Object Oriented Development Analysis and design

- **Texts:**
  - **Pressman Ch 19, 20, 21**
  - **Schach Ch 11 and 12**
  - **Notes of a complete case study will be provided**
- **Reference:**
  - **Bahrami Ch 6, 7, 8 9, 10**
- **Assigned readings**
- **Case Tool:**
  - Rational Rose

- To start learning object oriented software engineering
- knowledge of the following is a pre-requisite:
  - OO Programming
  - Information hiding
  - Classes and objects
  - Inheritance
  - Object relationships
  - Polymorphism
- Any introductory text on oop will provide the above information
- Another suggestion: Ch 2 and 3 of "A book of OO knowledge" by Henderson-Sellers.

Object Oriented Analysis

- **OOA consists of three steps**
  - **Use case modelling**
  - **Class modelling**
  - **Dynamic modelling**

- **Use case modelling**
  - **Identify actors**
  - **Identify use cases**
  - **Identify extensions/uses]**
  - **Write use case specifications**
  - **Make diagram**

- **Class modeling:**
  - **Identify Objects and Classes**
  - **Identify Hierarchies**
  - **Identify the Important attributes**
  - **Identify Relationships**
  - **Identify behavior/services/messages/functions**

- **Dynamic modeling**
  - **Time dependent behavior**
    - **Object Interaction Diagrams**
    - **State charts**

Use Case Modeling

Clear and Consistent Requirements with Use Cases

- **The use cases take high level system requirements, domain knowledge, business rules and work flow information and create a set of documents and diagrams to build a solid foundation of system functionality.**
- **These requirements are suitable for downstream object-oriented analysis and design phases.**
- **Use Cases are also an integral part of the Lifecycle Process. They are the first expression of business processes which are to be automated.**

What's Needed From Requirements

- Requirements need to be right
- Requirements need to be more focused to the user
- They should involve the users more

User Centered Analysis

- Is a process of capturing requirement from the users’ perspective
- Followed by
  - Analysis, which explores the connectivity and the consequences of different and potentially conflicting user requirements
  - Design, which maps the requirements into the software application to meet its needs

So How Do We Get the Requirements Right?

Back to Basics

- The Customer must tell you what they want
- You must gain an understanding of the customer’s business
- You have to know everyone and everything that will interact with the system
- If the customer does not know what they want, you’ve got to take time, observe and record how they are working today
- You’ve got to structure information in a specific way so it can be used later for analysis and design
- Uncover critical business rules from experts
- Stay focused on the what not the how

Back to Basics

- Then, you must write this information in such a way that the customer will understand
- It would be nice to create some pictures about the customer’s business
  - Helps comprehension
  - Gives you a starting point to confirm real requirements

Jacobson Proposed Just That

- His work originate around 1967 to help Ericcson
- Ideas were applied for gathering requirements on a large telecommunications switching system
- Skeleton method completed around 1971
- In 1985 a major refinement was proposed and the Objectory method was released
  - Heart of Objectory is Use Case Modeling
- Most OO methods have adopted use cases as their front-end
- Use cases have been widely adopted in the industries

Use Cases Model Capture of Requirements

- Depicts what services the system provides to the user
- Provides information about the users (actors) of the system
- Shows the nature of interactions between the actor and the system (use cases)
- Relates Actors and Use Cases

Use Case Reflects System Functionality for its User(s)

- Actors
  - Human
  - Other Systems
  - Machines
- System
  - Contains Use Cases
    - Sequence of transactions in an interaction with the system.

Actor

- External to the system
- Details about the actor is un-necessary
  - Capture if active or passive
- Actor represents anything outside the system that will interact with the system
  - Can be Human
  - Can be Machine

Actor is not a User

- Actor represents a role that a user plays
- User is someone playing a role while using the system
  - e.g. Ahmed is a booking clerk

Actor

- **Each actor uses the system in different ways, otherwise they should be the same actor**
- **Each way the actor uses the system is a use case**

Use Case

- Describes transactions offered by the system
- Initiated by an actor
- A use case may be called by another use case
- Use cases can be combined for greater functionality
- Use case represents what the system must provide, rather than how
  - How the system will provide a service is unimportant to the user
  - When using the telephone, how the connection is made to another party is unimportant.
  - We just want to use the phone when necessary

Use Case

- **Use cases are not design documents or analysis documents**
  - **Use cases do support analysis and design efforts, though**
- **Use cases are not scenarios**
  - **They are not a record of a specific set of interactions between the user and the system**
  - **Use cases do come from scenarios**

Scenario

- A session that an actor has with the system
- Has details of real data and actual expected output
- Potentially hundreds to thousands in an application
- Each scenario may be slightly different than the previous one, even though you did essentially the same thing
- Scenarios are important as background information for discovering use cases

Example Scenarios

- Ahmed enters his account# 404504
- Ahmed enters his pin# 9342
- Ahmed requests his average balance from
- 1/1/99- 7/31/99
- System gives the average balance

- Asma enters her account# 34334
- Asma enters her pin# 4343
- Asma requests her average balance from
  - 6/30/99 - 7/31/99
- System gives the average balance

Use Case

- Use cases represent a set of potential scenarios
- Looking at a family of similar scenarios, you can gather the essence of what is typically done
- Similar scenarios will follow similar patterns of work and provide similar types of results
- Normally each use case focuses on a specific purpose
  - e.g. To obtain the current account balance

Use Cases Describe the System

- **A system is described by a finite set of use cases**
- **A system potentially has an infinite number of scenarios**
- **Every use case of a system must be enumerated, otherwise the system will not be functionally complete**

Use Cases are Nice Testable Units

- **Each use case specifies the system functionality**
  - **Its inputs**
  - **Its outputs**
  - **Nature of interaction**
-

Objective

- Be able to discover use cases
- Be able to discover actors
  - Primary Actors
  - Secondary Actors
- Understand use case diagram notation
- Get an introduction to use case specification

Lets Look at Some Sample Use Cases

- – Actor switches directory on a system
- – Actor withdraws cash from an ATM
- – Actor inquires about flight status
- – Actor obtains order information
- – Actor calls another phone

Uses of Mobile Phone

- Display Last Dialed Number
- Display Date & Time
- (List other uses)
- ?
- ?
- ?
- ?
- ?

Who Uses Mobile Phones

- **(List Actors)**
- **?**
- **?**
- **?**

Steps for Making a Call

- Steps for Making a Call

- ?
- ?
- ?
- ?

Uses of a Payroll System

- Add a new employee
- Enter worked hours
- (List other uses)
- ?
- ?
- ?
- ?

Users of a Payroll System

- **(List Actors)**
- **?**
- **?**
- **?**
- **?**

Use Case

- **No “repeats” - only a single operation.**
- **No “ifs” - describe what it will do in any given instance.**
- **Do the ordinary situations first, worry about special cases 2nd.**
- **The set of use cases is your requirements statement.**

Use Case Specification

- Contains all important details about use cases
- Captures critical steps in the actor - system interaction
  - It sequences the interactions
- Declares information about actor
- It partitions the responsibility of the system and the actor
- It lists some pre-conditions that have to be met prior to proper system functionality
- It provides some technical requirements for the service to be of acceptable quality

Simple use case example

- **Point of sales system**
  - **computerized system used to record sales and handle payments**
  - **typically used in retail stores**
  - **hardware and software components:**
    - **computer**
    - **barcode scanner**
    - **software to run the system**

Simple Use Case Example

Actor action

- 1) Customer arrives at POST with items to purchase
- 2) Cashier scans the barcode of each item. If there is more than one of the same item the cashier can enter the quantity as well

System response

- 3) Determines the item price and adds the item information to the running sales transaction.

Simple Use Case Example

Actor action

- 5) on completion of item entry the cashier indicates to the POST that item entry is complete

System response

- 4) The description and price of the current item are presented

- 6) Calculates and presents the sales total

Simple Use Case Example

Actor action

- 7) Cashier tells the customer the total
- 8) Customer gives cash- greater than the sales total
- 9) Cashier records the cash received amount

System response

- 10) Shows the balance due back to the customer

Simple Use Case Example

Actor action	**System response**
• 11) Cashier deposits the cash received and extracts the balance	
	• 12) Logs the completed sales
• 13) Cashier gives the balance and receipt to the customer	
• 14) Customer leaves with items	

Types of Actors

- **Primary Actors**
  - **Payroll Clerk**
  - **Payroll Supervisor**
  - **Tax Accountant**
- **Secondary actors called by the system to complete a use case**
  - **Printer**
  - **Direct Deposit Facility**

Primary Actors

- Active
- Initiate activity with the system
  - Computer user with computer
  - Phone user with phone
  - Payroll clerk with Payroll system
  - Internet subscriber with Web Browser
- Get some value in return

Secondary Actors

- They are passive
- Do not initiate any activity with the system
- Available when the system needs their help
- Usually, machines or other systems
  - Printers, Plotters, Modems, ...
  - Applications
- Possibly human actors, as in work flow
- There to ultimately fulfill the needs of a primary actor

Use Case Diagram is Drawn with Simple Graphics

- **Actors**
  - **Depicted as a stick figure**
    -
- **Use cases**
  - **Depicted as an ellipse with title inside or just below the ellipse**

Basic Use Case Diagram

- System is represented by the box with a title
- Actor is represented by the stick figure outside the box
- Use cases are ellipses inside the box
- Actor to Use Case interactions are shown as a double-ended arrow

Primary Actors on UC Diagram

- **Primary actors shown on the left side of the diagram**
- **Put the name of the use case inside the ellipse**

Secondary Actors on Diagram

- **Secondary actors shown on the right side of the diagram**

Sample Use Case Diagrams