Requirements Engineering

Lecture 3

Ruzanna Chitchyan, Jon Bird, Pete Bennett

Overview

- What are requirements?
- Stakeholder Identification
- Requirements with Agile
- Requirements with UML
  - Use Case Diagram
  - Use Case Specification

Requirements

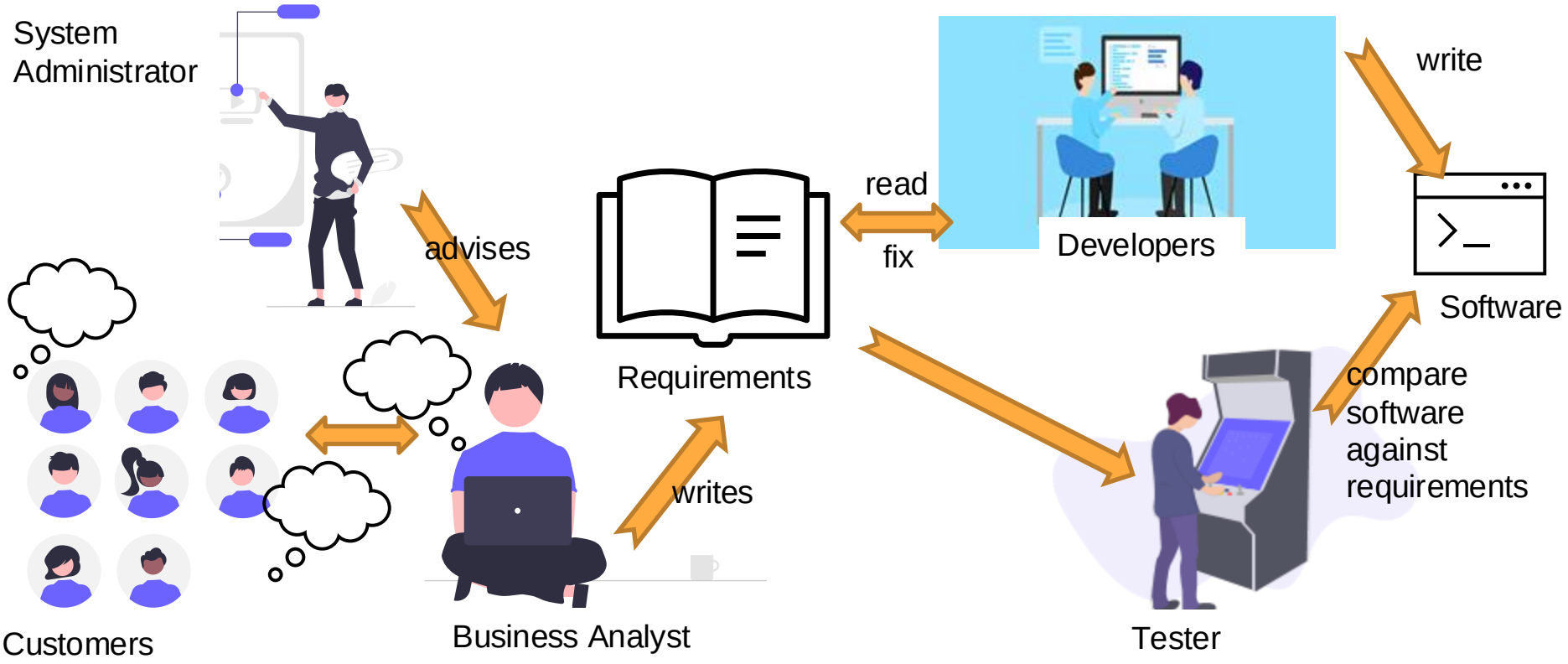
- **System requirements** specify a system, not in terms of system implementation, but in terms of user observation. Requirements record description of the system features and constraints.
  - **Functional requirements** specify user interactions with the system, they say **what** the system is supposed 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**



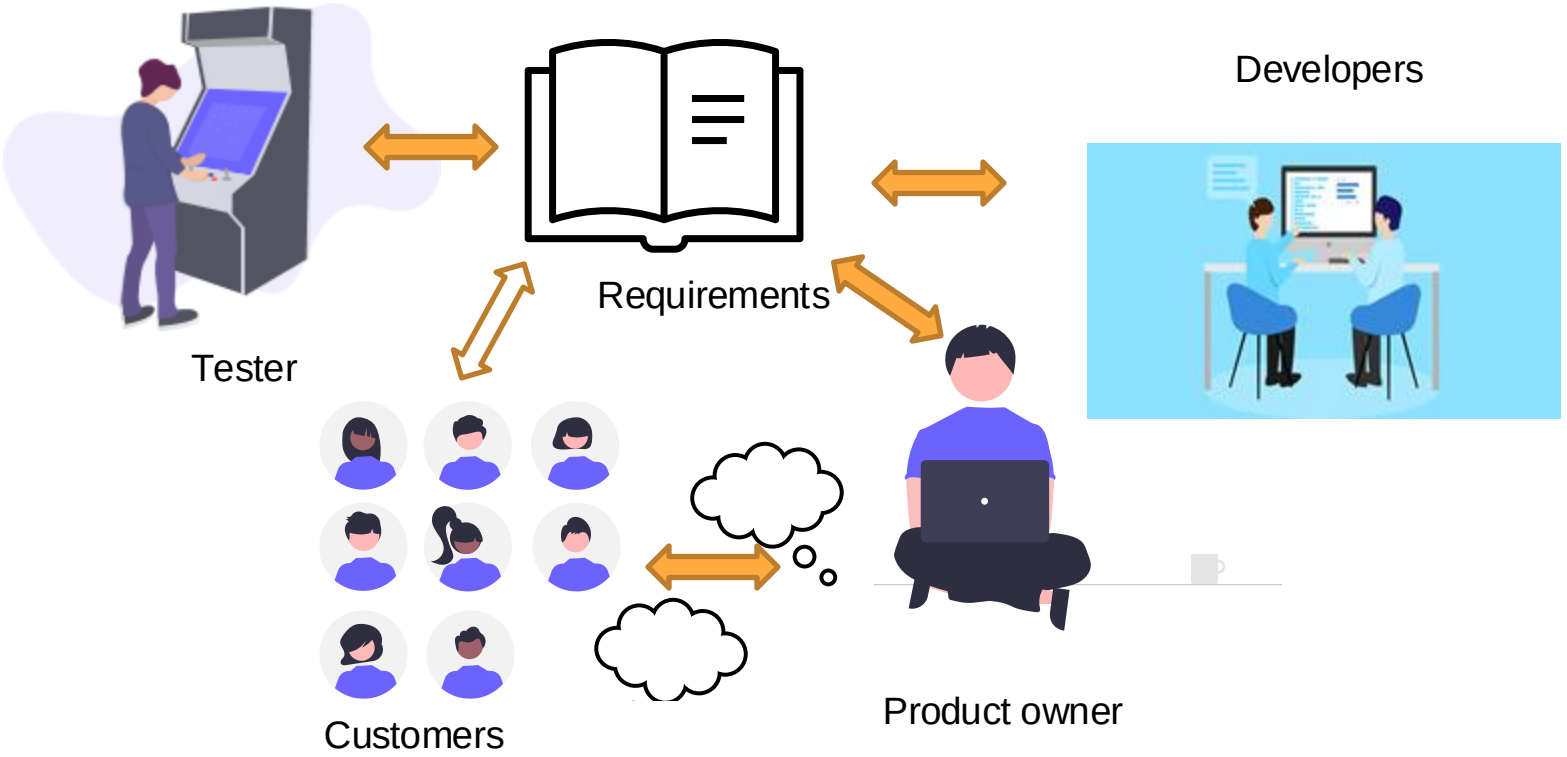
Requirements are **instructions**

- Explain what do do!

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
  - Sufficiently accessible / Measuirable
  - Understandable / Clear

Requirements drive agile process



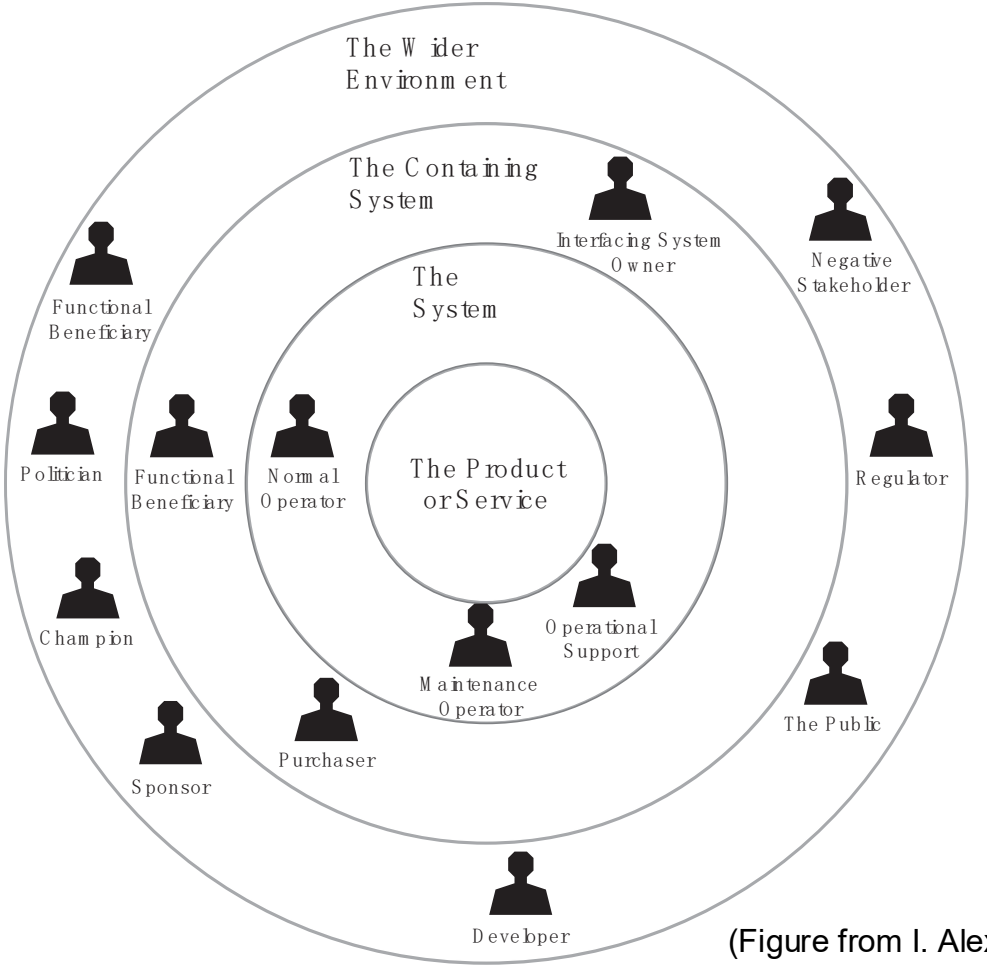
Analysing Requirements

1. Identify stakeholders involved with the system
2. Identify top-level user needs (e.g., as NFRs or initiatives/epics)
3. Break down needs into individual stories / Refine requirements
4. Specify atomic requirements (e.g., formal specification)
5. Some UML: use cases (use case diagram, use case description)

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:

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

2. Identify top level needs/concerns

Identify Epics

- High level requirement (no detail)
- Focused on business or user value
- Large and/or complex at scope
- Needs to be broken down into smaller pieced to implement
    - The pieced are grouped under that epic

Examples:

- Enhance visual accessibility
- Implement alternative input methods
- Build accessibility settings options

All part of an **Initiative**:

“***Increase usability for disabled users”***

3. Break down requirements into smaller steps / Refine requirements

Break into “User Stories” - Template

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

For my epic: Enhance visual accessibility:

- *As a **user with low vision**, I want to resize text so that I can read comfortably.*
- *As a **colour-blind user**, I want to customise the colour scheme so that I can see all content regardless of colour perception.*
- *As a **user**, I want sufficient contrast between text and background so that I can read all sections of the app.*

INVEST for Good User Stories

- **Independent**: Can be worked on separately from other stories.
- **Negotiable**: Flexible and open to discussion.
- **Valuable**: Delivers clear value to the user.
- **Estimable**: Can be estimated for effort.
- **Small**: Small enough to complete within a sprint.
- **Testable**: Has clear criteria to determine if it’s done.

Acceptance Criteria for User Stories

- **Clear**: easy to understand and unambiguous.
- **Testable**: should be able to test to verify that it is met.
- **Measurable**: can be measured quantitatively or qualitatively.
- **Atomic**: each criteria is independent, can be checked by itself.

Template:

Given <initial context or precondition>, **when** <action or event>, **then** <expected outcome>.

Example Acceptance Criteria

As a user with low vision, I want to resize text so that I can read comfortably.

- **Text resizing: Given** I am on the website or application, **when** I navigate to the accessibility settings, **then** I should see an option to adjust text size (e.g., small, medium, large, extra large).
- **No impact on non-text elements: Given** I resize the text to any size, **when** I view non-text elements (e.g., images or icons), **then** they should not distort or change in size.
- **Default reset option: Given** I have resized the text to a different size, **when** I want to return to the default size, **then** I should see and be able to use a "Reset to Default" option in the accessibility settings.

Initiative vs Epic vs User Story

Initiative	Epic	User Story
A strategic objective for the company, with important business outcome.	A large, strategic goal.	A specific feature or functionality.
Spans multiple epics and teams/departments.	Spans multiple sprints.	Completed within a sprint.
Example: “Improve experience of disabled users."	Example: ”Enhance visual accessibility."	Example: "As a user with low vision, I want to resize text so that I can read comfortably."

Prioritise User Stories: what to implement?

- Considering on:
  - User needs
  - Business value
  - Technical considerations
- **M**o**SC**o**W**:
  - **M**ust-Have: essential
  - **S**hould-Have: important
  - **C**ould-Have: nice to have
  - **W**on’t-Have: out of scope at present
- Value vs Effort:
  - High value, Low effort (do first)
  - High value, High effort (do next)
  - Low value, Low effort (do if there is time)
  - Low value, High Effort (avoid)

Make a perfect sandwich

Customer: Alex is gluten intolerant, vegetarian and prefers spicy food.

Work in groups of 3.

Teams have **3 minutes** to **prioritise the ingredients and crate a Perfect Sandwich recipe for Alex**.

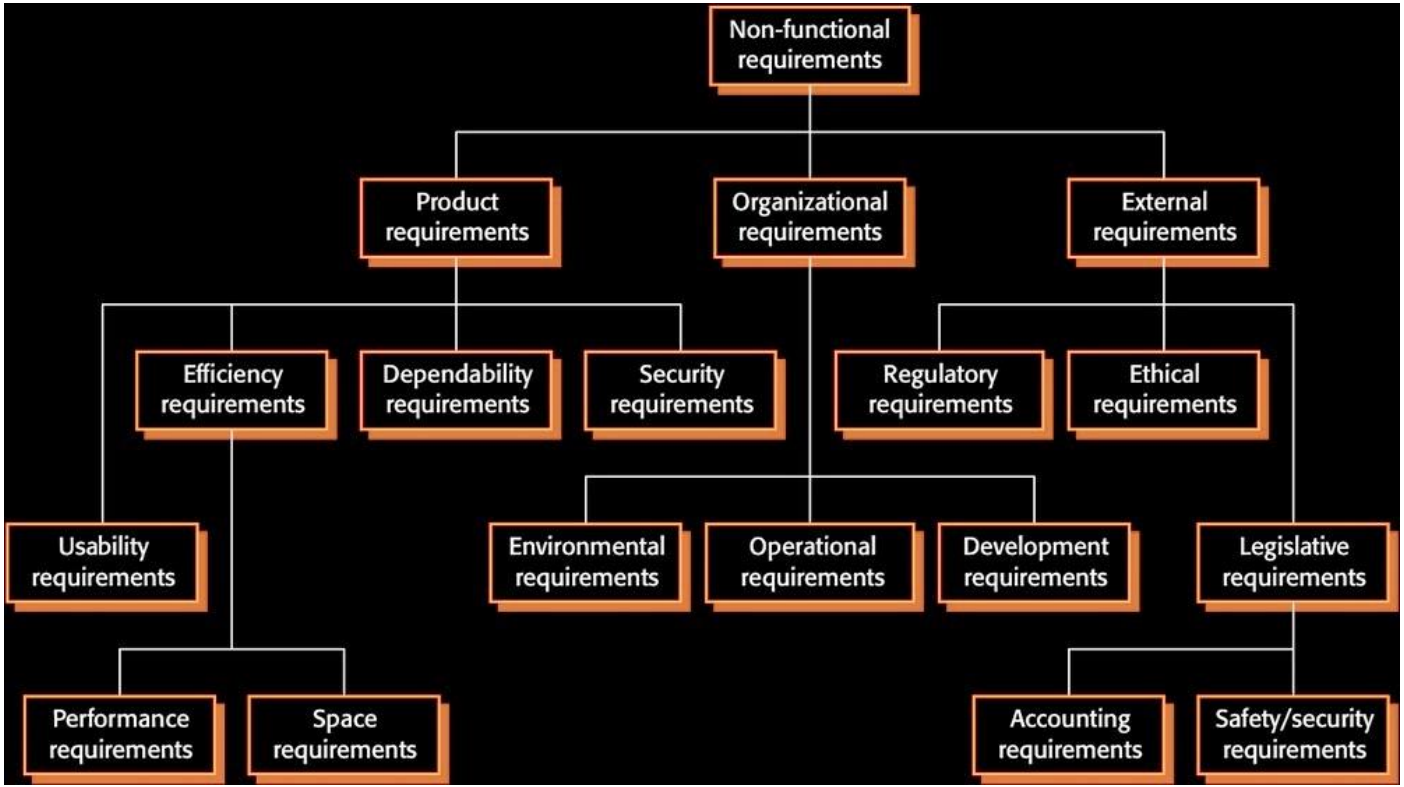
1. Grilled chicken breast
2. Turkey slices
3. Grilled Tofu
4. Smoked salmon
5. Chickpea patties

1. Hummus
2. Spicy mustard
3. Basil pesto
4. Sriracha
5. Mayo
6. Cream cheese

1. Rice cakes
2. Whole wheat bread
3. Ciabatta rolls
4. Regular flour tortillas
5. Gluten-free sandwich bread

1. Tomatoes
2. Cucumbers
3. Bell peppers
4. Avocado
5. Jalapeños (adds spice)

Non-functional Requirements



4. Specify atomic requirements

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

Requirements Elicitation Techniques

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

A Bit of UML

What is a use-case model?

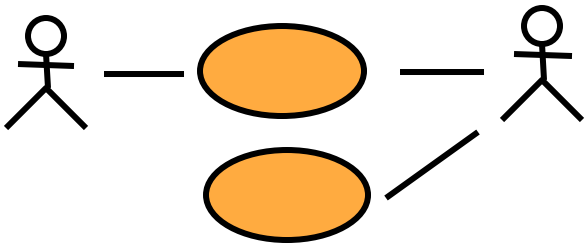
- System behavior is how a system acts and reacts. Use cases describe the interactions between the system and (parts of) its environment.

- 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**

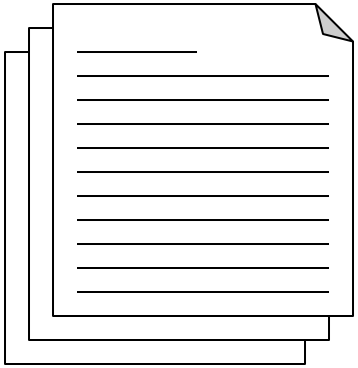
Capture a use-case model

- A use-case model is comprised of:

Use-case diagrams
(visual representation)

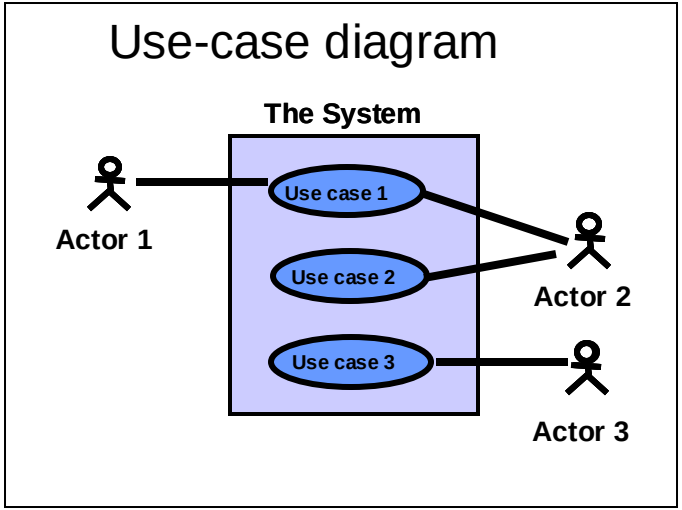


Use-case specifications
(text representation)



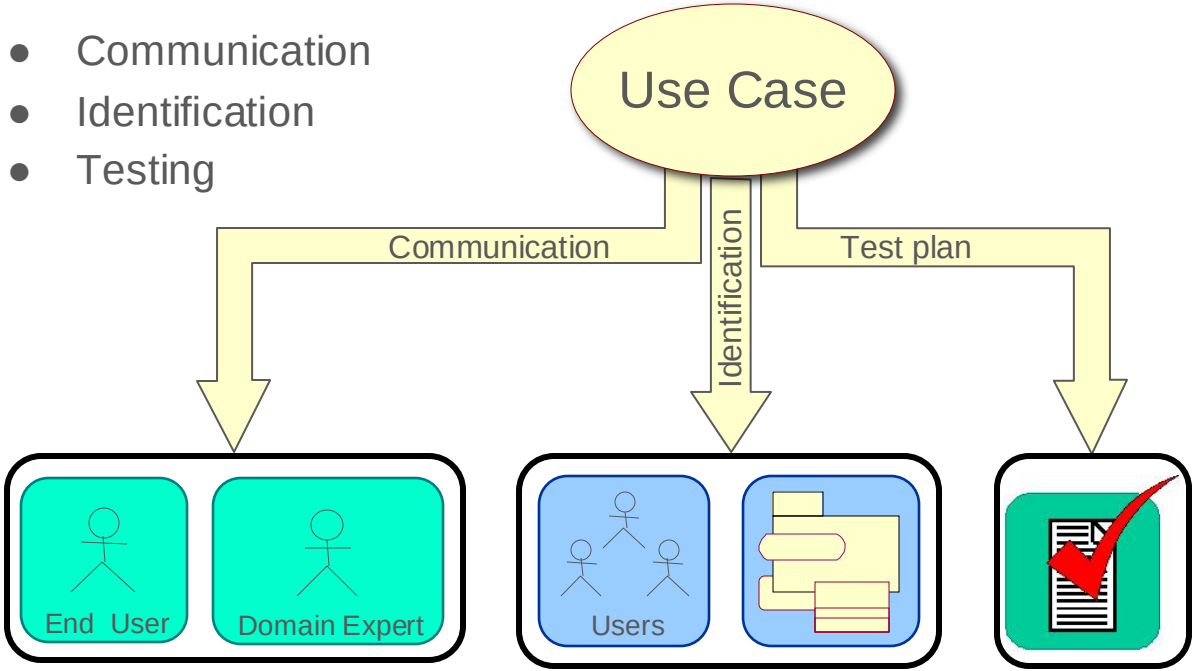
Use-case diagram

- 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



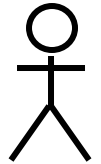
What Are the Benefits of a Use-Case Model?

- Communication
- Identification
- Testing

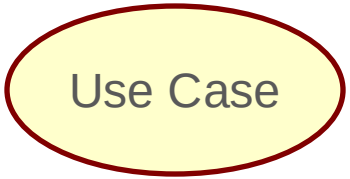


Major Concepts in Use-Case Modeling

- 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.
- Association: Shows that a use case is initiated by an actor.

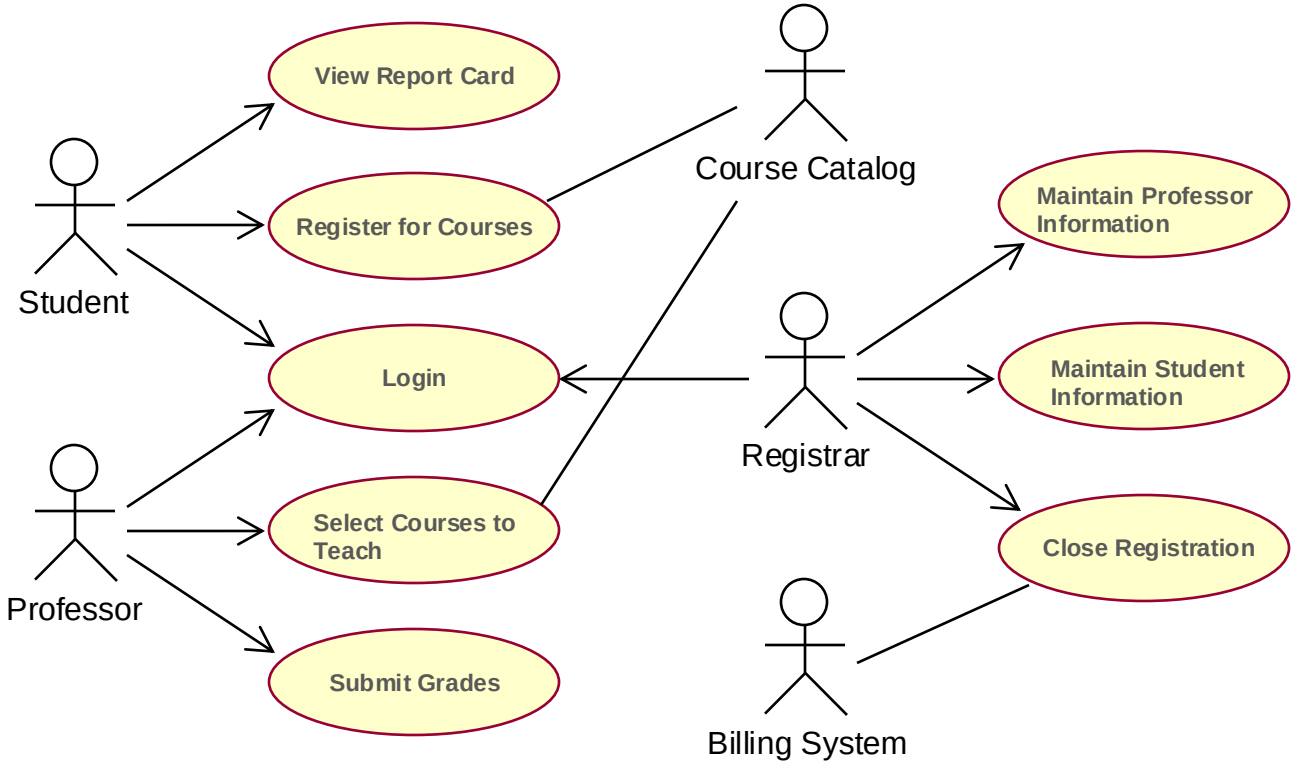


Actor



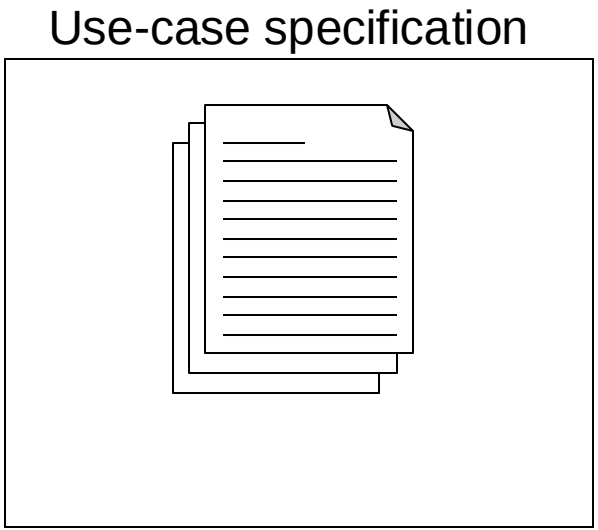
Association

How Would You Read This Diagram?



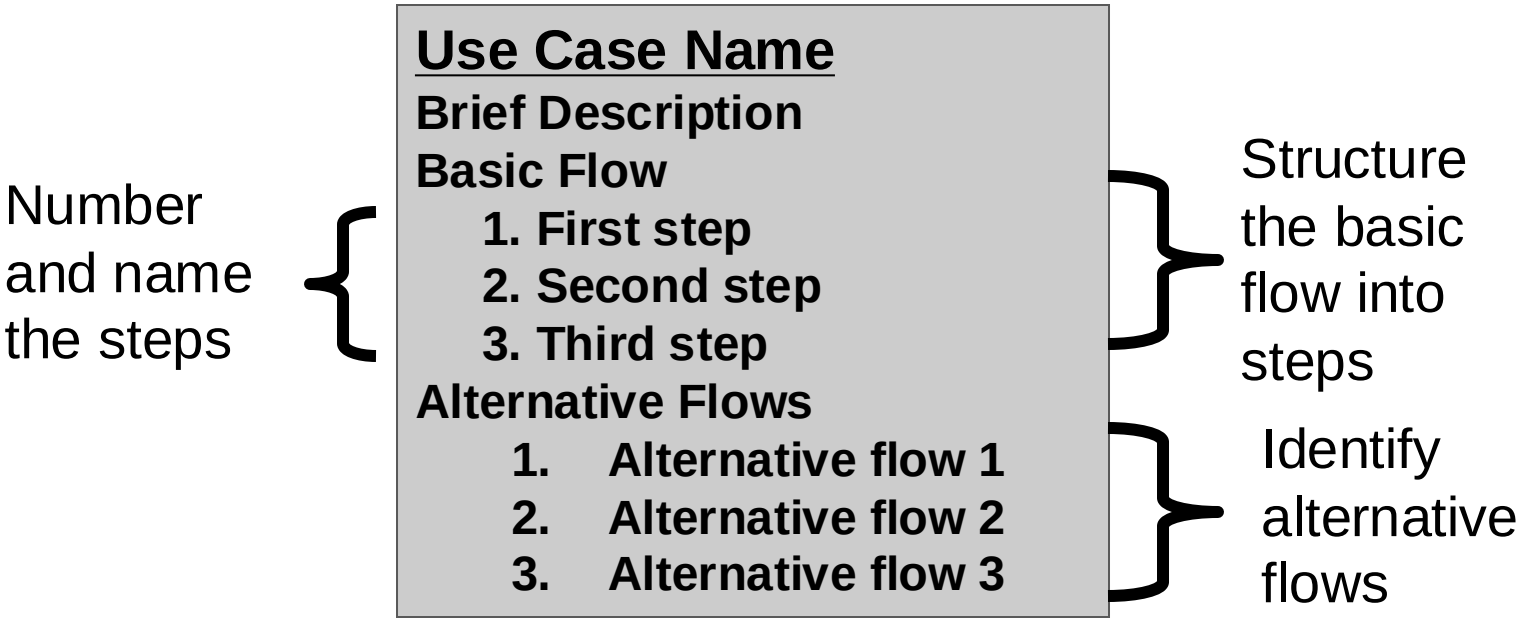
Use-case specification

- 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
    - Postconditions
    - Special requirements
    - Key scenarios
    - Subflows



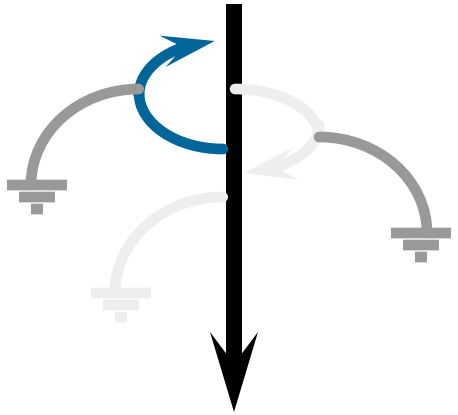
Outline each use case

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



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



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

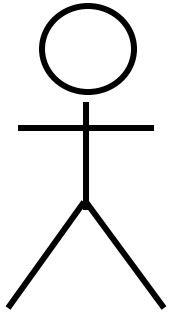
Review

- 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?

A Bit UML: Look at these at your own time!

What Is an Actor?

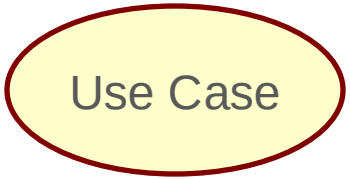
- Actors represent roles a user of the system can play.
- They can represent a human, a machine, or another system.
- They can actively interchange information with the system.
- They can be a giver of information.
- They can be a passive recipient of information.
- Actors are not part of the system.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



Actor

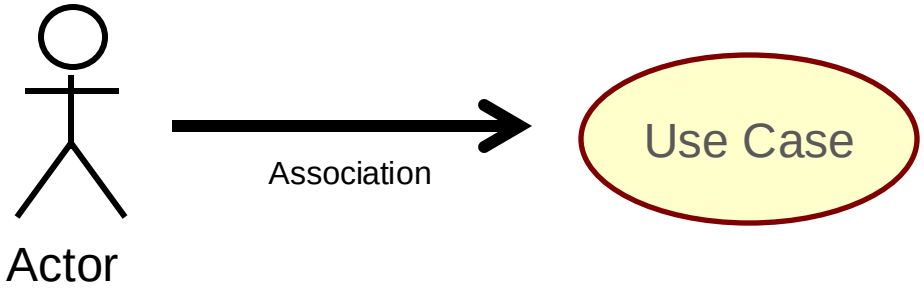
What Is a Use Case?

- 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

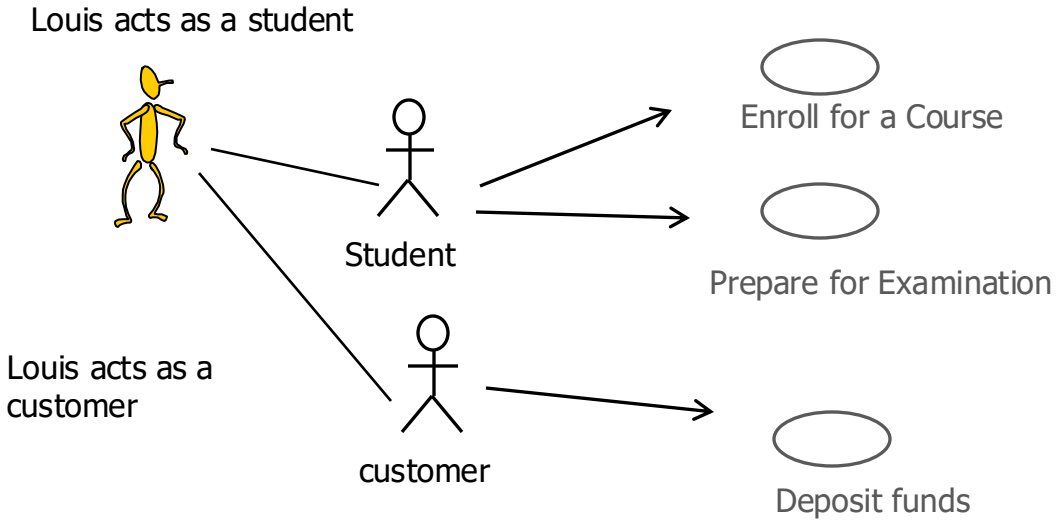


Use Cases and Actors

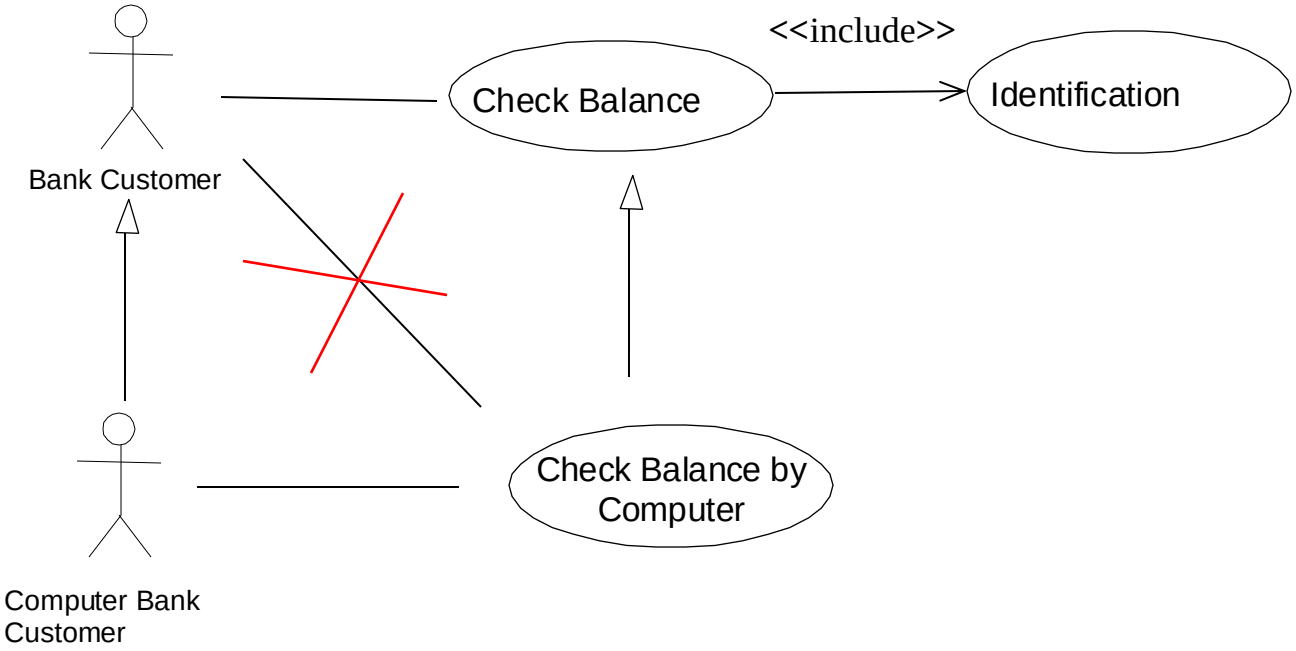
- A use case models a dialog between actors and the system.
- A use case is initiated by an actor to invoke a certain functionality in the system.



Actors

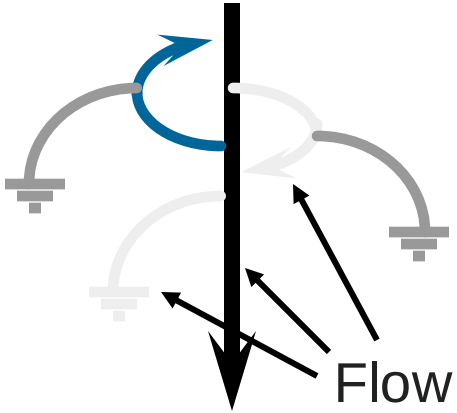

Louis acts as a student
Student
Enroll for a Course
Prepare for Examination
Louis acts as a customer
customer
Deposit funds

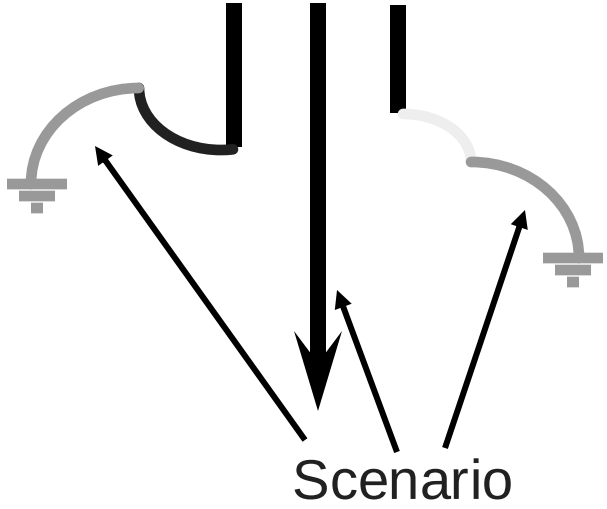
Use Case Diagram: Structuring



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.