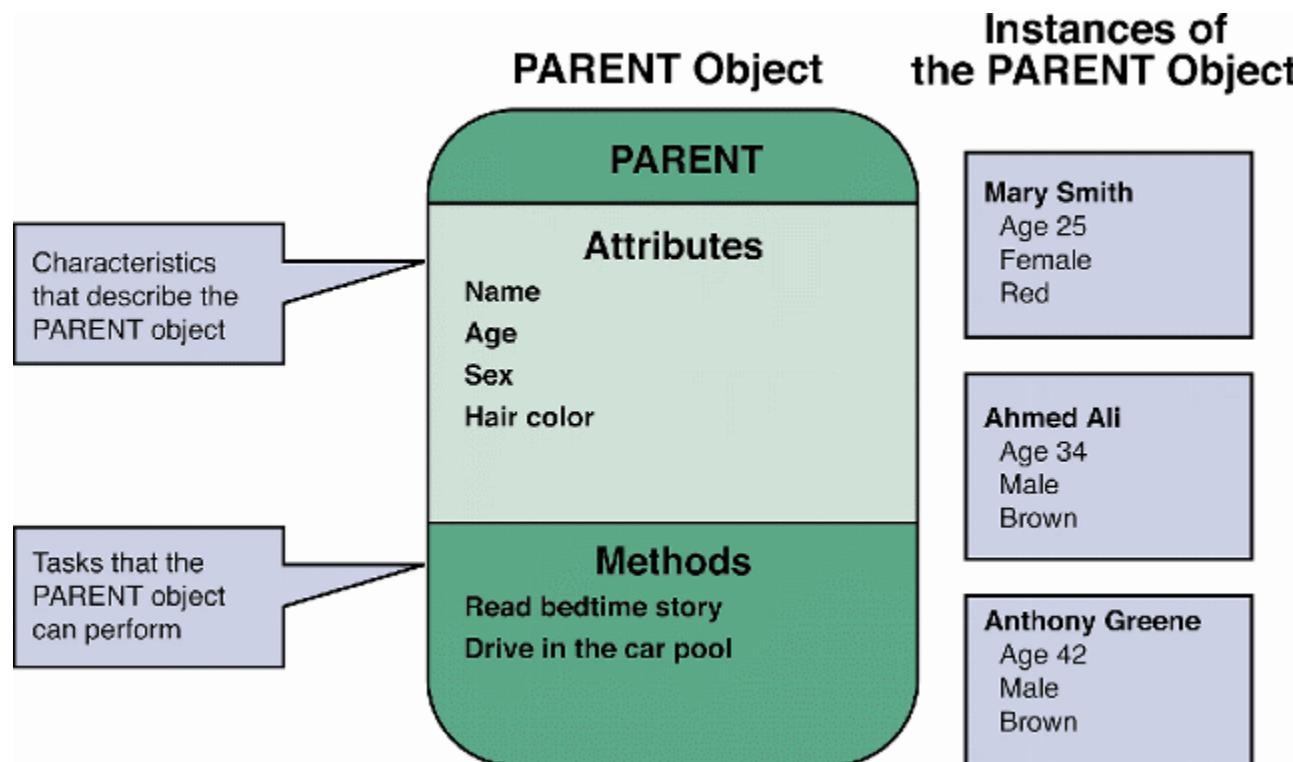
Overview of Object-Oriented Analysis

- Messages
 - A major advantage of O-O designs is that systems analysts can save time and avoid errors by using modular objects, and programmers can translate the designs into code, working with reusable program modules that have been tested and verified



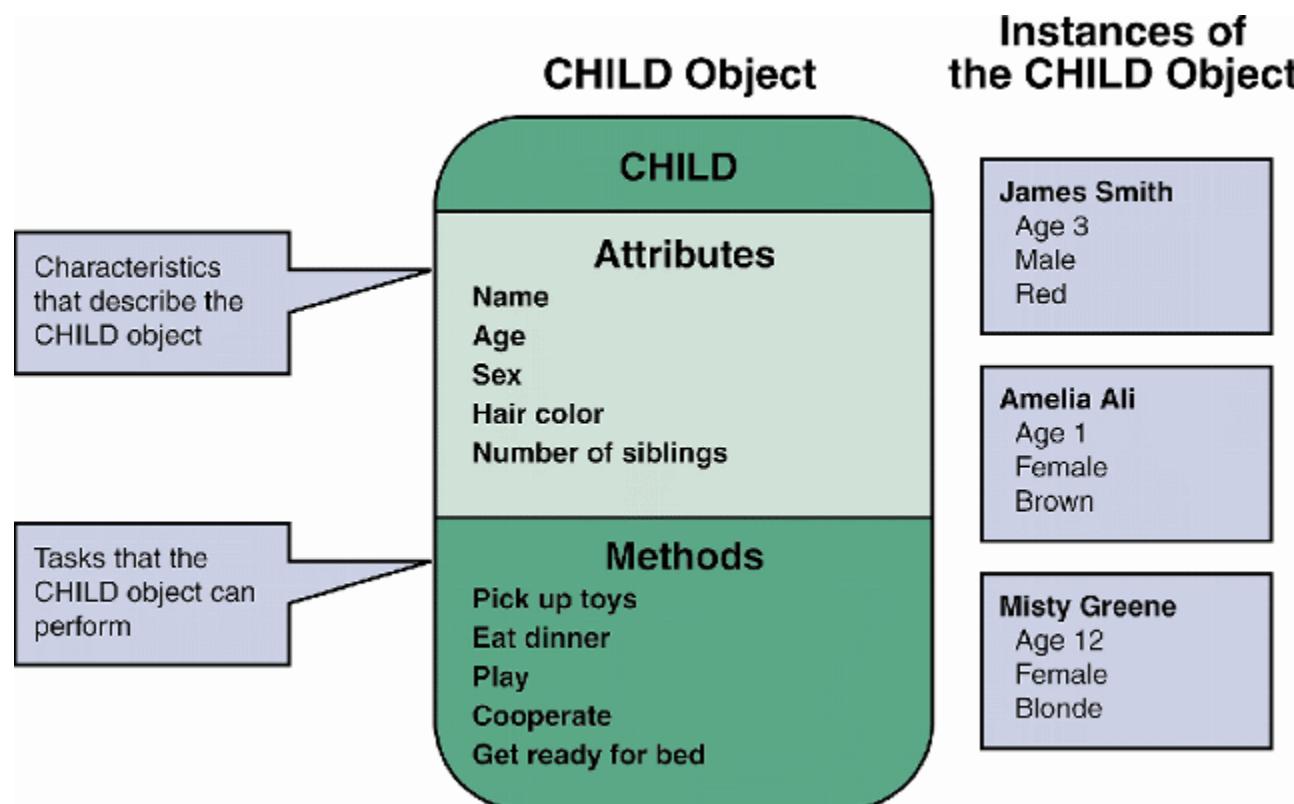
Overview of Object-Oriented Analysis

- Objects



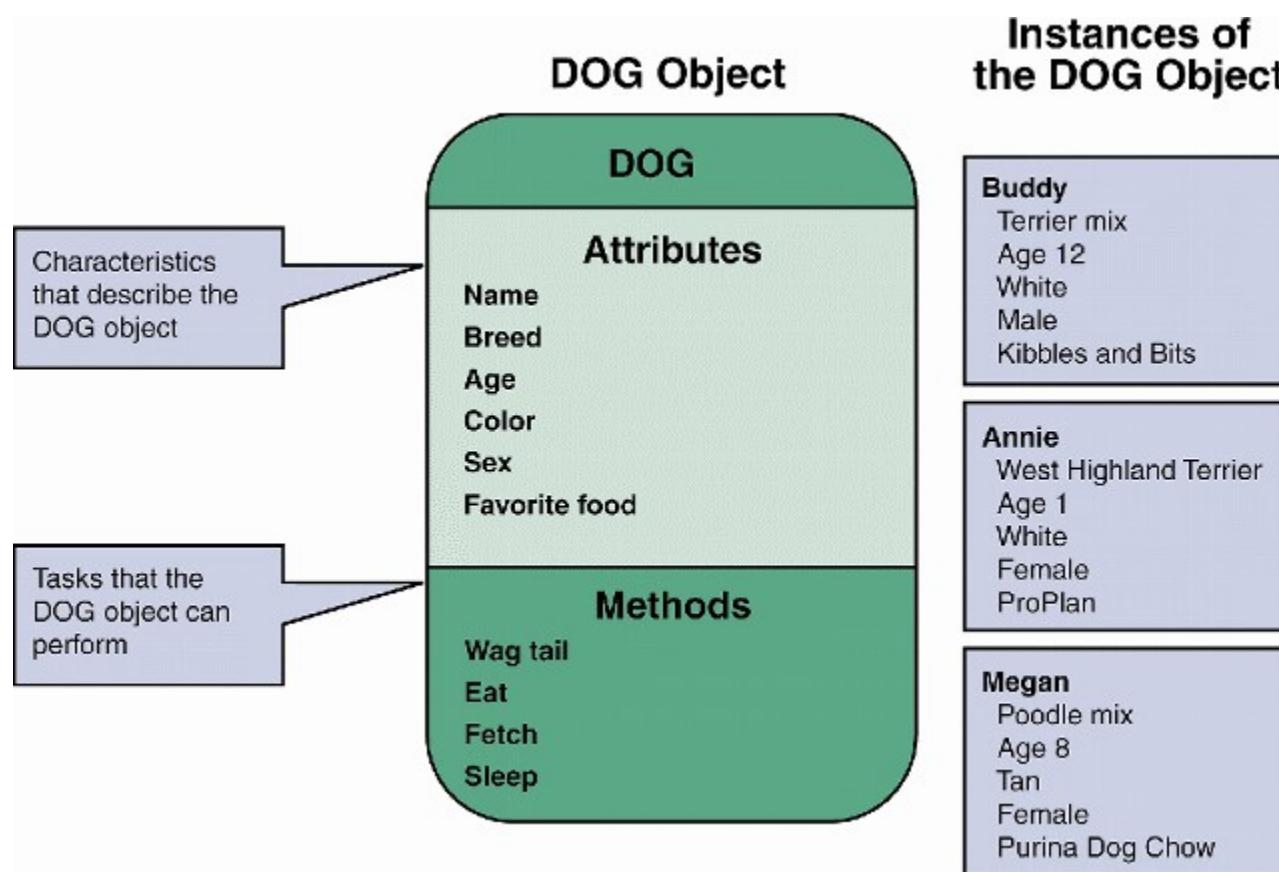
Overview of Object-Oriented Analysis

- Objects



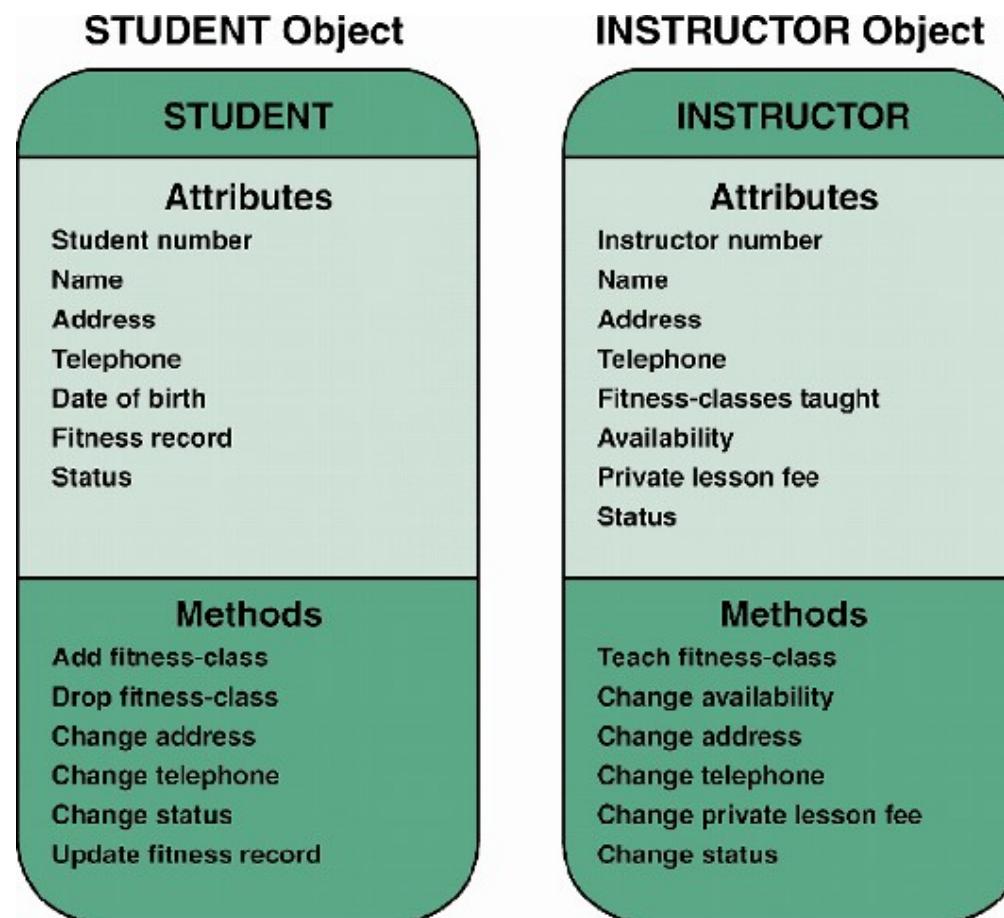
Overview of Object-Oriented Analysis

- Objects



Overview of Object-Oriented Analysis

- Objects

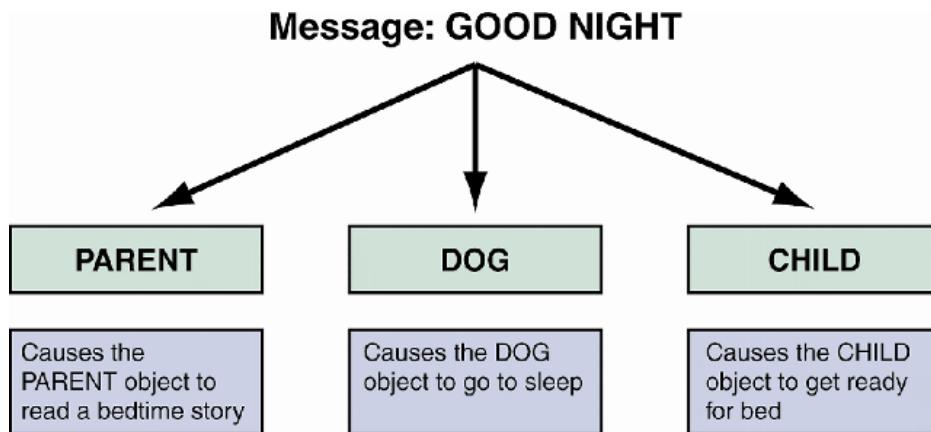


Overview of Object-Oriented Analysis

- Methods
 - A method defines specific tasks that an object can perform
 - Just as objects are similar to nouns and attributes are similar to adjectives, **methods** resemble **verbs** that describe what and how an object does something

Method: MORE FRIES	Steps: <ol style="list-style-type: none">1. Heat oil2. Fill fry basket with frozen potato strips3. Lower basket into hot oil4. Check for readiness5. When ready raise basket and let drain6. Pour fries into warming tray7. Add salt
------------------------------	---

Overview of Object-Oriented Analysis



- Messages
 - Polymorphism
 - Black box
 - Encapsulation

Polymorphism

Polymorphism is the ability of objects of **different types** to be treated as objects of a **common super type**. It allows a single method or function to operate in different ways based on the object it is applied to.

This leads to more flexible and **reusable code** by allowing methods to be overridden or used across different classes

Example: Consider a base **class Shape** with a method **draw()**. Different derived classes like **Circle**, **Rectangle**, and **Triangle** override the **draw()** method to implement their specific drawing logic. A piece of code that calls **draw()** on a **Shape** object can work with any subclass without knowing its exact type

Black Box

Black box refers to the concept of **hiding the internal implementation details of an object and only exposing its behavior or interface.**

This means that the user of an object **only needs to know how to interact with it** (i.e., which methods are available) without needing to understand how those methods are implemented.

Example: A Car class may have methods like startEngine() and accelerate(). The user of the Car class **doesn't need to know how the engine starts internally**; they only need to call startEngine()

Encapsulation

Encapsulation is the practice of bundling the data (attributes) and methods (functions) that operate on that data **within a single unit, typically a class.**

It restricts direct access to some of an object's components, which means the internal representation of the object is hidden from the outside.

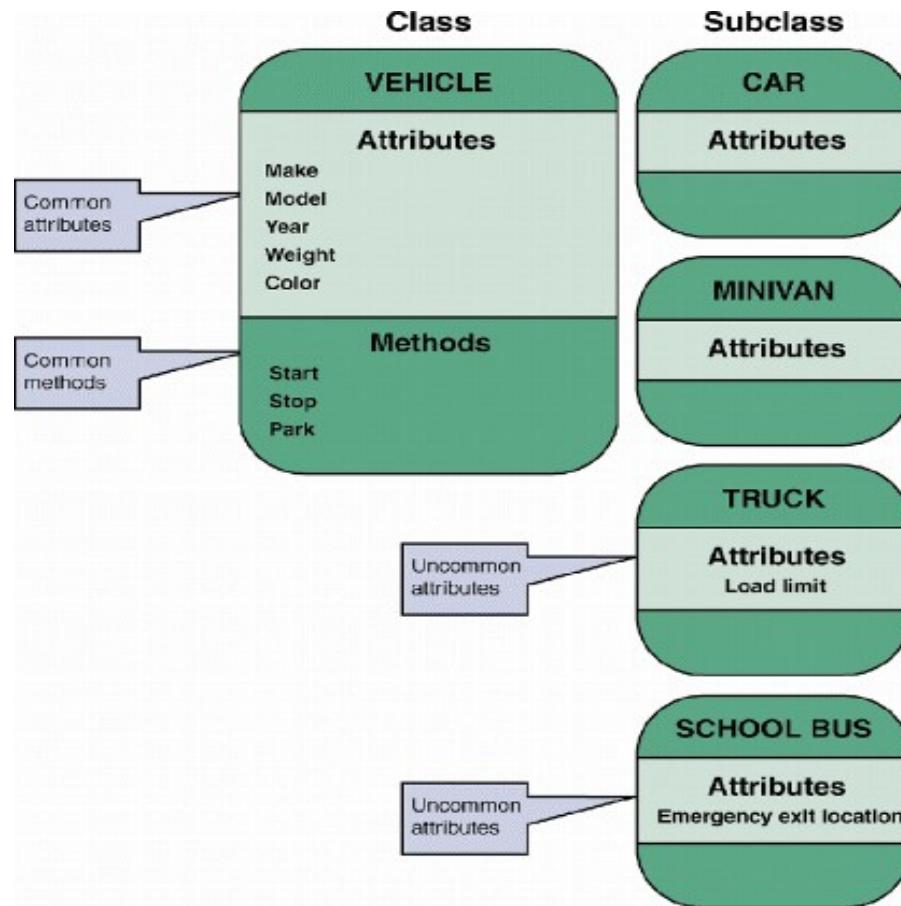
Example: A BankAccount class might have private data members such as balance and methods deposit() and withdraw(). **External code cannot directly access or modify balance;** it must use the provided methods, which can include checks and safeguards

Overview of Object-Oriented Analysis

- Classes
 - An object belongs to a group or category called a class
 - All objects within a class share common attributes and methods
 - Subclasses
 - Superclass

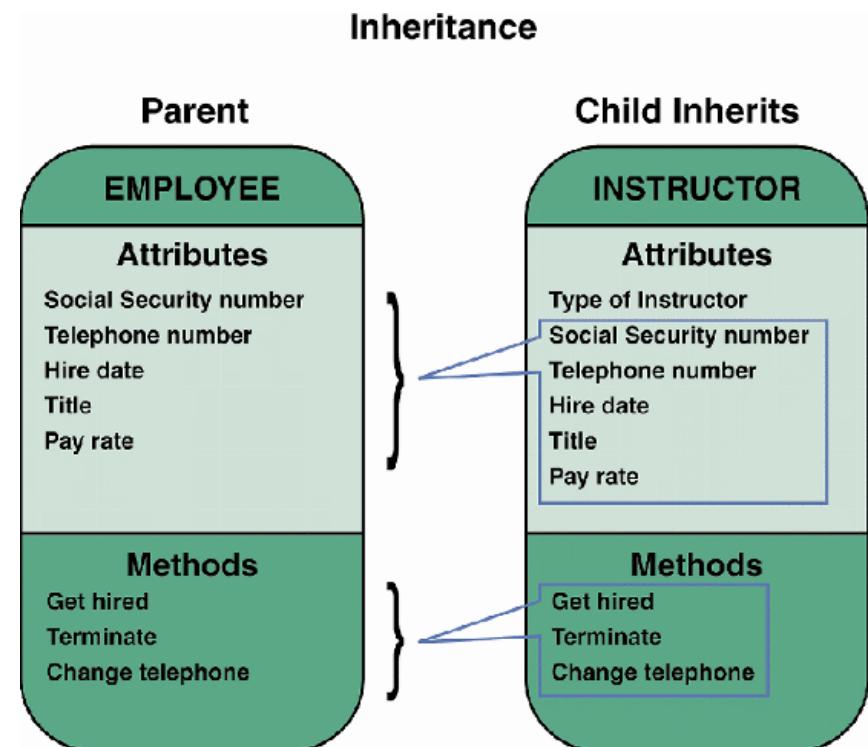
Overview of Object-Oriented Analysis

- Classes



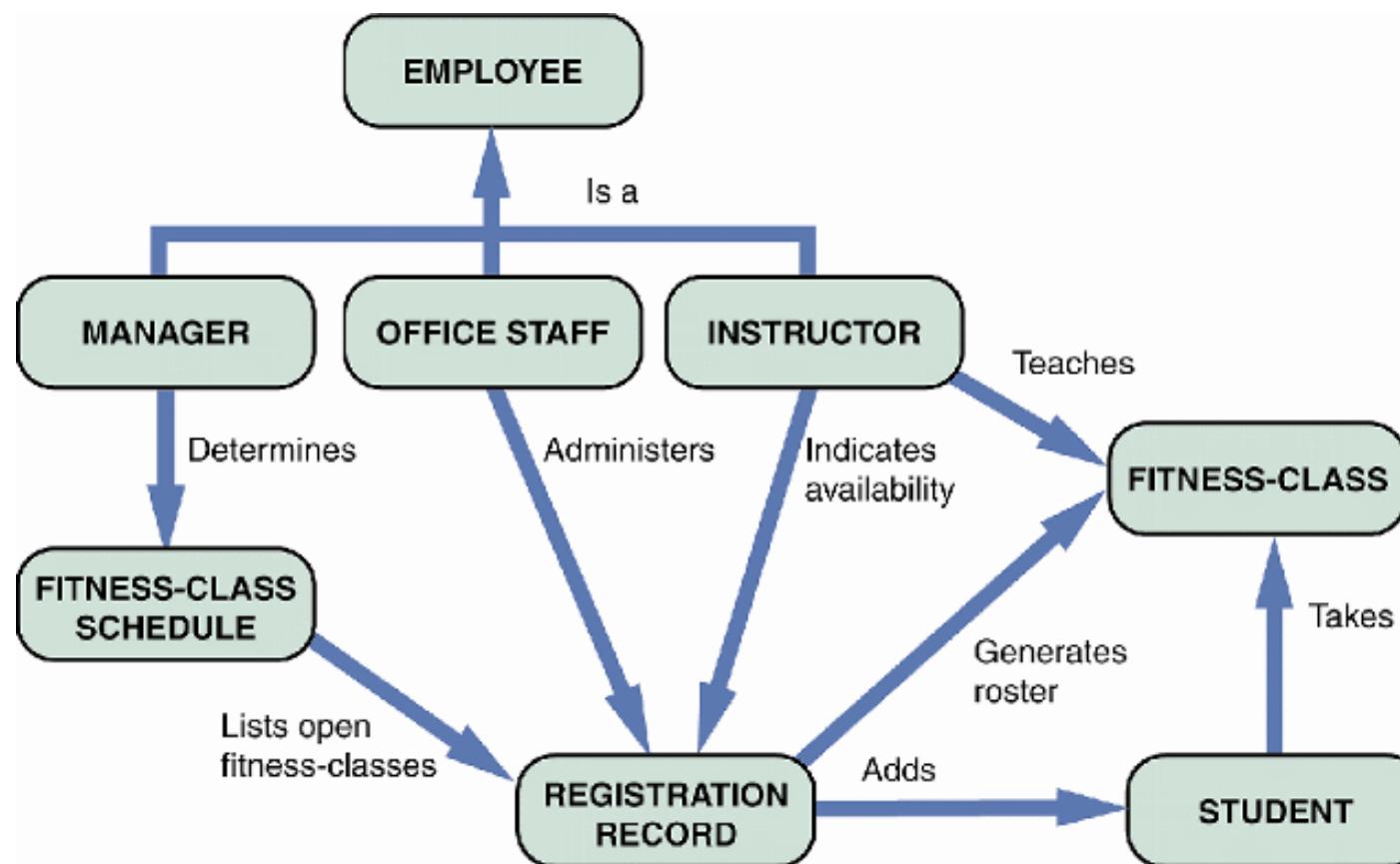
Relationships Among Objects and Classes

- Inheritance
- Child
- Parent



Relationships Among Objects and Classes

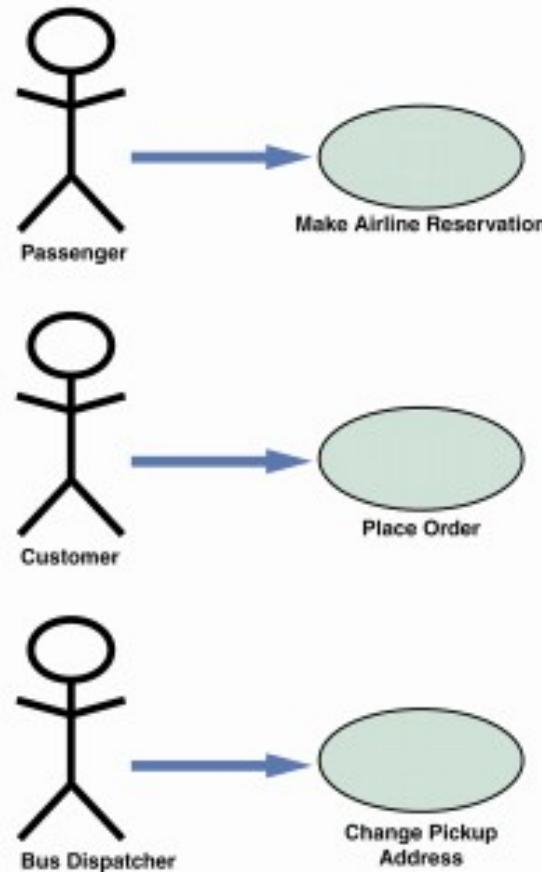
- Object Relationship Diagram



Object Modeling with the Unified Modeling Language

- The UML uses a set of symbols to represent graphically the various components and relationships within a system
- It mainly is used to support object-oriented systems analysis and to develop object models

Object Modeling with the Unified Modeling Language



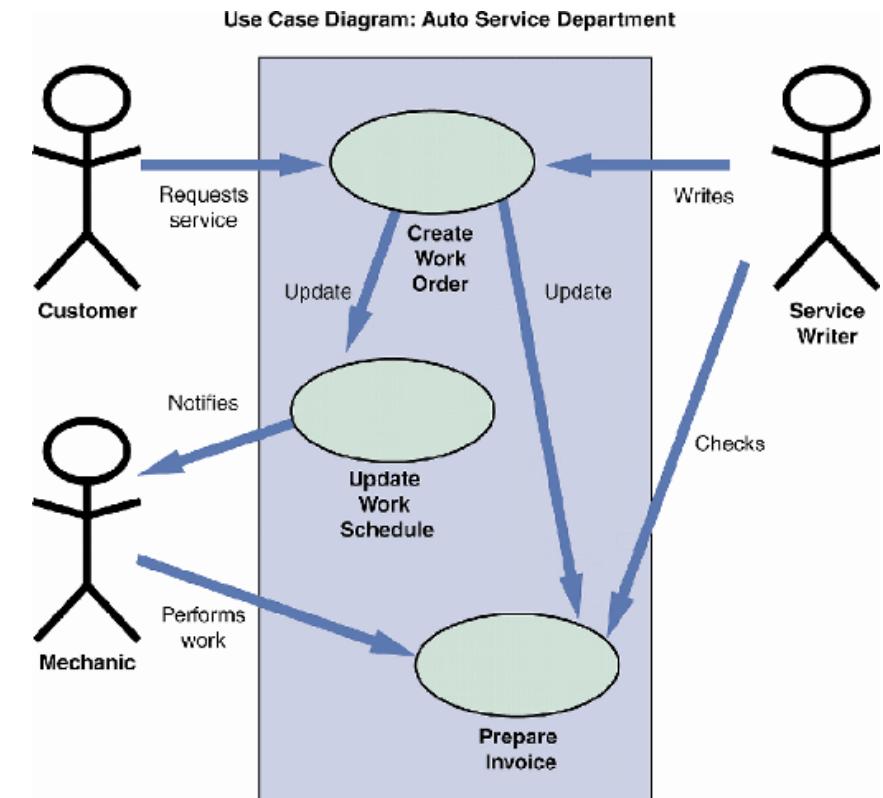
- Use Case Modeling
 - Actor
 - Symbol for a use case is an oval with a label that describes the action or event
 - Use cases also can interact with other use cases

Object Modeling with the Unified Modeling Language

- Use Case Modeling
 - When the outcome of one use case is incorporated by another use case, we say that the second case uses the first case
 - Use case description
 - When you identify use cases, try to group all the related transactions into a single use case

Object Modeling with the Unified Modeling Language

- Use Case Diagrams
 - Use case diagram
 - System boundary
 - After you identify the system boundary, you place the use cases on the diagram, add the actors, and show the relationships

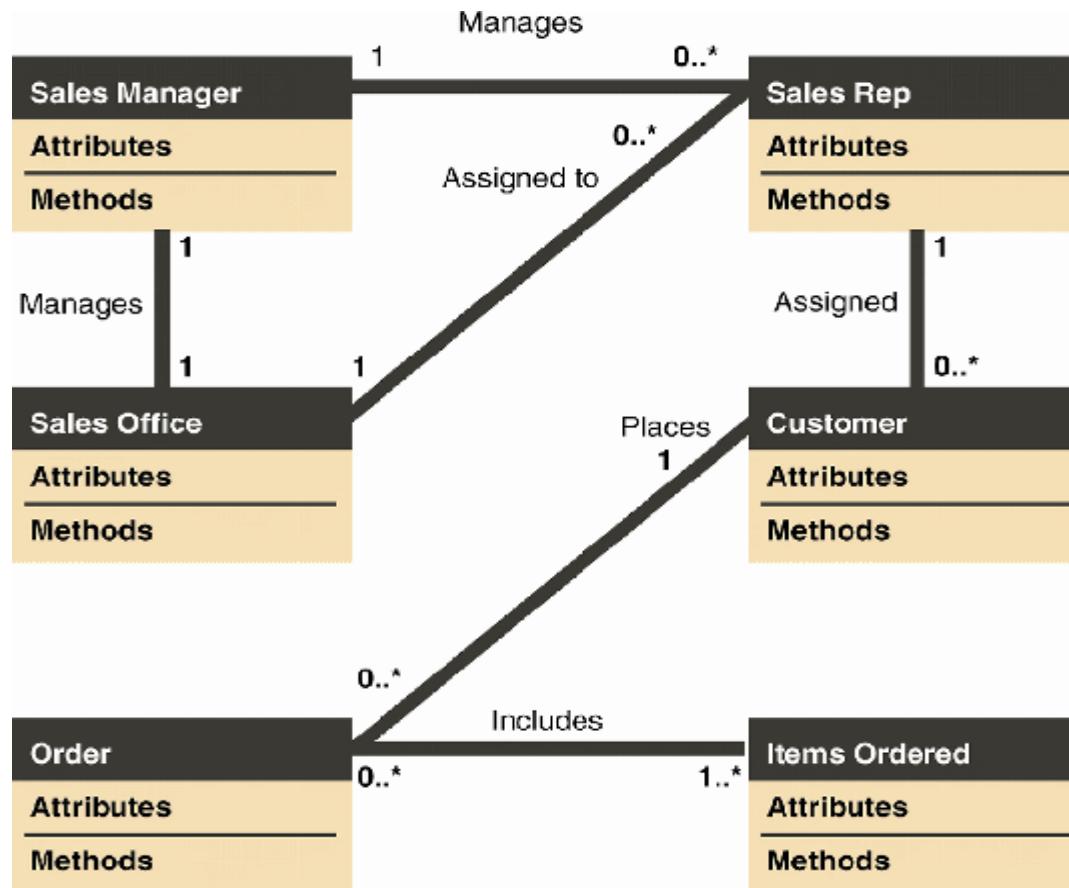


Object Modeling with the Unified Modeling Language

- Class Diagrams
 - Class Diagram
 - Evolves into a physical model and finally becomes a functioning information system
 - Each class appears as a rectangle, with the class name at the top, followed by the class's attributes and methods
 - Cardinality

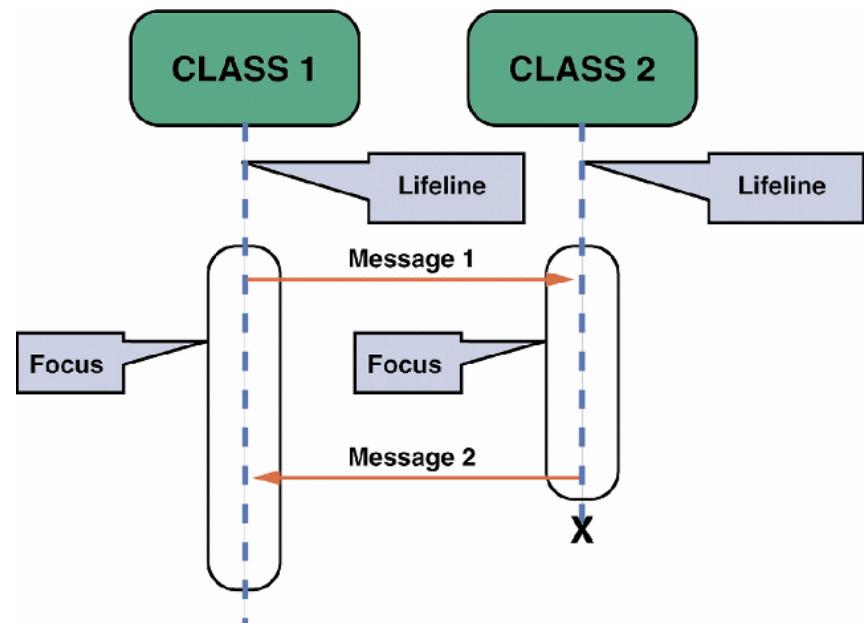
Object Modeling with the Unified Modeling Language

- Class Diagrams



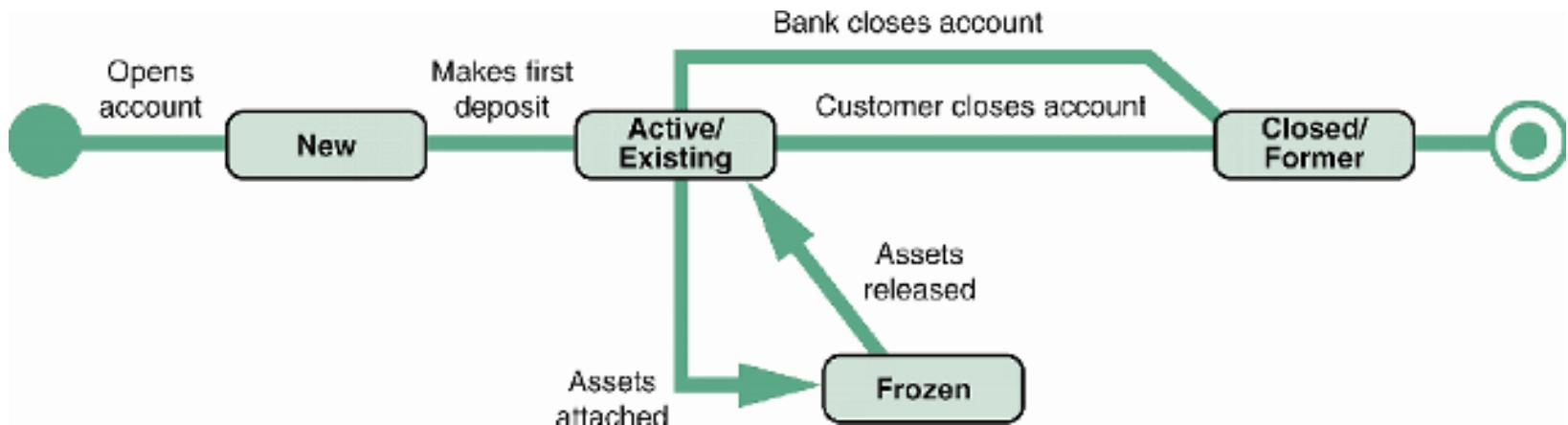
Object Modeling with the Unified Modeling Language

- Sequence Diagrams
 - Sequence diagram
 - Include symbols that represent
 - Classes
 - Lifelines
 - Messages
 - Focuses



Object Modeling with the Unified Modeling Language

- State Transition Diagrams

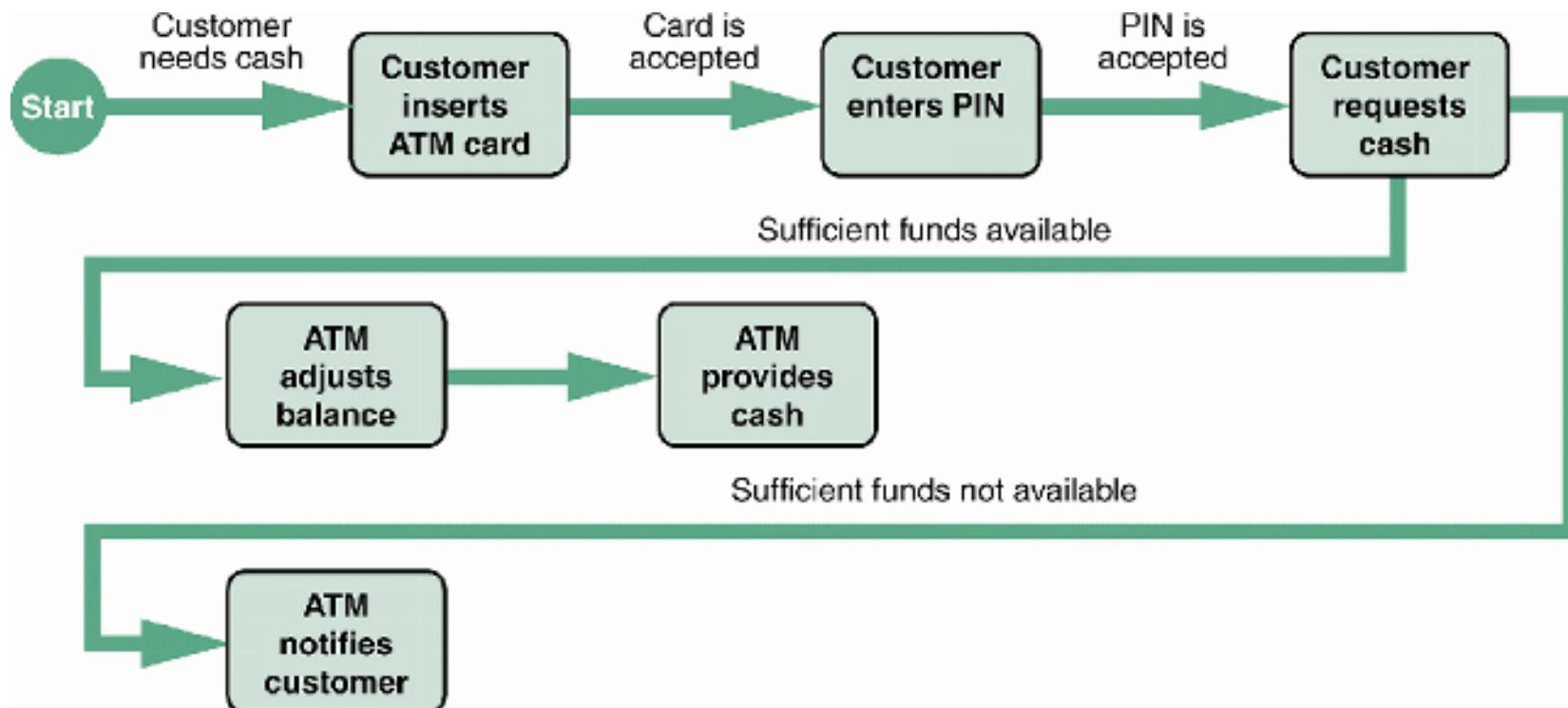


Object Modeling with the Unified Modeling Language

- State Transition Diagrams
 - The small circle to the left is the initial state, or the point where the object first interacts with the system
 - Reading from left to right, the lines show direction and describe the action or event that causes a transition from one state to another
 - The circle at the right with a hollow border is the final state

Object Modeling with the Unified Modeling Language

- Activity Diagrams



Object Modeling with the Unified Modeling Language

- Activity Diagrams
 - Sequence diagrams, state transition diagrams, and activity diagrams are dynamic modeling tools that can help a systems analyst understand how objects behave and interact with the system

Object Modeling with the Unified Modeling Language

- CASE Tools
 - Object modeling requires many types of diagrams to represent the proposed system
 - Creating the diagrams by hand is time-consuming and tedious, so systems analysts rely on CASE tools to speed up the process and provide an overall framework for documenting the system components

Organizing the Object Model

- You should develop an object relationship diagram that provides an overview of the system
- You should organize your use cases and use case diagrams so they can be linked to the appropriate class, state transition, sequence, and activity diagrams
- It is much easier to repair a diagram now than to change the software later

Chapter Summary

- This chapter introduces object modeling, which is a popular technique that describes a system in terms of objects
- The Unified Modeling Language (UML) is a widely used method of visualizing and documenting an information system
- At the end of the object modeling process, you organize your use cases and use case diagrams and create class, sequence, state transition, and activity diagrams

Chapter Summary

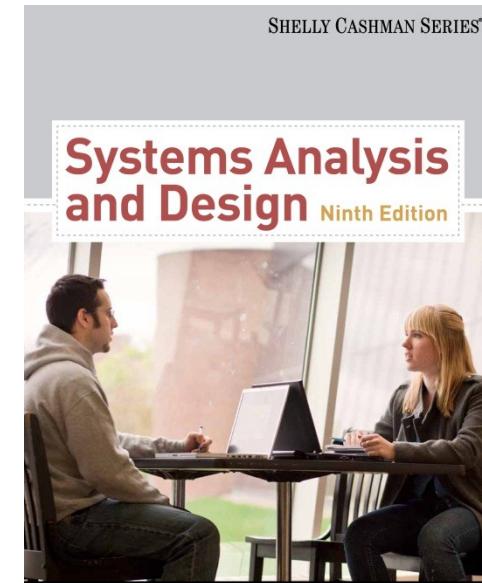
- Chapter 6 complete

Systems Analysis and Design 9th Edition

Chapter 7

Development Strategies

SHELLY CASHMAN SERIES



SHELLY | ROSENBLATT

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

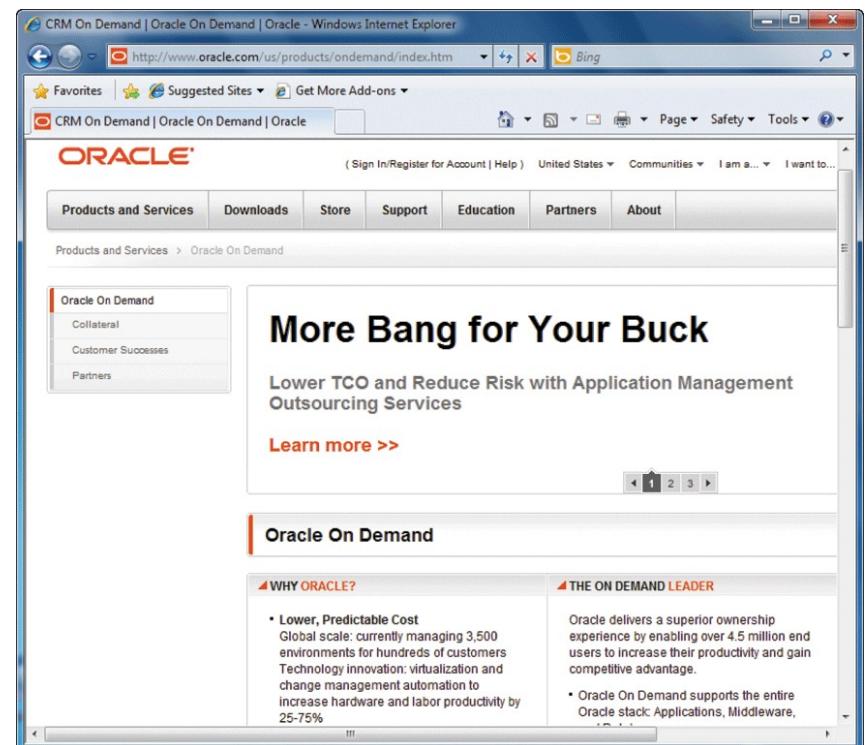
Example of Web-Based Development: A Customer Relationship Management (CRM) tool like Salesforce, which is fully web-based, allows sales and customer support teams to access customer data from anywhere with an Internet connection. The system is inherently designed to be accessed online, with frequent 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

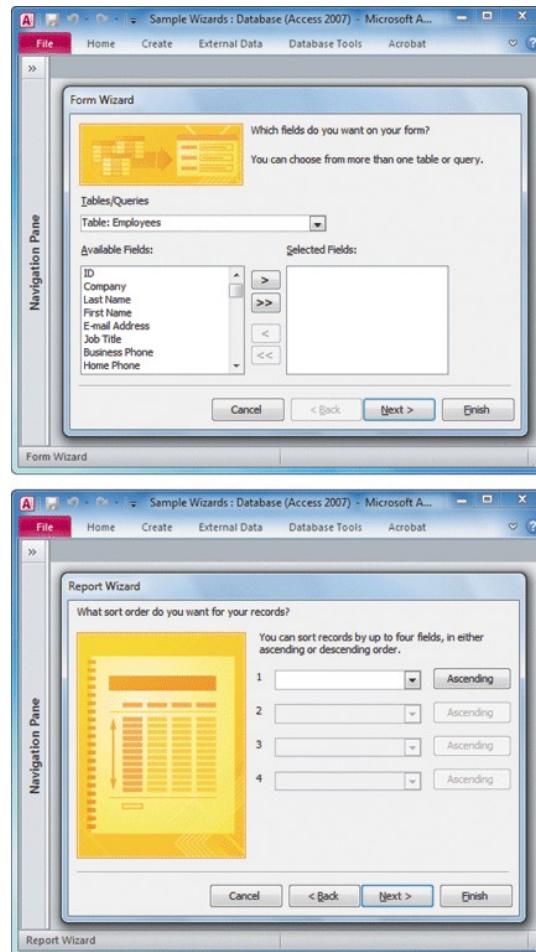
- Purchasing a Software Package
 - Lower costs
 - Requires less time to implement
 - Proven reliability and performance benchmarks
 - Requires less technical development staff
 - Future upgrades provided by the vendor
 - Input from other companies

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

The Value of Linux on HP

TCO comparison

	Sun Solaris	Oracle Linux on HP	Savings with Oracle Linux on HP
Hardware	\$ 180,906	\$ 63,220	\$ 97,686
Software	\$ 27,430	\$ 36,800	\$ -9,400
IT Operations	\$ 54,964	\$ 32,445	\$ 22,519
IT Administration	\$ 3,355	\$ 3,535	\$ -182
Facilities and Overhead	\$ 5,000	\$ 4,000	\$ 1,000
Downtime	\$ 23,908	\$ 23,908	\$ 0
TCO summary	\$ 275,532	\$ 163,309	\$ 111,823
Savings with HP			41 %
TCO per system	\$ 275,532	\$ 81,955	\$ 193,577
TCO per user	\$ 55	\$ 33	\$ 22

TCO Comparison

Category	Sun Solaris	Oracle Linux on HP
Hardware	\$ 180,906	\$ 63,220
Software	\$ 27,430	\$ 36,800
IT Operations	\$ 54,964	\$ 32,445
IT Administration	\$ 3,355	\$ 3,535
Facilities and Overhead	\$ 5,000	\$ 4,000
Downtime	\$ 23,908	\$ 23,908

- 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

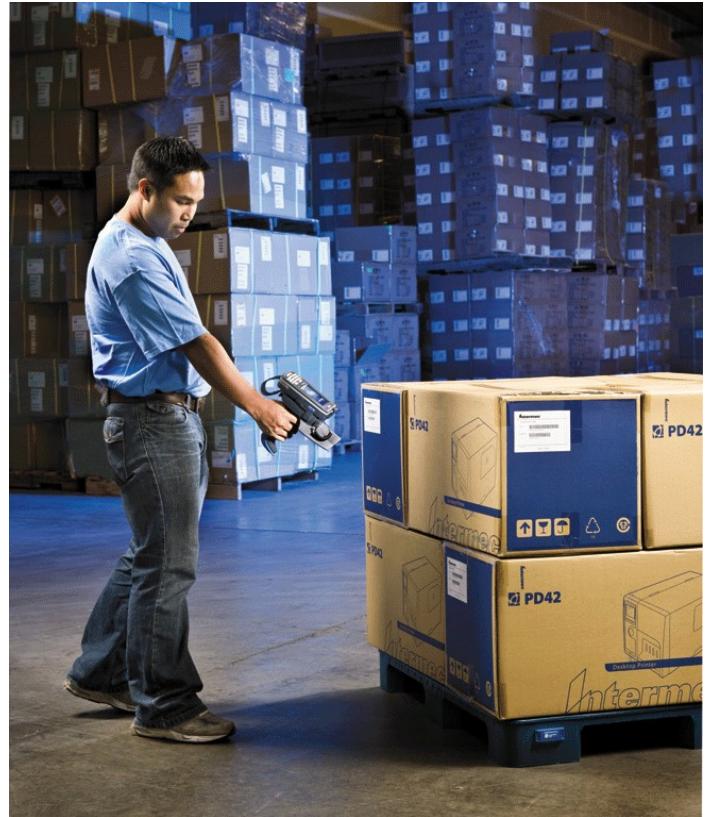
- Overview
 - A system is effective if it supports business requirements and meets user needs
 - A system is reliable if it handles input errors, processing errors, hardware failures, or human mistakes
 - A system is maintainable if it is flexible, scalable, and easily modified

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

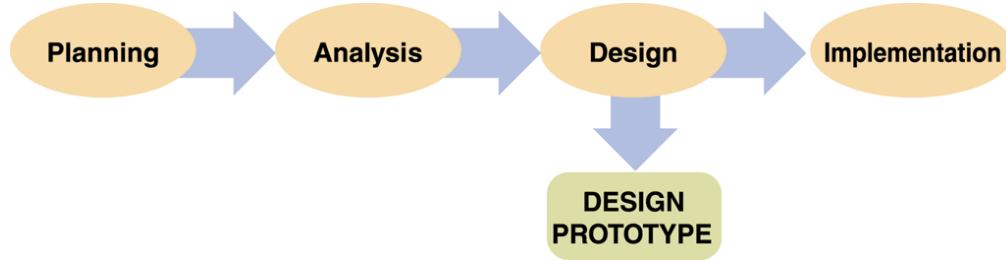
Design Modules that Perform a Single Function

- 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