Requirements Engineering

Lecture 3

Ruzanna Chitchyan, Jon Bird, Pete Bennett
TAs: Alex Elwood, Alex Cockrean, Casper Wang

Overview

- What are requirements?
- Stakeholder Identification
- Functional and Non-Functional Requirements
- Describe system behavior and capture it in a model with Use Case Model
    - Use Case Diagram
    - Use Case Specification
- Requirements Quality

Requirements

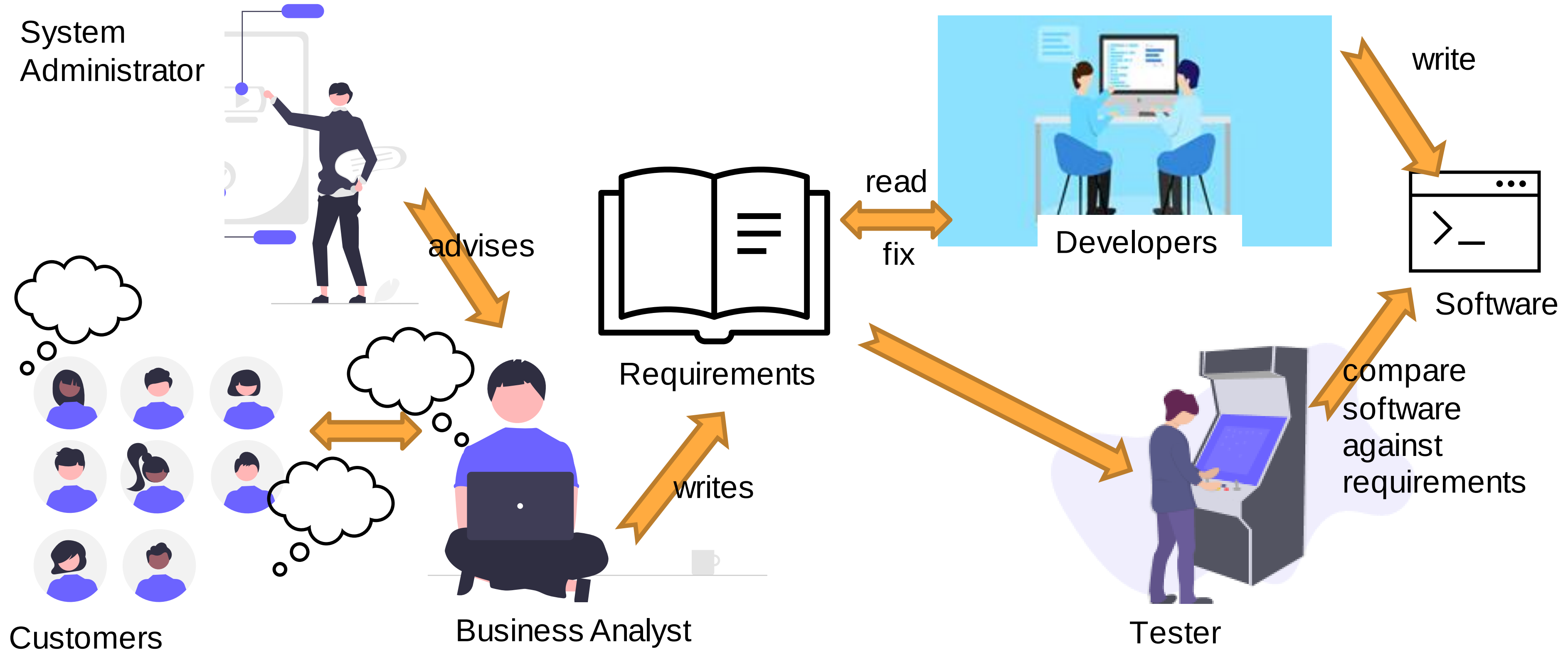
- **System requirements** specify a system, not in terms of system implementation, but in terms of user observation. Requirements record description of the system features and constraints.
  - **Functional requirements** specify user interactions with the system, they say **what** the system is supposed to do:
    - Statements of services the system should provide
    - How the system should react to particular inputs
    - How the system should behave in particular situations
    - May also state what the system should NOT do
  - **Non-functional requirements** specify other system properties, they say **how** the functional requirements are realised:
    - Constraints ON the services or functions offered by system
    - Often apply to whole system, not just individual features

Why do we need Requirements Engineering?

General Problems and Requirements Engineering

- Inconsistent terminology: people express needs in **their own words**
- **Conflicting** needs for the same system
- People frequently **don’t know** what they want (or at least can't explain !)
- Requirements **change** quite frequently
- Relevant people/information may **not be accessible**

Requirements are **communication mechanism**



Requirements are **instructions**

- On your own or in pairs
- Follow some instructions to draw.
- I'll then judge whether you followed the instructions correctly.
- NOT your artistic skill!

Drawing Activity: 5 min.

Requirements are **acceptance criteria**

- To be able to fairly assess whether the team have produced something that matches what you asked for, the thing that you asked for must be:
  - Unambiguous / Precise
  - Complete
  - Understandable / Clear

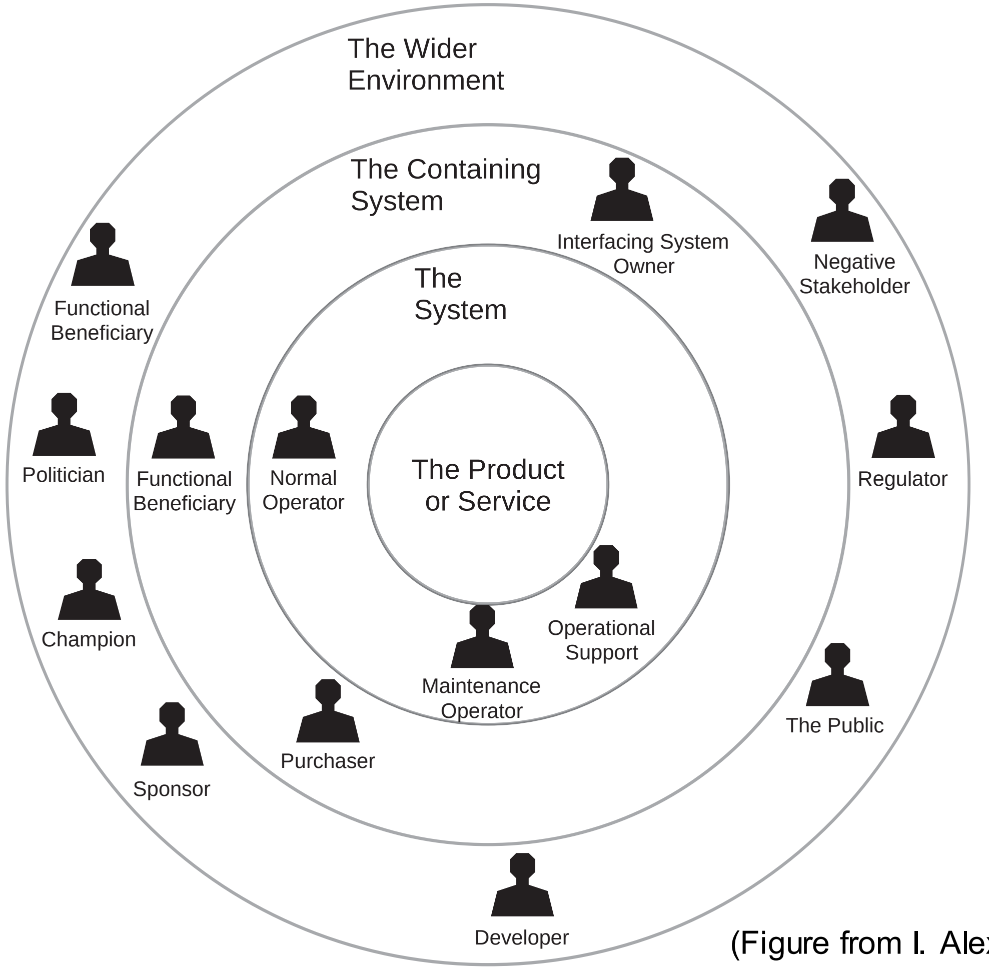
Analysing Requirements

1. Identify <u>stakeholders</u> involved with the system
2. Identify <u>top-level user needs</u> (e.g., as NFRs or “user stories”)
3. Break down stories into individual steps / Refine requirements
4. Specify atomic requirements (e.g., for each step in user stories)

Let’s look at each of these stages in turn…

1. Identify Stakeholders

The Onion model



(Figure from I. Alexander's book)

Stakeholders

Identification

- Clients
- Documentation, e.g., organisation chart
- Templates (e.g., onion model)
- Similar projects
- Analysing the context of the project

Keeping in mind:

代理的

- Surrogate stakeholders (e.g., legal, unavailable at present, mass product users)
- Negative stakeholders

2. Identify top level needs/concerns

Identify "User Stories"

A popular way to record user needs…

As a < type of user >, I want to < some goal >
so that < some reason >.

As a student, I want to be able to register for a module, so that I can learn about topics of interest to me.

As a customer, I want to be able to pay for a university course, so that I can attend the lectures to get a degree.

As a customer, I want to have my data kept securely, so that my privacy is protected.

What Is System Behavior?

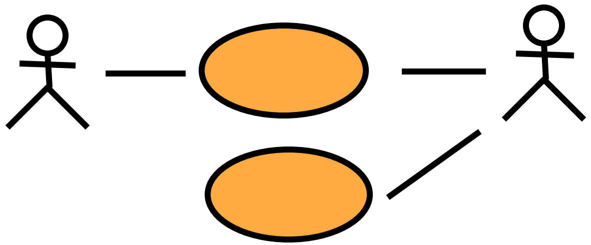
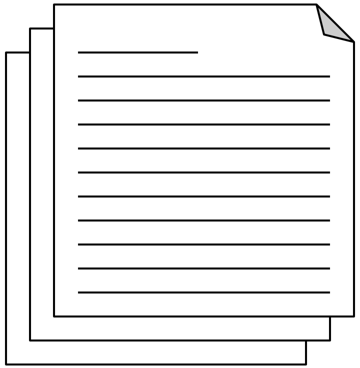
- System behavior is how a system acts and reacts.
  - It comprises the actions and activities of a system.
- System behavior is captured in use cases.
  - Use cases describe the interactions between the system and (parts of) its environment.

What is a use-case model?

- Describes the functional requirements of a system in terms of use cases
- Links stakeholder needs to software requirements
- Serves as a planning tool
- Consists of **actors** and **use cases**

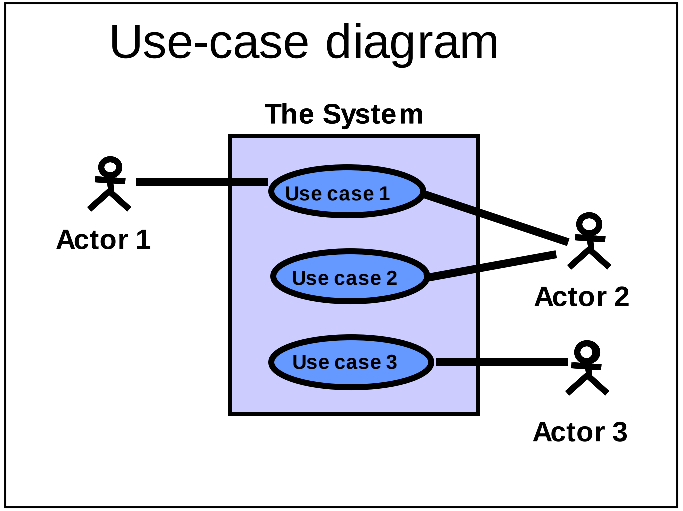
Capture a use-case model

- A use-case model is comprised of:

Use-case diagrams (visual representation)	**Use-case specifications** (text representation)
	

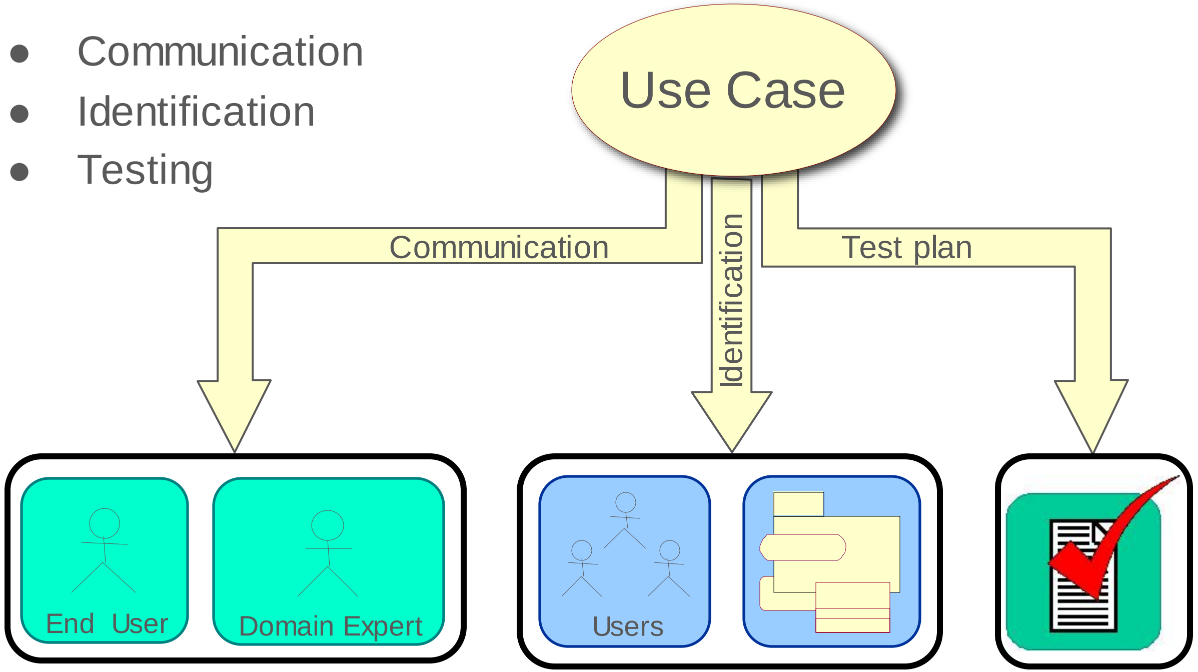
Use-case diagram

- Shows a set of use cases and actors and their relationships
- Defines clear boundaries of a system
- Identifies who or what interacts with the system
- Summarizes the behavior of the system



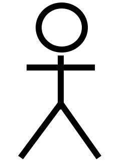
What Are the Benefits of a Use-Case Model?

- Communication
- Identification
- Testing

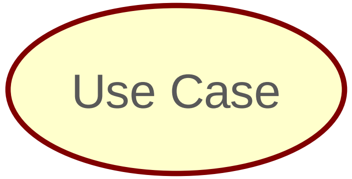


Major Concepts in Use-Case Modeling

- An actor represents anything that interacts with the system.

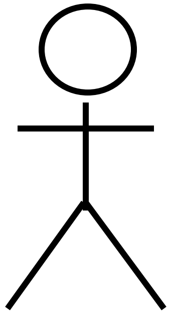


- A use case describes a sequence of events, performed by the system, that yields an observable result of value to a particular actor.



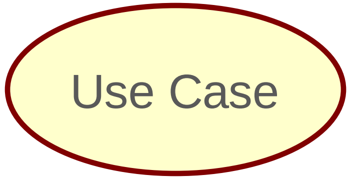
What Is an Actor?

- Actors represent roles a user of the system can play.
- They can represent a human, a machine, or another system.
- They can actively interchange information with the system.
- They can be a giver of information.
- They can be a passive recipient of information.
- Actors are not part of the system.
  - Actors are EXTERNAL.


Actor

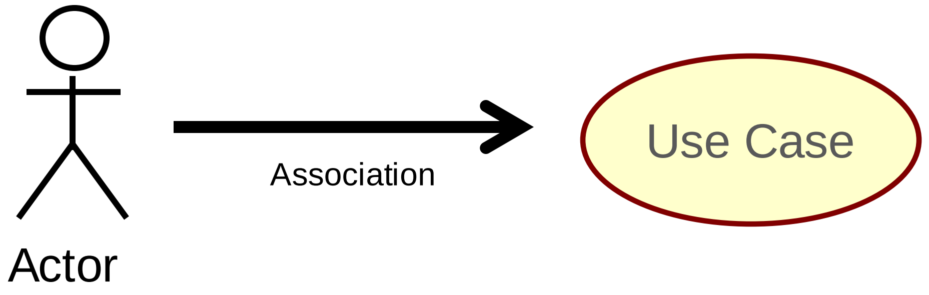
What Is a Use Case?

- Defines a set of use-case instances, where each instance is a sequence of actions a system performs that yields an observable result of value to a particular actor.
  - A use case models a dialogue between one or more actors and the system
  - A use case describes the actions the system takes to deliver something of value to the actor



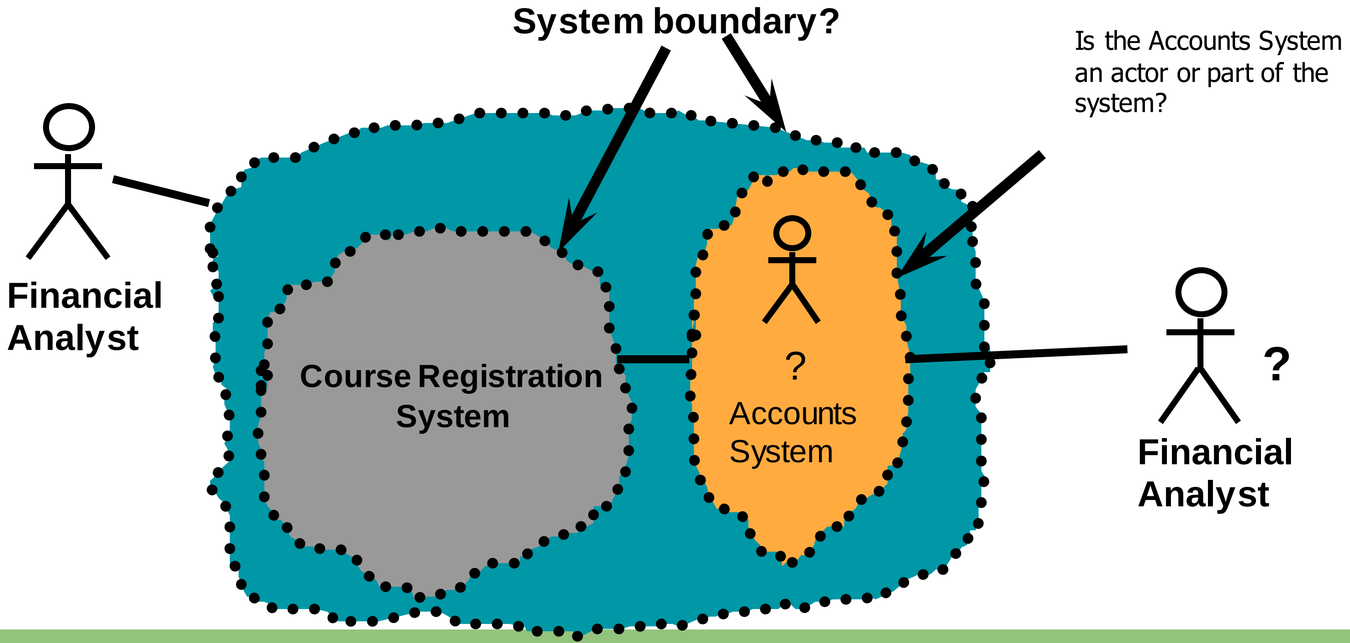
Use Cases and Actors

- A use case models a dialog between actors and the system.
- A use case is initiated by an actor to invoke a certain functionality in the system.



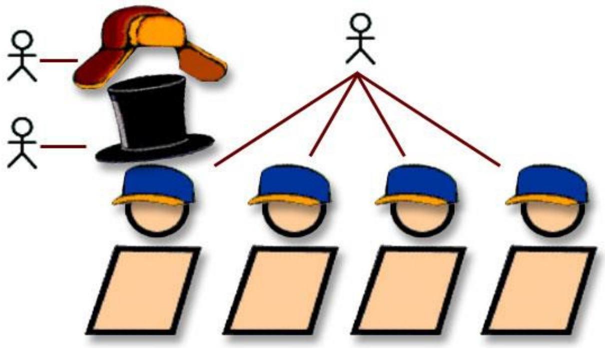
Actors and the system boundary

- Determine what the system boundary is
- Everything beyond the boundary that interacts with the system is an instance of an actor

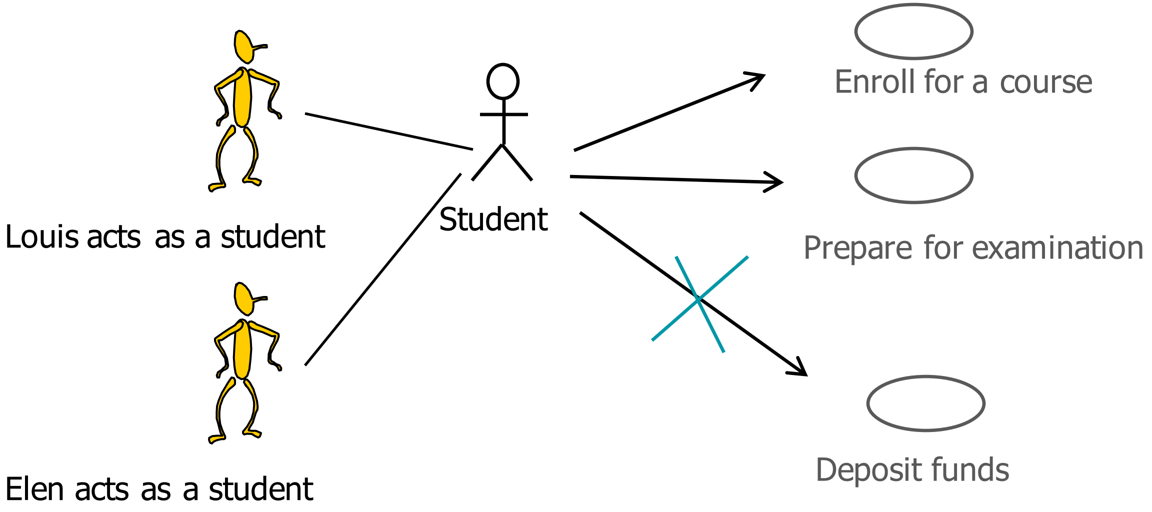

System boundary?
Is the Accounts System an actor or part of the system?
Financial Analyst
Course Registration System
?
Accounts System
Financial Analyst
?

Actors and roles

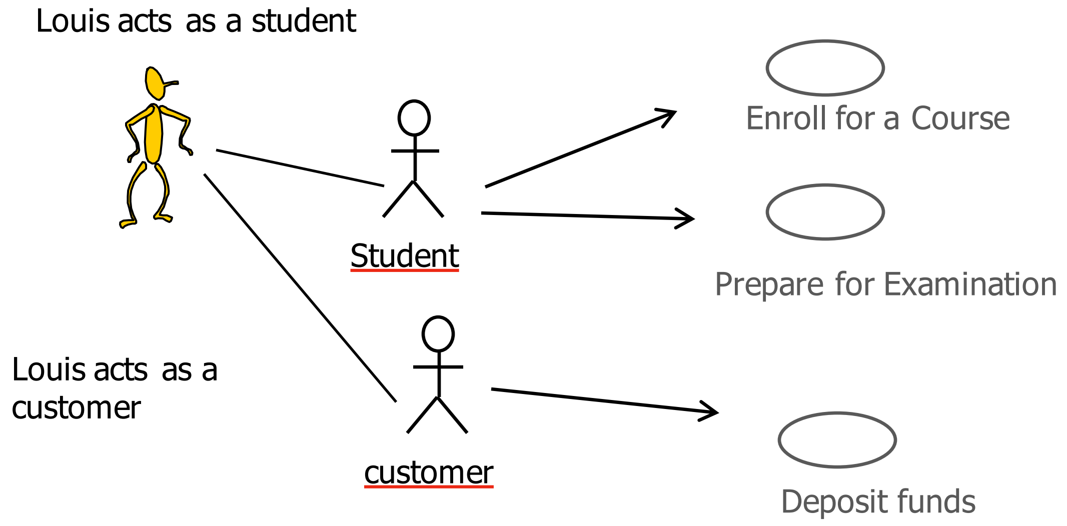
- An actor represents a role that a human, hardware device, or another system can plan in relation to the system.



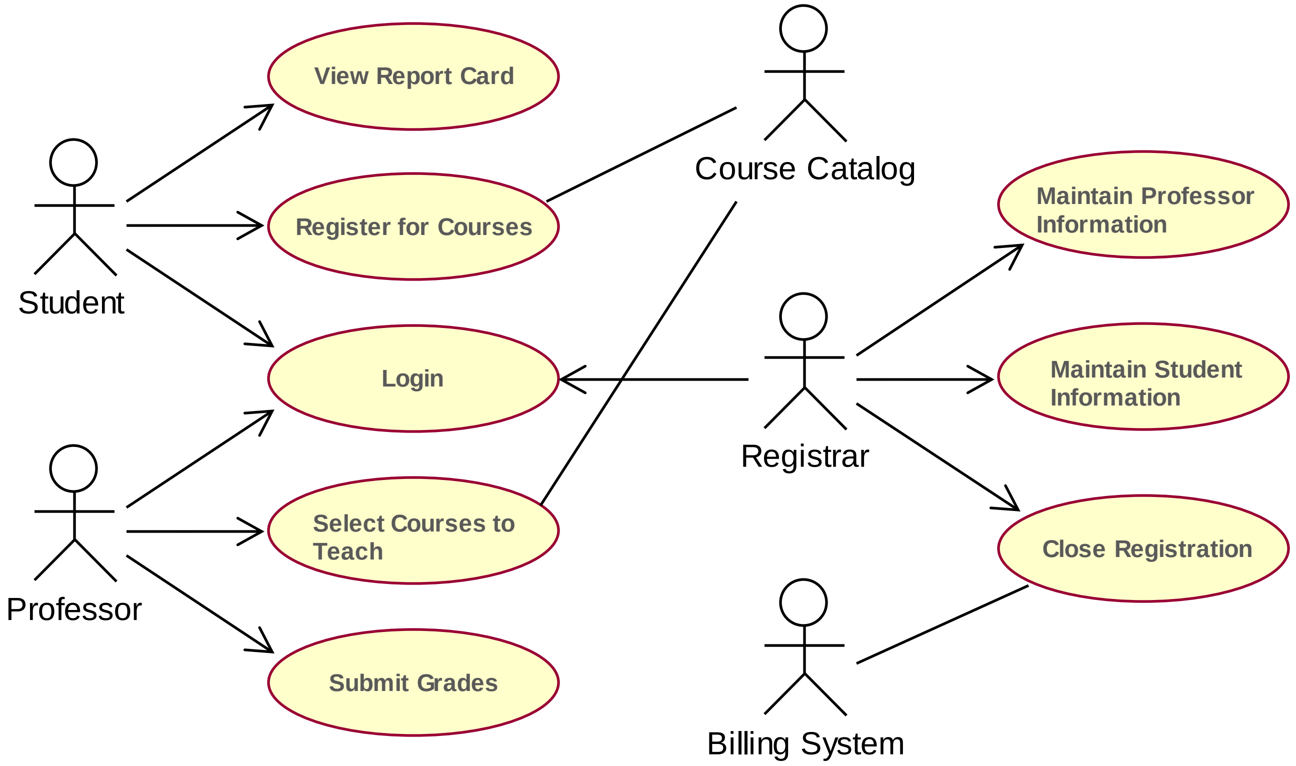
Actors


Louis acts as a student
Elen acts as a student
Student
Enroll for a course
Prepare for examination
Deposit funds

Actors

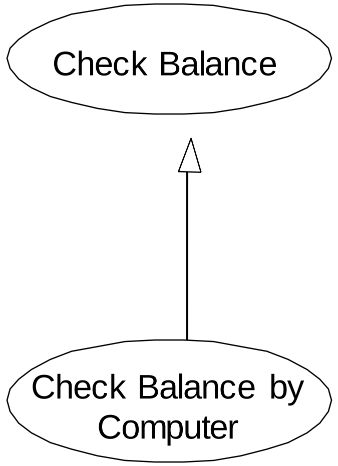

Louis acts as a student
Student
Enroll for a Course
Prepare for Examination
Louis acts as a customer
customer
Deposit funds

How Would You Read This Diagram?

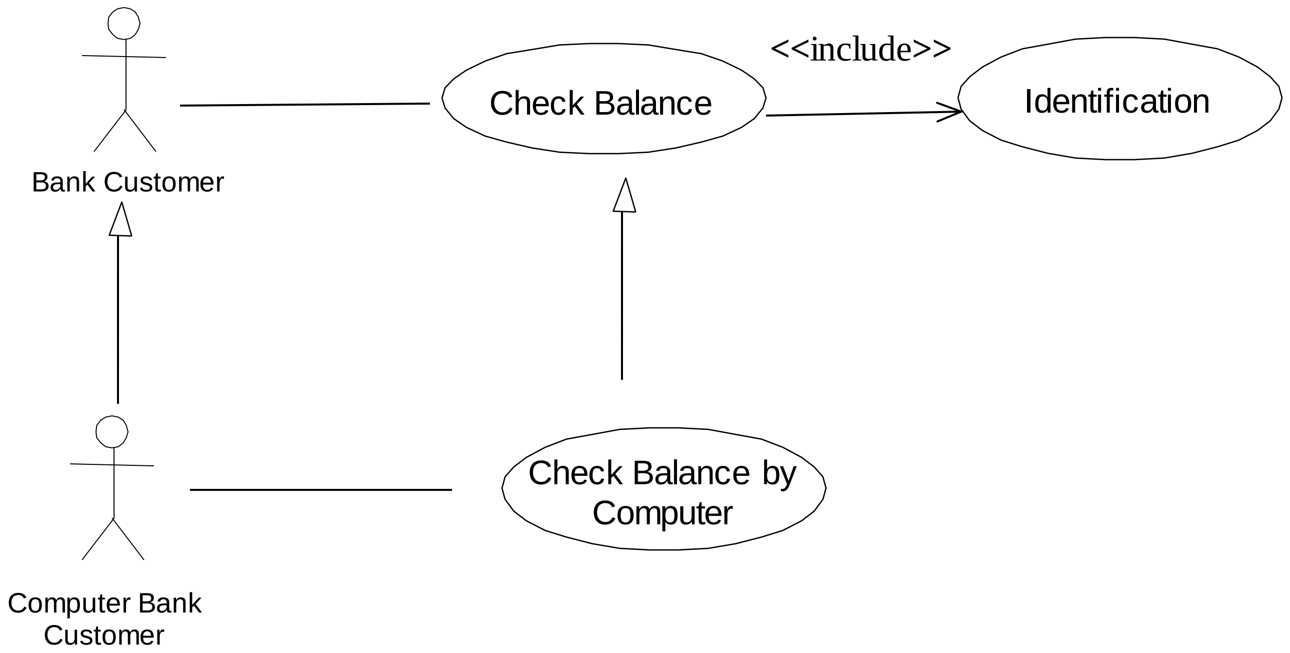


Modelling with Use Cases

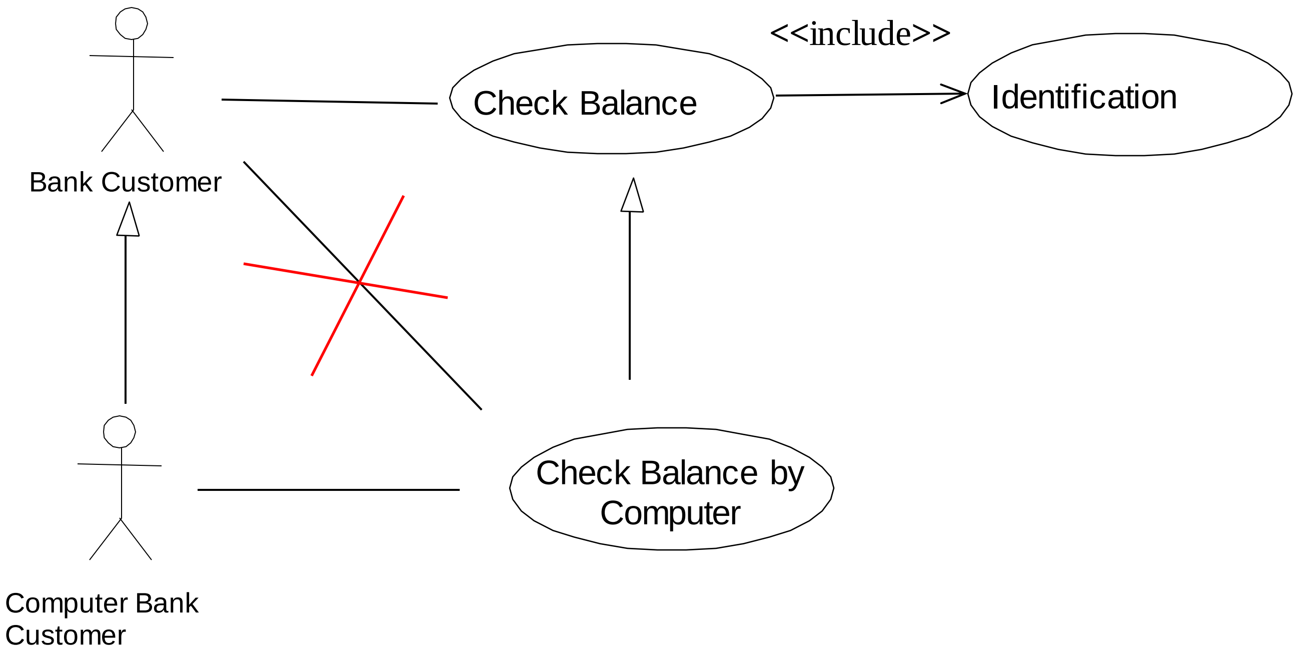
- Generalization between Use Cases means that the child is a more specific than the parent; the child inherits all attributes and associations of the parent, but may add new features



Structuring Use Case Diagrams



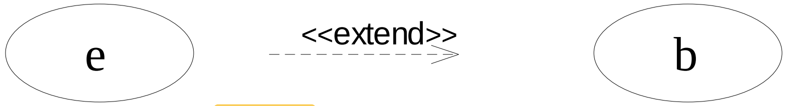
Use Case Diagram: Structuring



Use Cases

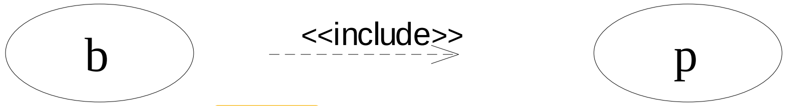
Specifies how the behaviour of the extension use cases `e` can be inserted into the behaviour of the base use case `b`.
`e` is optional.

嵌入



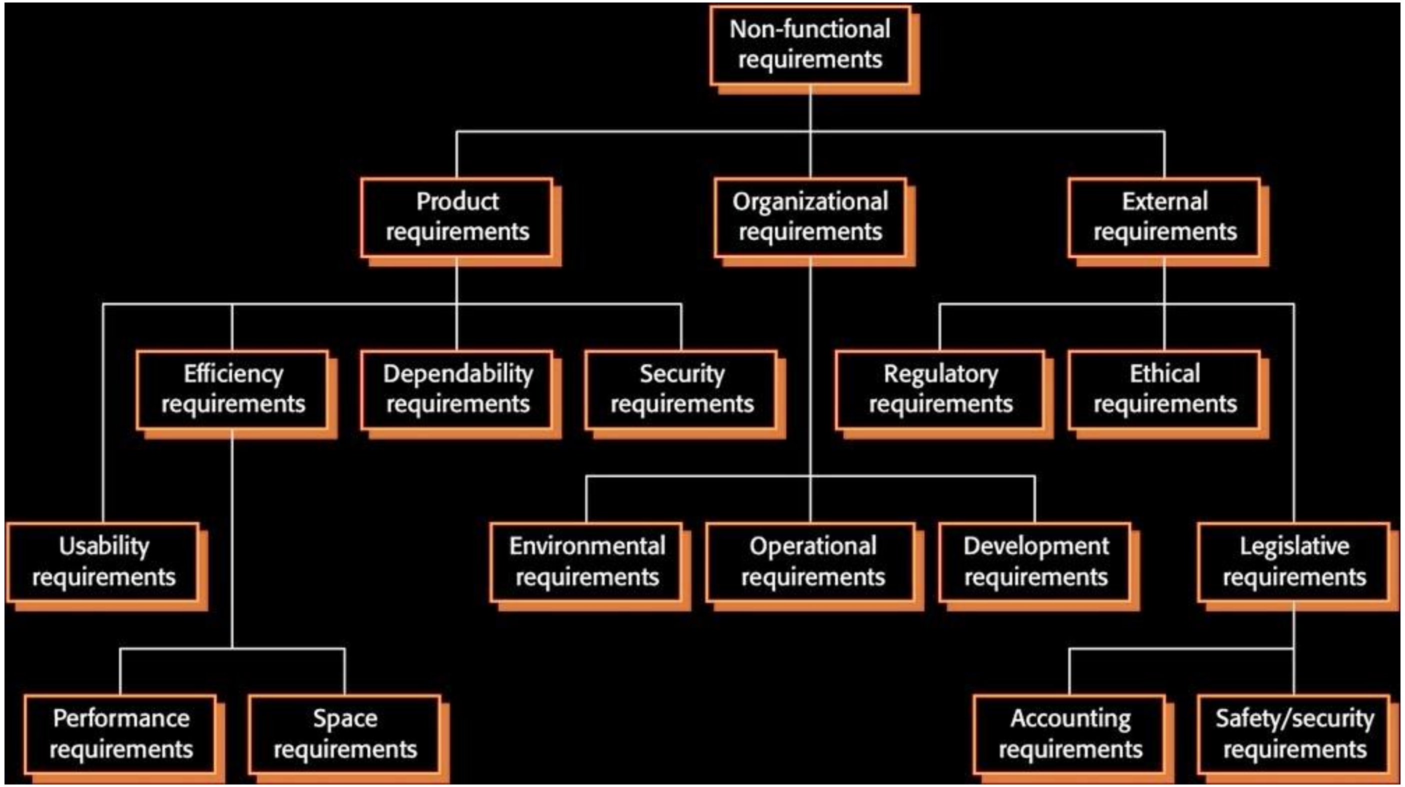
Extend ralationship

Specifies how the behaviour of the included Use Case `p` contributes to the behaviour of the base use case `b`.



Include relationship

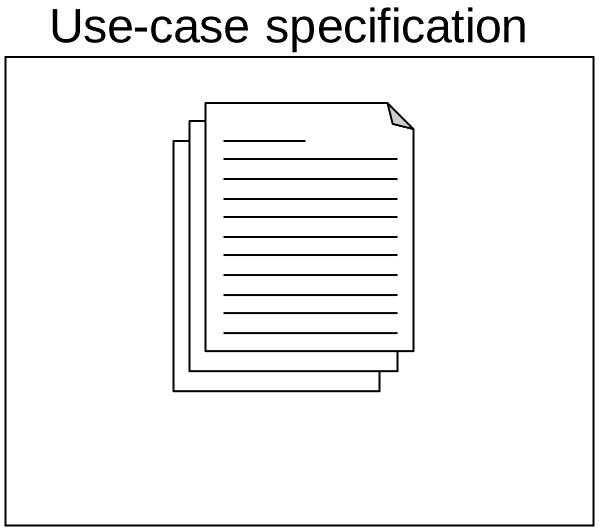
But also: Non-functional Requirements



3. Break down stories into individual steps / Refine requirements

Use-case specification

- A requirements document that contains the text of a use case, including:
  - A description of the flow of events describing the interaction between actors and the system
  - Other information, such as:
    - Preconditions n. 前提，先决条件
    - Postconditions n. 后[置]条件
    - Special requirements
    - Key scenarios
    - Subflows



Outline each use case

- An outline captures use case steps in short sentences, organized sequentially

<u>Use Case Name</u>

Brief Description

Basic Flow

1. **First step**
2. **Second step**
3. **Third step**

Alternative Flows

1. **Alternative flow 1**
2. **Alternative flow 2**
3. **Alternative flow 3**

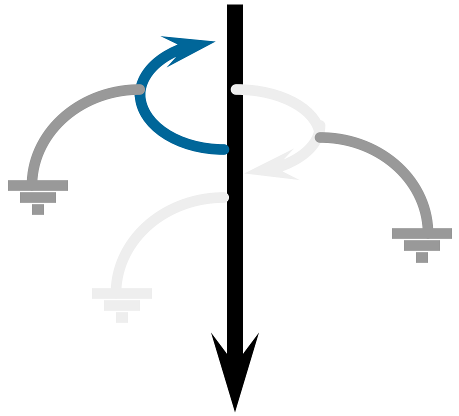
Number and name the steps

Structure the basic flow into steps

Identify alternative flows

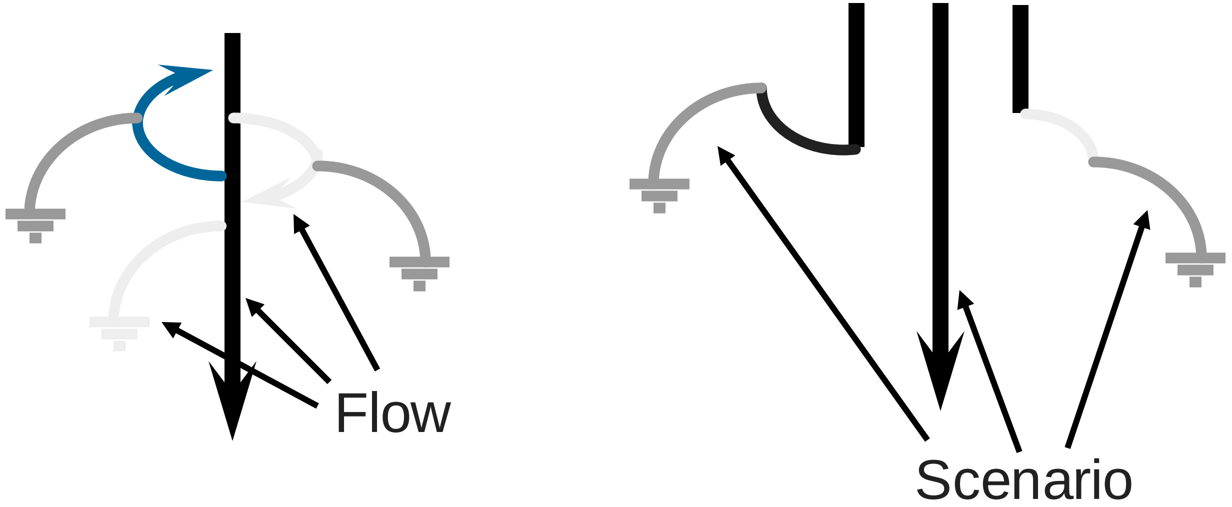
Flows of events (basic and alternative)

- A flow is a sequence of steps
- One basic flow
  - Successful scenario from start to finish
- Many alternative flows
  - Regular variants
  - Odd cases
  - Exceptional (error) flows



What is a use-case scenario?

- An instance of a use case
- An ordered set of actions from the start of a use case to one of its end points



Note: This diagram illustrates only some of the possible scenarios based on the flows.

Checkpoints for use cases

- ✓ Each use case is independent of the others
- ✓ No use cases have very similar behaviors or flows of events
- ✓ No part of the flow of events has already been modeled as another use case

4. Specify atomic requirements (e.g., for each step in user stories)

- ✓ Not detailed in this course, e.g.:
  - ✓ Structured language
  - ✓ Formal methods

Quality Attributes of Requirements

- Consistency: Are there conflicts between requirements ?
- Completeness: Have all features been included ?
- Comprehendability: Can the requirement be understood ?
- Traceability: Is the origin of requirement clearly recorded ? n. 可描写,可追溯
- Realism: Can the requirements be implemented given available resources and technology ?
- Verifiability: Can requirements be “ticked off” ? 勾选，列举

Verifiability of Requirements

We should ensure that requirements are ***verifiable***
No point specifying something that can't be "ticked off"

For example, the following is an ***unverifiable*** "objective":
The system must be easy to use by waiting staff and should be organised so that user errors are minimised.

In comparison, the following is a ***testable*** requirement:
Waiting staff shall be able to use all system functions after four hours of training. After this training, the average number of errors made by experienced users shall not exceed two per hour of system use.

Requirements Elicitation Techniques

n. 引出,诱出,抽出,启发

- Interviews
- Observations
- Surveys
- Current documentation
- Similar products and solutions
- Co-design
- Prototyping
- …

Review

- What are models for?
- What is system behavior?
- What is an actor?
- A use case?
- What is a role?
- How do we know if our requirements are of good quality?