Introduction to
Software System Analysis and Design

Part 1
About the Course

■ Course Materials
  ■ My over 20 years of IT technology strategy and business application software development experiences
  ■ Business application project management experiences
  ■ Software industry best practices

■ Classroom Style
  ■ Lecture
  ■ Discussion
  ■ Practice
  ■ Questions and answers

■ Exams
  ■ Test (30% of total grade)
  ■ Design project (70% of total grade)
Topics Covered in This Course

- **Software Development Lifecycle**
  - Development Methodology
  - Software Engineering Goals and Roles

- **Project Development Planning**
  - Project scope
  - Project management

- **System Analysis**
  - Requirement Gathering
  - Use Case Modeling
  - Structural Analysis
  - Behavior Analysis

- **System Design Approach**
  - System Architecture Design
  - User Interface Design
  - Business Logic Design
  - Persistence Data Design

- **Implementation**
  - Programming construction
  - Testing
<table>
<thead>
<tr>
<th>Time</th>
<th>6/29 (Monday)</th>
<th>6/30 (Tuesday)</th>
<th>7/1 (Wednesday)</th>
<th>7/2 (Thursday)</th>
<th>7/3 (Friday)</th>
<th>7/6 (Monday)</th>
</tr>
</thead>
<tbody>
<tr>
<td>8:00 am -</td>
<td>Self Introduction</td>
<td>Part 5 System Analysis</td>
<td>Part 8 Analysis Modeling</td>
<td>Classroom Test</td>
<td>Part 16 Data Persistence Design</td>
<td>Final Project Report</td>
</tr>
<tr>
<td>8:45 am</td>
<td>Part 1 Introduction</td>
<td>System Analysis</td>
<td>Analysis Modeling</td>
<td>Part 11 System Design Approach</td>
<td>- Data Persistence Design</td>
<td>(Student course grading)</td>
</tr>
<tr>
<td></td>
<td>About the course</td>
<td>Analysis Process</td>
<td>Concepts</td>
<td>- Design and development strategy</td>
<td>- Database</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Business and technology system</td>
<td>Techniques</td>
<td></td>
<td>Classroom Workshop</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>UML</td>
<td></td>
<td>- Student Practice</td>
<td></td>
</tr>
<tr>
<td>8:55 am -</td>
<td>Part 1 Introduction</td>
<td>Part 5 System Analysis</td>
<td>Part 9 Structural Analysis</td>
<td>Design and Development Reviews</td>
<td>Design and Development Reviews</td>
<td></td>
</tr>
<tr>
<td>9:40 am</td>
<td>Software system development lifecycle</td>
<td>System Analysis</td>
<td>Structural Analysis</td>
<td>- Basic system design review</td>
<td>(Same as above)</td>
<td>(Student course grading)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Current System</td>
<td>Conceptual system structures</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Business process automation</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Business Process reengineering</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10:35 am</td>
<td>Information Engineering</td>
<td>Get requirements</td>
<td>System interactions for business</td>
<td>Concepts</td>
<td>- Programming Testing</td>
<td>(Same as above)</td>
</tr>
<tr>
<td></td>
<td>Object-oriented</td>
<td>Joint Application Design</td>
<td>data and processing</td>
<td>Technique</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Component-based</td>
<td></td>
<td></td>
<td>Applying concepts</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1:00 pm -</td>
<td>Part 3 Software engineering Goals</td>
<td>Part 7 Use Case Modeling</td>
<td>Part 14 User Interface Design</td>
<td>Part 18 Final Course Review</td>
<td></td>
<td>(Same as above)</td>
</tr>
<tr>
<td>1:45 pm</td>
<td>and Roles</td>
<td>Concepts</td>
<td>- Student Practice</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Roles in business system</td>
<td>Technique</td>
<td>- User interface design process</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Roles in technology system</td>
<td></td>
<td>- Design Consideration</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1:55 pm -</td>
<td>Part 4 Project Development Planning</td>
<td>Classroom Workshop</td>
<td>Part 15 Business Logic Design</td>
<td></td>
<td>Student Project Assignment</td>
<td></td>
</tr>
<tr>
<td>2:40 pm</td>
<td>Project Initiation</td>
<td>- Student Practice</td>
<td>Application logic</td>
<td>- Complete during 6/28-6/29</td>
<td>- Submit Final Report by Monday</td>
<td>(Same as above)</td>
</tr>
<tr>
<td></td>
<td>Project Management</td>
<td></td>
<td>Component dependency</td>
<td>Morning 6/30</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Grading

Test 30
Final Project 70
Total 100

Grading

Pass
86-100 A
75-85 B
60-74 C

Not Pass
< 60 F
Course Objectives

What will you learn from this course?
- Concepts about system architecture
- Concepts about project management
- System analysis and design concepts
- System analysis and design techniques

What is the value to you?
- Gain general knowledge about software engineering
- Learn about system analysis and design skill
- Improve key design communication skill
- Increase your system thinking skill
- More importantly, prepare yourself with knowledge and ability for the future
Main Learning Objectives

We will take a top-down approach for this course with the following subjects as the main learning objectives:

- **Software system analysis and design**
  - Software analysis and design concepts
  - Analysis and design artifacts

- **Software system development lifecycle**
  - Fundamental analysis and design process
  - Software system architect basic skills
What is “System Analysis and Design”? 

- Technology solutions for business
  - Understand the business needs
  - Enable business with technology solutions

- Software engineering
  - Follow a systematic development process
  - Communicate artifacts of system transformation

- Technology automation creation
  - Define system architecture
  - Create design models
Conceptual View of System Development Process

New ideas

Re-engineering

Define Business Process

Change Business Process

Business Process Automation

Development Planning

System Analysis

System Design

Production System Installation

System Implementation

Our study focus

Production System Support
Part 2

System Development Methodologies
What is Software System Development Methodology?

A Software System Development Methodology should:

- describe an approach for software system development.
- guide development teams using common steps.
- define consistent artifacts (e.g. design models).
- support a system development lifecycle.
- support a team development environment.
- be supported by a standardized set of tools.
Different Software Development Methodologies

- Information Engineering Development Methodology
  - Water-fall methodology
  - Modular structure
  - Procedure processing

- Object-oriented Development Methodology
  - Iterative methodology
  - Class structure
  - Event driven and messaging

- Component-based Development Methodology
  - Component integration
  - Component reuse
Information Engineering Development Methodology

Pros:
- Simple process
- Fixed requirements
- Clear phase changes

Cons:
- Less flexible for changes
- High degree module dependencies
- High degree data dependencies
Object-Oriented Development Methodology

- **Pros:**
  - Iterative analysis and design process
  - Flexible design changes with incremental requirement changes
  - Promote functional and data independence
  - High degree of development productivity
  - Object reuse

- **Cons:**
  - Difficulty of managing project increases
  - Software complexity increases
Component-based Development Methodology

**Pros:**
- Flexible design changes with incremental requirement changes
- Promote functional and data independence
- High degree of development productivity
- Requirements managed independently
- Clear development boundaries
- Parallel development
- Component reuse

**Cons:**
- Complex component development management process
## System Development Lifecycle – Planning

<table>
<thead>
<tr>
<th>Phase</th>
<th>Step</th>
<th>Technique</th>
<th>Artifacts</th>
</tr>
</thead>
<tbody>
<tr>
<td>Planning (Why build the system?)</td>
<td>Identifying Business Value</td>
<td>System request and dependencies</td>
<td>Business request and requirements</td>
</tr>
<tr>
<td></td>
<td>Analyze feasibility</td>
<td>Technical feasibility</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Economic feasibility</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Organizational feasibility</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Feasibility study</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Develop work plan</td>
<td>Task Identification</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Time estimation</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Work plan</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Staff the project</td>
<td>Creating a staffing plan</td>
<td>Staffing plan</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Creating a project charter (goals and plan)</td>
<td>Project plan</td>
</tr>
<tr>
<td></td>
<td>Control and direct project (throughout the project)</td>
<td>Refine estimates</td>
<td>GANTT chart</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Track tasks</td>
<td>CASE tool</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Coordinate project</td>
<td>Standards list</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Manage scope</td>
<td>Project binder(s)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Mitigate risk</td>
<td>Risk assessment</td>
</tr>
</tbody>
</table>
# System Development Lifecycle - Analysis

## System Analysis

<table>
<thead>
<tr>
<th>Phase</th>
<th>Step</th>
<th>Technique</th>
<th>Artifacts</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Analysis</strong></td>
<td>System analysis</td>
<td>Problem analysis</td>
<td>Analysis report</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Benchmarking</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reengineering</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Information gathering</td>
<td>Interviews</td>
<td>Requirement information</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Questionnaires</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Use case modeling</td>
<td>Use cases</td>
<td>Functional models</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Use case models</td>
<td>(business automation)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Activity diagram</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Structural modeling</td>
<td>Class diagrams</td>
<td>System structural models</td>
</tr>
<tr>
<td></td>
<td>(static)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Behavioral modeling</td>
<td>Sequence diagram</td>
<td>System interaction models</td>
</tr>
<tr>
<td></td>
<td>(dynamic)</td>
<td>Collaboration diagram</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Statechart diagram</td>
<td></td>
</tr>
</tbody>
</table>
### System Development Lifecycle - Design

<table>
<thead>
<tr>
<th>Phase</th>
<th>Step</th>
<th>Technique</th>
<th>Artifacts</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Design</strong></td>
<td>System design</td>
<td>Custom development</td>
<td>Design strategy</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Package development</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Outsourcing</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Network design</td>
<td>Hardware design</td>
<td>Technical architecture</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Network design</td>
<td>Infrastructure design</td>
</tr>
<tr>
<td></td>
<td>Interface design</td>
<td>Interface structure</td>
<td>Interface design</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Input design</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Output design</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Database design</td>
<td>Data schema</td>
<td>Physical data structure/表</td>
</tr>
<tr>
<td></td>
<td>(including any files)</td>
<td>Data storage</td>
<td>Data storage design</td>
</tr>
<tr>
<td></td>
<td>Object design</td>
<td>Class diagrams</td>
<td>Program structure chart</td>
</tr>
<tr>
<td></td>
<td>(refined physical design)</td>
<td>Sequence diagrams</td>
<td>Program specifications</td>
</tr>
</tbody>
</table>
## System Development Lifecycle - Implementation

<table>
<thead>
<tr>
<th>Phase</th>
<th>Step</th>
<th>Technique</th>
<th>Artifacts</th>
</tr>
</thead>
<tbody>
<tr>
<td>Implementation</td>
<td>Construction</td>
<td>Programming Testing</td>
<td>Programs Test plan</td>
</tr>
<tr>
<td><em>(System delivery)</em></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Installation</td>
<td>Direct Online</td>
<td>Configured system Training plan</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Creating Training plan</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Support</td>
<td>Support strategy</td>
<td>Support/service plan</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Post-implementation review</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Software Engineering Goals and Roles

Part 3
Software Engineering Goals

- **Productivity**
  - To easily develop and modify system designs with efficient and effective development methodology

- **Extendibility**
  - To minimize the dependencies between data and functions to increase flexibility of making modifications
  - To design modular functions for easy of adding new business functions

- **Reusability**
  - To reuse data and functions for other business opportunities
Software Engineering Key Design Considerations

- Modularity
  - by data encapsulated with processing functions (i.e. methods)

- Extendibility
  - by object independence and inheritance

- Testability
  - by high degree of object independence
  - Component integration

- Maintainability
  - by high degree of modularity

- Reusability
  - by reuse of objects, components, and object-oriented frameworks
Team Work with Software Development Process

- Planning
  - Project Manager
  - Business Analyst / End-user Liaison
  - System / Network / Database Architect
  - Software Engineer / Developer / Programmer
  - System Engineer / Integrator / Usability Tester

- Design

- Programming

- Delivering

Analysis

Testing
Part 4
Project Development Planning

- Project Initiation
  - Identify business value
  - Feasibility analysis

- Project Management
  - Develop project plan
  - Staff the project
  - Manage the project
Identify Business Value of using Technology

- Project sponsors
  - Business owners
  - Customers

- Business needs
  - New business capabilities
  - Business re-engineering

- Functionality
  - Business functions
  - Business processes
Identify Business Value of using Technology (continued)

- Expected value
  - Financial gains
  - Increasing market shares
  - Customer satisfactions

- Special issues/constrains
  - Business limitations
  - Regulation limitations
  - Competitions in the marketplace
  - Market supply/demand limitations
Feasibility Analysis

- Technical Feasibility
  *i.e., can we build the system?*
  - Familiarity with business application
  - Familiarity with technology to be used
  - Manageable project size
Feasibility Analysis (continued)

- Economical feasibility
  *i.e., is it worth to build the system?*

  - Cash flow
    - Pay for the development resources (people, hardware, software, etc.)
  - Total Cost (TC) of years
    - Development costs
    - Operating costs
  - Total Benefits (TB) of years
    - Reduced cost of labor reduction and efficiency
    - Additional revenues generated

- Total Net Benefit (TNB) = TB – TC
- Return on Investment = TNB/TC, *(ROI is % value)*
- Net Present Value = TNB / (1+interests)^years

- Intangible costs and benefits
  - Market presences
  - Reputation
  - Brand recognition
### Example

<table>
<thead>
<tr>
<th></th>
<th>Year 1</th>
<th>Year 2</th>
<th>Year 3</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>New Development Cost</strong></td>
<td>$2500 (software and labor)</td>
<td></td>
<td></td>
<td>$2500</td>
</tr>
<tr>
<td><strong>New System Support Cost</strong></td>
<td></td>
<td>$4000 (people and system)</td>
<td>$3500 (people and system)</td>
<td>$7500</td>
</tr>
<tr>
<td><strong>New Business (benefits)</strong></td>
<td>$200</td>
<td>$300</td>
<td></td>
<td>$500</td>
</tr>
<tr>
<td><strong>Current System Support Cost</strong> (no change)</td>
<td>$5000</td>
<td>$5000</td>
<td></td>
<td>$10000</td>
</tr>
</tbody>
</table>

- **TNB = TB –TC**
  - = Cost Saving + new business – initial cost of development
  - = [($10000 - $7500)+ $500] - $2500 = $500
- **ROI = 500/2500 = 20%**
- **Net Present Value = $500 / (1+4%)^3 = $444.5** (value of today)
Organizational Feasibility

*i.e., who will support the system?*

- Project champion (e.g. key business or technology management)
- Senior management (e.g. company’s leadership management)
- Users (e.g. customers)
- Other stakeholders (e.g. internal and external business partners)
Identifying tasks
- Activities
- Deliverables
- Task hours
- Assignments

Time estimate
- Project milestones
- Resource planning (i.e., when/how many people on project)

Create an overall project plan
- Project effort staffed with appropriate resources:
  - Resource (number of staffs) matched Time (milestones)
- Overhead increased situations (project costs increased)
  - Under staffing (project time will be longer, overhead increased)
  - Overstaffing (project productivity reduced, overhead increased)
Iterative and Incremental Process

Plan
Analyze
Design
Build
Test

Iteration 1   Iteration 2   ... ...

Project should start with 20% of each use case out of 20% of all use cases = 4% total development effort

20% of all use cases
20% of each use case
Staff the Project

- Staffing plan
  - Team members (what are the roles and the needs)
  - Skill requirements (what expertise the project team needs)
  - Project team structure (who plays what role)
  - Number resources according to the project plan schedule
Project Management Activities

- Manage the scope
  - What is (or is not) to be delivered

- Refine the estimates
  - When will tasks be done

- Track the tasks
  - Who is doing what

- Coordinate the project
  - Who is responsible for what to happen and when

- Mitigate the risks
  - How can contingency plans reduce the failure risk of the project
Part 5
About System Analysis

- Purpose
- Analysis Process
- Business System and Technology System
- Business process automation
- Business reengineering
What is the Purpose of Analysis?

- Capture the business requirements with models
- Transform the business needs into technology implications
- Understand what functions to be built for the business
- Model the business information for future design solutions
Analysis Process

- Understand as-is system (current environment)
  - Understand the current system
  - Capture the current business and technology environment
- Identify improvement opportunities (changes needed)
  - What are problems that should be solved
  - What are the priorities
  - What are the cost-effective opportunities
- Develop to-be system (future environment)
  - Revise the as-is system
    - Modify existing and add new processes
    - Modify existing and add new data
  - Model and recommend the to-be system
What is a business system?
- A depiction of a real world business including:
  Business process, Business data, Business organization,
  Business operation rules, Business policies, etc. ……

What is a technology system?
- A depiction of a technology system (solution) for a business system including:
  System process, System data, System structure, etc. ……

Technology systems support the business systems and automate the business processes.
Business Process Automation

- Automate the business with technology (why to change)
  - Change from manual to automation – faster (e.g. on-line banking vs. paper)
  - Use technology to improve business tasks – more accurate

- Analysis methods to determine an automation area (where to change)
  - Duration analysis
  - Activity-based costing
  - Informal benchmarking
  - Formal benchmarking

- Follow the analysis process (about how)
  - Understand as-is system (current environment)
  - Identify improvement opportunities (changes needed)
  - Develop to-be system (future environment)

Copyright © 2003-2015 Steve Wu
Business Reengineering

- Change current business process
  - Increasing business value
  - Create business opportunities

- Apply the analysis process to
  - Outcome analysis – change value produced for customers
  - Breaking assumptions – change reasoning/rules
  - Technical analysis – leverage technology
  - Activity elimination – reduce steps in the process
  - Proxy benchmarking – borrow ideas from different industry
Developing an Analysis Report

- Justify the cost for the benefit of business value

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Potential Business Value of Return</td>
<td>Low-Moderate</td>
<td>Moderate</td>
<td>High</td>
</tr>
<tr>
<td>Project Cost</td>
<td>Low</td>
<td>Low-Moderate</td>
<td>High</td>
</tr>
<tr>
<td>Breadth of Analysis Needed</td>
<td>Narrow</td>
<td>Narrow-Moderate</td>
<td>Very broad</td>
</tr>
<tr>
<td>Risk of Change</td>
<td>Low-Moderate</td>
<td>Low-Moderate</td>
<td>Very high</td>
</tr>
</tbody>
</table>
Requirement Gathering

Part 6
Business Requirement Gathering

Selecting appropriate techniques to gather business requirements:

- **Document Analysis**
  - Based on the exiting documentations

- **Questionnaires**
  - Information in writing from individual(s)

- **Interviews**
  - Information from meeting of individual(s)

- **Joint Application Design (JAD)**
  - Information from meeting of group(s)
Document Analysis

- Research existing documentation
  - Business process and data
  - Business management

- Synthesizes the information
  - Findings
  - Identify gaps

- Make recommendations as requirement
  - Changes needs
  - Suggestions
Questionnaires

- Selecting Participants
  - Business representatives
  - Subject matter experts

- Design the Questionnaires
  - Categorize the questionnaires
  - List important items
  - Straight forward and unbiased
  - Set them at a right level without ambiguity

- Administrating the Questionnaire
  - Get participants to complete questions
  - Explain why the questionnaire is conducted
  - Clarify any questions

- Follow-up
  - Get back to the participants to make sure work completions
Interviews

- Selecting Interviewees
  - Name
  - Position
  - Purpose of interview
  - Meeting

- Designing Questions
  - Close-ended questions – Specific needs
  - Open-ended questions – Uncover needs
  - Probing questions – Confirm the needs’ certainty or clarity
  - Level of questions:
    - High-level: very general
    - Medium-level: moderately specific
    - Low-level: very specific
Preparing for the interview
- Plan - topic
- Questions – purpose
  - Open-ended questions are easier to prepare than the close-ended
- Possible answers – confirmation

Conducting the interview
- Build trust
- Professional
- Unbiased
- Document accurately

Follow-up
- Written interview report
- Confirm interview points
- Clarify additional questions
Joint Application Design (JAD)

- Selecting Participants
  - Who should join the JAD session

- Designing the Session
  - What topics to go over and get results

- Preparing for the Session
  - What materials to prepare before the session

- Conducting the Session
  - Facilitate the discussion and engage with the participants

- Follow-Up
  - Any additional information to get from the participants identified at the session to complete the tasks
## Selecting Appropriate Techniques

<table>
<thead>
<tr>
<th>Techniques</th>
<th>Document Analysis</th>
<th>Questionnaires</th>
<th>Interviews</th>
<th>JAD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gathered information</td>
<td>As-is</td>
<td>As-is, Improvement</td>
<td>As-is, Improvement, To-be</td>
<td>As-is, Improvement, To-be</td>
</tr>
<tr>
<td>Depth of information</td>
<td>Low</td>
<td>Medium</td>
<td>High</td>
<td>High</td>
</tr>
<tr>
<td>Breadth of information</td>
<td>Medium (depends on availability)</td>
<td>High</td>
<td>Low</td>
<td>Medium</td>
</tr>
<tr>
<td>Integration of information</td>
<td>Low</td>
<td>Low</td>
<td>Low</td>
<td>High</td>
</tr>
<tr>
<td>User involvement</td>
<td>Low</td>
<td>Low</td>
<td>Medium</td>
<td>High</td>
</tr>
<tr>
<td>Cost</td>
<td>Low</td>
<td>Low</td>
<td>Medium</td>
<td>Medium-High</td>
</tr>
</tbody>
</table>
Use Case Modeling

Part 7
Modeling

- What does modeling mean?
  - General purposes

- What is to model?
  - Subject contents

- Use case modeling
  - Concepts
  - Structures
  - Development
What does Modeling Mean?

Modeling is to depict the real world information with artifact (representations) so that …

- We capture meaningful information in the scope of interests
  i.e. Useful model with relevant information
  e.g. Business processes, technology systems

- We know how to utilize the information for developing solutions
  i.e. Meaningful model with purpose
  e.g. Designs for building system

- We can communicate and share it with development team
  i.e. Understandable model with communicable notation
  e.g. Graphical and textual descriptions for users to understand
What is to Model?

- Model what is in the context
  
  e.g. business rules, technology constrains

- Model information at a right level
  
  e.g. analysis level, design level

- Model complex information with different level of abstractions
  
  e.g. subsystems, packages
Use Case Modeling

What is Use Case?

“A use case is a sequence of transactions in a system whose task is to yield a result of measurable value to an individual actor of the system.”

Ivar Jacobson

In other words

A use case describes a sequence of “system” activities from its user’s perspective.

Therefore

- Use cases describe how a system will be used
- Use cases specify what business requirements are
- Use cases define system’s functionality and scope
- Use cases provide specifications of a system capability
- Use cases create basis for system testing
Use Case Modeling Concepts

- **What is use case for?**
  Define what is to be performed by the “system”.

- **Who develops use case?**
  Jointly developed by the end-users and the “system” development team.

---

**Library System**

- **Check book (Borrower’s ID)**
- **Handle Checkbook**
- **Update book record**

---

*Check book (Borrower's ID)*

*<<include>>*

*Handle Checkbook*

*Update book record*
Use Case Model Structure

Use Case Model Contains:

- Use case diagram (graphical representation)
  - A graphical depiction of how a system is used in a particular context with the involved actors.

- Actor
  - Role in the context of use case
  - Initiator or receiver of service
  - External Systems

- Use case (textual description)

- Use case event

- Use case relationship
  - Communications: between actors and use cases
  - Includes: between a use case and a shared use case by another use case
  - Extends: between a core use case and an extended use case (extension or alternate course of the core use case)
Use Case Model Structure (continued)

- Use Case Modeling Level
  - Main use cases
  - Detail use cases

![Diagram of use cases]

- Borrow book (Student ID)
- Check-out book
- Update book record
- Charge late-return fee
- Notify book return
- Check-in book
- Recall book (Book ID)
- Return book (Borrower's ID)

Library System

Copyright © 2003-2015 Steve Wu
More about Use Case

- Use case is an analysis technique, which can be used for different contexts of modeling “system” and at different levels (conceptual or logical). This system can be application, component, etc.

- Use case gives the abstraction of the system from the perspective of its “actor” (i.e., business users, applications, or components interfacing the system).

- Use case describes the process (activities) performed by the system. It is generally triggered by actor(s) along with some pre-conditions or post-conditions. Alternate steps may be described if certain conditions are met. This may include any exception handlings.

- Use case diagram is a collection of actors, use cases visually depicting their relationships (e.g., actors trigger use cases or use cases associate with other use cases).
Develop Use Case Model

- Identify the Main Use Case
  - Find the system’ boundaries
  - List the primary actors
  - List the goals of the primary actors
  - Identify and write the main use cases
  - Review the main use cases

- Expand the Main Use Case to Detail Use Cases
  - Choose main use cases to expand (for further analysis)
  - Fill in details of the chosen use cases (becomes detail use cases)
  - Write the normal flow of the events of the use case
    - Complex or long flow can be sub-flows
    - Identify alternate or exceptional flows
    - Write the use case in terms of “who does what to whom” to provide clarity
    - Write specific condition, rules and any technical requirements as part of the activities
Develop Use Case Model (continued)

- Confirm the main use case
  - Review the main use cases to make any adjustments
    - Use packages to manage complex business functional requirements

- Create use case diagram along the way
  - Determine the system boundary
  - Place the use cases on the diagram
  - Place the actors on the diagram
  - Draw the associations
    - Use case events
    - Use case relationships
How to write a use case?

■ Identify (discover) an appropriate level of use cases to write
  – Main use cases: Abstract or generalized sequence
  – Detailed/Expanded use cases: Specific sequence

■ Start with an actor
  – A person who initiates the event
  – A system which triggers the event

■ End with a completion of an activity thread performed by the system
Telemedicine System Use Case Descriptions

Manage Services <<Home>>

This is the home page for users to browse the general information about Telemedicine, which has links to the activities that patients and physicians can perform after having registered and sign on successfully. The intent is to have as much flexibility as possible in principle. The activities may be performed by the patients and physicians are independent until they are in a session. The following links will be shown on the home page to access medical service or medical data. They are described by each use case respectively:

- Register
- Sign on Service Session
- Manage Patient Profile
- Manage Physician Profile
- Update Patient Treatment History
- Update Patient Health Record

Manage User Registration / Identification

This use case manage users’ registration and sign-on authentication:

- Register user and/or
- Authenticate users at the beginning of accessing the services and data.
Use Case Model Example – Use Case Description

Telemedicine System Use Case Descriptions

Sign on Service Session

After successful sign-on verification, a patient will start a session for service. There are two types of sessions:

- On-demand: (request on the fly and waiting for a physician online to response)
- Scheduled (pre-schedule time slot with a physician signed up for, see use case Manage Service Schedule)

Start a session with a service request that will lead to Make Payment before a service can be granted.

Once a successful payment is made, a session then can be started, including a set of physical conditions will be entered by the patient.

In a case of On-demand:
Patient will be put on a waiting list until a doctor is sign on/response to the request.
The waiting list may be sorted by time, treatment need or by name. (go to step 3)

In a case of scheduled:
See use case Manage Service Schedule
Scheduled time can be modified any time hereafter for change.

A session gets started when a doctor who has signed in and respond to a service request. When both the patient and the doctor are ready for the session with options available (they are described in each use cases respectively):

1. Chat
2. Audio
3. Video

A generic note of SOAP is created for the physician to capture the information pertain this service:

- Subject - the patient name
- Objective – Service request purpose
- Assessment - the record of diagnoses including text, image
- Plan – treatment plan and medication prescribed.
- The note will be saved in the system for future reference. Patient will be notified for information.

A physician may schedule a follow up of treatment services by opening the scheduler for future appointments, which put both physician and patient to a specific date/time through Manage Service Schedule.

Use case ends with session closed.
How do We Specify Software Systems?

- Need a software development process
- Need design models to describe the systems

Development Process

Software Systems

Design Models
What do We Use to Model Software Systems?

Unified Modeling Language (UML)

- Requirements Analysis
- Design
- Implementation
- Testing
- Realized
- Implemented
- Verified
- Test Plan
UML Offers Features for System Modeling

- Model Elements
  - Representations for system structures and interactions

- Relationships
  - Representations for system static and dynamic relationships

- Diagrams
  - Graphical models of static and dynamic information about the system

- Common Mechanisms
  - Annotations and user defined information

- Architecture Views
  - Models for different system perspectives
Model Elements in UML

- **Use Case** - A way in which an external actor uses a system
- **Class** - The definition of objects that share a common structure and common behavior
- **State** - A stage of lifecycle in which an object instance is in
- **Package** - A logical grouping of model elements
- **Component** - Physical implementation of a software unit
- **Node** - A hardware processor on which the software units execute
Relationships in UML

Association - A semantic connection between two classes/instances of objects

Generalization - A relationship between an element and the elements that specialize it

Interface - Mechanism offered by an element for others to access its functions

Dependency - An element is need by another

Aggregation - An elements is part of another element

Navigability - An unidirectional association

Multiplicity - Specification of an (+) integer range of allowable cardinalities
Diagrams in UML

- **Use Case Diagram (UML 1.x and 2.x)**
  shows the external view of how a system is used
- **Composite Diagram (UML 2.x)**
  shows the system structural parts, ports and interfaces
- **Class Diagram (UML 1.x and 2.x)**
  shows the class hierarchy and static relationships
- **Communication Diagram (UML 2.x)**
  shows interaction among objects organized in spatial network
- **Sequence Diagram (UML 1.x and 2.x)**
  shows interaction among objects organized in temporal order
- **Statechart Diagram (UML 1.x and 2.x)**
  shows states and conditions that cause state changes of an object
- **Activity Diagram (UML 1.x and 2.x)**
  shows activity flows or algorithms and controls of use cases or objects
- **Component Diagram (UML 1.x and 2.x)**
  shows the implementation components in a system and their dependencies
- **Deployment Diagram (UML 1.x and 2.x)**
  shows a system placement of nodes, networking and process distribution

*Get more information about UML: www.omg.org/uml*
Common Mechanisms in UML

- **Specifications**
  A textual description of additional details about an element

- **Adornments**
  Details rendered as graphical attachment to basic symbols

- **Notes**
  Arbitrary text comments attached to elements

- **Constraints**
  Textual specifications of or conditions imposed on relationships

- **Stereotypes**
  Creation of a new kind of model elements adapted from an existing model element

- **Properties**
  Tag-value pairs to attach arbitrary information to model elements
Use UML to Model Applications Analysis and Designs

Conceptual Representations

Variants of Applications

Distinct Applications

Specific Business Domain Components (business portfolio-level)

Cross Business Domain Components (enterprise-level)

System Software

UML Representations

Conceptual View

Logical View

Physical View

Use Case View

System Structure View

System Process View

Implementation Component View

Component Deployment View

Copyright © 2003-2015 Steve Wu
UML Can Model Business and Technology Systems

- Business Event
- Application Capability
- Application Flow
- Component Structure
- Data Model
- System Component
- Technology Deployment
Structural Modeling

Part 9
Create System Components using Objects

- **Business Component**
  - Logically grouped business functions with objects
  - For business applications
  - Objects (integrated together) interacting to complete a set of business processing

- **System Component**
  - Non-business specific
  - Technical utilities to support lower-level system functions
  - Built or supplied by technology vendors for supporting business component developments
Object Oriented View

- View the world by objects
  - Analyze information by objects
  - Relate objects by associations
  - Communicate to objects by messages

- Build systems with objects
  - User interface
  - Business logic
  - Information brokers
  - Data storage
Objects are ……

- Defined by classes with object-oriented approach
  - Contains information about itself attributes and/or states.
  - Has behaviors internally (private methods) and interface (public methods) available for other objects.

- Supported by object-oriented programming languages, JAVA, C++, C#, etc.

- Implemented as software program units of component or applications
Object Oriented Concepts

- **Object**
  - Packaged with data and behavior

- **Class**
  - Object classifications

- **Encapsulation**
  - Information hiding

- **Inheritance**
  - Information generalization and specialization

- **Polymorphism** (multiple forms)
  - Objects (and all subclassed objects) can be referenced and methods can be involved through dynamic bindings.

  e.g. in OO Programming: SHAPE has \(<<\text{subclass}>>\) SQUIRE, TRIANGLE, CIRCLE
  
  for i := 1 to 10
  
  \{a\text{Shape} := \text{SHAPE}[i]; // \text{SHAPE}[i] can be SQUIRE or TRIANGLE
  \text{shapeArea}[i] := a\text{Shape}.computeArea\} // SQUIRE.computeArea or TRIANGLE. computeArea
Class Diagram for Structure Modeling

- **Class**
  - Attributes - *Data*
  - Operations - *Function*

- **Associations**
  - Static – Cardinality, Containment, Inheritance
  - Dynamic – Messaging (more discussed in Behavior Modeling)

- **Class Diagrams (not just for objects)**
  - System-level
  - Component-level
  - Object-level
Class Diagram – Library Example

Library Personnel
- Library ID
- Name
- Get Library ID()

Book Management
- Check-in Book()
- Check-out Book()
- Record New Book()
- Check Book Status()

Book
- Book Name
- Book ID (ISBN)
- Book Type
- Lending Status
- Set Lending Status()
- Get Lending Status()

Library Personnel
- Librarian
  - Authorization Level
  - Get Authorization Level()
- Student
  - Class Level
  - Get Class Level()
- Teacher
  - Teaching Class
  - Get Teaching Class()

Borrower
- Book Borrowed
- Borrower Name
- Borrower ID
- Get Books Borrowed()
- Set Books Borrowed()
- Borrow Book()
- Return Book()
- Recall Book()

Reference Book
- Subject Type

Text Book
- Course

Novel
- Category

Library System Structural Model

Copyright © 2003-2015 Steve Wu
Part 10
Develop System Analysis Models

Use Case Model
(External view)

Analysis Model
(Internal view)

Business Functions

Use Case Package
Use Case Package
Use Case Diagram
Use Case
Actor

Collaboration Diagram
Sequence Diagram
Class Diagram
Class
State Diagram

Business Processes / Activities

Realization
(Business domain objects)

Business Requirements
(Rules and constraints)
Model the System Behaviors

- **Use Case View (system external view)**
  - What business needs
  - What business conditions
  - What business functions

- **Interaction Diagram (system internal view)**
  - How the need should be satisfied
  - How the conditions are applied
  - How the functions are carried out
System Interactions

- **Collaboration Diagram**
  (Communication Diagram in UML 2.0)
  - For functional analysis
    - Complexity
    - Distribution

- **Sequence Diagram**
  - For sequencing analysis
    - Operation order
    - System availability
    - Data availability
Using Collaboration Diagram
(Communication Diagram in UML 2.0)

Collaboration Diagram - Check-out Book

1: Check-out Book(ISBN)
2: Check Book Status(ISBN)
3: Get Lending Status(ISBN)
4: Check-out Book(ISBN)
5: Set Lending Status(Out)
6: Borrow Book(ISBN)
7: Set Books Borrowed()
Using Sequence Diagram

School : Book Management
Student : Borrower
Drama : Novel

Check Book Status()

Recall Book (ISBN)
Get Books Borrowed(ISBN)
Set Lending Status(Recalled)

Return Book(ISBN)
Set Books Borrowed(ISBN)
Set Lending Status(In)

Check-in Book()

Sequence Diagram - Recall Book
Model System, Component, Object States

- **What to model?**
  - Initial state
  - Trigger event
  - Every new state
  - All possible end-states

- **When to model?**
  - States represent different business conditions
  - States represent different business processing steps
  - States represent different business values
Statechart Diagram - Book Lending Status
Model System, Component, Object Activities

What to model?
- Trigger event
- All activities
- Conditional/un-conditional steps
- Messaging to other systems, components, objects

When to model?
- Clarify business activities
- Clarify business processing steps
- Clarify system, component, object control flow
Activity Diagram - Manage Book Circulation

System Trigger

Daily Book Lending Maturity

Matured Book?

Student : Borrower
- Books Borrowed

Recall Book

Reference : Book
- Lending Status

If Pass Due?

Process Late Fee

End Process

Copyright © 2003-2015 Steve Wu
System Design Approach

Part 11
From Analysis to Design

Development Strategies
- Custom development
- Packaged software
- Outsourcing

Transform Analysis Models into Design Models
- Factoring
- Partitions and collaborations

Structure Application Components
- Layers
- Packaging
Application Development Strategy

- **Custom Development**
  - In house software development teams
  - Develop software products
  - Develop special purpose applications

- **Packaged Software**
  - Off-shelf commercial software products
  - General purpose software products for customization
  - Business components for development
  - Non-business components for development
  - Application frameworks for development

- **Outsourcing**
  - Software vendors
  - In-house consulting and development
Selecting Development Strategy

Select a development strategy that meets the business needs and the time table

<table>
<thead>
<tr>
<th>Determining Factors</th>
<th>Choose Custom (in-house) Development if …</th>
<th>Select Package Solution if …</th>
<th>Look for Out-sourcing Development, if …</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Business Need</strong></td>
<td>unique need</td>
<td>Common need</td>
<td>Not core technology for the business</td>
</tr>
<tr>
<td><strong>In-house Experience</strong></td>
<td>Existing business and technical experience</td>
<td>Existing business experience</td>
<td>Both business and technical experience do not exist</td>
</tr>
<tr>
<td><strong>Project Skill And Management</strong></td>
<td>Ability to build</td>
<td>Not strategic skill</td>
<td>Strategic outsource direction</td>
</tr>
<tr>
<td><strong>Time Table</strong></td>
<td>Flexible solution</td>
<td>Quick solution</td>
<td>Flexible skill available for development, but may not as quick</td>
</tr>
</tbody>
</table>
Transform Analysis Models into Design Models

For custom software development:

- Factoring – generalization
  - Abstract functional behaviors
  - Abstract common data relationships
  - Group coherent functions and data

- Partitions – reduce complexity
  - Components
  - Subsystems
Create System Design Models

Analysis Model

Business Requirements
(Rules and constraints)

Transformation
(Change/Add to Implementation Level Classes)

Technology
Requirements /Implications

Design Model

Dynamic

Static
Structure Your Application

- **Architecture Layers (top to bottom)**
  - Business application specific
  - Non-business application specific
  - Business domain components
  - Non-business domain components
  - System components

- **Application Tiers (front to back)**
  - User interface – presentation
  - Business functional logic
  - Back-end business data management

- **Package Application Components**
  - Cohesive business functions in scope
  - Implementation of coherent processes and data
  - Runtime component distribution on platforms
Design Applications

- **Applications**
  - Each Line of business
  - Business domain

- **Components**
  - High granular units of software
  - High cost-effective, harder to build

- **Objects**
  - Low granular units of software
  - Good reusability, easy to build
System Architecture

Part 12
System Architecture and Infrastructure

- System Architecture Designs
- Making an Architecture Choice
- Design System with Quality and Reusability
- Infrastructure Design
- Application Deployment Model
System Architecture Designs

- Server-based
- Client-based
- Client-server
- Distributed multi-tiered
Server-based

- Easy to manage
- Less flexible and versatile
Client-based

- Versatile user experiences
- Difficult to change and upgrade
- Higher cost
Client-server

- Versatile user experiences
- Scalable for data accessing
- Harder upgrade
- Higher cost

Presentation Logic
Application Logic

Data Access Logic
Data Storage
Distributed multi-tiered

- Versatile user experiences
- Easy to manage and upgrade
- Scalable for application and data processing
- Low cost
# Making Architecture Choice

Web-enablement are all possible in these architecture configurations

<table>
<thead>
<tr>
<th>Measurement</th>
<th>Server-based (Mainframe)</th>
<th>Client-based (Desktop)</th>
<th>Client(rich)-server (2-tiers)</th>
<th>Thin/Rich-Client (N-tiers)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cost of Infrastructure</td>
<td>Very High</td>
<td>Medium</td>
<td>Low</td>
<td>Medium/High</td>
</tr>
<tr>
<td>Cost of Development</td>
<td>Medium</td>
<td>Low</td>
<td>High</td>
<td>Medium/High</td>
</tr>
<tr>
<td>Ease of Development</td>
<td>Low</td>
<td>High</td>
<td>Low-Medium</td>
<td>Medium</td>
</tr>
<tr>
<td>Interface Capabilities</td>
<td>Low</td>
<td>High</td>
<td>High</td>
<td>High</td>
</tr>
<tr>
<td>Control and Security</td>
<td>High</td>
<td>Low</td>
<td>Medium</td>
<td>Low</td>
</tr>
<tr>
<td>Scalability</td>
<td>Low</td>
<td>Medium</td>
<td>Medium</td>
<td>High</td>
</tr>
</tbody>
</table>
Design System with Quality and Reusability

- **Granularity** - Modula and functional decoupling
- **Interface** - Consistent and interoperable
- **Security** - Protect the business
- **Scalability** – able to handle large business data volume and users
- **Performance** - Efficient and high throughput
- **Extensibility** - Easy to modify and add new functionality
- **Reliability** - High degree of fault-tolerance
- **Testability** - Less programmable errors
- **Portability** - Develop once use everywhere
Infrastructure Design

- Technology Deployment Model
  - System topology (locations)
  - System connectivity (LAN, WAN, Wireless)
  - System process distributions (business functions)

- Hardware and Software Specification (requirements)
  - Choices of hardware
  - Platforms/OS (operating system)
  - System operational environment
  - User/client environment
Application Deployment Model (UML)

Customer

<<External Business System>>
Credit Report System

DB2

<<Local Database>>

Credit Report System

Web Server

Build Web pages
Handle Client Request

Application Server

Access Customer Record
Credit Evaluation
Serialize Object
Handle Service Request

Corp Net

Campus

<<Intranet>>

<<Internet>>

<<Local Network>>

<<Internet>>

<<Client>>
Customer

<<Client>>
Business User
Guiding Your Designs

- Architectural Planes
- Purposes of Design Models
- Design Principles
- Design Best Practice
- Design Heuristic
- Design Techniques and Considerations
Architectural Planes

- **Business Architecture**
  - Business processes
  - Business requirements

- **Application Architecture**
  - Application analyses
  - Application designs

- **Data Architecture**
  - Business data
  - Physical designs

- **Technology Architecture**
  - Infrastructure
  - System software and hardware
System Model Transformation

Use Case Model
(business logic - user’s view)

Analysis & Design Model
(business logic - system’s view)

Component Model
(software code)

Deployment Model
(infrastructure)
Purposes of Design Models

- **To show how a system is designed to work**
  - Transformed the business activities captured from the analysis object model.

- **To show what a system needs to have**
  - Implementation-level classes and relationships
  - Decomposed and added transient classes to support the implementation.

- **To meet needs of all business use cases**
  - All the design models satisfies the use case requirements
  - Transform all analysis object models into design object models
Design Principles

- A class should have a key abstraction
  - Group coherent functions, not just a collections of data and functions
- Hide (encapsulate) all data within its class
  - Use object interfaces to get data
- Keep classes independent as much as possible
  - For potential reuse and easier to extend
- Distribute system functions among the component evenly
  - Balanced functional distribution among the component structure
Design Best Practice

- Keep related data and behavior in one place
- Factor the commonality of data, behavior, and/or interface as much as possible in an inheritance hierarchy
- Use public interfaces of a class (or a component) for accessing data
- Implement a minimal public interface that all classes understand.
- Make roles as attributes of a class instead of each role as different classes, if they have common responsibilities/behaviors (e.g. father and mother is attribute value of parent)
- An class should minimize the number of class it collaborates with for loose-coupling
- A container class should know what it contains, and the contained classes should not have to know who contains it (so that the class is reusable elsewhere)
- The derived-class should not know anything about its supper class (so that it is independent for reuse)
Design Heuristic – Best Practices

- Do not use global (constant/variable) data. Instead, use objects variables and methods as bookkeeping information for other objects.
- Do not create global class in your system that the most of classes have relationships with it, or are depend on it.
- Do not access or change the data of an object without going through its public interfaces.
- A class with many <<accessor>> methods in its public interface that should not be a class or should be more than one class.
- Application business logic (i.e., functions) should be decoupled with the user interface.
- Physical design criteria should not corrupt the logical designs.
Design Techniques and Considerations

- Take the analysis models and create a system design models:
  - Architecture Layers
  - Design Principles
  - Design Best Practice

- Consider the following when define the system architecture:
  - Current environment
  - System to be interfaced
  - Future business needs
  - Possibility of system expansion

- Use collaboration/sequence diagrams to model the system behaviors.
- Use class diagrams to model the relationships among the objects discovered.
- Use statechart diagrams to model an object class behavior as needed.
User Interface Design Process

- **User Scenario Development**
  - Outline steps of user’s workflow

- **Interface Structure Design**
  - Navigation (links) of interface flows

- **Interface Standards Design**
  - Interface metaphor – real world concept
  - Interface objects – views, forms, pages
  - Interface actions – navigation commands, operations
  - Interface icons – visual placement
  - Interface templates – appearance of page and screens or forms
User Interface Design Process (continued)

- Interface Design Prototyping
  - Storyboard
  - Web-page prototype
  - Language prototype

- Interface Evaluation
  - Heuristic – compare basic design principles and best practice
  - Walk-through – present to users for feedback
  - Interactive – experimental evaluation
  - Formal usability testing – real environment of user’s evaluation
User Interface Design Considerations

- **Layout**
  - Areas for commands, navigation, inputs, outputs, status

- **Content Awareness**
  - Informative for users to know where they are in the system

- **Appealing**
  - Functional and inviting with good use of color space and fonts

- **User Experience**
  - Easy to learn and easy to use for users

- **Consistency**
  - Consistent design for clarity and predictable functional settings

- **Minimize Effort**
  - Simple to use the system with simple user interface
User Interface Navigation Design

- Basic Principles
  - Prevent user mistakes
  - Simplify recover from mistakes
  - Use consistent grammar order

- Types of Navigation Controls
  - Languages (commands)
  - Menus (lists)
  - Direct Manipulations (control widgets)

- Messages
  - Responses feedback
  - Guidance instructions (lists)

- Navigation Design Documentation
Input Design

- Basic Principles
  - On-line processing vs. batch processing
  - Capture data at the source
  - Minimize the keystrokes

- Types of Inputs
  - Text
  - Numbers
  - Selection - check box, radio button, list box, drop-down, combo box, slider
  - Graphs

- Input Validation
  - Completion
  - Format
  - Range
  - Digit
  - Consistency
Output Design

- Basic Principles
  - Understand report usage
  - Manage information load
  - Minimize bias

- Types of Outputs
  - Detail reports
  - Summary report
  - Document
  - Graphs

- Media
  - Online (voice, video)
  - Paper (image, text)
  - Electronic (control/message notification)
Best Practice for User Interface Design

- Separate the user interface objects from domain business objects so they can change independently.
- Avoid sequence of modal dialog windows at anytime if possible as they tend to “lock-up” the windows while users may need to access other information.
- Design user interfaces with users in mind - how it is used and how convenient it is, etc. Minimize user learning curve.
- Considering business rules and work flows to make design trade-off between “flat” vs. “deep” windows, web pages, or menus.
- Prototype user interface and validate the usability with the business users as earlier as possible.
- Balance of interactivity and frequent updates cost
Business Application Logic

- **Design Criteria**
  - Object coupling
  - Class cohesion
  - Method coherence
  - Design goals

- **Design Activities**
  - Design specifications
  - Identifying opportunities for reuse
  - Restructuring the design
  - Optimizing the design

- **Design Application Components with Objects**
  - Design models for business logic
## Object Coupling (between objects)

<table>
<thead>
<tr>
<th>Level of Coupling</th>
<th>Type of Object Relationship</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>loose</strong></td>
<td>No direct coupling between objects</td>
</tr>
<tr>
<td></td>
<td>Passing reference to another object</td>
</tr>
<tr>
<td></td>
<td>Passing value to another object</td>
</tr>
<tr>
<td></td>
<td>Passing control data to another object</td>
</tr>
<tr>
<td></td>
<td>Common or global referenced by both objects</td>
</tr>
<tr>
<td><strong>Tight</strong></td>
<td>Content being referenced directly by another object</td>
</tr>
</tbody>
</table>
## Class Cohesion

A class should have consistent information.

<table>
<thead>
<tr>
<th>Cohesion</th>
<th>If a class has …</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Good cohesion</strong></td>
<td>Single business concept (attributes related to the business)</td>
</tr>
<tr>
<td></td>
<td>Mixed business concepts (different business attributes)</td>
</tr>
<tr>
<td></td>
<td>Mixed business domain (different business areas)</td>
</tr>
<tr>
<td><strong>Not good cohesion</strong></td>
<td>Unrelated types of business (different concepts)</td>
</tr>
</tbody>
</table>
Method Coherence

Methods of a class should be related

<table>
<thead>
<tr>
<th>Coherence</th>
<th>If methods of a class have …</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Good Coherence</strong></td>
<td>only a single function</td>
</tr>
<tr>
<td></td>
<td>connected/related functions</td>
</tr>
<tr>
<td></td>
<td>functions using same attributes</td>
</tr>
<tr>
<td></td>
<td>directly related functions</td>
</tr>
<tr>
<td></td>
<td>support indirectly related functions</td>
</tr>
<tr>
<td></td>
<td>supports other related functions of different objects</td>
</tr>
<tr>
<td><strong>Not Good Coherence</strong></td>
<td>has unrelated functions</td>
</tr>
</tbody>
</table>
Design Goals

- Maximize the functional coherence within an encapsulation boundary (e.g. within a component or an object)

- Minimize the functional coupling between the encapsulation boundaries (e.g. between two components or objects)
Design Application Components with Objects

- Understand Business Requirements
- Design Business Class Structure
- Design Business Application Logic Flow
- Design Statechart of Business Process Rules
- Design Components of Business Application
Understand Business Requirements

Customer

Profile

Customer Record System

System Process Request

<<communicate>> Request Service

<<include>>

<<system event>>

Access Customer Record

<<extend>>

Get Credit Record

Customer Record System

<<communicate>>

process new case

Business Person

<<extend>>

Check Credit

External Business System

<<system event>>

Business User

Copyright © 2003-2015 Steve Wu
Design Business Class Structure

- Customer
- FinancialServiceCustomer
  - Service Application Form: 0..n
  - Account Record: 1..n
  - Service Log: 1
- Customer Record

Copyright © 2003-2015 Steve Wu
Design Business Application Logic Flow

1: Submit Application
2: Handle Request
3: Authentication
4: Handle Service Request
5: Get Customer Record
6: Access Customer Record
7: Send to Process Queue
8: Send Service Result
9: Display Result Page
10: Inform Customer

- Customer Record System
- SingleSignOn
- Process Queue
- CreditEvaluation
- External Business System

Copyright © 2003-2015 Steve Wu
Part 16
Data Persistence

- Data Persistence Formats
- Mapping Objects to a Relational Format
- Optimizing RDBMS-based Object Storage
- Design Considerations of Data Persistency
Data Persistence Formats

- Sequential and Random Access Files
- Relational Databases
- Object-Relational Databases
- Object-Oriented Databases
- Selecting an Data Persistence Format
Mapping Objects to Relational Format

- Create Persistent Object Models
  - Extracted business data from the class hierarchy for data persistency to a relational database

- Mapping Table

<table>
<thead>
<tr>
<th>Object</th>
<th>Entity (Relational)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class</td>
<td>Entity table</td>
</tr>
<tr>
<td>Object</td>
<td>Entity (Row)</td>
</tr>
<tr>
<td>Attribute</td>
<td>Column</td>
</tr>
<tr>
<td>Object identity attribute</td>
<td>Primary key (Entity key column)</td>
</tr>
<tr>
<td>Methods</td>
<td>Stored procedures</td>
</tr>
<tr>
<td>Association</td>
<td>Foreign key</td>
</tr>
<tr>
<td>Collections (whole-parts)</td>
<td>Association tables</td>
</tr>
</tbody>
</table>
Managed by the Book Management

First-Normal Form:
Eliminate the repeating fields (columns)
Optimizing RDBMS-based Object Storage

**Second-Normal Form:**
Remove field depending only on partial key

- **Borrower**
  - **P**
  - **Borrower ID**
  - **Borrower Name**
  - **1**

- **Book**
  - **P**
  - **Book Name**
  - **Book ID (ISBN)**
  - **Book Type**
  - **Lending Status**
  - **Book Quantity**
  - **Book Description**
  - **1..***

- **Book Borrowing**
  - **P**
  - **Lending Transaction ID**
  - **Date**
  - **Borrower Name**
  - **Borrower ID**
  - **Book ID (ISBN)**
  - **0..***

- **Book Borrowed**
  - **0..***
  - **Lending Transaction ID**
  - **Date**
  - **Borrower ID**

- **Borrow Book**
  - **P, F**
  - **Lending Transaction ID**
  - **Book ID (ISBN)**
  - **0..***

Identify joint keys for M:M relationships

P = Primary Key
F = Foreign Key
Third-Normal Form:
Remove field depending only on non primary key

They are fields that only depending on non-primary key “book name”
Optimizing Data Access Speed

- **De-normalization**
  - Reversing the normalization for performance

- **Clustering**
  - Adding data fields (columns) to other tables for faster data retrieval (e.g. Borrower Name are in both Borrower and Book Borrowed)

- **Indexing**
  - An indexing table has pre-sorted index based on a type for faster search (e.g. by “Text-book”, “Reference-book” and “novel” pointing at each data record as groups)
Design Considerations of Data Persistency

- Store data in local memory for high access-frequency data that requires less update needs.

- Do not store data in local memory for low access-frequency data that may require frequent update needs.

- Make a design trade-off when data are high access-frequency and frequent update needs. The bottom line is the balance of time vs. space.
Manage Implementation

- Resource Assignments
  - GUI development
  - Specialized business function
  - Foundation/Shared function
  - Data accessing/persistence
  - Networking/infrastructure communication

- Manage the schedule
  - Development selections
  - Development dependencies
  - Change management
  - Incremental testing
Avoid Mistakes

- Using “Bleeding Edge” Technology
  - Inexperience

- Using Low-cost Personnel
  - Inexperience
  - Low quality

- Lack of Code Control
  - No change management
  - Lack of development code standards

- Inadequate Testing
  - No rigger test
  - No early test cost later major testing difficulties
Test Planning

- Unit/Component Testing
- Integration Testing
- System Testing
- Acceptance Testing
Unit/Component Testing

- **White-box Testing**
  - Code
  - Error handling

- **Black-box Testing**
  - Methods
  - Interfaces
  - Error test
Integration Testing

- **User Interface Testing**
  - User interface functions
  - Error conditions
- **Use Case Testing**
  - Use cases
  - Extended use cases
  - Special conditions
- **Interaction Testing**
  - Business logic
  - Business transaction (data processing)
- **System Interfacing Testing**
  - Data exchange
System Testing

- Requirement Testing
  - Business requirements
  - Business process status/conditions
- Usability Testing
  - Easy of use
  - Performance
  - Error reporting
- Security Testing
  - Authorization
  - Authentication
- Reliability Testing
  - System Monitoring
  - Disaster recover
- Performance Testing
  - Stress
  - Scalability
- Documentation Testing
  - Accuracy of HELP, procedure, tutorials
Acceptance Testing

- **Beta Testing (pre-release)**
  - User environments
  - User data

- **Alpha Testing (post-release)**
  - Conducted by users
  - Acceptance test
Example of Test Plan Format

<table>
<thead>
<tr>
<th>Use Case Name</th>
<th>Test Type</th>
<th>Test Conditions Settings / Field Values</th>
<th>Test Pattern (steps)</th>
<th>Expected Results</th>
<th>Actual Results</th>
<th>Pass / Not Pass</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Final Course Review

Part 18
Review of System Analysis and Design

- Capture the business function process
  - Create use cases
  - Use activity models for detail process for each use case as needed
  - Identify requirements

- Analyze the system components to realize the business needs
  - Class structures to support the business data and functions
  - Model their interactions
  - Model their internal behaviors
Create system design models
- Transform the Analysis-level models as base for creation of design
  - Decompose the high-level classes
  - Add design structures to support business objects
  - Add interfaces with underlying development framework, library, system component, utilities
- Design specific components using class diagrams, interaction diagrams, state diagrams:
  - User interfaces
  - Business components/objects
  - Interfaces with other applications or components
  - Interfaces to the back-end data accessing capabilities
- Define deployment model:
  - User environments
  - Business process distributions
  - Operational environment, network, security
  - Hardware specifications
Review of System Development Process

New idea

Define Business Process

Business Process Automation

Change Business Process

Development Planning

System Analysis

System Design

System Implementation

Production System Installation

Production System Support

Re-engineering
Key Learning from This Course

You have learned ...

- **Software system analysis and design**
  - Software analysis and design concepts
  - Analysis and design artifacts

- **Software system development lifecycle**
  - Fundamental analysis and design process
  - Software system architect basic skills
Thank you …

Wish you all have great accomplishment in your academic studies and feature success!
Appendix
Student Practice – *Create Use Case Model*

- Add a use case of “Renew book”
- Write use case descriptions for “Check-out Book”
- Students present models
Student Practice – *Create Analysis Model*

- Create a collaboration diagram for “Check-in book”

- Add new class structures to support
  - “Magazine”
  - “Staffs” of the school

- Student presents models
Student Practice - *Create Design Models*

- Create application flow for “check-in book” and “check-out book”:
  - User interface
  - System application logic flow (UI, flow control, data access)
  - Create database table
  - Show deployment diagram

- Students present models