Defining Functional Requirements with Actors, Scenarios and Use Cases

Software Requirements and Design
CITS 4401
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 (OOSE)

- We will focus on Object Oriented Software Processes, where documentation is the main 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 – not covered in this unit
- User Manual – not covered in this unit

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

Actors
- 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

FRIEND Use Case Diagram
- Dispatcher
- FieldOfficer
- ReportEmergency
- OpenIncident
- AllocateResources
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.

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, 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.

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, BuyMonthlyCard, UpdateFares.

Scenarios vs. use cases

- Scenario
  - an instance of a use case
  - a sequence of steps describing an interaction between a user and a system
  - used to gather stories and build requirements
  - focus is on understandability

- Use case
  - an abstraction that describes all possible scenarios involving the described functionality
  - focus is on completeness
Scenarios and Use Cases. Why?

- Comprehensible by all system stakeholders
  - Describe functional requirements from the users’ point of view
- Great tools to manage a project. They 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 example (from FRIEND system)

- Scenario name: warehouseOnFire
- Participating actors:
  - bob – FieldOfficer
  - james - Dispatcher
- Flow of events:
  - bob is driving down main street in a patrol car and notices smoke coming out of a warehouse. He activates the ReportEmergency function of FRIEND
  - bob enters the details into the system
  - james is alerted by a several beeps. He reviews the information and records the incident using the OpenIncident function
  - james allocates a fire unit and two paramedic units to the scene with the AllocateResources function

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
Use-Case Textual Description

<table>
<thead>
<tr>
<th>Use Case Section</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>An appropriate name for the use case – a short active verb phrase e.g. RegisterForCourses</td>
</tr>
<tr>
<td>Goal</td>
<td>A brief description of the use case’s role and purpose, that is its goal</td>
</tr>
<tr>
<td>Flow of Events</td>
<td>A textual description (understandable to the customer) of what the system does with regard to the use case (not how specific problems are solved by the system).</td>
</tr>
<tr>
<td>Special Require-</td>
<td>Collections 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.</td>
</tr>
<tr>
<td>Precon-Conditions</td>
<td>A textual description that defines any constraints on the system at the time the use case may start.</td>
</tr>
<tr>
<td>Post con-Conditions</td>
<td>A textual description that defines any constraints on the system at the time the use case will terminate.</td>
</tr>
</tbody>
</table>

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.

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.

Writing 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