Chapter 13
Software Engineering
Objectives

• In this chapter you will:
  – Learn how software engineering is used to create applications
  – Learn some software engineering process models
  – Understand how a design document is used during software development
  – Review the steps for formulating a design document
Objectives (cont’d.)

• In this chapter you will (cont’d.):
  – Learn how Unified Modeling Language (UML) diagrams can be used as a blueprint for creating an application
  – See some pitfalls in developing software, and learn how to avoid them
  – Understand how teams are used in application development
Why You Need to Know About... Software Engineering

• Defining a project involves the need to define the project’s scope before beginning
  – Must find out exactly what’s required before starting

• Software engineering
  – Enables a student to design programs and communicate with clients and other team members
    • Essential elements of writing applications
What Is Software Engineering?

• Producing software applications
  – Requires source code and associated documentation
    • UML diagrams, screen prototypes, reports, software requirements, future development issues, and data needed to make programs operate correctly

• End user
  – Driving force behind software development
  – Someone or something needing the program to perform a function or meet a need
    • Determines program’s required functionality
  – Piece of machinery or task to be accomplished
What Is Software Engineering? (cont’d.)

• Major part of software engineering:
  – Designing, writing, and producing software applications
    • Based on end users’ needs
• End users’ needs might change over time
  – Need for application might even disappear
    • Makes application obsolete
• Constant need to communicate with end users
  – Make software applicable to their needs
Software Development Life Cycle

• Model describing application life
  – Including all stages in program development, testing, installation, and maintenance

• SDLC elements
  – Project feasibility
  – Software specifications
  – Software design and implementation
  – Software validation
  – Software evolution
Software Development Life Cycle (cont’d.)

• Software development process models
  – Waterfall
    • Fundamental processes in creating the program represented as phases
      – Output from each phase used as input for next phase
  – Build and fix (evolutionary)
    • Developer writes a program and continues to modify it until functional
  – Rapid prototyping
    • Uses tools allowing end users to work with prototypes of program screens and other interfaces
Software Development Life Cycle (cont’d.)

- Software development process models (cont’d.)
  - Incremental
    - Development with a series of software releases
  - Spiral
    - Cycles through the waterfall approach until all functionality is completed and delivered to end user
  - Agile
    - Tasks carried out in small increments with minimal planning
    - Two methods: scrum and extreme programming (XP)
Software Development Life Cycle (cont’d.)

- Each model varies in the steps needed to complete the development tasks
  - This chapter focuses on the waterfall model
    - Widely used
    - Around since 1970
    - Resembles process of building a house
Software Development Life Cycle (cont’d.)

• Waterfall model steps
  – Gather all requirements
  – Design the system and software
  – Build and implement the application
  – Test after the application is finished
  – Put into operation

• Software need not become obsolete
  – Modify to meet end users’ changing needs
Software Development Life Cycle (cont’d.)

Figure 13-1, The waterfall model of software development
Creating the Design Document

• Design document
  – Details all design issues for an application
    • Screen layouts, colors, reports, security, paths for files, online help, user documentation, future plans, and more

• Advantages of software development environment application development tool:
  – Prototypes screens and reports without writing source code
  – Serves as a blueprint for the system
  – Process is based on good communication with end users
Creating the Design Document (cont’d.)

Figure 13-2, The process of creating a design document

- learn the current system and needs
- create UML diagrams
- create a data dictionary
- design reports
- structure the application’s logical flow
- start building the prototype
- put all the pieces together
Step 1: Learn the Current System and Needs

• Initial task
  – Establish end user needs and goals
    • Ask client for samples of desired reports
  – Document information client provides
  – Come up with solutions or suggestions to address standard application factors
    • Security, colors, printing, etc.
  – Keep digging for information
  – Write project objectives or introduction, specifications, and requirements
### Figure 13-3, A design document includes objectives, specifications, and requirements

<table>
<thead>
<tr>
<th>1. Introduction</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>1.1. Purpose</strong></td>
</tr>
<tr>
<td>1.1.1. This document lists all software requirements for the creation and implementation of a Fantasy Basketball Web site. It defines the feasibility study, operational requirements, algorithms, databases, user interfaces, error systems, help systems, cost analysis, and supporting diagrams. The intended audience for this document is the end user or client, development team, project manager, and any other stakeholders in the system.</td>
</tr>
<tr>
<td><strong>1.2. Terms</strong></td>
</tr>
</tbody>
</table>
| • **League Owner**: The creator of the league  
• **Commissioner**: The person responsible for overseeing league actions  
• **Team Owner**: Any person who owns a team in the specified league  
• **Team**: Consists of 12 players, each playing in the position of guard, forward, or center  
• **User**: Any person who registers to play in a league of Fantasy Basketball |
| **1.3. Scope** |
| 1.3.1. The users of this product are the participants in the Fantasy Basketball game. Users can create their own league or participate in an established league. |
| **1.4. Overview** |
| 1.4.1. This product enables people to create leagues and organize teams by letting them manage and follow their teams through a basketball season. This product is Web based and requires a server, an Internet connection, and a Web browser. Every night, basketball statistics are downloaded to the server. These statistics are then updated throughout the league teams to determine a team's final score for a specific game. |
| **2. Specifications** |
| 2.1. ... |
| **3. System Requirements** |
| 3.1. ... |
Step 2: Create UML Diagrams

- Unified Modeling Language (UML)
  - Enables creation of diagrams (included in the blueprint) showing overall functionality of program
    - Provides client and developer communication
  - Visual modeling approach
    - Visual diagrams created before source code
  - Provides many types of diagrams
    - Explains different parts of a system
  - Tools can create UML diagrams
    - Microsoft Visio
Step 2: Create UML Diagrams (cont’d.)

Figure 13-4, Creating UML diagrams in Microsoft Visio
Step 2: Create UML Diagrams (cont’d.)

• Types of UML diagrams and their uses
  – Class
    • Shows how different object classes relate to each other
  – Object
    • Gives details of an object created from a class
  – Use case
    • Describes a system’s behavior from a user’s standpoint
  – State
    • Shows an object’s particular state at any given time
Step 2: Create UML Diagrams (cont’d.)

- Types of UML diagrams and their uses (cont’d.)
  - Sequence
    - Shows how one class communicates with another class by sending messages back and forth
  - Activity
    - Shows activities occurring within a use case or within an object’s behavior
  - Component
    - Shows how system components relate to each other
  - Deployment
    - Shows computer-based system’s physical architecture
Step 2: Create UML Diagrams (cont’d.)

- Each type of UML diagram serves a specific purpose in defining a system’s functionality
  - Client’s viewpoint
  - Describes object-oriented functionality
- Developers use these visual models as a blueprint when writing actual source code
Step 2: Create UML Diagrams (cont’d.)

Figure 13-5, Use case diagram for the music inventory application
Step 2: Create UML Diagrams (cont’d.)

Figure 13-6, Class diagram for the music inventory application
Step 2: Create UML Diagrams (cont’d.)

Figure 13-7, Sequence diagram for the music inventory application
Step 3: Create the Data Dictionary

- Data dictionary
  - Describes type of data used in the program
    - Table definitions, indexes, and other data relationships
  - Defines database structure
- Database administrator (DBA)
  - In charge of data dictionary
- If database is already in place, review it for accuracy
  - Compare with meeting notes, project’s objectives, specifications, and requirements
Step 3: Create the Data Dictionary (cont’d.)

• If database is needed:
  – Review reports provided by end user
  – Devise list of data tables to use in the application

• Data dictionary becomes a schematic
  – Describes type of data used in the program
Step 3: Create the Data Dictionary (cont’d.)

![Figure 13-8, Creating a data dictionary](image)

---

**Music Inventory Data Dictionary**

*Database is MIToeTappin written in Oracle 11g*

**Table: Artist**

<table>
<thead>
<tr>
<th>Indexes</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ByCode</td>
<td>Artist_CD</td>
</tr>
<tr>
<td>ByName</td>
<td>Artist_NM</td>
</tr>
</tbody>
</table>

*Use:* This table contains all the music artists.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARTIST_CD</td>
<td>Unique code identifying the record</td>
</tr>
<tr>
<td>ARTIST_NM</td>
<td>Artist name</td>
</tr>
</tbody>
</table>

**Table: Inventory**

<table>
<thead>
<tr>
<th>Indexes</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ByCode</td>
<td>Media_CD</td>
</tr>
<tr>
<td>ByType</td>
<td>Media_Type</td>
</tr>
</tbody>
</table>

*Use:* This table contains all the music items in the store’s inventory.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MEDIA_CD</td>
<td>Unique code identifying record</td>
</tr>
<tr>
<td>MEDIA_TYPE</td>
<td>Media type (CD, tape, album, and so on)</td>
</tr>
<tr>
<td>ON_HAND</td>
<td>Quantity on hand</td>
</tr>
<tr>
<td>MRP</td>
<td>Minimum reorder point</td>
</tr>
<tr>
<td>COST</td>
<td>Store’s cost</td>
</tr>
<tr>
<td>PRICE</td>
<td>Retail price</td>
</tr>
</tbody>
</table>

---
Step 4: Design Reports

- Gathering information
  - Meet with users
    - Allow users to help in report design
    - Review data dictionary
  - Integrated development environment (IDE)
    - Contains design tools and wizards, making application development easier
  - Sit down with the end user
    - Design reports interactively by using reporting tools
Step 4: Design Reports (cont’d.)

Music CD Catalog
ToeTappin’ Tunes
Sorted by Artist and Song Title

<table>
<thead>
<tr>
<th>Artist Name</th>
<th>Song Title</th>
<th>CD Title</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>COUNTING CROWS</strong></td>
<td>ANGELS OF THE SILENCES</td>
<td>RECOVERING THE SATELLITES</td>
</tr>
<tr>
<td></td>
<td>MR. JONES</td>
<td>AUGUST AND EVERYTHING AFTER</td>
</tr>
<tr>
<td></td>
<td>RECOVERING THE SATELLITES</td>
<td>RECOVERING THE SATELLITES</td>
</tr>
<tr>
<td></td>
<td>TIME AND TIME AGAIN</td>
<td>AUGUST AND EVERYTHING AFTER</td>
</tr>
<tr>
<td><strong>ERIC CLAPTON</strong></td>
<td>BLUES BEFORE SUNRISE</td>
<td>ERIC CLAPTON UNPLUGGED</td>
</tr>
<tr>
<td></td>
<td>HEY HEY</td>
<td>ERIC CLAPTON UNPLUGGED</td>
</tr>
<tr>
<td></td>
<td>HOOCHIE COOCHIE MAN</td>
<td>ERIC CLAPTON UNPLUGGED</td>
</tr>
<tr>
<td></td>
<td>LAYLA</td>
<td>ERIC CLAPTON UNPLUGGED</td>
</tr>
<tr>
<td></td>
<td>TEARS IN HEAVEN</td>
<td>ERIC CLAPTON UNPLUGGED</td>
</tr>
<tr>
<td><strong>HOWARD JONES</strong></td>
<td>CONDITIONING</td>
<td>HUMAN’S LIB</td>
</tr>
<tr>
<td></td>
<td>LOOK MAMA</td>
<td>BEST OF HOWARD JONES</td>
</tr>
<tr>
<td></td>
<td>NEW SONG</td>
<td>BEST OF HOWARD JONES</td>
</tr>
<tr>
<td></td>
<td>PEARL IN THE SHELL</td>
<td>BEST OF HOWARD JONES</td>
</tr>
<tr>
<td></td>
<td>WHAT IS LOVE?</td>
<td>BEST OF HOWARD JONES</td>
</tr>
<tr>
<td><strong>MANHATTAN TRANSFER</strong></td>
<td>BIRDLAND</td>
<td>THE MANHATTAN TRANSFER ANTHOLOGY</td>
</tr>
<tr>
<td></td>
<td>BOY FROM NEW YORK CITY</td>
<td>THE MANHATTAN TRANSFER ANTHOLOGY</td>
</tr>
<tr>
<td></td>
<td>JAVA JIVE</td>
<td>THE MANHATTAN TRANSFER ANTHOLOGY</td>
</tr>
</tbody>
</table>

Figure 13-9, Example of a report created with a report generator
Step 5: Structuring the Application’s Logical Flow

- Create logical flow of application before writing source code
  - Details main functionality of the system and the relationship of tasks to be completed
- Flowcharts
  - Combination of symbols and text
  - Provide a visual description of a process
Step 5: Structuring the Application’s Logical Flow (cont’d.)

Figure 13-10, Flowchart example
Step 5: Structuring the Application’s Logical Flow (cont’d.)

**Figure 13-11, Flowchart symbols**

- **starts or ends the program flow**
- **a task to be performed**
- **get input**
- **make a decision**
- **display data**
- **document that can be read**
Step 5: Structuring the Application’s Logical Flow (cont’d.)

• Pseudocode
  – Description of program logic written in human language
    • Sample pseudocode for the process of starting a car:

  Start
  Put the key in the ignition and turn
  If the car does not start, call a friend to take you to school
  Else if the car does start
    Release the brake
    Put the car in drive
    Drive to school carefully
  End if
  Remove key
  Get out of car
  Stop
Step 6: Start Building the Prototype

• Opening screen must reflect user’s goals and program’s main function
  – Design opening screen
  – Wait for end user opening screen approval
  – Design data input screens
  – Include end user in the design process

• Prototype
  – Provides end users with a good idea of completed application
  – Not the final product
Step 7: Putting All the Pieces Together

- Take all information and create the design document
  - Provide realistic dates and price estimates
  - Make user sign the design document
  - Create addendum detailing any new items
Step 7: Putting All the Pieces Together (cont’d.)

• Design document items:
  – Header page describing the contents
  – Project objective
  – Defined terms related to the project
  – Feasibility study
  – Project specifications and requirements
  – Project cost analysis
  – Data dictionary
  – Copies of screens (or prototypes) and reports
Step 7: Putting All the Pieces Together (cont’d.)

• Design document items (cont’d.):
  – Diagrams (UML diagrams and flowcharts of all business processes)
  – Plans to test the software after it is written
  – Plans to gather user feedback about the application’s functionality
  – Notes from meetings

• May also include other appropriate information
  – Employee bios and company profiles
  – Place for user to sign and addendums
Avoiding the Pitfalls

• Userphobia
  – Fear of failure if user is included in design process
    • Solution: keep communication open

• Too much work
  – “Heap on the work” syndrome
    • Solution: be assertive and honest with the manager, and document everything
Avoiding the Pitfalls (cont’d.)

• Scope creep
  – Making continual changes and extensions
    • Solution: use phased approach to development
  – Gold plating
    • Software engineers add their own unnecessary features to the design
    • Not approved by end user
The Project Development Team

- **Project manager**
  - Chooses right players for the right positions
  - Determines risk and cost
  - Schedules tasks using project management software
  - Pulls together the design document
- **Database administrator**
  - Creates and maintains database structure
The Project Development Team (cont’d.)

Figure 13-12, Project management software helps a manager keep track of the project’s status
The Project Development Team (cont’d.)

• Software developers (programmers)
  – Writes source code to meet end user functional requirements

• Client (end user)
  – Driving force behind project
  – Has the need that can be met by team

• Tester
  – Ensures program functions correctly and meets all functional requirements in the design document
    • Tests all possible situations and keeps a log of errors
The Project Development Team (cont’d.)

• Customer relations representative
  – Interface between testers, developers, and end users during product creation and early release

• Generator of installation media
  – May be task of customer relations person
  – Ensures all necessary files included on the media

• Installer of the application
  – Installs program on the end user’s machine
  – Gives the user a guided tour of the application
One Last Thought

• Good design results in good programs
  – Results of skipping steps:
    • Poor performance, unmet client needs, a project running over budget and over schedule
• Project manager’s main responsibilities:
  – Build a team that can work well together
  – Keep project on schedule and within budget
• Include a thorough testing cycle
Summary

• Software engineering includes many steps to follow in order
• Create applications meeting end user’s needs
• Software development life cycle (SDLC) process
  – Describes application life, including program development, testing, installation, and maintenance
• Design document
  – Blueprint for software development
  – Requires several steps
Summary (cont’d.)

• UML provides a visual view of system functionality
  – Several types of UML diagrams
• Using reports and the data dictionary can help a developer assess any design “holes”
• Software development is a team effort
• Once the application is developed, installation disks are generated
• After a system is installed on client’s system, provide initial end user training