Use Cases

Massimo Felici

Room 1402, JCMB, KB

0131 650 5899

mfelici@inf.ed.ac.uk

Use Cases

- Specify only 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**
  - High level view of the **system**
  - Capture some **structure**
- Capture how the system coordinates **human action**
  - Link to scenarios keeps the activity concrete
- Rapid **change** allows exploratory approach
- Comprehensible by **users**

Use Cases

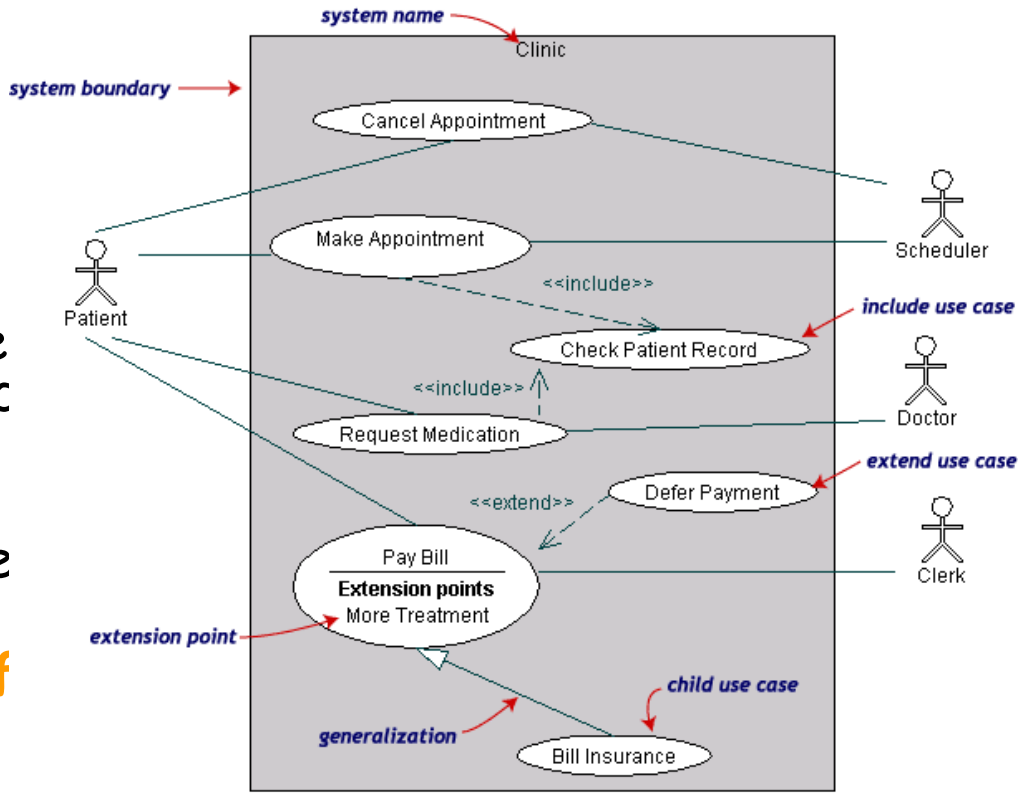
- Support Requirements Engineering
  - It is an effective means of communicating with users and other stakeholders about the system and what is intended to do
- **Strengths**
  - capture different actors views of the system
  - comprehensible by naïve users
  - capture some elements of structure in requirements
- **Weaknesses**
  - **Fail to capture non- functional requirements**
  - Fail to capture what the system shall not do
  - Do not support analysis particularly well

The Benefits of Use Cases

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

Use Cases at a Glance

- A use case describes **sequences of actions** a **system performs** that yield **an observable result of value** to **a particular actor**
  - **Sequence of actions**: se of functions, algorithmic procedures, internal processes, etc.
  - **System performs**: syste functionalities
  - **An observable result of value** to a user
  - **A particular actor**: individual or device



Anatomy of Use Cases

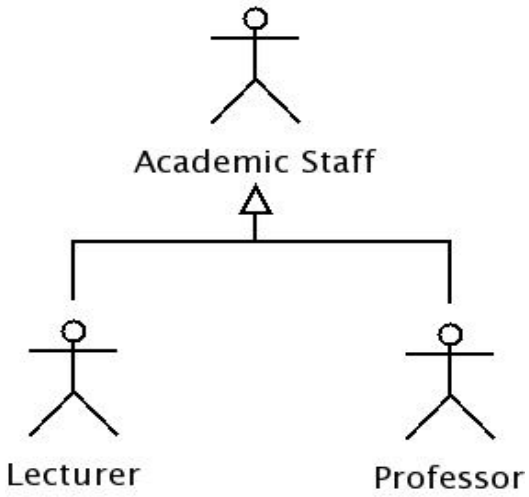
- **Basic Diagrams**
  - actors are represented as stick figures
  - use cases as ellipses
  - lines represent associations between these things
  - basic use case diagrams show who is involved with what.
- Can be used to help in **structuring systems**
  - For example, the scheduler and patient more or less form a sub-system – look at delegating appointment management to a single component or sub-system.
- Take care to identify generic actors who do a particular **task**
  - **Do not** get confused with job titles, etc.
- Use case diagrams should not be too complex
  - Aim for reasonably generic use cases
  - try not be too detailed at first

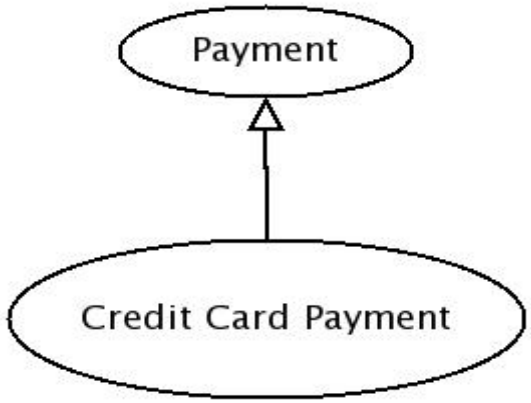
Use Case Basics

- **Actors**
  - An Actor is **external** to a system, **interacts** with the system, may be a **human user or another system**, and has a **goals** and **responsibilities** to satisfy in interacting with the system.
- **Use Cases**
  - identify **functional requirements**, which are described as a sequence of steps
  - describe **actions** performed by a system
  - capture **interactions** between the system and actors.
- **Relationships**
  - Actors are connected to the use cases with which they interact by a line which represents a relationship between the actors and the use cases.
- **System Boundaries**
  - Identify an implicit separation between actors (external to the system) and use cases (internal to the system)

Generalizations

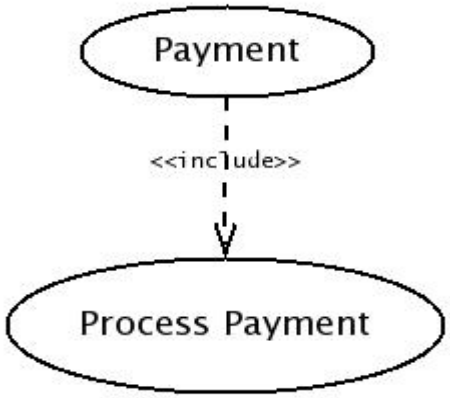
- **Actor Generalizations**:
  - Actors may be similar in how they use the system (e.g., project and system managers)
  - An Actor generalization indicates that instances of the more specific actor may be substituted for instances of the more general actor
  - E.g., A "health worker" is a generalization of "nurse", "doctor" etc.
- **Use Case Generalizations**:
  - Indicate that the more specific use case receives or inherits the actors, behavior sequences, and extension points of the more general use case
  - E.g., pay bill is a generalization of bill insurance

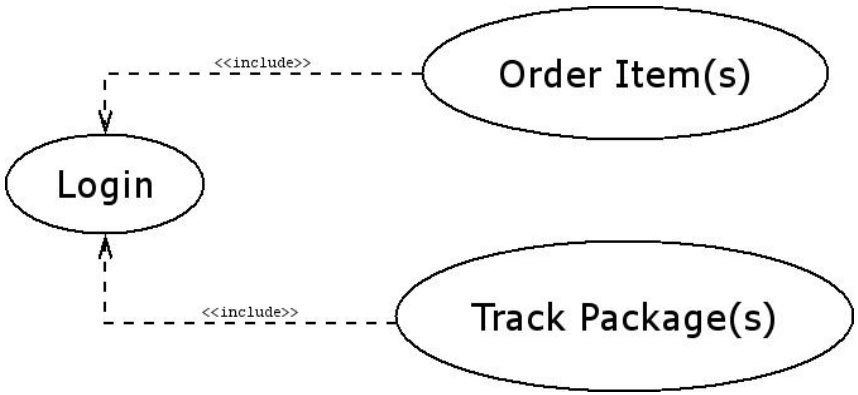




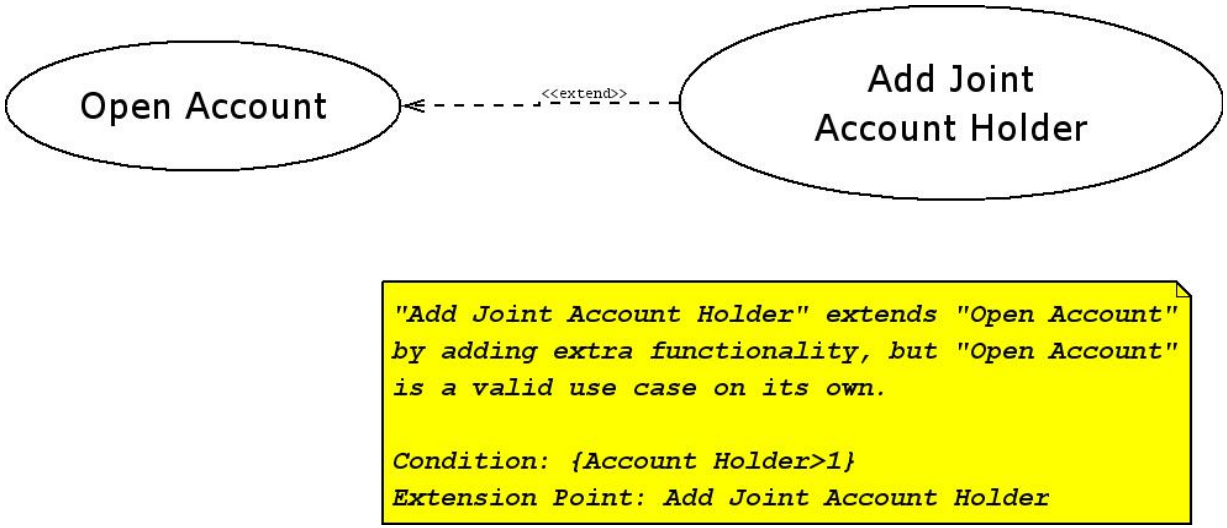
The <<include>> Relationship

- 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





The <<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**

Use Case Descriptions

- A use case description should be attached to each case in the diagram
- Provide further details for design
- Identify use case information
- Complement use case diagrams
- Provide generic test scenarios for the full system
- **Templates** capture/structure use case information
  - Some types of information, e.g., actors, related requirements, preconditions, successful/failed end conditions, etc.

- A use case main flow is a generic sequence of actions undertaken in using the system
  1. **Patient**: request appointment to scheduler
  2. **Scheduler**: queries System for available times
  3. **System**: responds with times
  4. **Scheduler**: negotiates with Patient on suitable time
  5. **Scheduler**: confirms time with system
  6. **System**: responds with confirmation of appointment (e.g. booking number)
  7. **Scheduler**: communicates confirmation to Patient

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

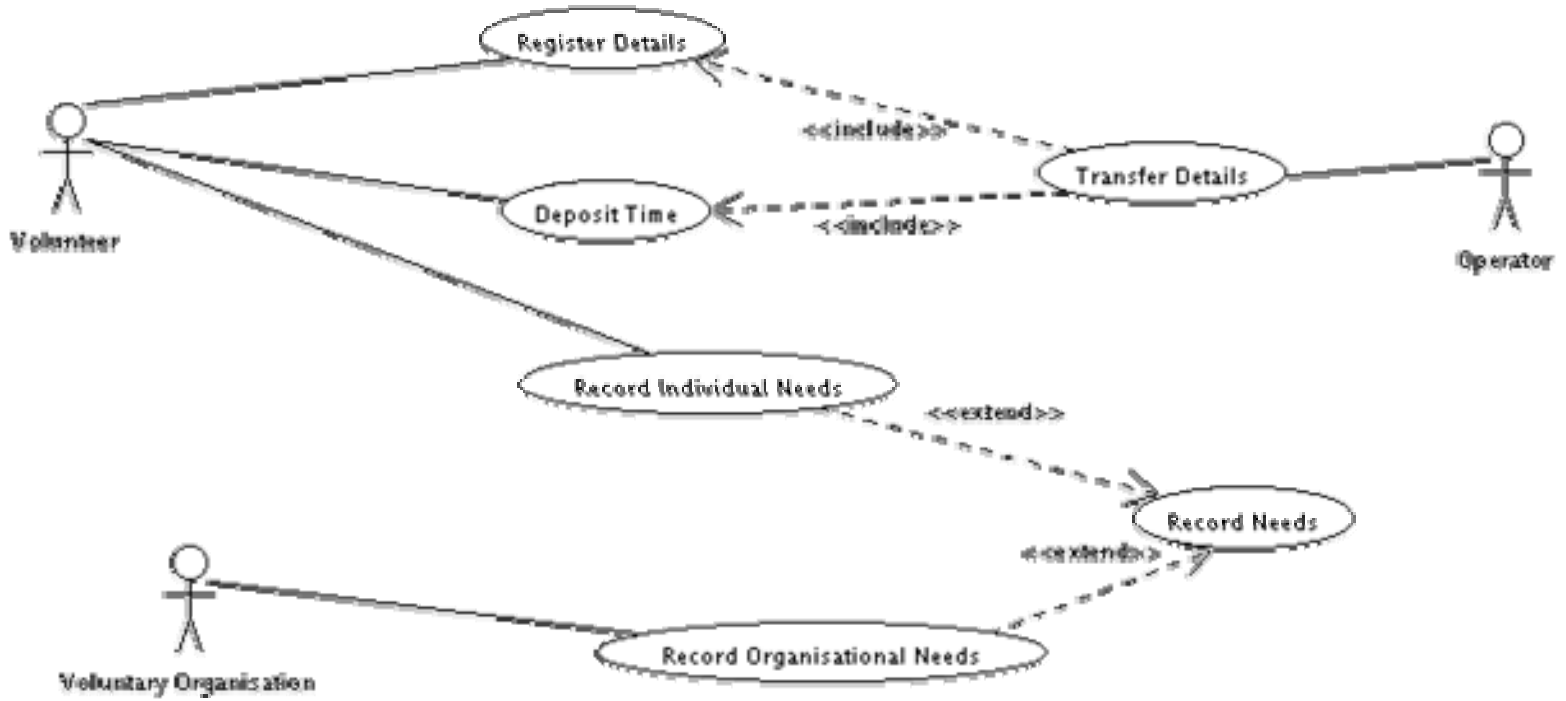
Creating Use Cases

- **Step 1**. Identify and Describe the **Actors**:
  - can use checklists: who uses the system? who manages the system? who maintains the system? Who provides information to the system? Who gets information from the system? etc.
- **Step 2**. Identify and Describes the **Use Cases**:
  - What will the actor use the system for? Will the actor create, store, change, remove or read information in the system? etc.
- **Step 3**. Identify the Actor and the Use Case **Relationships**
- **Step 4**. Outline the individual **Use Cases**
- **Step 5**. **Prioritize** the use cases
  - for instance, on the basis of utility or frequency of use
  - depending on the process this may be closely linked to what is needed in the process
- **Step 6**. **Refine** the Use Cases
  - Develop each use case (starting with the priority ones)
  - develop the associated use case structure the use case

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*

Building the Right System

- **Tracing Requirements**
  - From Use Cases to **Implementation**
    - Mapping requirements to design and code
    - **Orthogonality problem**: the structure of requirements and the structure of design and implementation are different
  - System architecture
  - From Use Cases to **Test Cases**
    - A scenario, or an instance of use case, is an use case execution wherein a specific user executes the use case in a specific way
  - Requirements dependencies
- Managing **Change**
  - Stakeholders interaction, business constraints, implementation issues, system usage and so on may trigger requirements changes
- Assessing Requirements Quality in **Iterative Development**
  - Successive refinement, rather than absolute completeness, or specificity, is the goal

Reading/Activity

- Please read
  - The **Volere template**
    - **You may want to use the Volere Template as support to structure your course project's requirements**
  - Alistair Cockburn's paper **Structuring Use Cases with Goals**
    - **You may want to use a Use Case Template to collect and represent your course project's use cases**
  - Articles on Modeling