Systems Analysis and Design I
Prof. Dr. Mohamed M. Elgazzar
IS 312

Course Information
- **Text Book:** Modern Systems Analysis and Design, Sixth Edition, Jeffrey A. Hoffer, Joey F. George, Joseph S. Valacich
The Vision

The Faculty of Computers and Artificial Intelligence aims to be a pioneer educational institution and research center in the field of Computer Science and Information Technology. This makes it possible to achieve competition and uniqueness at the local, regional, and international levels.
The Faculty Mission:
The Faculty of Computers and Artificial Intelligence committed to: Providing academic educational services that lead to a distinguished graduate capable of competing locally, regionally, internationally, and highly qualified in the field of computers and information technology. In addition, graduates will have high professional skills and ethics and will be capable of self-learning to cope with rapid developments in his/her field of specialization.

Providing continuous education, training and consulting services to achieve national development and improve the local environment.

Conducting scientific research in various fields of computer science and information technology.
Course Specification
IS 312: Analysis & Design of Information Systems I

A- Affiliation:

Relevant Program: Computer Science - BSc program. Information System - BSc program.
Department offering the program: Information System Department
Department offering the course: Information System Department
Date of Specification Approval: 19th February 2018

B- Basic Information:

Title: Analysis & Design of Information Systems I
Pre-requisite: Software Engineering (1)
Lectures: 3H
Credit Hours: 3H
Level: 3
Code: IS 312
Semester: Spring
Tutorial/Practical: 2H

Professional Information

This course provides students with advanced tools and techniques essential for the analysis and design of information systems. Course topics include: introduction to information systems, outline of different information system development methodologies, managing information system project, structuring system requirements (using DFD and use cases), data dictionary specification, process specification, modeling logic (using decision tables and UML diagrams), system data base design (using E/R and class diagram), I/O interface design (menus, forms and reports), system implementation and maintenance, and case studies.

1- Course Learning Objectives:

By the end of this course the students should demonstrate the knowledge and understanding of the different development approaches for information systems (including traditional SDLC, prototyping, JAD, and object oriented approaches). They should be able to analysis, design, implementation, testing, and operation of large information systems.
2- Intended Learning Outcomes (ILOs)  

a- Knowledge and Understanding:  
On successful completion of the course, the students should demonstrate knowledge and sound understanding of:  
a1- Basic principles of information systems including different types of information systems, different development methodologies, phases of system development life cycle, project management phases, I/O interface guidelines (A1, A13, A14)  
a2- Techniques and methods of analysis and design to meet the special needs of information systems (A1, A4, A5, A18, A20)  
a3- Build information system applications using software tools for analysis, design, and implementation (A2, A3, A19).  

b- Intellectual Skills:  
On successful completion of the course, the students should be able to:  
b1- Realize the development principles of information systems (B1, B2).  
b2- Analyze and compare different techniques and methods of analysis and design to meet the special needs of information systems (B11, B14)  
b3- Design, implement, and test complete software system for a specified information system application domain. (B14, B17, B18)  

c – Professional and Practical Skills:  
On successful completion of the course, the students should be able to:  
c1- Deploy project management tools, analysis DFD tools, database design tools, and I/O interface tools, and CASE tools to improve the process of analysis and design of information systems (C1, C2, C3, C9, C19).  
c2- Implement an information system using database and I/O interface software packages (C1, C2, C6, C14, C15).  
c3- Test and maintain information system applications (C21).  

d- General and Transferable Skills:  
On successful completion of the course, the students should be able to:  
d1- Work effectively alone and as a member of development team and involve in group discussions and in seminars (D1, D2)
d2- Communicate effectively and present data results orally and in written form (D6)
d3- Search for information in references and in internet (D3, D7).
d4- Practice self-learning (D7).

Course Contribution in the program ILO’S

<table>
<thead>
<tr>
<th>ILO’S</th>
<th>program ILO’S</th>
</tr>
</thead>
<tbody>
<tr>
<td>A Knowledge and understanding</td>
<td>A1, A2, A3, A4, A5, A13, A14, A18, A19, A20</td>
</tr>
<tr>
<td>B Intellectual skills</td>
<td>B1, B2, B11, B14, B17, B18</td>
</tr>
<tr>
<td>C Professional and practical skills</td>
<td>C1, C2, C3, C6, C9, C14, C15, C19, C21</td>
</tr>
<tr>
<td>D General and transferable skills</td>
<td>D1, D2, D3, D6, D7</td>
</tr>
</tbody>
</table>

3- Contents:

<table>
<thead>
<tr>
<th>Topic</th>
<th>Lecture</th>
<th>Tutorial/ Practical</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction to information systems</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>Systems development environment</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>Managing the Information Systems Project</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>Initiating and Planning Systems Development Projects</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>Structuring System Requirements: Process modeling using Data Flow Diagram (DFD)</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>Structuring System Requirements Data Dictionary</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>Structuring System Requirements : Modeling Logic with Structured English and Decision Tables</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>System Database Design : Conceptual Data Modeling using EER</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>I/O Interface Design: Designing Forms and Reports</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>I/O Interface Design: Designing Menus and Dialogs</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>System Implementation &amp; Maintenance</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>Total</td>
<td>45</td>
<td>30</td>
</tr>
</tbody>
</table>
## 4-Teaching and Learning and Assessment methods:

<table>
<thead>
<tr>
<th>Course IL OS</th>
<th>Teaching Methods</th>
<th>Learning Methods</th>
<th>Assessment Method</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Lecture</td>
<td>Presentations and Movies</td>
<td>Discussions &amp; Seminar Seminars</td>
</tr>
<tr>
<td>Knowledge</td>
<td>a1</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td></td>
<td>a2</td>
<td>√</td>
<td></td>
</tr>
<tr>
<td></td>
<td>a3</td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>intellectual</td>
<td>b1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>b2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>b3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Applied</td>
<td>c1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>c2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>c3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>General</td>
<td>d1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>d2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>d3</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>d4</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
5- Assessment Timing and Grading:

<table>
<thead>
<tr>
<th>Assessment method</th>
<th>Timing</th>
<th>Grade (Decrees)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Semester Work : seminars, quizzes</td>
<td>Bi-Weekly</td>
<td>20%</td>
</tr>
<tr>
<td>assignment and reports</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mid-Term Exam</td>
<td>7-th Week</td>
<td>20%</td>
</tr>
<tr>
<td>Practical Exam</td>
<td>15-th Week</td>
<td>10%</td>
</tr>
<tr>
<td>Written Exam</td>
<td>16-th Week</td>
<td>50%</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td>100%</td>
</tr>
</tbody>
</table>

6 List of references
1. – Text Books:

6.2 - Course notes:
Available and is handed to the students on first day of classes.

6.3- Required Books:

6.4- Websites:
• https://www.pearsonhighered.com/sign-in.html?null

7 Facilities required for teaching and learning:
   1. – White board and markers.
   2. – Overhead projector (transparencies).
   3. – Data show.

Course Coordinator: Prof. Dr. Hanafty M. Ismail
Head of Department: Prof. Dr. Hafez M.S Abdel-Wahab
# TABLE OF CONTENTS

## CHAPTER 1 Introduction to Systems Analysis and Design
- The Impact of Information Technology 5
- Information System Components 12
- What Information Do Users Need? 22
- Systems Development Tools 32

## CHAPTER 2 The Systems Development Environment
- Introduction 45
- Developing Information Systems 48
- Systems Development Life Cycle (SDLC) 49
- The Heart of the Systems Development Process 56
- Traditional Waterfall SDLC 57
- Different Approaches to Improving Development 59

## CHAPTER 3 Managing the Information Systems Project
- Importance of Project Management 81
- Managing the Information Systems Project 85
- Project Management Activities 87
- Phases of Project Management Process 88
- Representing and Scheduling Project Plans 102
TABLE OF CONTENTS (Cont.)

CHAPTER 4 Structuring System Process Requirements
- Process Modeling 119
- Deliverables and Outcomes 122
- Data Flow Diagramming Mechanics 124
- Definitions and Symbols 126
- Developing DFDs 129
- Data Flow Diagramming Rules 133
- Decomposition of DFDs 136
- Balancing DFDs 140
- Guidelines for Drawing DFDs 148
- Using DFDs as Analysis Tools 154

CHAPTER 5 Structuring System Data Requirements
- Conceptual Data Modeling 167
- Deliverables and Outcome 171
- Gathering Information for Conceptual Data Modeling 174
- Introduction to Entity-Relationship (E-R) Modeling 178
- Naming and Defining Entity Types 182
- Candidate Keys and Identifiers 189
- Relationships 195
TABLE OF CONTENTS (Cont.)

CHAPTER 6 Designing Forms and Reports
   - Designing Forms and Reports 219
   - Formatting Forms and Reports 228
   - Usability Success Factors 251
   - Measures of Usability 253

CHAPTER 7 Designing Interfaces and Dialogues
   - Designing Interfaces and Dialogues 261
   - Interaction Methods and Devices 265
   - Object-Based Interaction 274

CHAPTER 8 System Implementation
   - The Process of Coding, Testing, and Installation 303
   - Software Application Testing 307
   - Acceptance Testing by Users 319

CHAPTER 9 Maintaining Information Systems
   - Maintaining Information Systems 352
   - Types of System Maintenance 358
   - The Cost of Maintenance 363
   - Measuring Maintenance Effectiveness 369
Chapter 1

Introduction to Systems Analysis and Design
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, software, and services that people use to manage, communicate, and share information.

- Business success depends on information technology. IT is driving a new digital economy, where advances in hardware, software, and connectivity can provide enormous benefits to businesses and individuals.
The Impact of Information Technology

- **Systems Analysis and Design**
  - Step-by-step process for developing high-quality information systems.

Hardware, software, data, people, and procedures that work together to produce quality information.

**System**—Set of components that interact to achieve common goal.

Businesses use many types of systems.
The Impact of Information Technology

- **Who develops Information Systems?**

  - Traditionally:
    - In-house applications
    - Software packages
  
  - Today, the choice is much more complex,
    - Internet-based application services
    - Outsourcing
    - Custom solutions from IT consultants
    - Enterprise-wide software strategies
  
  - It is always important for companies to plan the system carefully before considering implementation options.
The Impact of Information Technology

- **Systems Development**
  
  - Business information systems are developed by people who are technically qualified, business-oriented, and highly motivated.
  
  - Successful developers also must be good communicators with strong analytical and critical thinking skills.
The Impact of Information Technology

What are guidelines for system development?

- Arrange tasks into **phases**
- Involve **users** (anyone for whom system is being built)
- Develop clearly defined **standards** (procedures company expects employees to follow)
The Impact of Information Technology

Who participates in the system development life cycle?
The Impact of Information Technology

What is a systems analyst?

- Responsible for designing and developing information system
- Liaison between users and IT professionals
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 System Components

Information systems have five key components: hardware, software, data, processes, and people as shown in Figure 1-1.

Figure 1-1 The five main components of an information system.
Information System Components

- **Hardware**
  - Is the physical layer of the information system.
Information System Components

Software

- refers to the programs that control the hardware and produce the desired information or results.

- Software consists of system software and application software.
Information System Components

Software (Cont.)

- **System software** manages the hardware components.

- **Examples of system software:**
  - Operating system
  - Security software that protects the computer from intrusion
  - Utility programs that handle specific tasks such as data backup and disk management.
Information System Components

- **Software** (Cont.)
  - Application software consists of programs that support day-to-day business functions and provide users with the information they require.
Information System Components

Software (Cont.)

- Examples of application software:
  - Enterprise applications (order processing systems, payroll systems)
  - Horizontal system: adapted for use in many different types of companies (an inventory application)
  - Vertical system: meet the unique requirements of a specific business or industry (Web-based retailer, a medical practice, or a video chain)
  - Legacy systems: how a new system will interface with older systems (new human resources system might need to exchange data with an older payroll application)
Data

Is the raw material that an information system transforms into useful information.
Information System Components

Processes

- Define the tasks and business functions that users, managers, and IT staff members perform to achieve specific results.
People

People who have an interest in an information system are called stakeholders.

- **Users**, or end users, are the people who interact with an information system, both inside and outside the company.
- IT staff members, such as systems analysts, programmers, and network administrators who develop and support the system.
What Information Do Users Need?

A systems analyst must understand the company’s organizational model in order to recognize who is responsible for specific processes and decisions and to be aware of what information is required by whom.
What Information Do Users Need?

- Top managers develop long-range plans, called **strategic plans**, which define the company’s overall mission and goals.

- To plot a future course, top managers ask questions such as:
  - “How much should the company invest in information technology?” or
  - “How much will Internet sales grow in the next five years?” or
What Information Do Users Need?

- Top managers (Cont.)
  - “Should the company build new factories or contract out the production functions?”
  - Strategic planning affects the company’s future survival and growth, including long-term IT plans.
  - Top managers focus on the overall business enterprise and use IT to set the company’s course and direction.
What Information Do Users Need?

- Top managers (Cont.)
  - To develop a strategic plan, top managers also need information from outside the company, such as economic forecasts, technology trends, competitive threats, and governmental issues.
What Information Do Users Need?

- Middle Managers and Knowledge Workers
  - Middle managers provide direction, necessary resources, and performance feedback to supervisors and team leaders.
  - Middle managers need more detailed information than top managers, but somewhat less than supervisors who oversee day-to-day operations.
What Information Do Users Need?

Middle Managers and Knowledge Workers

- Middle managers need more detailed information than top managers, but somewhat less than supervisors who oversee day-to-day operations.

- Example:
  - A middle manager might review a weekly sales summary for a three-state area, whereas a local sales team leader would need a daily report on customer sales at a single location.
What Information Do Users Need?

Middle Managers and Knowledge Workers

- Every company has people called knowledge workers.
- Knowledge workers include professional staff members such as systems analysts, programmers, accountants, researchers, trainers, and human resource specialists.
- Knowledge workers also use business support systems, knowledge management systems, and user productivity systems.
What Information Do Users Need?

- **Middle Managers and Knowledge Workers**
  
  - Knowledge workers provide support for the organization’s basic functions.
  
  - Just as a military unit requires logistical support, a successful company needs knowledge workers to carry out its mission.
What Information Do Users Need?

- Supervisors and Team Leaders
  - Supervisors, often called team leaders, oversee operational employees and carry out day-to-day functions. They coordinate operational tasks and people, make necessary decisions, and ensure that the right tools, materials, and training are available.
  - Supervisors and team leaders need decision support information, knowledge management systems, and user productivity systems to carry out their responsibilities.
What Information Do Users Need?

- **Operational Employees**
  - *Operational employees* include users who rely on TP systems to enter and receive data they need to perform their jobs.
  - In many companies, *operational users also need information to handle tasks and make decisions that were assigned previously to supervisors.*
    - This trend, called *empowerment*, gives employees more responsibility and accountability.
Systems Development Tools

- Systems analysts must know how to use a variety of techniques such as modeling, prototyping, and computer-aided systems engineering tools to plan, design, and implement information systems.

- Systems analysts work with these tools in a team environment.
Modeling

- Modeling produces a graphical representation of a concept or process that systems developers can analyze, test, and modify.

- A systems analyst can describe and simplify an information system by using a set of business, data, object, network, and process models.
Systems Development Tools

Modeling (Cont.)

- A **business model**, or **requirements model**, describes the information that a system must provide.

- A **data model** describes data structures and design.

- An **object model** describes objects, which combine data and processes.
Systems Development Tools

- **Modeling (Cont.)**
  - A network model describes the design and protocols of telecommunications links.
  - A process model describes the logic that programmers use to write code modules.
Systems Development Tools

- Prototyping
  - What is a prototype?
    - Working model of proposed system
    - Beginning a prototype too early may lead to problems
Prototyping (Cont.)

- Prototyping tests system concepts and provides an opportunity to examine input, output, and user interfaces before final decisions are made.

- A prototype is an early working version of an information system.

- A prototype can be an extremely valuable tool.
Systems Development Tools

Iterative development process:
- Requirements quickly converted to a working system
- System is continually revised
- Close collaboration between users and analysts
Prototyping (Cont.)

Advantage:

- Speeds up the development process significantly (prototype can serve as an initial model that is used as a benchmark to evaluate the finished system, or the prototype itself can develop into the final version of the system).
Systems Development Tools

- Prototyping (Cont.)

  □ Disadvantage:

    ➢ Important decisions might be made too early, before business or IT issues are thoroughly understood.
Computer-Aided Systems Engineering (CASE) Tools

Computer-Aided Systems Engineering (CASE), also called computer-aided software engineering, is a technique that uses powerful software, called CASE tools, to help systems analysts develop and maintain information systems.
Systems Development Tools

- **Computer-Aided Systems Engineering (CASE) Tools (Cont.)**
  - CASE tools provide an overall framework for systems development and support a wide variety of design methodologies, including structured analysis and object-oriented analysis.
  - Diagramming tools
  - Example products: Oracle Designer, Rational Rose
Systems Development Tools
What is computer-aided software engineering (CASE)?

Software tools designed to support activities of system development cycle (Software tools providing automated support for systems development).
Chapter 2
The Systems Development Environment
Introduction

Information Systems Analysis and Design

- Complex organizational process
- Used to develop and maintain computer-based information systems
- Used by a team of business and systems professionals
FIGURE 1-1 An organizational approach to systems analysis and design is driven by methodologies, techniques, and tools.
A Modern Approach to Systems Analysis and Design (Cont.)

- **Application Software**
  - Computer software designed to support organizational functions or processes

- **Systems Analyst**
  - Organizational role most responsible for analysis and design of information systems
Developing Information Systems

**System Development Methodology** is a standard process followed in an organization to conduct all the steps necessary to analyze, design, implement, and maintain information systems.
Systems Development Life Cycle (SDLC)

- Traditional methodology used to develop, maintain, and replace information systems.
- Phases in SDLC:
  - Planning
  - Analysis
  - Design
  - Implementation
  - Maintenance
Standard and Evolutionary Views of SDLC

**FIGURE 1-2**
The systems development life cycle

**FIGURE 1-3** Evolutionary model
Systems Development Life Cycle (SDLC) (Cont.)

- **Planning** – an organization’s total information system needs are identified, analyzed, prioritized, and arranged

- **Analysis** – system requirements are studied and structured
Design – a description of the recommended solution is converted into logical and then physical system specifications

Logical design – all functional features of the system chosen for development in analysis are described independently of any computer platform
Systems Development Life Cycle (SDLC) (Cont.)

- **Physical design** – the logical specifications of the system from logical design are transformed into the technology-specific details from which all programming and system construction can be accomplished
Systems Development Life Cycle (SDLC) (Cont.)

- **Implementation** – the information system is coded, tested, installed and supported in the organization

- **Maintenance** – an information system is systematically repaired and improved
<table>
<thead>
<tr>
<th>Phase</th>
<th>Products, Outputs, or Deliverables</th>
</tr>
</thead>
<tbody>
<tr>
<td>Planning</td>
<td>Priorities for systems and projects; an architecture for data, networks, and selection hardware, and IS management are the result of associated systems</td>
</tr>
<tr>
<td></td>
<td>Detailed steps, or work plan, for project</td>
</tr>
<tr>
<td></td>
<td>Specification of system scope and planning and high-level system requirements or features</td>
</tr>
<tr>
<td></td>
<td>Assignment of team members and other resources</td>
</tr>
<tr>
<td></td>
<td>System justification or business case</td>
</tr>
<tr>
<td>Analysis</td>
<td>Description of current system and where problems or opportunities are with a general recommendation on how to fix, enhance, or replace current system</td>
</tr>
<tr>
<td></td>
<td>Explanation of alternative systems and justification for chosen alternative</td>
</tr>
<tr>
<td>Design</td>
<td>Functional, detailed specifications of all system elements (data, processes, inputs, and outputs)</td>
</tr>
<tr>
<td></td>
<td>Technical, detailed specifications of all system elements (programs, files, network, system software, etc.)</td>
</tr>
<tr>
<td></td>
<td>Acquisition plan for new technology</td>
</tr>
<tr>
<td>Implementation</td>
<td>Code, documentation, training procedures, and support capabilities</td>
</tr>
<tr>
<td>Maintenance</td>
<td>New versions or releases of software with associated updates to documentation, training, and support</td>
</tr>
</tbody>
</table>
The Heart of Systems Development Process

Current practice combines analysis, design, and implementation into a single iterative and parallel process of activities.
Traditional Waterfall SDLC

One phase begins when another completes, with little backtracking and looping.

FIGURE 1-9
A traditional waterfall SDLC
Problems with Waterfall Approach

- System requirements “locked in” after being determined (can't change)
- Limited user involvement (only in requirements phase)
- Too much focus on milestone deadlines of SDLC phases to the detriment of sound development practices
Different Approaches to Improving Development

- CASE Tools
- Rapid Application Development (RAD)
- Agile Methodologies
- eXtreme Programming
Computer-Aided Software Engineering (CASE) Tools

- Diagramming tools enable graphical representation.
- Computer displays and report generators help prototype how systems “look and feel”.
Computer-Aided Software Engineering (CASE) Tools (Cont.)

- Analysis tools automatically check for consistency in diagrams, forms, and reports.
- A central repository provides integrated storage of diagrams, reports, and project management specifications.
Computer-Aided Software Engineering (CASE) Tools (Cont.)

- Documentation generators standardize technical and user documentation.
- Code generators enable automatic generation of programs and database code directly from design documents, diagrams, forms, and reports.
FIGURE 1-10
A class diagram from IBM’s Rational Rose
(Source: IBM)
### Table 1-2: Examples of CASE Usage within the SDLC

<table>
<thead>
<tr>
<th>SDLC Phase</th>
<th>Key Activities</th>
<th>CASE Tool Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project identification and selection</td>
<td>Display and structure high-level organizational information</td>
<td>Diagramming and matrix tools to create and structure information</td>
</tr>
<tr>
<td>Project initiation and planning</td>
<td>Develop project scope and feasibility</td>
<td>Repository and documentation generators to develop project plans</td>
</tr>
<tr>
<td>Analysis</td>
<td>Determine and structure system requirements</td>
<td>Diagramming to create process, logic, and data models</td>
</tr>
<tr>
<td>Logical and physical design</td>
<td>Create new system designs</td>
<td>Form and report generators to prototype designs; analysis and documentation generators to define specifications</td>
</tr>
<tr>
<td>Implementation</td>
<td>Translate designs into an information system</td>
<td>Code generators and analysis, form and report generators to develop system; documentation generators to develop system and user documentation</td>
</tr>
<tr>
<td>Maintenance</td>
<td>Evolve information system</td>
<td>All tools are used (repeat life cycle)</td>
</tr>
</tbody>
</table>
Rapid Application Development (RAD)

- Methodology to radically decrease design and implementation time
- Involves: extensive user involvement, prototyping, JAD sessions, integrated CASE tools, and code generators
Rapid Application Development (RAD) (Cont.)

FIGURE 1-11
RAD life cycle
Service-Oriented Architecture (SOA)

- An approach to systems development based on building complete systems through assembling software components, each of which model generic business functions
Service-Oriented Architecture (SOA) (Cont.)

FIGURE 1-12
Illustration of a service, a credit check, used by applications and other services
Agile Methodologies

- Motivated by recognition of software development as fluid, unpredictable, and dynamic
- Three key principles
  - Adaptive rather than predictive
  - Emphasize people rather than roles
  - Self-adaptive processes
The Agile Methodologies group argues that software development methodologies adapted from engineering generally do not fit with real-world software development.
When to use Agile Methodologies

- If your project involves:
  - Unpredictable or dynamic requirements
  - Responsible and motivated developers
  - Customers who understand the process and will get involved
eXtreme Programming

- Short, incremental development cycles
- Automated tests
- Two-person programming teams
eXtreme Programming (Cont.)

- Coding and testing operate together
- Advantages:
  - Communication between developers
  - High level of productivity
  - High-quality code
Object-Oriented Analysis and Design (OOAD)

- Based on objects rather than data or processes

  **Object**: a structure encapsulating attributes and behaviors of a real-world entity
Object-Oriented Analysis and Design (OOAD) (Cont.)

- **Object class**: a logical grouping of objects sharing the same attributes and behaviors

- **Inheritance**: hierarchical arrangement of classes enable subclasses to inherit properties of superclasses
Rational Unified Process (RUP)
عملية موحدة عقلانية

- An object-oriented systems development methodology
- RUP establishes four phase of development: inception, elaboration, construction, and transition.
- Each phase is organized into a number of separate iterations.
FIGURE 1-13
Phases of OOSAD-based development
Chapter 3
Managing the Information Systems Project
Importance of Project Management

- Project management may be the most important aspect of systems development.
- Effective PM helps to ensure
  - The meeting of customer expectations.
  - The satisfying of budget and time constraints.
- PM skills are difficult and important to learn.
Three computer applications at Pine Valley Furniture: Order filling, invoicing, and payroll.

Deciding on Systems Projects

- **System Service Request (SSR)**
  A standard form for requesting or proposing systems development work within an organization

- **Feasibility study**
  A study that determines whether a requested system makes economic and operational sense for an organization
FIGURE 3-2
System Service Request for purchasing a fulfillment system with name and contact information of the person requesting the system, a statement of the problem, and the name and contact information of the liaison and sponsor.
Managing the Information Systems Project

- **Project**
  - A planned undertaking of related activities to reach an objective that has a beginning and an end

- **Project management**
  - A controlled process of initiating, planning, executing, and closing down a project
Managing the Information Systems Project (cont.)

- Project manager
  - Systems analyst with management and leadership skills responsible for leading project initiation, planning, execution, and closedown

- Deliverable
  - The end product of an SDLC phase
Project Management Activities

FIGURE 3-4
A project manager juggles numerous Activities
Phases of Project Management Process

- Phase 1: Initiation
- Phase 2: Planning
- Phase 3: Execution
- Phase 4: Closedown
PM Phase 1: Project Initiation

- Assess size, scope and complexity, and establish procedures.
  تقييم الحجم والنطاق والتعقيد ، ووضع الإجراءات.

- Establish:
  - Initiation team
  - Relationship with customer
  - Project initiation plan
  - Management procedures
  - Project management environment
  - Project workbook
The project workbook for the Purchase Fulfillment System project contains nine key documents in both hard-copy and electronic form.
PM Phase 2: Project Planning

- Define clear, discrete activities and the work needed to complete each activity

Tasks
- Define project scope, alternatives, feasibility
- Divide project into tasks
- Estimate resource requirements
- Develop preliminary schedule
- Develop communication plan
- Determine standards and procedures
- Identify and assess risk
- Create preliminary budget
- Develop a statement of work
- Set baseline project plan
Level of project planning detail should be high in the short term, with less detail as time goes on.

FIGURE 3-8
Some Components of Project Planning

- **Statement of Work (SOW)**
  - “Contract” between the IS staff and the customer regarding deliverables and time estimates for a system development project

- **The Baseline Project Plan (BPP)**
  - Contains estimates of scope, benefits, schedules, costs, risks, and resource requirements

- **Preliminary Budget**
  - Cost-benefit analysis outlining planned expenses and revenues

التحليل التكلفة والعائد يحدد النفقات والإيرادات المخططة
Some Components of Project Planning (cont.)

- **Work Breakdown Structure (WBS)**
  - Division of project into manageable and logically ordered tasks and subtasks

- **Scheduling Diagrams**
  - Gantt chart: horizontal bars represent task durations
  - Network diagram: boxes and links represent task dependencies
Special-purpose project management software is available for this.

FIGURE 3-10
Gantt chart showing project tasks, duration times for those tasks, and predecessors
Scheduling Diagams Network Diagram

**FIGURE 3-13**
A network diagram illustrating tasks with rectangles (or ovals) and the relationships and sequences of those activities with arrows.

Special-purpose project management software is available for this.
Spreadsheet software is good for this.

FIGURE 3-15
A financial cost and benefit analysis for a systems development project
PM Phase 3: Project Execution

- Plans created in prior phases are put into action.

- Actions
  - Execute baseline project plan
  - Monitor progress against baseline plan
  - Manage changes in baseline plan
  - Maintain project workbook
  - Communicate project status

خطة المشروع الاساسية
احتفظ بمصنف المشروع
أبلغ عن حالة المشروع
Monitoring Progress with a Gantt Chart

FIGURE 3-17
Gantt chart with tasks 3 and 7 completed

Red bars indicate critical path; lines through bars indicate percent complete.
Communication Methods

- **High Formality**
  - Project workbook
  - Newsletters
  - Status reports
  - Specification documents
  - Meeting minutes

- **Medium Formality**
  - Meetings
  - Seminars and workshops
  - Memos

- **Low Formality**
  - Bulletin boards
  - Brown bag lunches
  - Hallway discussions
PM Phase 4: Project Closedown

- Bring the project to an end

- Actions
  - Close down the project.
  - Conduct post-project reviews.
  - Close the customer contract.
Representing and Scheduling Project Plans

- Gantt Charts
- Network Diagrams
- PERT Calculations
- Critical Path Scheduling
- Project Management Software
Gantt Charts vs. Network Diagrams

- **Gantt charts**
  - Show task durations.
  - Show time overlap.
  - Show slack time in duration.

- **Network diagrams**
  - Show task dependencies.
  - Do not show time overlap, but show parallelism.
  - Show slack time in boxes.
Estimating Task Duration

- PERT: Program Evaluation Review Technique

- Technique that uses optimistic \((o)\) متفائل, pessimistic \((p)\) متشائم, and realistic \((r)\) واقعية time estimates to determine expected task duration.

- Formula for Estimated Time:
  \[ ET = (o + 4r + p)/6 \]
# Example PERT Analysis

**FIGURE 3-22**

Estimated time calculations for the SPTS project

<table>
<thead>
<tr>
<th>ACTIVITY</th>
<th>TIME ESTIMATE (in weeks)</th>
<th>EXPECTED TIME (ET)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>o</td>
<td>r</td>
</tr>
<tr>
<td>1. Requirements Collection</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>2. Screen Design</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>3. Report Design</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>4. Database Design</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>5. User Documentation</td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>6. Programming</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>7. Testing</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>8. Installation</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Critical Path Scheduling

- A scheduling technique whose order and duration of a sequence of task activities directly affect the completion

- **Critical path:** the shortest time in which a project can be completed

- **Slack time:** the time an activity can be delayed without delaying the project
Critical Path Example (dependencies between tasks)

PRECEDING ACTIVITIES indicate the activities that must be completed before the specified activity can begin.

<table>
<thead>
<tr>
<th>ACTIVITY</th>
<th>PRECEDING ACTIVITY</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Requirements Collection</td>
<td>—</td>
</tr>
<tr>
<td>2. Screen Design</td>
<td>1</td>
</tr>
<tr>
<td>3. Report Design</td>
<td>1</td>
</tr>
<tr>
<td>4. Database Design</td>
<td>2,3</td>
</tr>
<tr>
<td>5. User Documentation</td>
<td>4</td>
</tr>
<tr>
<td>6. Programming</td>
<td>4</td>
</tr>
<tr>
<td>7. Testing</td>
<td>6</td>
</tr>
<tr>
<td>8. Installation</td>
<td>5,7</td>
</tr>
</tbody>
</table>

**FIGURE 3-23** Sequence of Activities within the SPTS project
Critical Path Example

FIGURE 3-25
A network diagram that illustrates the activities (circles) and the sequence (arrows) of those activities
Determining the Critical Path

- Calculate the earliest possible completion time for each activity by summing the activity times in the longest path to the activity. This gives total expected project time.

- Calculate the latest possible completion time for each activity by subtracting the activity times in the path following the activity from the total expected time. This gives slack time for activities.

- Critical path – contains no activities with slack time.
Critical Path Calculation

FIGURE 3-26
A network diagram for the SPTS project showing estimated times for each activity and the earliest and latest expected completion time for each activity.

Early and late time calculations are determined and critical path established. (Note: Activity #5 can begin late without affecting project completion time).
Critical Path Calculation (cont.)

FIGURE 3-27
Activity slack time calculations for the SPTS project; all activities except number 5 are on the critical path.

Note the slack time in Activity #5.
Using Project Management Software

Many powerful software tools exist for assisting with project management.

Example: Microsoft Project can help with

- Entering project start date.
- Establishing tasks and task dependencies.
- Viewing project information as Gantt or Network diagrams.
Project Start Date

FIGURE 3-28
Establishing a project starting date in Microsoft Project for Windows
Entering Tasks

FIGURE 3-29
Entering tasks and assigning task relationships in Microsoft project for Windows
Viewing Network Diagram

Hexagon shape indicates a milestone.

Red boxes and arrows indicate critical path (no slack).

FIGURE 3-30 Viewing project information as a network diagram in Microsoft Project for Windows
Viewing Gantt Chart

FIGURE 3-31
Gantt chart showing progress of activities (right frame) versus planned activities (left frame)

Black line at top indicates a summary activity (composed of subtasks). Diamond shape indicates a milestone.
Chapter 4
Structuring System Process Requirements
Process Modeling

FIGURE 7-1
Systems development life cycle with the analysis phase highlighted
Process Modeling (Cont.)

Graphically represent the processes that capture, manipulate, store, and distribute data between a system and its environment and among system components.
Process Modeling (Cont.)

- Utilize استفاده information gathered during requirements determination.
- Processes and data structures are modeled.
Deliverables and Outcomes

- Context data flow diagram (DFD)
  - Scope of system
- DFDs of current physical system
  - Adequate detail only
- DFDs of current logical system
  - Enables analysts to understand current system
Deliverables and Outcomes (Cont.)

- DFDs of new logical system
  - Technology independent
  - Show data flows, structure, and functional requirements of new system

- Thorough description of each DFD component
Data Flow Diagramming Mechanics

- Represent both physical and logical information systems
- Only four symbols are used
Data Flow Diagramming Mechanics (Cont.)

- Useful for depicting purely logical information flows
- DFDs that detail physical systems differ from system flowcharts which depict details of physical computing equipment
Definitions and Symbols

FIGURE 7-2
Comparison of DeMarco and Yourdon and Gane and Sarson DFD symbol sets
Definitions and Symbols (Cont.)

- **Process**: work or actions performed on data (inside the system)
- **Data store**: data at rest (inside the system)
Definitions and Symbols (Cont.)

- **Source/sink**: external entity that is origin or destination of data (outside the system)

- **Data flow**: arrows depicting movement of data
Developing DFDs

- **Context diagram** is an overview of an organizational system that shows:
  - the system boundaries.
  - external entities that interact with the system.
  - major information flows between the entities and the system.

- Note: only one process symbol, and no data stores shown
Context Diagram

FIGURE 7-4
Context diagram of Hoosier Burger’s food-ordering system
Developing DFDs (Cont.)

- **Level-0 diagram** is a data flow diagram that represents a system’s major processes, data flows, and data stores at a high level of detail.
  - Processes are labeled 1.0, 2.0, etc. These will be decomposed (lower-level) DFDs.
Level-0 Diagram

FIGURE 7-5
Level-0 DFD of Hoosier Burger’s food-ordering system
Data Flow Diagramming Rules

- There are two DFD guidelines that apply:
  - *The inputs to a process are different from the outputs of that process.*
    - Processes purpose is to transform inputs into outputs.
  - *Objects on a DFD have unique names.*
    - Every process has a unique name.
### TABLE 7-2 Rules Governing Data Flow Diagramming

<table>
<thead>
<tr>
<th>Process:</th>
</tr>
</thead>
<tbody>
<tr>
<td>A. No process can have only outputs. It is making data from nothing (a miracle). If an object has only outputs, then it must be a source.</td>
</tr>
<tr>
<td>B. No process can have only inputs (a black hole). If an object has only inputs, then it must be a sink.</td>
</tr>
<tr>
<td>C. A process has a verb phrase label.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Data Store:</th>
</tr>
</thead>
<tbody>
<tr>
<td>D. Data cannot move directly from one data store to another data store. Data must be moved by a process.</td>
</tr>
<tr>
<td>E. Data cannot move directly from an outside source to a data store. Data must be moved by a process that receives data from the source and places the data into the data store.</td>
</tr>
<tr>
<td>F. Data cannot move directly to an outside sink from a data store. Data must be moved by a process.</td>
</tr>
<tr>
<td>G. A data store has a noun phrase label.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Source/Sink:</th>
</tr>
</thead>
<tbody>
<tr>
<td>H. Data cannot move directly from a source to a sink. It must be moved by a process if the data are of any concern to our system. Otherwise, the data flow is not shown on the DFD.</td>
</tr>
<tr>
<td>I. A source/sink has a noun phrase label.</td>
</tr>
</tbody>
</table>
### TABLE 7-2 Rules Governing Data Flow Diagramming (cont.)

<table>
<thead>
<tr>
<th>Data Flow:</th>
</tr>
</thead>
<tbody>
<tr>
<td>J. A data flow has only one direction of flow between symbols. It may flow in both directions between a process and a data store to show a read before an update. The latter is usually indicated, however, by two separate arrows because these happen at different times.</td>
</tr>
<tr>
<td>K. A fork in a data flow means that exactly the same data goes from a common location to two or more different processes, data stores, or sources/sinks (this usually indicates different copies of the same data going to different locations).</td>
</tr>
<tr>
<td>L. A join in a data flow means that exactly the same data come from any of two or more different processes, data stores, or sources/sinks to a common location.</td>
</tr>
<tr>
<td>M. A data flow cannot go directly back to the same process it leaves. There must be at least one other process that handles the data flow, produces some other data flow, and returns the original data flow to the beginning process.</td>
</tr>
<tr>
<td>N. A data flow to a data store means update (delete or change).</td>
</tr>
<tr>
<td>O. A data flow from a data store means retrieve or use.</td>
</tr>
<tr>
<td>P. A data flow has a noun phrase label. More than one data flow noun phrase can appear on a single arrow as long as all of the flows on the same arrow move together as one package.</td>
</tr>
</tbody>
</table>

(Source: Adapted from celko, 1987.)
Decomposition of DFDs

- **Functional decomposition** is an iterative process of breaking a system description down into finer and finer detail.
  - Creates a set of charts in which one process on a given chart is explained in greater detail on another chart.
  - Continues until no subprocess can logically be broken down any further.
Decomposition of DFDs (Cont.)

- *Primitive DFD* is the lowest level of a DFD.
- *Level-1 diagram* results from decomposition of Level-0 diagram.
- *Level-n diagram* is a DFD diagram that is the result of *n* nested decompositions from a process on a level-0 diagram.
Level-1 DFD

Processes are labeled 4.1, 4.2, etc. These can be further decomposed in more primitive (lower-level) DFDs if necessary.

**FIGURE 7-8**
Level-1 diagram showing the decomposition of Process 4.0 from the level-0 diagram for Hoosier Burger’s food-ordering system

Level-1 DFD shows the sub-processes of one of the processes in the Level-0 DFD.

This is a Level-1 DFD for Process 4.0.
Level-\( n \) DFD

**FIGURE 7-9**
Level-2 diagram showing the decomposition of Process 4.3 from the level-1 diagram for Process 4.0 for Hoosier Burger’s food-ordering system.

Processes are labeled 4.3.1, 4.3.2, etc. If this is the lowest level of the hierarchy, it is called a *primitive DFD*.

Level-\( n \) DFD shows the sub-processes of one of the processes in the Level \( n-1 \) DFD.

This is a Level-2 DFD for Process 4.3.
Balancing DFDs

- **Conservation Principle**: conserve inputs and outputs to a process at the next level of decomposition.

- **Balancing**: conservation of inputs and outputs to a data flow diagram process when that process is decomposed to a lower level.
Balancing DFDs (Cont.)

Balanced means:

- Number of inputs to lower level DFD equals number of inputs to associated process of higher-level DFD
- Number of outputs to lower level DFD equals number of outputs to associated process of higher-level DFD
Balancing DFDs (Cont.)

FIGURE 7-10 An unbalanced set of data flow diagrams

(a) Context diagram

1 input
1 output

(b) Level-0 diagram

2 inputs
1 output

This is unbalanced because the process of the context diagram has only one input but the Level-0 diagram has two inputs.
Balancing DFDs (Cont.)

- **Data flow splitting** is when a composite data flow at a higher level is split and different parts go to different processes in the lower level DFD.

- The DFD remains balanced because the same data is involved, but split into two parts.
Balancing DFDs (Cont.)

**FIGURE 7-11**
Example of data flow splitting

(a) Composite data flow

(b) Disaggregated data flows
Balancing DFDs: More DFD Rules

TABLE 7-3  Advanced Rules Governing Data Flow Diagramming

Q. A composite data flow on one level can be split into component data flows at the next level, but no new data can be added and all data in the composite must be accounted for in one or more subflows.

R. The inputs to a process must be sufficient to produce the outputs (including data placed in data stores) from the process. Thus, all outputs can be produced, and all data in inputs move somewhere: to another process or to a data store outside the process or onto a more detailed DFD showing a decomposition of that process.

S. At the lowest level of DFDs, new data flows may be added to represent data that are transmitted under exceptional conditions; these data flows typically represent error messages (e.g., “Customer not known; do you want to create a new customer?”) or confirmation notices (e.g., “Do you want to delete this record?”).

T. To avoid having data flow lines cross each other, you may repeat data stores or sources/sinks on a DFD. Use an additional symbol, like a double line on the middle vertical line of a data store symbol or a diagonal line in a corner of a sink/source square, to indicate a repeated symbol.

(Source: Adapted from Celko, 1987.)
Four Different Types of DFDs

- **Current Physical**
  - Process labels identify technology (people or systems) used to process the data.
  - Data flows and data stores identify actual name of the physical media.

- **Current Logical**
  - Physical aspects of system are removed as much as possible.
  - Current system is reduced to data and processes that transform them.
Four Different Types of DFDs (Cont.)

- **New Logical**
  - Includes additional functions.
  - Obsolete functions are removed.
  - Inefficient data flows are reorganized.

- **New Physical**
  - Represents the physical implementation of the new system.
Guidelines for Drawing DFDs

- Completeness
  - DFD must include all components necessary for system.
  - Each component must be fully described in the project dictionary or CASE repository.

- Consistency
  - The extent to which information contained on one level of a set of nested DFDs is also included on other levels.
Guidelines for Drawing DFDs (Cont.)

- **Timing**
  - Time is not represented well on DFDs.
  - Best to draw DFDs as if the system has never started and will never stop.

- **Iterative Development**
  - Analyst should expect to redraw diagram several times before reaching the closest approximation to the system being modeled.
Guidelines for Drawing DFDs (Cont.)

- **Primitive DFDs**
  - Lowest logical level of decomposition
  - Decision has to be made when to stop decomposition
Guidelines for Drawing DFDs (Cont.)

- Rules for stopping decomposition
  - When each process has been reduced to a single decision, calculation or database operation
  - When each data store represents data about a single entity
Guidelines for Drawing DFDs (Cont.)

- Rules for stopping decomposition, cont.
  - When the system user does not care to see any more detail
  - When every data flow does not need to be split further to show that data are handled in various ways
Guidelines for Drawing DFDs (Cont.)

- Rules for stopping decomposition, cont.
  - When you believe that you have shown each business form or transaction, online display and report as a single data flow
  - When you believe that there is a separate process for each choice on all lowest-level menu options
Using DFDs as Analysis Tools

- **Gap Analysis** is the process of discovering discrepancies between two or more sets of data flow diagrams or discrepancies within a single DFD.

- Inefficiencies in a system can often be identified through DFDs.
Using DFDs in BPR

FIGURE 7-16
IBM Credit Corporation’s primary work process before BPR (Source: Based on Hammer and Champy, 1993.)
Using DFDs in BPR (Cont.)

FIGURE 7-17
IBM Credit Corporation’s primary work process after BPR
(Source: Based on Hammer and Champy, 1993.)
Modeling Logic with Decision Tables

- **Decision table**: a matrix representation of the logic of a decision which specifies the possible conditions for the decision and the resulting actions.

- Best used for complicated decision logic.
### FIGURE 7-18
Complete decision table for payroll system example

<table>
<thead>
<tr>
<th>Condition Stubs</th>
<th>Conditions/Courses of Action</th>
<th>Rules</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Employee type</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>Hours worked</td>
<td>S</td>
</tr>
<tr>
<td></td>
<td></td>
<td>H</td>
</tr>
<tr>
<td></td>
<td></td>
<td>S</td>
</tr>
<tr>
<td></td>
<td></td>
<td>H</td>
</tr>
<tr>
<td></td>
<td></td>
<td>S</td>
</tr>
<tr>
<td></td>
<td></td>
<td>H</td>
</tr>
<tr>
<td>Action Stubs</td>
<td>Pay base salary</td>
<td>&lt;40</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;40</td>
</tr>
<tr>
<td></td>
<td></td>
<td>40</td>
</tr>
<tr>
<td></td>
<td></td>
<td>40</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&gt;40</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&gt;40</td>
</tr>
<tr>
<td></td>
<td>Calculate hourly wage</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>Calculate overtime</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Produce absence report</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- S: Standard
- H: Hourly
Modeling Logic with Decision Tables (Cont.)

- **Condition stubs**: that part of a decision table that lists the conditions relevant to the decision

- **Action stubs**: that part of a decision table that lists the actions that result for a given set of conditions
Modeling Logic with Decision Tables (Cont.)

- **Rules**: that part of a decision table that specifies which actions are to be followed for a given set of conditions

- **Indifferent condition**: in a decision table, a condition whose value does not affect which actions are taken for two or more rules
Modeling Logic with Decision Tables (Cont.)

- Procedure for Creating Decision Tables
  - Name the condition and the values that each condition can assume.
  - Name all possible actions that can occur.
  - List all possible rules.
  - Define the actions for each rule.
  - Simplify the table.
Modeling Logic with Decision Tables (Cont.)

**FIGURE 7-19**
Reduced decision table for payroll system example

<table>
<thead>
<tr>
<th>Conditions/Courses of Action</th>
<th>Rules</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Rule 1</td>
</tr>
<tr>
<td>Employee type</td>
<td>S</td>
</tr>
<tr>
<td>Hours worked</td>
<td>–</td>
</tr>
<tr>
<td>Pay base salary</td>
<td>X</td>
</tr>
<tr>
<td>Calculate hourly wage</td>
<td>X</td>
</tr>
<tr>
<td>Calculate overtime</td>
<td></td>
</tr>
<tr>
<td>Produce Absence Report</td>
<td>X</td>
</tr>
</tbody>
</table>
Electronic Commerce Application: Process Modeling using Data Flow Diagrams

- Process modeling for Pine Valley Furniture’s Webstore
  - Completed JAD session.
  - Began translating the Webstore system structure into data flow diagrams.
    - Identified six high-level processes.
Electronic Commerce Application: Process Modeling using Data Flow Diagrams (Cont.)

**TABLE 7-4 System Structure of the WebStore and Corresponding Level-0 Processes**

<table>
<thead>
<tr>
<th>WebStore System</th>
<th>Processes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Main Page</td>
<td>Information Display (minor/no processes)</td>
</tr>
<tr>
<td>• Product Line (Catalog)</td>
<td>1.0 Browse Catalog</td>
</tr>
<tr>
<td>✓ Desks</td>
<td>2.0 Select Item for Purchase</td>
</tr>
<tr>
<td>✓ Chairs</td>
<td></td>
</tr>
<tr>
<td>✓ Tables</td>
<td></td>
</tr>
<tr>
<td>✓ File Cabinets</td>
<td></td>
</tr>
<tr>
<td>• Shopping Cart</td>
<td>3.0 Display Shopping Cart</td>
</tr>
<tr>
<td>• Checkout</td>
<td>4.0 Check Out Process Order</td>
</tr>
<tr>
<td>• Account Profile</td>
<td>5.0 Add/Modify Account Profile</td>
</tr>
<tr>
<td>• Order Status/History</td>
<td>6.0 Order Status Request</td>
</tr>
<tr>
<td>• Customer Comments</td>
<td>Information Display (minor/no processes)</td>
</tr>
<tr>
<td>❑ Company Information</td>
<td></td>
</tr>
<tr>
<td>❑ Feedback</td>
<td></td>
</tr>
<tr>
<td>❑ Contact Information</td>
<td></td>
</tr>
</tbody>
</table>
Electronic Commerce Application: Process Modeling using Data Flow Diagrams

FIGURE 7-22
Level-0 data flow diagram for the WebStore
Chapter 5
Structuring System Data Requirements
Conceptual Data Modeling

- **Conceptual data modeling**: a detailed model that captures the overall structure of data in an organization
  - Independent of any database management system (DBMS) or other implementation considerations
Conceptual Data Modeling (Cont.)

FIGURE 8-1
Systems development life cycle with analysis phase highlighted
The Conceptual Data Modeling Process

- Develop a data model for the current system.
- Develop a new conceptual data model that includes all requirements of the new system.
- In the design stage, the conceptual data model is translated into a physical design.
- Project repository links all design and data modeling steps performed during SDLC.
Conceptual Data Modeling (Cont.)

- Enterprise-wide data model (E-R with only entities)
- Conceptual data model (E-R with only entities for specific project)

FIGURE 8-2
Relationship between data modeling and the systems development life cycle
Deliverables and Outcome

- Primary deliverable is an entity-relationship (E-R) diagram or class diagram.
- As many as 4 E-R or class diagrams are produced and analyzed:
  - E-R diagram that covers data needed in the project’s application
  - E-R diagram for the application being replaced
  - E-R diagram for the whole database from which the new application’s data are extracted
  - E-R diagram for the whole database from which data for the application system being replaced is drawn
FIGURE 8-3
Sample conceptual data model
Deliverables and Outcome (cont.)

- Second deliverable is a set of entries about data objects to be stored in repository or project dictionary.
  - Repository links data, process, and logic models of an information system
  - Data elements included in the DFD must appear in the data model and vice versa
  - Each data store in a process model must relate to business objects represented in the data model
Two perspectives on data modeling:

- *Top-down approach* for a data model is derived from an intimate understanding of the business.

- *Bottom-up approach* for a data model is derived by reviewing specifications and business documents.
Gathering Information for Conceptual Data Modeling (Cont.)

- Requirements Determination Questions for Data Modeling:
  - What are subjects/objects of the business?
    - Data entities and descriptions
  - What unique characteristics distinguish between subjects/objects of the same type?
    - Primary keys
Gathering Information for Conceptual Data Modeling (Cont.)

- What characteristics describe each subject/object?
  - Attributes and secondary keys
- How do you use the data?
  - Security controls and user access privileges
- Over what period of time are you interested in the data?
  - Cardinality and time dimensions
Gathering Information for Conceptual Data Modeling (Cont.)

- Are all instances of each object the same?
  - Supertypes, subtypes, and aggregations

- What events occur that imply associations between objects?
  - Relationships and cardinalities

- Are there special circumstances that affect the way events are handled?
  - Integrity rules, cardinalities, time dimensions
Introduction to Entity-Relationship (E-R) Modeling

- **Entity-Relationship data model (E-R model):** a detailed, logical representation of the entities, associations and data elements for an organization or business area

- **Entity-relationship diagram (E-R diagram):** a graphical representation of an E-R model
Introduction to Entity-Relationship (E-R) Modeling

The E-R model is expressed in terms of:

- Data entities in the business environment.
- Relationships or associations among those entities.
- Attributes or properties of both the entities and their relationships.
Introduction to E-R Modeling (Cont.)

- **Entity**: a person, place, object, event or concept in the user environment about which data is to be maintained

- **Entity type**: collection of entities that share common properties or characteristics

- **Entity instance**: single occurrence of an entity type
FIGURE 8-5 Basic E-R notation

Entity Types
- Strong
- Weak

Attributes
- ENTITY NAME
  - Identifier
  - Partial identifier
  - Optional
  - [Derived]
  - {Multivalued}
  - Composite( , , )

Relationship Degrees
- Unary
- Binary
- Ternary

Relationship Cardinality
- Mandatory One
- Mandatory Many
- Optional One
- Optional Many

ترابطي
Composite
associate
unary
ternary
مandatory
إلزامي
أحادي
ثلاثي
إلزامي
Naming and Defining Entity Types

- An entity type name should be:
  - A singular noun.
  - Descriptive and specific to the organization.
  - Concise.

- Event entity type should be named for the result of the event, not the activity or process of the event.
An entity type definition should:

- Include a statement of *what the unique characteristic(s) is (are) for each instance.*
- Make clear *what entity instances are included and not included* in the entity type.
- Often include a description of *when an instance of the entity type is created or deleted.*
For some entity types the definition must specify:

- When an instance might change into an instance of another entity type.
- What history is to be kept about entity instances.
Attributes

- **Attribute** (السمات): a named property or characteristic of an entity that is of interest to the organization

  - Naming an attribute: i.e. Vehicle_ID
  - Place its name inside the rectangle for the associated entity in the E-R diagram.
Naming and Defining Attributes

- An attribute name is a *noun* and should be *unique*.
- To make an attribute name unique and for clarity, *each attribute name should follow a standard format*.
- *Similar attributes of different entity types should use similar but distinguishing names*. 
An attribute definition:

- States what the attribute is and possibly why it is important.
- Should make it clear what is included and what is not included.
- Contain any aliases or alternative names.
- States the source of values for the attribute.
An attribute definition should indicate:

- If a value for the attribute is required or optional.
- If a value for the attribute may change.
- Any relationships that attribute has with other attributes.
Candidate Keys and Identifiers.

- **Candidate key** مفتاح مرشح: an attribute (or combination of attributes) that uniquely identifies each instance of an entity type.

- **Identifier**: a candidate key that has been selected as the unique, identifying characteristic for an entity type.
Candidate Keys and Identifiers (Cont.)

- Selection rules for an identifier
  - Choose a candidate key that will not change its value.
  - Choose a candidate key that will never be null.
  - Avoid using intelligent keys.
  - Consider substituting single value surrogate keys for large composite keys.
Other Attribute Types

- **Multivalued attribute**: an attribute that may take on more than one value for each entity instance

- **Repeating group**: a set of two or more multivalued attributes that are logically related
FIGURE 8-8
Multivalued attributes and repeating groups

(a) Multivalued attribute skill

(b) Repeating group of dependent data

(c) Weak entity for dependent data
Other Attribute Types

- **Required attribute**: an attribute that must have a value for every entity instance
- **Optional attribute**: an attribute that may not have a value for every entity instance
- **Composite attribute**: an attribute that has meaningful component parts
- **Derived attribute**: an attribute whose value can be computed from related attribute values
FIGURE 8-11
Examples of relationships of different degrees

Unary relationships

- One-to-one:
  - PERSON
  - EMPLOYEE
  - TEAM

Binary relationships

- One-to-one:
  - EMPLOYEE - PARKING SPACE
  - STUDENT - COURSE

- Many-to-many:
  - PART - VENDOR - WAREHOUSE

Ternary relationships

- One-to-many:
  - PRODUCT LINE - PRODUCT

- Contains:
  - PRODUCT LINE - PRODUCT
Relationships

- **Relationship**: an association between the instances of one or more entity types that is of interest to the organization

- **Degree**: the number of entity types that participate in a relationship
Conceptual Data Modeling and the E-R Model

- **Unary relationship**: a relationship between the instances of one entity type
  - Also called a *recursive relationship*

- **Binary relationship**: a relationship between instances of two entity types
  - Most common type of relationship encountered in data modeling

- **Ternary relationship**: a simultaneous relationship among instances of three entity types
Cardinalities in Relationships

- **Cardinality**: the number of instances of entity B that can (or must) be associated with each instance of entity A

- Minimum Cardinality
  - The minimum number of instances of entity B that may be associated with each instance of entity A

- Maximum Cardinality
  - The maximum number of instances of entity B that may be associated with each instance of entity A
Cardinalities in Relationships (Cont.)

- Mandatory vs. Optional Cardinalities
  - Specifies whether an instance must exist or can be absent in the relationship.
FIGURE 8-14 Examples of cardinality constraints

(a) Mandatory cardinalities

(b) One optional, one mandatory cardinality

(c) Optional cardinalities
Naming and Defining Relationships

- A relationship name is a verb phrase; avoid vague names.
- A relationship definition:
  - Explains what action is to be taken and possibly why it is important.
  - Gives examples to clarify the action.
A relationship definition should:

- Explain any optional participation.
- Explain the reason for any explicit maximum cardinality other than many.
- Explain any restrictions on participation in the relationship.
- Explain the extent of history that is kept in the relationship.
- Explain whether an entity instance involved in a relationship instance can transfer participation to another relationship instance.
Associative Entities

**Associative Entity**: an entity type that associates the instances of one or more entity types and contains attributes that are peculiar to the relationship between those entity instances

- Sometimes called a gerund

The data modeler chooses to model the relationship as an entity type.
FIGURE 8-15 An associative entity

An associative entity using Microsoft Visio®
The purpose of E-R diagramming is to capture the richest possible understanding of the meaning of the data necessary for an information system or organization.
Representing Supertypes and Subtypes

- **Subtype**: a subgrouping of the entities in an entity type
  - Is meaningful to the organization
  - Shares common attributes or relationships distinct from other subgroupings

- **Supertype**: a generic entity type that has a relationship with one or more subtypes
Representing Supertypes and Subtypes (Cont.)

- Business Rules for Supertype/subtype Relationships:
  - **Total specialization** specifies that each entity instance of the supertype must be a member of some subtype in the relationship.
  - **Partial specialization** specifies that an entity instance of the supertype does not have to belong to any subtype, and may or may not be an instance of one of the subtypes.
Representing Supertypes and Subtypes (Cont.)

- **Disjoint rule** specifies that if an entity instance of the supertype is a member of one subtype, it cannot simultaneously be a member of any other subtype.

- **Overlap rule** specifies that an entity instance can simultaneously be a member of two (or more) subtypes.
FIGURE 8-19
Example of supertype/subtype hierarchy
Business Rules

- **Business rules**: specifications that preserve the integrity of the logical data model
  - Captured during requirements determination
  - Stored in CASE repository as they are documented
Four basic types of business rules are:

- *Entity integrity*: unique, non-null identifiers
- *Referential integrity constraints*: rules governing relationships between entity types
- *Domains*: constraints on valid values for attributes
- *Triggering operations*: other business rules that protect the validity of attribute values
Domains

- **Domain**: the set of all data types and values that an attribute can assume

- Several advantages
  - Verify that the values for an attribute are valid
  - Ensure that various data manipulation operations are logical
  - Help conserve effort in describing attribute characteristics
Triggering Operations

**Trigger**: an assertion or rule that governs the validity of data manipulation operations such as insert, update and delete
Triggering Operations

- Includes the following components:
  - **User rule**: statement of the business rule to be enforced by the trigger
  - **Event**: data manipulation operation that initiates the operation
  - **Entity Name**: name of entity being accessed or modified
  - **Condition**: condition that causes the operation to be triggered
  - **Action**: action taken when the operation is triggered
Chapter 6
Designing Forms and Reports
Designing Forms and Reports

FIGURE 10-1
Systems development life cycle with logical design phase highlighted
Designing Forms and Reports (Cont.)

- **Form**: a business document that contains some predefined data and may include some areas where additional data are to be filled in.
  - An instance of a form is typically based on one database record.
Designing Forms and Reports (Cont.)

- **Report**: a business document that contains only predefined data
  - It is a passive document used solely for reading or viewing data.
- A report typically contains data from many unrelated records or transactions.
Common Types of Reports:

- **Scheduled**: produced at predefined time intervals for routine information needs
- **Key-indicator**: provides summary of critical information on regular basis
- **Exception**: highlights data outside of normal operating ranges
- **Drill-down**: provides details behind summary of key-indicator or exception reports
- **Ad-hoc**: responds to unplanned requests for non-routine information needs
The Process of Designing Forms and Reports

- Is user-focused activity.
- Follows a prototyping approach.
- First steps are to gain an understanding of the intended user and task objectives by collecting initial requirements during requirements determination.
The Process of Designing Forms and Reports

Requirements determination:

- Who will use the form or report?
- What is the purpose of the form or report?
- When is the report needed or used?
- Where does the form or report need to be delivered and used?
- How many people need to use or view the form or report?
The Process of Designing Forms and Reports (Cont.)

- Prototyping
  - Initial prototype is designed from requirements.
  - Users review prototype design and either accept the design or request changes.
  - If changes are requested, the construction-evaluation-refinement cycle is repeated until the design is accepted.
A coding sheet is an “old” tool for designing forms and reports, usually associated with text-based forms and reports for mainframe applications. Visual Basic and other development tools provide computer aided GUI form and report generation.
The Process of Designing Forms and Reports (Cont.)

FIGURE 10-2
The layout of a data input form using a coding sheet

<table>
<thead>
<tr>
<th>SYSTEM</th>
<th>Customer Information Entry</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROGRAM</td>
<td>STAN</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>CUSTOMER INFORMATION</td>
<td></td>
</tr>
<tr>
<td>CUSTOMER NUMBER:</td>
<td></td>
</tr>
<tr>
<td>NAME:</td>
<td></td>
</tr>
<tr>
<td>ADDRESS:</td>
<td></td>
</tr>
<tr>
<td>CITY:</td>
<td></td>
</tr>
<tr>
<td>STATE:</td>
<td></td>
</tr>
<tr>
<td>ZIP:</td>
<td></td>
</tr>
</tbody>
</table>
The Process of Designing Forms and Reports (Cont.)

FIGURE 10-3
A data input screen designed in Microsoft’s Visual Basic .NET
Deliverables and Outcomes

- Design specifications are the major deliverables and inputs to the system implementation phase.
Deliverables and Outcomes (Cont.)

- Design specifications have three sections:
  - *Narrative overview*: characterizes users, tasks, system, and environmental factors
  - *Sample design*: image of the form (from coding sheet or form building development tool)
  - *Testing and usability assessment*: measuring test/usability results (consistency, sufficiency, accuracy, etc.)
Formatting Forms and Reports

- **Meaningful titles** — use clear, specific, version information, and current date
- **Meaningful information** — include only necessary information, with no need to modify
Formatting Forms and Reports (Cont.)

- *Balanced layout* — use adequate spacing, margins, and clear labels
- *Easy navigation system* — show how to move forward and backward, and where you are currently
FIGURE 10-5
Contrasting customer information forms (Pine Valley Furniture)

(a) Poorly designed form
FIGURE 10-5 (continued)

(b) Improved design for form

Formatitng Forms and Reports (Cont.)
Highlighting Information

- Notify users of errors in data entry or processing.
- Provide warnings regarding possible problems.
- Draw attention to keywords, commands, high-priority messages, unusual data values.
Highlighting can include use of

- upper case
- bold
- italics
- underlining
- boxing
- size and color differences
- all capital letters
- blinking
- reverse video
- audible tones
- intensity differences
- offsetting nonstandard information
Highlighting Information (Cont.)

FIGURE 10-6
Customer account status display using various highlighting techniques

(Pine Valley Furniture)
Color vs. No Color

- Benefits of Using Color
  - Soothes or strikes the eye.
  - Accents an uninteresting display.
  - Facilitates subtle discriminations in complex displays.
  - Emphasizes the logical organization of information.
  - Draws attention to warnings.
  - Evokes more emotional reactions.
Problems from Using Color

- Color pairings may wash out or cause problems for some users.
- Resolution may degrade with different displays.
- Color fidelity may degrade on different displays.
- Printing or conversion to other media may not easily translate.
Displaying Text

- **Case**: mixed upper and lower case, use conventional punctuation
- **Spacing**: double spacing if possible, otherwise blank lines between paragraphs
- **Justification**: left justify text, ragged right margins
- **Hyphenation**: no hyphenated words between lines
- **Abbreviations**: only when widely understood and significantly shorter than full text
Displaying Text (Cont.)

FIGURE 10-7
Contrasting the display of textual help information

(a) Poorly designed help screen with many violations of the general guidelines for displaying text
Displaying Text (Cont.)

FIGURE 10-7 (continued)

(b) An improved design for a help screen
Designing Tables and Lists

Labels

- All columns and rows should have meaningful labels.
- Labels should be separated from other information by using highlighting.
- Redisplay labels when the data extend beyond a single screen or page.
Designing Tables and Lists (Cont.)

- Formatting columns, rows and text:
  - Sort in a meaningful order.
  - Place a blank line between every five rows in long columns.
  - Similar information displayed in multiple columns should be sorted vertically.
  - Columns should have at least two spaces between them.
Designing Tables and Lists (Cont.)

- Allow white space on printed reports for user to write notes.
- Use a single typeface, except for emphasis.
- Use same family of typefaces within and across displays and reports.
- Avoid overly fancy fonts.
Designing Tables and Lists (Cont.)

- Formatting numeric, textual and alphanumERIC data:
  - Right justify numeric data and align columns by decimal points or other delimiter.
  - Left justify textual data. Use short line length, usually 30 to 40 characters per line.
  - Break long sequences of alphanumERIC data into small groups of three to four characters each.
Designing Tables and Lists (Cont.)

FIGURE 10-8
Contrasting the display of tables and lists (Pine Valley Furniture)

(a) Poorly designed form
FIGURE 10-8 (continued)
(b) Improved design for form

Clear and separate column labels for each data type

Numeric data are right justified
Use tables for reading individual data values.

Use graphs for:
- Providing quick summary.
- Displaying trends over time.
- Comparing points and patterns of variables.
- Forecasting activity.
- Simple reporting of vast quantities of information.
Designing Tables and Lists (Cont.)

![Table Illustrating Design Guidelines]

**FIGURE 10-9**
Tabular report illustrating numerous design guidelines

(Pine Valley Furniture)
Designing Tables and Lists (Cont.)

FIGURE 10-10
Graphs for comparison

(a) Line graph

(b) Bar graph
Assessing Usability

Objective for designing forms, reports and all human-computer interactions is usability.

There are three characteristics:

- **Speed** — Can you complete a task efficiently?
- **Accuracy** — Does the output provide what you expect?
- **Satisfaction** — Do you like using the output?
Assessing Usability (Cont.)

- **Usability**: an overall evaluation of how a system performs in supporting a particular user for a particular task.
Usability Success Factors

- **Consistency** — of terminology, formatting, titles, navigation, response time
- **Efficiency** — minimize required user actions.
- **Ease** — self-explanatory outputs and labels.
- **Format** — appropriate display of data and symbols.
- **Flexibility** — maximize user options for data input according to preference.
Usability Success Factors (Cont.)

Characteristics for consideration:

- **User**: experience, skills, motivation, education, personality
- **Task**: time pressure, cost of errors, work durations
- **System**: platform
- **Environment**: social and physical issues
Measures of Usability

- Time to learn
- Speed of performance
- Rate of errors
- Retention over time
- Subjective satisfaction
- Layout of information should be consistent, both within and across applications
Measures of Usability (Cont.)

- Layout of information should be consistent both within and across applications, whether information is delivered on screen display or on a hard-copy report.
Designing Forms and Reports for Pine Valley Furniture Web Store

- General guidelines for rapid deployment of Internet Web sites have resulted

- Three possible solutions to the problem:
  - Make it possible to design reasonably usable sites without having UI experience.
  - Train more people in good Web design.
  - Live with poorly designed sites that are hard to use.
Designing Forms and Reports at Pine Valley Furniture

- PVF established the following guidelines:
  - Use lightweight graphics.
  - Establish forms and data integrity rules.
  - Use template-based HTML.
Lightweight Graphics

- **Lightweight Graphics**: the use of small, simple images to allow a Web page to more quickly be displayed
  - Quick image download
  - Quick feedback from the Web site will help to keep customers at the PVF WebStore longer
Forms and Data Integrity Rules

- All forms that request information should be clearly labeled and provide adequate room for input.
- Specific fields requiring specific information must provide a clear example.
- Must designate which fields are optional, required, and which have a range of values.
Template-Based HTML

- **Template-based HTML**: templates to display and process common attributes of higher-level, more abstract items
  - Creates an interface that is very easy to maintain
  - Advantageous to have a “few” templates that could be used for entire product line
  - Not every product needs its own page
Chapter 7
Designing Interfaces and Dialogues
Designing Interfaces and Dialogues

- User-focused activity
- Prototyping methodology of iteratively:
  - Collecting information
  - Constructing a prototype
  - Assessing usability
  - Making refinements
- Must answer the who, what, where, and how questions
Designing Interfaces and Dialogues (Cont.)

FIGURE 11-1
Systems development life cycle (SDLC)
Deliverables and Outcomes

- Creation of a design specification
  - A typical interface/dialogue design specification is similar to form design, but includes multiple forms and dialogue sequence specifications.
Deliverables and Outcomes (Cont.)

- The specification includes:
  - Narrative overview
  - Sample design
  - Testing and usability assessment
  - Dialogue sequence

*Dialogue sequence* is the ways a user can move from one display to another.
Interaction Methods and Devices

- **Interface**: a method by which users interact with an information system

- All human-computer interfaces must:
  - have an interaction style, and
  - use some hardware device(s) for supporting this interaction.
Methods of Interacting

- Command line
  - Includes keyboard shortcuts and function keys
- Menu
- Form
- Object-based
- Natural language
Command Language Interaction

- **Command language interaction**: a human-computer interaction method whereby users enter explicit statements into a system to invoke operations

- Example from MS DOS:
  - COPY C:PAPER.DOC A:PAPER.DOC
  - Command copies a file from C: drive to A: drive
Menu Interaction

- **Menu interaction**: a human-computer interaction method in which a list of system options is provided and a specific command is invoked by user selection of a menu option.

- **Pop-up menu**: a menu-positioning method that places a menu near the current cursor position.
Menu Interaction (Cont.)

- **Drop-down menu** is a menu-positioning method that places the access point of the menu near the top line of the display.
  - When accessed, menus open by dropping down onto the display.
  - Visual editing tools help designers construct menus.
Guidelines for Menu Design

- **Wording** — meaningful titles, clear command verbs, mixed upper/lower case
- **Organization** — consistent organizing principle
- **Length** — all choices fit within screen length
- **Selection** — consistent, clear and easy selection methods
- **Highlighting** — only for selected options or unavailable options
Menu Interaction (Cont.)

FIGURE 11-8
Menu building with Microsoft Visual Basic .NET
Form Interaction

- **Form interaction**: a highly intuitive human-computer interaction method whereby data fields are formatted in a manner similar to paper-based forms.
  - Allows users to fill in the blanks when working with a system.
Form Interaction (Cont.)

FIGURE 11-9
Example of form interaction from the Google Advanced Search Engine (Source: Google.)
Object-Based Interaction

- **Object-based interaction**: a human-computer interaction method in which symbols are used to represent commands or functions

- **Icons**: graphical pictures that represent specific functions within a system
  - Use little screen space and are easily understood by users
Object-Based Interaction (Cont.)

FIGURE 11-10
Object-based (icon) interface from Microsoft Visual Basic .NET
Natural Language Interaction

- **Natural language interaction**: a human-computer interaction method whereby inputs to and outputs from a computer-based application are in a conventional spoken language such as English
- Based on research in artificial intelligence.
- Current implementations are tedious and difficult to work with, not as viable as other interaction methods.
Usability Problems with Hardware Devices

- Visual Blocking
  - touch screen, light pen
- User Fatigue
  - touch screen, light pen
- Movement Scaling
  - keyboard, mouse, joystick, trackball, graphics tablet, voice
- Durability
  - trackball, touch screen
- Adequate Feedback
  - keyboard, mouse, joystick, trackball, graphics tablet, voice
- Speed
  - keyboard
- Pointing Accuracy
  - joystick, touch screen, light pen, voice
Natural Language Interaction (Cont.)

- Usability problems with hardware devices:
  - Visual Blocking
    - touch screen, light pen
  - User Fatigue
    - touch screen, light pen
Natural Language Interaction (Cont.)

- **Movement Scaling**
  - keyboard, mouse, joystick, trackball, graphics tablet, voice

- **Durability**
  - trackball, touch screen
Natural Language Interaction (Cont.)

- Adequate Feedback
  - keyboard, mouse, joystick, trackball, graphics tablet, voice

- Speed
  - keyboard

- Pointing Accuracy
  - joystick, touch screen, light pen, voice
Natural Language Interaction (Cont.)

<table>
<thead>
<tr>
<th>Task</th>
<th>Most Accurate</th>
<th>Shortest Positioning</th>
<th>Most Preferred</th>
</tr>
</thead>
<tbody>
<tr>
<td>Target Selection</td>
<td>trackball, graphics tablet, mouse, joystick</td>
<td>touch screen, light pen, mouse, graphics tablet, trackball</td>
<td>touch screen, light pen</td>
</tr>
<tr>
<td>Text Selection</td>
<td>mouse</td>
<td>mouse</td>
<td>—</td>
</tr>
<tr>
<td>Data Entry</td>
<td>light pen</td>
<td>light pen</td>
<td>—</td>
</tr>
<tr>
<td>Cursor Positioning</td>
<td>—</td>
<td>light pen</td>
<td>—</td>
</tr>
<tr>
<td>Text Correction</td>
<td>light pen, cursor keys</td>
<td>light pen</td>
<td>light pen</td>
</tr>
<tr>
<td>Menu Selection</td>
<td>touch screen</td>
<td>—</td>
<td>keyboard, touch screen</td>
</tr>
</tbody>
</table>

(Source: Based on Blattner & Schultz, 1988.)

Key:
- Target Selection = moving the cursor to select a figure or item
- Text Selection = moving the cursor to select a block of text
- Data Entry = entering information of any type into a system
- Cursor Positioning = moving the cursor to a specific position
- Text Correction = moving the cursor to a location to make a text correction
- Menu Selection = activating a menu item
- — = no clear conclusion from the research
Designing Interfaces

Forms have several general areas in common:

- Header information
- Sequence and time-related information
- Instruction or formatting information
- Body or data details
- Totals or data summary
- Authorization or signatures
- Comments
Designing Interfaces (Cont.)

- Use standard formats similar to paper-based forms and reports.
- Use left-to-right, top-to-bottom navigation.
Designing Interfaces (Cont.)

- Flexibility and consistency:
  - Free movement between fields
  - No permanent data storage until the user requests
  - Each key and command assigned to one function
## Structuring Data Entry

<table>
<thead>
<tr>
<th>Entry</th>
<th>Never require data that are already online or that can be computed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Defaults</td>
<td>Always provide default values when appropriate</td>
</tr>
<tr>
<td>Units</td>
<td>Make clear the type of data units requested for entry</td>
</tr>
<tr>
<td>Replacement</td>
<td>Use character replacement when appropriate</td>
</tr>
<tr>
<td>Captioning</td>
<td>Always place a caption adjacent to fields</td>
</tr>
<tr>
<td>Format</td>
<td>Provide formatting examples</td>
</tr>
<tr>
<td>Justify</td>
<td>Automatically justify data entries</td>
</tr>
<tr>
<td>Help</td>
<td>Provide context-sensitive help when appropriate</td>
</tr>
</tbody>
</table>
Controlling Data Input

- Objective: Reduce data entry errors
- Common sources data entry errors in a field:
  - Appending: adding additional characters
  - Truncating: losing characters
  - Transcripting: entering invalid data
  - Transposing: reversing sequence of characters
Providing Feedback

Three types of system feedback:

- **Status information**: keep user informed of what’s going on, helpful when user has to wait for response
- **Prompting cues**: tell user when input is needed, and how to provide the input
- **Error or warning messages**: inform user that something is wrong, either with data entry or system operation
Providing Help

- Place yourself in user’s place when designing help.

- Guidelines for designing usable help:
  - **Simplicity** — Help messages should be short and to the point.
  - **Organize** — Information in help messages should be easily absorbed by users.
  - **Show** — It is useful to explicitly show users how to perform an operation.
Designing Dialogues

- **Dialogues**: the sequence of interaction between a user and a system

- Dialogue design involves:
  - Designing a dialogue sequence.
  - Building a prototype.
  - Assessing usability.
Designing the Dialogue Sequence

 Typical dialogue between user and Customer Information System:

- Request to view individual customer information.
- Specify the customer of interest.
- Select the year-to-date transaction summary display.
- Review the customer information.
- Leave system.
Designing the Dialogue Sequence (Cont.)

- **Dialogue diagramming**: a formal method for designing and representing human-computer dialogues using box and line diagrams
Designing the Dialogue Sequence (Cont.)

- Three sections of the box are used as:
  - *Top* contains a unique display reference number used by other displays for referencing it.
  - *Middle* contains the name or description of the display.
  - *Bottom* contains display reference numbers that can be accessed from the current display.
Designing the Dialogue Sequence (Cont.)

FIGURE 11-17
Sections of a dialogue diagramming box
Designing the Dialogue Sequence (Cont.)

- Dialogue diagrams depict the sequence, conditional branching, and repetition of dialogues.
Chapter 8
System Implementation
FIGURE 13-1
Systems development life cycle with the implementation phase highlighted
System Implementation

- Six major activities:
  - Coding
  - Testing
  - Installation
  - Documentation
  - Training
  - Support
System Implementation (Cont.)

- Purpose:
  - To convert final physical system specifications into working and reliable software
  - To document work that has been done
  - To provide help for current and future users
The Process of Coding, Testing, and Installation

- **Coding**
  - Physical design specifications are turned into working computer code.

- **Testing**
  - Tests are performed using various strategies.
  - Testing performed in parallel with coding.

- **Installation**
  - The current system is replaced by new system.
Chapter 9
Maintaining Information Systems
Maintaining Information Systems

FIGURE 14-1
Systems development life cycle
The Process of Maintaining Information Systems

- Process of returning to the beginning of the SDLC and repeating development steps focusing on system change until the change is implemented.
- Maintenance is the longest phase in the SDLC.
The Process of Maintaining Information Systems (Cont.)

Four major activities:
- Obtaining maintenance requests
- Transforming requests into changes
- Designing changes
- Implementing changes
Deliverables and Outcome

The maintenance phase of the SDLC is basically a subset of the activities of the entire development process.
Deliverables and Outcome (Cont.)

- The deliverables and outcomes from the process are the development of a new version of the software and new versions of all design documents created or modified during the maintenance effort.
FIGURE 14-3
Maintenance activities parallel those of the SDLC
Types of System Maintenance

- **Maintenance**: changes made to a system to fix or enhance its functionality
Types of System Maintenance (Cont.)

- **Corrective maintenance**: changes made to a system to repair flaws in its design, coding, or implementation
Types of System Maintenance (Cont.)

- Adaptive maintenance: changes made to a system to evolve its functionality to changing business needs or technologies
Types of System Maintenance (Cont.)

- **Perfective maintenance**: changes made to a system to add new features or to improve performance
1. What is information systems analysis and design?

2. What is systems thinking? How is it useful for thinking about computer-based information systems?

4. In what way are organizations systems?

5. What is prototyping?

6-LIST Phases of Project Management Process? and What is a systems analyst?

7-List Information System Components and Explain each one?

8-List and Explain Systems Development Tools?
9-What is computer-aided software engineering (CASE)?:

10) Briefly define and compare Gantt charts and network diagrams.

11) List and define the five major SDLC phases.

12) What is object-oriented analysis and design?

13) Briefly define and compare Gantt charts and network diagrams.

14) Define the **Context diagram**, **DFDs** of current physical system, **DFDs** of current logical system, **DFDs** of new logical system
**Problem 1:** Construct a network diagram using the following data. For each activity, identify its early start time, late start time, early finish time, late finish time, and slack. Identify the critical path.

<table>
<thead>
<tr>
<th>Activity</th>
<th>Optimistic Time</th>
<th>Pessimistic Time</th>
<th>Realistic Time</th>
<th>Expected Time</th>
<th>Preceding Task</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>4</td>
<td>8</td>
<td>6</td>
<td></td>
<td>--</td>
</tr>
<tr>
<td>B</td>
<td>7</td>
<td>11</td>
<td>9</td>
<td></td>
<td>A</td>
</tr>
<tr>
<td>C</td>
<td>3</td>
<td>7</td>
<td>5</td>
<td></td>
<td>A</td>
</tr>
<tr>
<td>D</td>
<td>9</td>
<td>13</td>
<td>11</td>
<td></td>
<td>A</td>
</tr>
<tr>
<td>E</td>
<td>6</td>
<td>10</td>
<td>8</td>
<td></td>
<td>B</td>
</tr>
<tr>
<td>F</td>
<td>3</td>
<td>9</td>
<td>6</td>
<td></td>
<td>C, D, E</td>
</tr>
</tbody>
</table>
Q1: Draw a context Data Flow Diagram and Level-0 diagram for a vehicle maintenance depot system. It contains a process that represents the system to model, in this case, the "vehicle maintenance depot system". It also shows the participants who will interact with the system. In this example, there are Customer and Mechanics. A Customer requests inspection and a Mechanics can Order Parts.

Q2 - Draw a context Data Flow Diagram and Level-0 diagram for a GRADING SYSTEM. It contains a process that represents the system to model, in this case, the "GRADING SYSTEM". It also shows the participants who will interact with the system.
In this example, there are STUDENT, STUDENT RECORDS SYSTEM, and PROFESSOR

Q3 Draw a context Data Flow Diagram and Level-0 diagram for a PATIENT INFORMATION SYSTEM. It contains a process that represents the system to model, in this case, the “PATIENT INFORMATION SYSTEM". It also shows the participants who will interact with the system. In this example, there are PATIENT, and DOCTOR.
Q1- The following Level-0 diagram represents an information system. Identify the errors in this diagram, and state the reason. (-- Pts.)
15) Differentiate between a form and a report.

16) Identify several general guidelines for the design of forms and reports.

17) Identify nine common errors that might occur when designing the interface and dialogues of Web sites.

18) Discuss the guidelines for drawing a DFD.

19) Briefly describe the data flow diagramming symbols. Provide one example of each.
20) Briefly discuss how DFDs can be used as analysis tools.

21) What is meant by DFD completeness? What is meant by DFD consistency?

22) What is gap analysis? Why is gap analysis useful?

23) What is process modeling? Identify three types of process models.

24) Identify six concrete rules for stopping the decomposition process.

25) Identify the deliverables for process modeling.