

## Defining Functional Requirements with Actors, Scenarios and Use Cases

Software Engineering Design  
Lecture 2

## Outline of Lecture 2

- Defining Functional Requirements
- Actors
  - Use case diagrams
  - Scenarios
  - Use cases

## Requirements Analysis...

- ... investigates the problem domain as far as possible before moving to the solution domain for design & implementation
- ... results in an analysis model of 3 parts
  - functional model: use cases & scenarios
  - analysis object model: class & object diagrams
  - dynamic model: statechart & sequence diagrams

## The Object Oriented Software Process

- We will focus on *Object Oriented Software Processes*, where documentation is the main artifact, or output, of the analysis and design stages.
- Later we will consider different methods such as Agile Development or Formal Methods which take a different approach to documentation.

## Types of OOSE Documentation

- Problem Statement
- Software Project Management Plan
- Requirements Analysis Document
- Software Design Document
- Object Design Document
- Test Manual
- User Manual

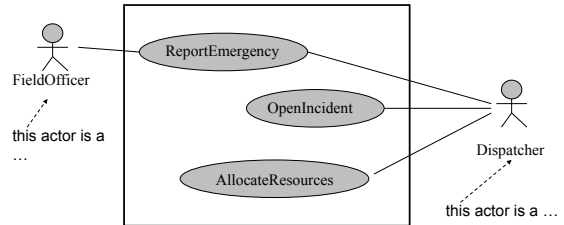


- external entities that interact with the system
- an actor can be a **user role**
- e.g. ...
- an actor can be **another system**
- e.g. ...
- actors have unique names and descriptions
- actors have a **goal** that must be satisfied by the system

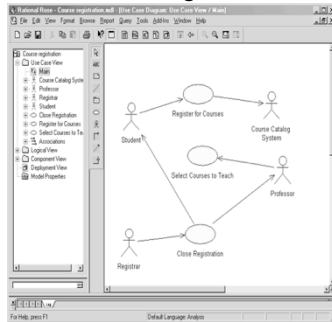
## Use Case Diagrams

- UML notation to
- describe the **functionality** of a system as seen by the users
- in terms of
  - actors and their goals
  - the system boundary: what is in scope and out of the scope of the system?
  - the names of use cases for the system

## FRIEND Use Case Diagram



## Use Case Diagrams



The ovals represent use cases, and the stick figures represent actors, either humans or other systems. The lines represent communication between an actor and a use case. A use-case diagram provides the big picture: Each use case represents a big chunk of functionality that will be implemented, and each actor represents someone or something outside our system that interacts with it.

## Draw a use case diagram based on the following problem statement

- A ticket distributor for a train system includes two actors:
  - a passenger who purchases different types of tickets
  - a central computer system which maintains a reference database for the fares.
- Use cases include BuyOneWayTicket, BuyWeeklyCard, BuyMonthlyCard, UpdateFares.

## Use cases – some history

- Projects have struggled for years to conceive the best approach to elicit, document, and trace functional requirements.
- Approaches range from mini-specifications -- a text narration of the requirements in paragraph form -- to diagrams that show each requirement's flow of control.
- Ivar Jacobson pioneered the notion of use cases while working on complex telecommunications projects at Ericsson

## Scenarios and Use Cases. What?

- ... describe the ways in which a user uses the system
- can be used to gather stories
- can be used to build requirements

## Scenarios and Use Cases. Why?

- Comprehensible by all system stakeholders
  - Use cases model a system from the users' point of view (functional requirements)
    - Define every possible event flow through the system
    - Description of interaction between objects
- Great tools to manage a project. Use cases can form basis for whole development process
  - User manual
  - System design and object design
  - Implementation
  - Test specification
  - Client acceptance test
- An excellent basis for incremental & iterative development

## Scenario (an example by Cockburn)

- System under discussion:
  - the insurance company
- Primary Actor:
  - me, the claimant
- Goal:
  - I get paid for my car accident
- Conditions:
  - Everything is in order
- Outcome:
  - Insurance company pays claim

## Scenario (cont)

1. Claimant submits insurance claim with substantiating data
2. Insurance company verifies claimant owns a valid policy
  - failure here probably means goal failure
3. Insurance company assigns agent to examine case
4. Agent verifies all details are within policy guidelines
  - an interaction between the agent and secondary actors
5. Insurance company pays claimant
  - implies all preceding goals were passed

## Use cases - overview

- The use-case approach focuses first on identifying the actors or users of the application
- Actors have a goal that needs to be satisfied by the system, and they rely on the use case to accomplish that
- Use cases represent major categories of functionality as perceived by the application's user
- A table and text-based template is used to describe each use case

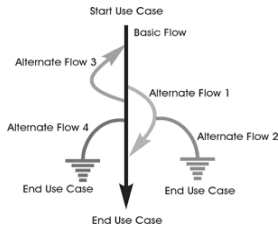
## Warning – don't get confused

- a use case diagram
- is not the same as
- a use case
- Beware mistake on B&D p737 and please correct your copy: the definition of use case diagram includes a definition of a use case

## Use-Case Textual Description

Use Case Section	Description
Name	An appropriate name for the use case – a short active verb phrase e.g. RegisterForCourses
Goal	A brief description of the use case's role and purpose, that is its goal
Flow of Events	A textual description (understandable to the customer) of what the system does with regard to the use case ( <i>not</i> how specific problems are solved by the system).
Special Requirements	Collects all requirements on the use case, e.g. non-functional reqs, that are not considered in the use-case model, but that need to be taken care of during design or implementation.
Preconditions	A textual description that defines any constraints on the system at the time the use case may start.
Post conditions	A textual description that defines any constraints on the system at the time the use case will terminate.

## Basic and Alternate Flow of Events



Use cases describe possible flows of events.

**basic flow of events** describes what "normally" happens when the use case is performed.

**alternate flows of events** covers behavior of an optional or exceptional character relative to normal behavior, and also variations of the normal behavior.

Think of the alternate flows of events as "detours" from the basic flow of events.

## RegisterForCourses Basic Flow of Events

- 1. Logon** This use case starts when a Student accesses the Wylie University Web site. The system asks for, and the Student enters, the student ID and password.
- 2. Select 'Create a Schedule'** The system displays the functions available to the student. The student selects "Create a Schedule."
- 3. Obtain Course Information** The system retrieves a list of available course offerings from the Course Catalog System and displays the list to the Student.
- 4. Select Courses** The Student selects four primary course offerings and two alternate course offerings from the list of available course offerings.
- 5. Submit Schedule** The student indicates that the schedule is complete. For each selected course offering on the schedule, the system verifies that the Student has the necessary prerequisites.
- 6. Display Completed Schedule** The system displays the schedule containing the selected course offerings for the Student and the confirmation number for the schedule.

## Some Alternate Flows of Events 1

### 1. Unidentified Student

In Step 1 of the Basic Flow, Logon, if the system determines that the student ID and/or password is not valid, an error message is displayed

### 2. Quit

The Course Registration System allows the student to quit at any time during the use case. The Student may choose to save a partial schedule before quitting. All courses that are not marked as "enrolled in" are marked as "selected" in the schedule. The schedule is saved in the system. The use case ends.

## Alternate Flow of Events 2

### 3. Unfulfilled Prerequisites, Course Full, or Schedule Conflicts

In Step 5 of the Basic Flow, Submit Schedule, if the system determines that prerequisites for a selected course are not satisfied, that the course is full, or that there are schedule conflicts, the system will not enroll the student in the course. A message is displayed that the student can select a different course. The use case continues at Step 4, Select Courses, in the basic flow

## Alternate Flow of Events 3

### 4. Course Catalog System Unavailable

In Step 3 of the Basic Flow, Obtain Course Information, if the system is down, a message is displayed and the use case ends

### 5. Course Registration Closed

If, when the use case starts, it is determined that registration has been closed, a message is displayed, and the use case ends.

## Using Use Cases - Summary

- Writing a use case involves a lot of work
- Ideally, the flows should be written as "dialogs" between the system and the actors
- Each step should explain what the actor does and what the system does in response
- It should also be numbered and have a title
- Alternate flows always specify where they start in the basic flow and where they go when they end