Use Cases

Massimo Felici

IF 3.46 0131 650 5899

mfelici@inf.ed.ac.uk

Use Cases

- Support **requirements engineering** activities and the requirement process
- Capture what a system is supposed to do, i.e., system's **functional requirements**
- Describe **sequences of actions** a system performs that yield an **observable result** of value to a **particular actor**
- Model actions of the system at its **external interface**
- Capture how the system coordinates human actions

The Benefits of Use Cases

- Relatively **easy** to write and easy to read
- Comprehensible by **users**
- Engage the **users** in the requirements process
- Force developers to think through the design of a system from a **user viewpoint**
- Identify a **context** for the requirements of the system
- Critical tool in the **design**, **implementation**, **analysis** and **testing process**
- Rapid **change** allows exploratory approach
- Serve as inputs to the user documentation

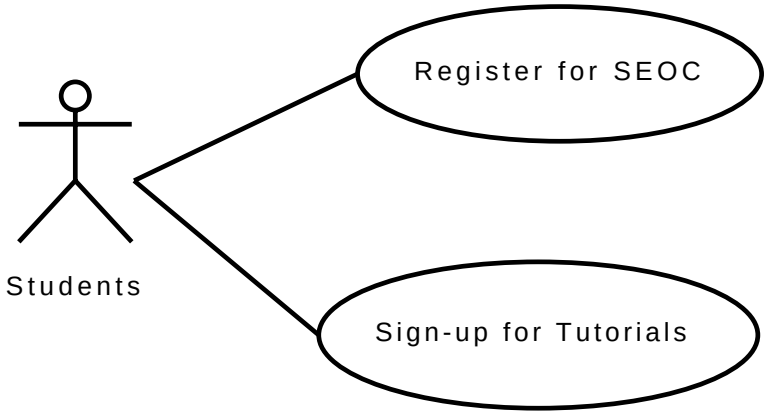
Use Cases: Strengths and Weaknesses

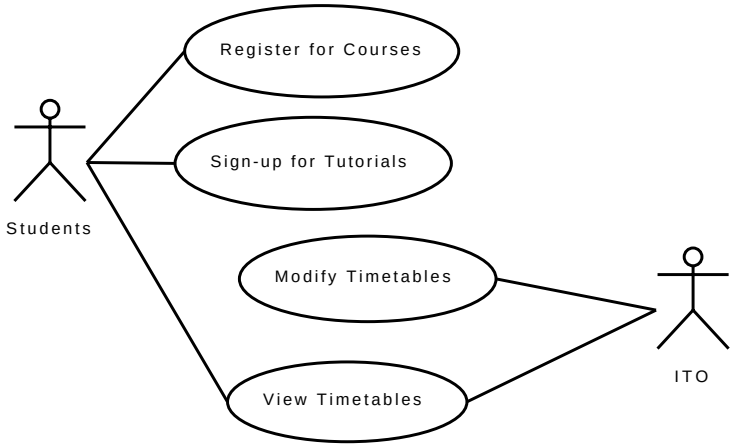
- **Strengths**
  - Capture different actors views of the system
  - Capture some structures in requirements
  - Are comprehensible by naïve users
- **Weaknesses**
  - Lack of non-functional requirements
  - Lack of what the system shall not do

Use Cases at a Glance

Use Cases	Relationships	
Use Case Use Case	Generalization	**Actors** Actor
Abstract Use case *Abstract Use Case*	<<include>> <<include>>	**System Boundaries**
Use Case with Extension Points Use Case with extension points Extension Point:	<<extend>> <<extend>>	**Extension Conditions** Extension Condition

Actors and Use Cases

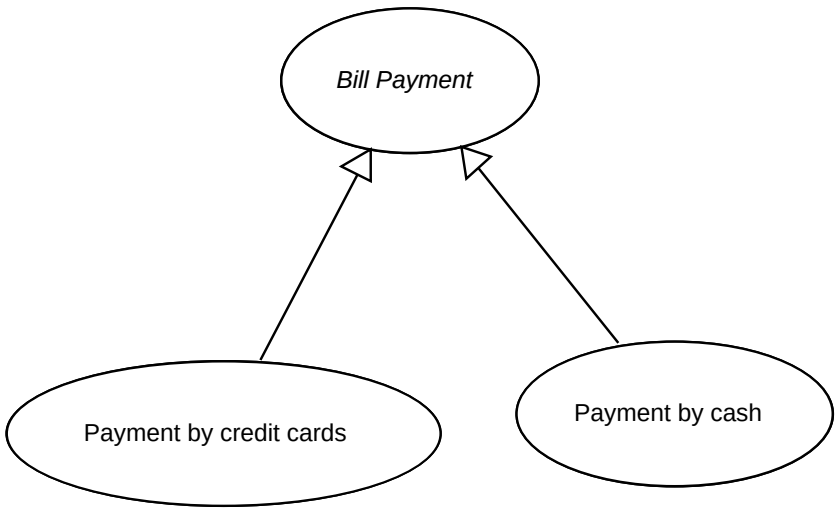




Warnings and Hints

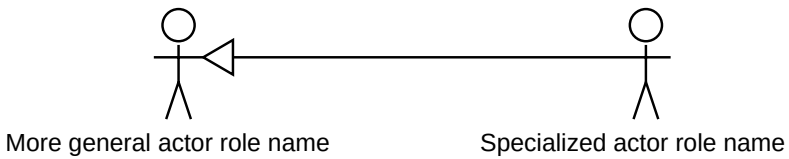
- Finding nonhuman actors
  - Incorporating other systems (e.g., databases)
  - Ignoring internal components
  - Input/Output Devices
- Roles of the Actors
- Naming the Actors

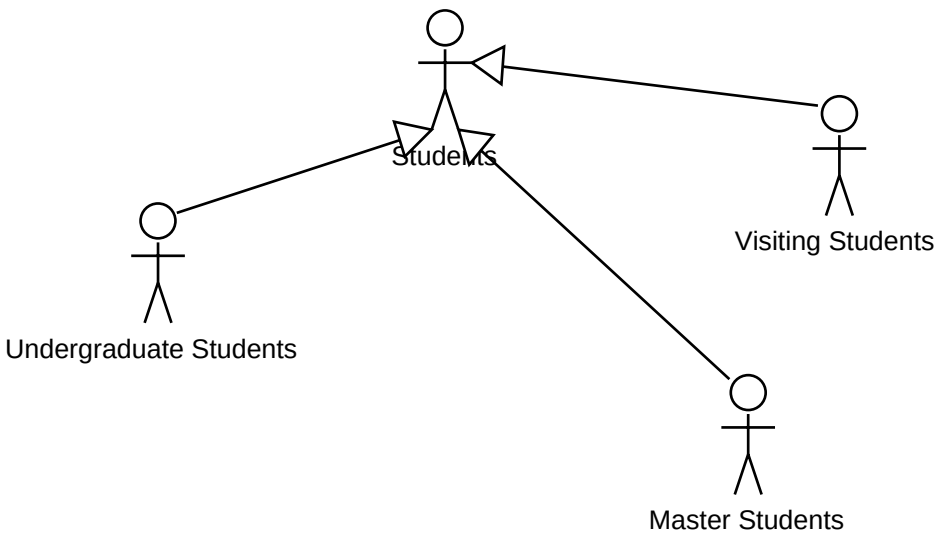
Generalizations between Use Cases



- Indicate that the more specific use case receives or inherits the actors, behavior sequences, and extension points of the more general use case
- Payment, for instance, is a generalization of Payment by credit cards and payment by cash

Generalization between Actors

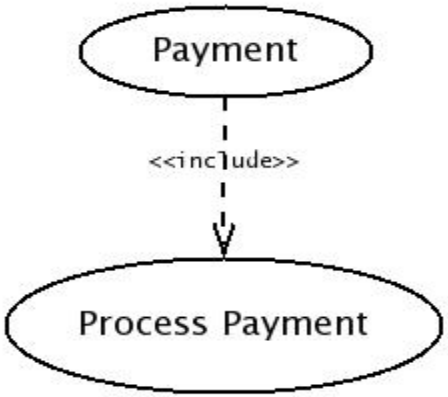


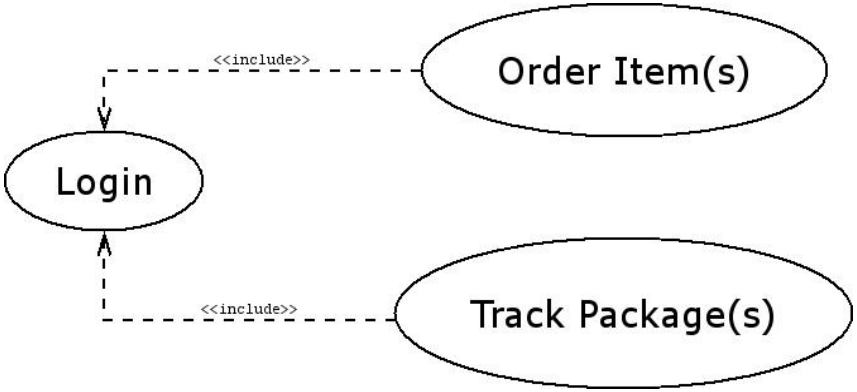


- Actors may be similar in how they use the system (e.g., project and system managers)
- An Actor generalization indicates that instances of the more specific actor may be substituted for instances of the more general actor

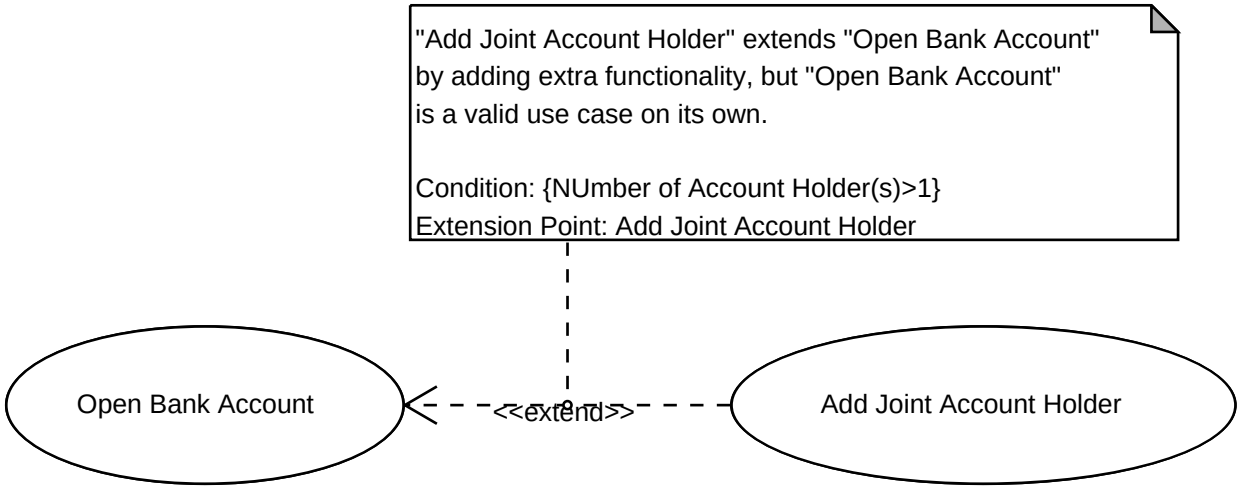
<<include>> Relationship

- The **<<include>>** relationship holds when one use case is included in others
- The **<<include>>** relationship declares that a use case reuses another one being included
- The included use case is (typically not complete on its own) a required part of other use cases



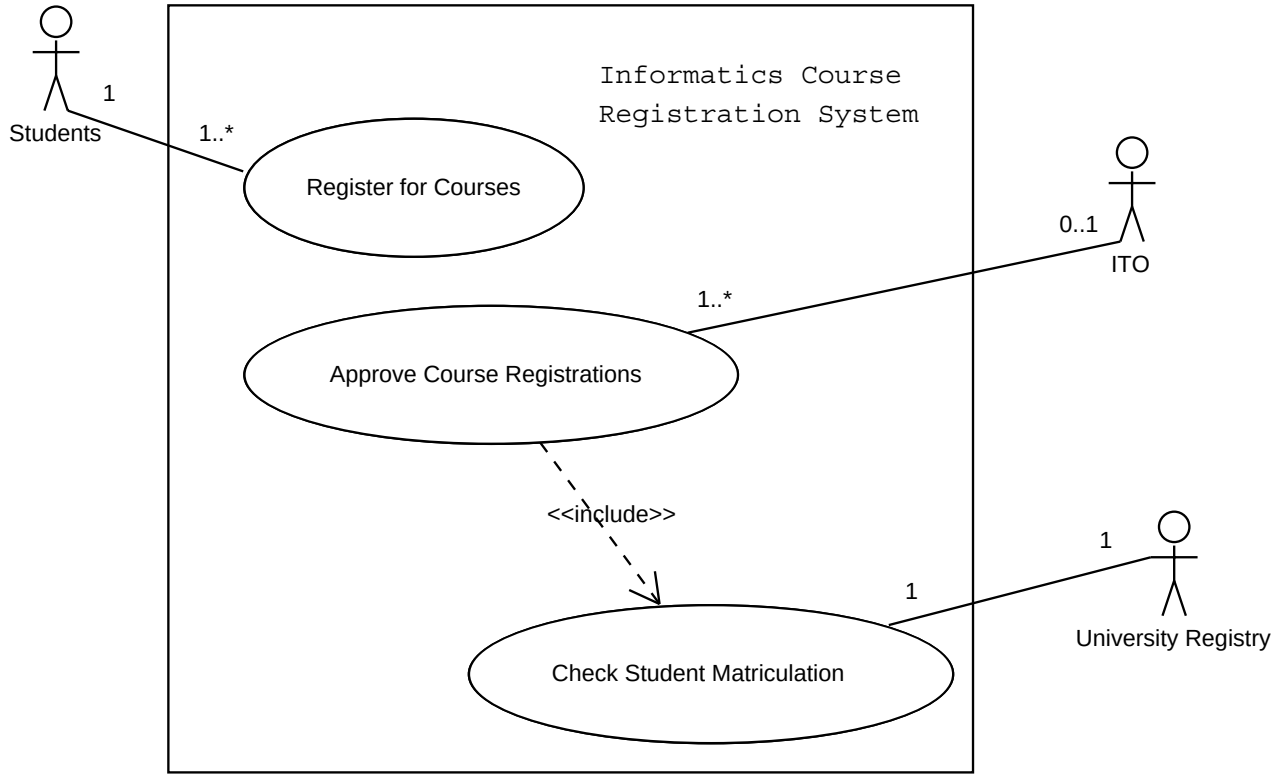


<<extend>> Relationship



- The <<extend>> relationship holds when use cases extend, i.e., optionally provide other functionalities to extended use cases
- A use case may be extend by (i.e., completely reuse) another use case, but this is optional and depends on runtime conditions or implementation decisions
- Any use case you want to extent must have clearly defined extensions points

System Boundaries



- Identify an implicit separation between actors (**external** to the system) and use cases (**internal** to the system)

Use Case Descriptions

- A use case description should be attached to each case in the diagram
- Complement use case diagrams
- Identify use case information
  - **Warnings: avoid to specify design information**
- A use case main course (of actions) is a generic sequence of actions undertaken in using the system
  - Identify pre and post conditions
  - Identify alternate courses
- Provide generic test scenarios for the full system
- Templates capture/structure use case information

Basic Use Case Template

Use Case: <number> <the name should be the goal as a short active verb phrase>
Goal in Context: <a longer statement of the goal, if needed>
Scope: <What system is being considered black-box under design>
Level: <one of Summary, Primary task, Subfunction>
Primary Actor: <A role name for the primary actor, or description>
Priority: <How critical to your system/organisation>
Frequency: <How often it is expected to happen>

Another Use Case Template

Use Case: Use case identifier and reference number and modification history
Description: Goal to be achieved by use case and sources for requirements
Actors: List of actors involved in use case
Assumptions: Conditions that must be true for use case to terminate successfully
Steps: Interactions between actors and system that are necessary to achieve the goal
Variations (optional): any variations in the steps of a use case
Non-Functional (optional): List of non-functional requirements that the use case must meet.
Issues: List of issues that remain to be solved

Using a Use Case Template

1. Learn to fill in all the fields of the template in several passes
2. Stare at what you have so far
3. Check your project's scope
4. Identify the open issues and a deadline for the implementation
5. Identify all the systems to which you have to build interfaces

Creating Use Cases

- **Step 1.** Identify and Describe the **Actors**
- **Step 2.** Identify and Describe the **Use Cases**
- **Step 3.** Identify the (Actor and Use Case) **Relationships**
- **Step 4.** individually **Outline** Use Cases
- **Step 5.** **Prioritize** the Use Cases
- **Step 6.** **Refine** the Use Cases

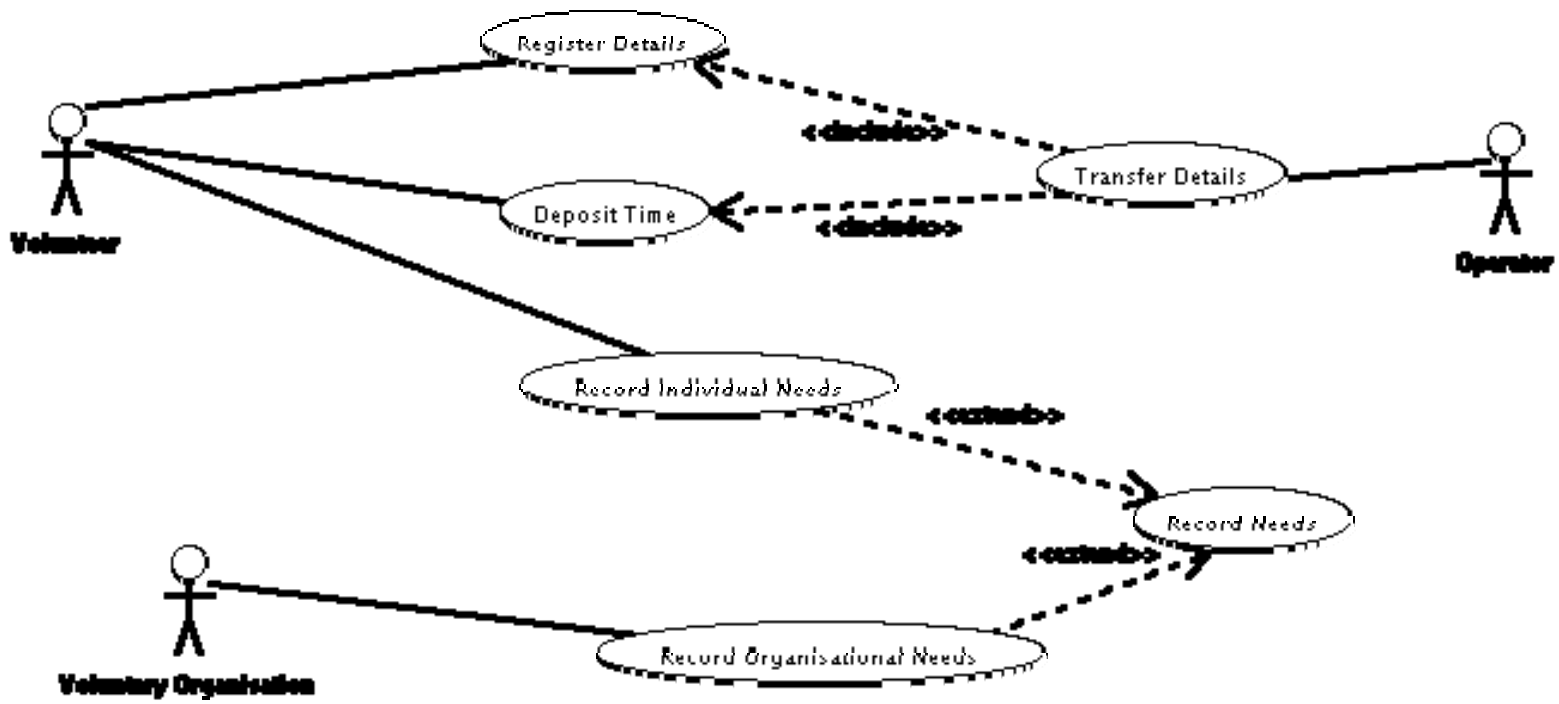
Building the Right System

- **Tracing** Requirements
- Managing **Changes**
- Assessing Requirements Quality in **Iterative Development**

VolBank: Creating Use Cases

- Who are the main actors in the VolBank example?
- Can you identify all the main use case names in the system?
- What opportunities are there to structure the use case diagram?
- Can you see any non-functional requirements that are present in the specification?
- How well are non-functional requirements represented in the use case diagram?

VolBank: Incomplete Use Cases



VolBank: Using Use Case Template

Use Case: 01 - deposit time
Goal in Context: *The VolBank system allows volunteers to deposit their availabilities in terms of time*
Scope: *volunteers' profiles are unavailable*
Level: Primary task
Primary Actor: Volunteers
Priority: *It supports one of the major functionalities of the VolBank system*
Frequency: *Every time volunteers provide information about their availability*

Readings

- **UML course textbook**
  - Chapter 3 on Use Cases
- Alistair Cockburn. Structuring Use Cases with Goals.
  - The paper introduces a **Basic Use Case Template**

Summary

- Use Cases in UML capture (to a certain extent) system requirements and support requirements engineering activities and processes
- Use Case notations and examples
- Describing use cases
- Developing use cases