Interaction Modelling: Use Cases

Marlon Dumas (replacing Fabrizio Maggi)
Institute of Computer Science

(These slides are derived from the book “Object-oriented modeling and design with UML”)
Object-Oriented Models & Code

- Domain Classes; Attributes; Relations
- Domain (Class) Model
- Interaction Modelling
- Application (Class) Model
- Code

Domain Classes; Attributes; Relations

- Domain (Class) Model
- Interaction Modelling
- Application (Class) Model
- Code
Domain (Class) Model

WHAT?
Object-Oriented Models & Code

Identifying the right interfaces/operations

Domain (Class) Model ➔ Interaction Modelling ➔ Application (Class) Model

Code

Systems modelling – Fabrizio Maria Maggi
Application (Class) Model
Software Development

Domain (Class) Model

Interaction Modelling

Application (Class) Model

Domain Classes; Application Classes (e.g., Patterns); Attributes; Relations; Operations

Code

Domain Classes; Application Classes (e.g., Patterns); Attributes; Relations; Operations
Interaction Modeling - Key Concepts

- Use case
  - Use case diagram
  - Use case description
- Scenario
  - Sequence diagram (next week)
Use cases

- A use case is a **contract** of an interaction between the system and its actor(s) to deliver a logical unit of functionality.

- A use case model usually comprises:
  - A diagram, describing relations between use-cases and actors.
  - A textual description of each use case.
Purpose of use case diagrams

- Create a semi-formal model of the functional requirements
- Analyze and define:
  - Scope
  - External interfaces
  - Scenarios and reactions
Use Case Diagram

- Actor
- Use case
- System boundary
- Association
Finding Actors

- External objects that produce/consume data:
  - Must serve as sources and destinations for data
  - Must be external to the system

Humans  Machines  External systems

Organizational Units  Sensors
Use Cases: Actors

- An actor is a direct external user of a system
  - An object or a set of objects that communicates directly with the system but that is not part of the system
- Examples
  - Customer and Repair Technician are actors of a vending machine
  - Traveler, Agent and Airline are actors of a travel agency system
  - User and Administrator are actors for a computer database system
Exercise

- Who are the actors in ŌIS (Study Information System)?
- For each actor, name at least three use cases.
Actor Generalization

- The child actor inherits all Use Case associations

Actor generalization should be used if the specific actor has more responsibility than the generalized one (i.e. associated with more Use Cases)
A use case is a coherent piece of functionality that a system can provide by interacting with actors.

- **Buy a beverage.** The vending machine delivers a beverage after a customer selects and pays for it.
- **Perform scheduled maintenance.** A repair technician performs the periodic service on the vending machine necessary to keep it in good working condition.
- **Make repairs.** A repair technician performs the unexpected service on the vending machine necessary to repair a problem in its operation.
- **Load items.** A stock clerk adds items into the vending machine to replenish its stock of beverages.

Figure 7.1 Use case summaries for a vending machine. A use case is a coherent piece of functionality that a system can provide by interacting with actors.

Each use case involves one or more actors as well as the system itself

Example: the use case “Buy a beverage” involves the Customer; the use case “Perform scheduled maintenance” involves the Repair Technician; in a telephone system the use case “Make a call” involves two actors, a Caller and a Receiver

An actor is not necessarily a person

Example: in an online shop the use case “Checkout” involves the Web Customer and the Credit Payment Service

A use case partitions the functionality of the system into a mainline behavior sequence, variations on normal behavior, exception conditions, error conditions, cancellations of a request

Use cases should all be at a comparable level of abstraction

Examples: “Make telephone call” and “Record voice mail message” are at a comparable level; “Set external speaker volume to high” is too narrow, “Set speaker volume” or even “Set telephone parameters” would be better
Linking Use cases

- Linking enables flexibility in requirements specification
  - Isolating functionality
  - Enabling functionality sharing
  - Breaking functionality into manageable chunks

- Three linking mechanism are available in UML:
  - Include
  - Extend
  - Generalization

- And one grouping mechanism:
  - Packages
INCLUDE

- Include is used when:
  - Decomposing complicated behavior
  - Centralizing common behavior
- The base use case explicitly incorporates the behavior of another use case at a location specified in the base
The base use case can incorporate another use case at certain points, called extension points.

Note the direction of the arrow.

The base use-case does not know which use-case extends it.
GENERALIZATION

- The child Use case inherits the behavior of the parent Use case:
  - The interaction (described in the textual description)
  - Use case links (associations, include, extend, generalization)
- The child Use case can substitute the parent Use case
  - Overriding occurs through the textual description
Packaging: use cases organized in packages

![Diagram of packaging use cases](image)
Exercise

- Define at least 3 packages of use cases in ÕIS (Study Information System)?
  - For each package, show at least five use cases.
- What criteria did you use for packaging and why?
Use Case Description

<table>
<thead>
<tr>
<th>Name</th>
<th>Actors</th>
<th>Triggers</th>
<th>Preconditions</th>
<th>Post-conditions</th>
<th>Success scenario</th>
<th>Alternative flows</th>
</tr>
</thead>
</table>
# Use Case Description

<table>
<thead>
<tr>
<th>Name</th>
<th>What starts the use-case?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actors</td>
<td>Examples:</td>
</tr>
<tr>
<td><strong>Triggers</strong></td>
<td>▶ Customer reports a claim</td>
</tr>
<tr>
<td>Preconditions</td>
<td>▶ Customer inserts card</td>
</tr>
<tr>
<td>Post-conditions</td>
<td>▶ System clock is 10:00</td>
</tr>
<tr>
<td>Success scenario</td>
<td></td>
</tr>
<tr>
<td>Alternative flows</td>
<td></td>
</tr>
</tbody>
</table>

- **Triggers**
  - What starts the use-case?
  - Examples:
    - Customer reports a claim
    - Customer inserts card
    - System clock is 10:00
Use Case Description

- What the system needs to be true before running the use-case.
- Examples
  - User account exists
  - User has enough money in his/her account
  - There is enough disk space
# Use Case Description

<table>
<thead>
<tr>
<th>Name</th>
<th>Actors</th>
<th>Triggers</th>
<th>Preconditions</th>
<th>Post-conditions</th>
<th>Success scenario</th>
<th>Alternative flows</th>
</tr>
</thead>
</table>

- **Post-conditions**
  - A post-condition is the outcome of the use-case.
  - **Examples**
    - Money was transferred to user account
    - User is logged in
  - The minimal things a system ensures (even in case of failures)
  - **Example**
    - Money is not transfer unless the customer
  - What happens when the Use Case concludes successfully?
  - **Example**
    - The money is transferred
# Use Case Description

<table>
<thead>
<tr>
<th>Name</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actors</td>
<td>Actors</td>
</tr>
<tr>
<td>Triggers</td>
<td>Triggers</td>
</tr>
<tr>
<td>Preconditions</td>
<td>Preconditions</td>
</tr>
<tr>
<td>Post-conditions</td>
<td>Post-conditions</td>
</tr>
<tr>
<td><strong>Success scenario</strong></td>
<td><strong>Success scenario</strong></td>
</tr>
<tr>
<td>Alternative flows</td>
<td>Alternative flows</td>
</tr>
</tbody>
</table>

- The success scenario is the main story-line of the use-case.
- It is written under the assumption that everything is okay, no errors or problems occur, and it leads directly to the desired outcome of the use-case.
- It is composed of a sequence of action steps.
- **Example**
  1. Administrator enters course name, code and description
  2. System validates course code
  3. System adds the course to the db and shows a confirmation message
Use Case Description

- **Name**
- **Actors**
- **Triggers**
- **Preconditions**
- **Post-conditions**
- **Success scenario**
- **Alternative flows**

- **Errors**
  - Case did not eject properly
  - Any network error occurred during steps 4-7
  - Any type of error occurred

- **Unusual or rare cases**
  - Credit card is defined as stolen
  - User selects to add a new word to the dictionary

- **Endpoints**
  - The system detects no more open issues

- **Shortcuts**
  - The user can leave the use-case by clicking on the “esc” key

- Used to describe exceptional functionality
Exercise

- Specify one (non-trivial) use case in ŌIS.
Alternatives to use cases

- User stories (agile methods)
  - Written in the language of the user
  - Captures: Who, what, why
  - As a student, I want to search all courses in my Masters that are offered in the current semester and that I have not yet passed, in order to register to some of them.

- Storyboards (i.e. “detailed” user stories)
  - Sketch of how a user will perform a task
  - Shows the interactions and relevant objects (or screens) at each step
  - Used in UI Design (but not only)
  - More on storyboards next week
Summary

- Use case models include
  - Use case diagrams
  - (Textual) use case descriptions
- Simple use case template available in the “Lectures” section of course web page