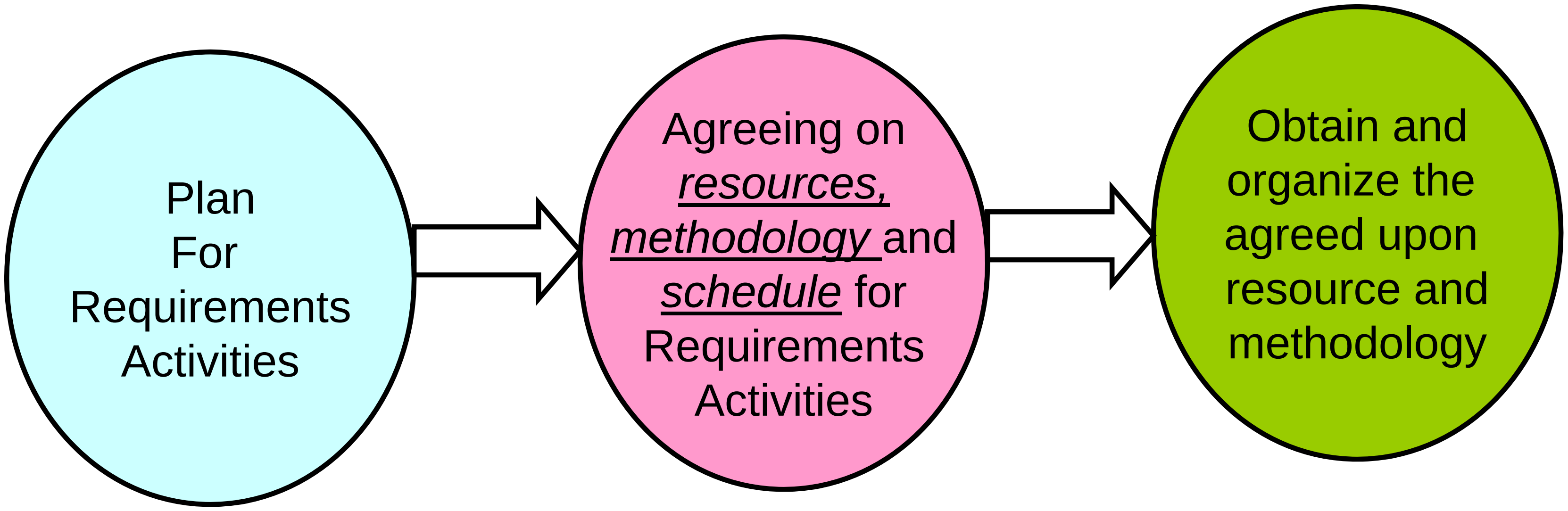
Requirements Engineering

Dr. Marouane Kessentini
Department of Computer Science

Preparation for Requirements Engineering


Plan For Requirements Activities
Agreeing on *resources*, *methodology* and *schedule* for Requirements Activities
Obtain and organize the agreed upon resource and methodology

1. Prior to actually performing the requirements engineering activities, it is important to plan for the resources, methodology and time needed to perform this crucial step in software engineering.

2. Some organizations even perform requirements engineering as a separate , stand-alone activity and price it separately, with the option of folding the cost Into the whole project if they get the call to complete the software project.

Requirements Analysis

- Requirements "analysis" is composed of:
  - Categorizing the requirements (by some criteria)
  - Prioritizing the requirements
- Also "*start to look*" for consistency & completeness

Types of Requirements

- Functional requirements
  - Describe what the system should do
- Quality requirements
  - Constraints on the design to meet specified levels of quality
- Platform requirements
  - Constraints on the environment and technology of the system
- Process requirements
  - Constraints on the project plan and development methods

Requirements Prioritization

- Most of the time we have some limitations in developing software:
  - Time
  - Resources
  - Technical capabilities (existing)
- We need to prioritize the requirements to satisfy these limitations

Requirements Definition/Prototyping/Review

- **Once the requirements are solicited, analyzed and prioritized, more details may/must be spelled out. Three major activities which may be intertwined must be performed:**
  - **Requirements definition**
  - **Requirements prototyping**
  - **Requirements reviewing**

Requirements Definitions/Documentation

- **Requirements definitions may be written in different forms:**

  1. **Simple Input/Process/Output (I-P-U) descriptions in English**
  2. **Dataflow diagrams (DFD)**
  3. **Entity Relations diagram (ERD)**
  4. **Use Case Diagram from Unified Modeling Language (UML)**

- **Once the requirements are defined in detail using any of the above forms, they still need to be *reviewed* by the users/customers and other stakeholders.**

Formal Review by Others

Requirements Analysis & Definition Methodology using UML

- Using OO's Use Case Methodology and Notation, which identifies:
  - Basic/Main functionalities
  - Pre-conditions of functionality
  - Flow of events or scenarios
  - Post-conditions for the functionality
  - Error conditions and alternative flows

- Using OO Use Case Diagram which identifies:
  - Actors (or all external interfaces with the system, including the users)
  - Related "use cases" (major functionalities)
  - Boundary conditions

Use Cases may be further refined during Design

Use-Cases: describing how the user will use the system

- **A use case is a typical sequence of actions that a user performs in order to complete a given task**
  - **The objective of use case analysis is to model the system from the point of view of**

    … how users interact with this system

    … when trying to achieve their objectives.

    It is one of the key activities in requirements analysis
  - **A use case model consists of**
    - <u>a set of use cases</u>
    - <u>an optional description or diagram indicating how they are related</u>

Use Cases

- A use case should
  - Cover the full sequence of steps from the beginning of a task until the end.
  - Describe the user's interaction with the system ...
  - Be written to be independent from any particular user interface design.
  - Only include actions in which the actor interacts with the computer.

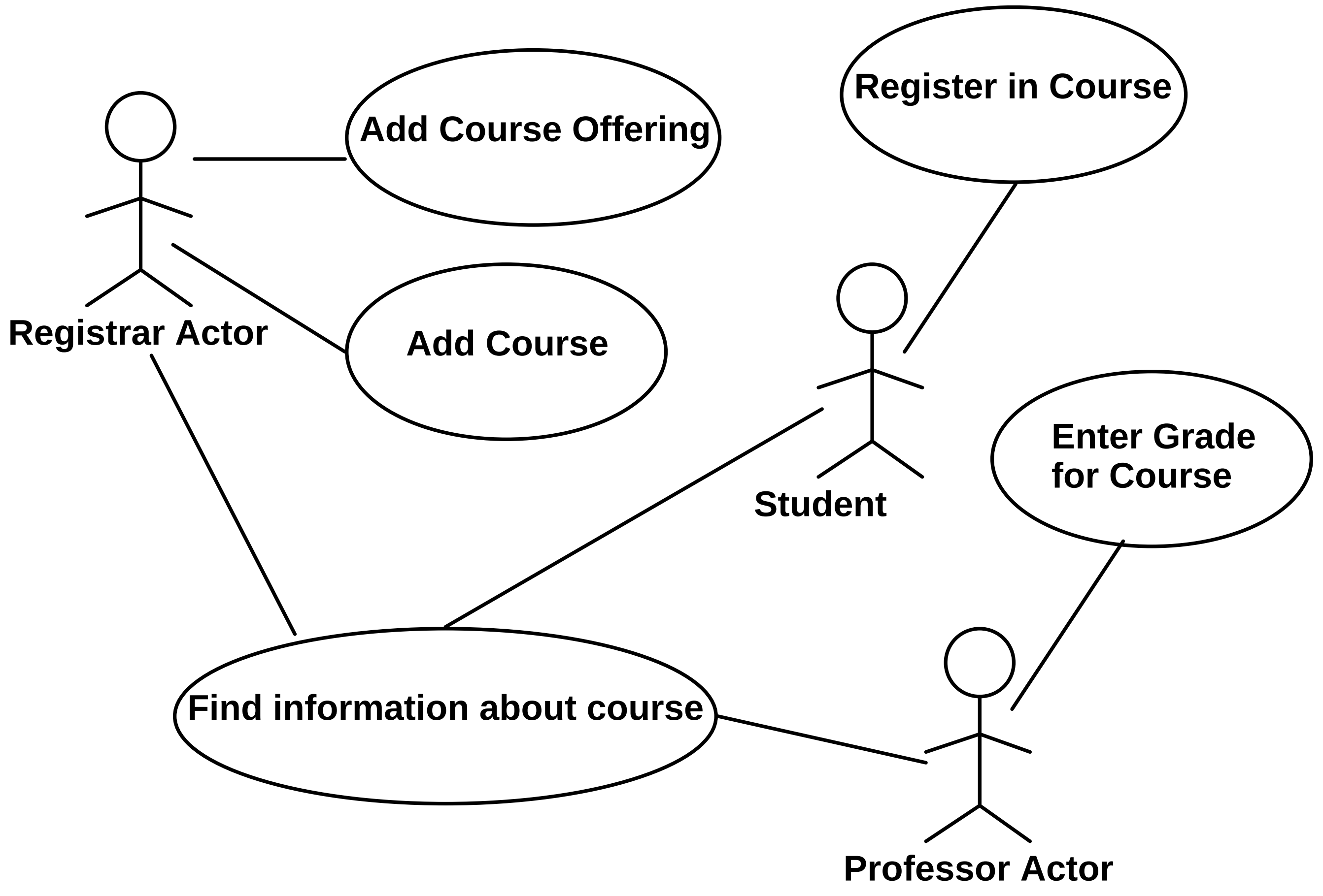
Scenarios

- **A scenario is an instance of a use case**
  - **A specific occurrence of the use case**
    - **a specific actor ...**
    - **at a specific time ...**

How to describe a single use case

- A. Name: Give a short, descriptive name to the use case.
- B. Actors: List the actors who can perform this use case.
- C. Goals: Explain what the actor or actors are trying to achieve.
- D. Preconditions: State of the system before the use case.
- E. Summary: Give a short informal description.
- F. Related use cases.
- G. Steps: Describe each step using a 2-column format.
- H. Postconditions: State of the system in following completion.

Use case diagrams



Extensions

– Used to make optional interactions explicit or to handle exceptional cases.
– Keep the description of the basic use case simple.

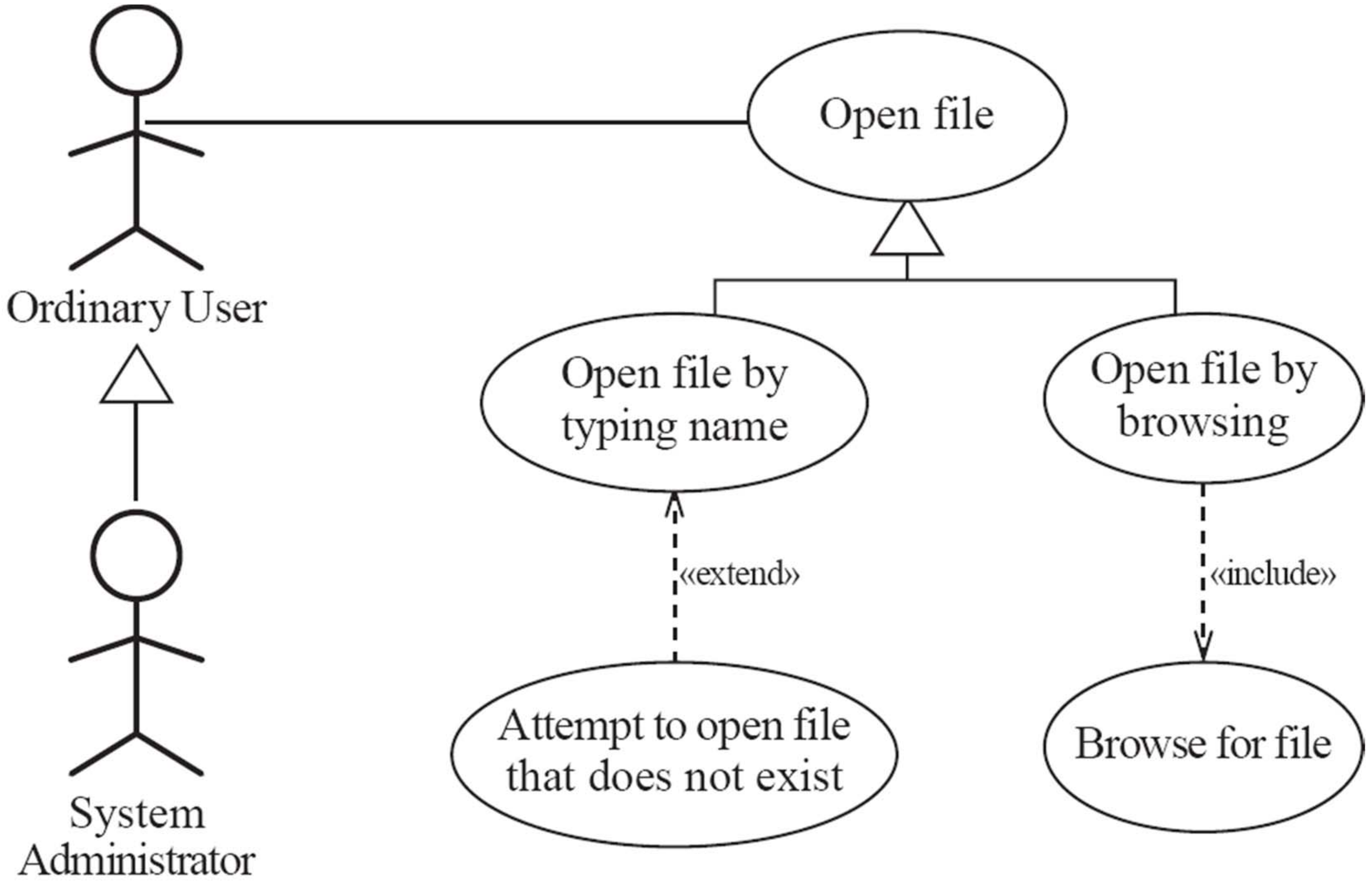
Generalizations

- Much like superclasses in a class diagram.
- A generalized use case represents several similar use cases.
- One or more specializations provides details of the similar use cases.

Inclusions

- **Allow one to express commonality between several different use cases.**
- **Are included in other use cases**
  - **Even very different use cases can share sequence of actions.**
  - **Enable you to avoid repeating details in multiple use cases.**
- **Represent the performing of a lower-level task with a lower-level goal.**

Example of generalization, extension and inclusion



Example description of a use case

Use case: **Open file**

Related use cases:

Generalization of:

- Open file by typing name
- Open file by browsing

Steps:

Actor actions	**System responses**
1. Choose 'Open…' command	2. File open dialog appears
3. Specify filename	
4. Confirm selection	5. Dialog disappears

Example (continued)

Use case: **Open file by typing name**

Related use cases:
Specialization of: Open file

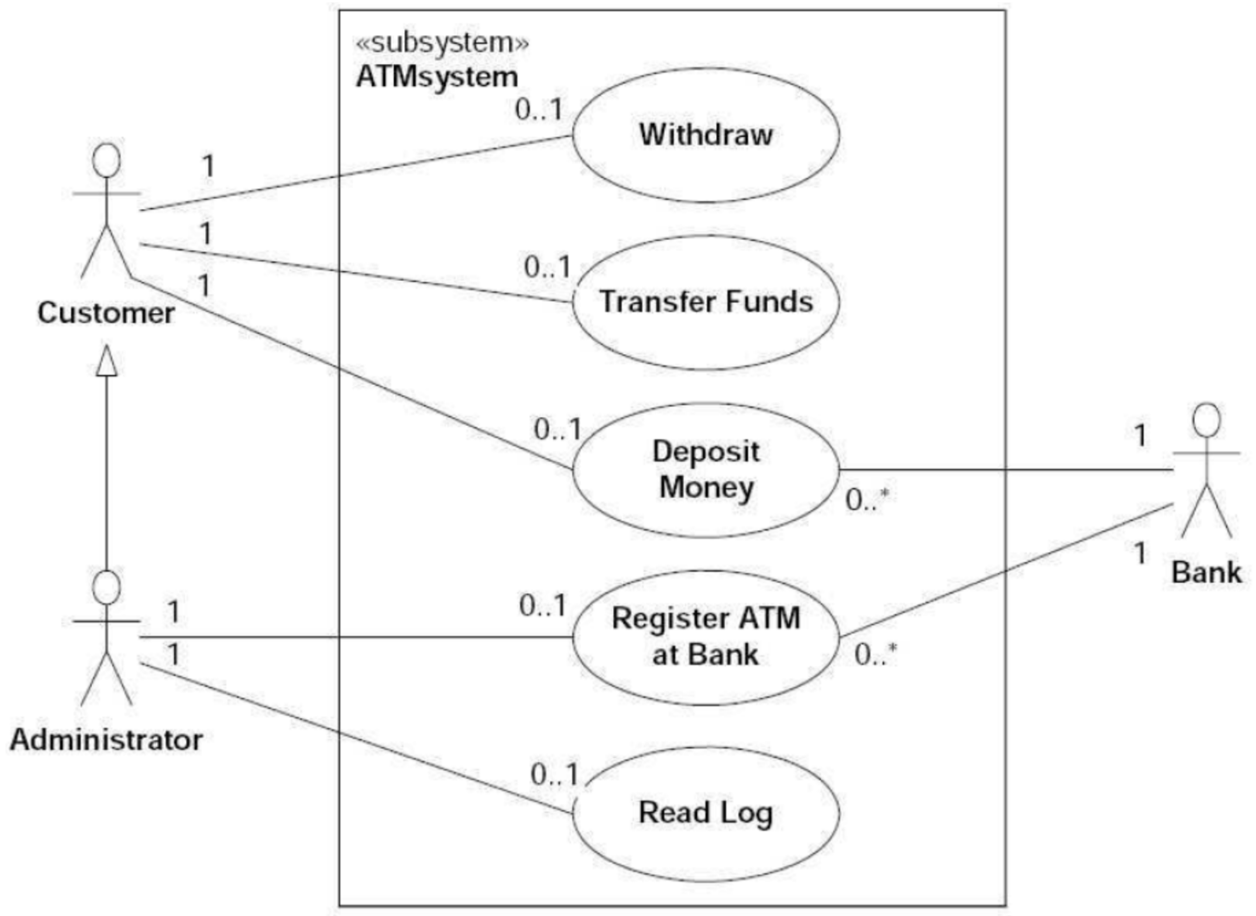
Steps:

Actor actions	**System responses**
1. Choose 'Open…' command	2. File open dialog appears
3a. Select text field	
3b. Type file name	
4. Click 'Open'	5. Dialog disappears

Simplified ATM machine

- Propose a use case diagram for an ATM machine for withdrawing cash. Make the use case simple yet informative; only include the major features.

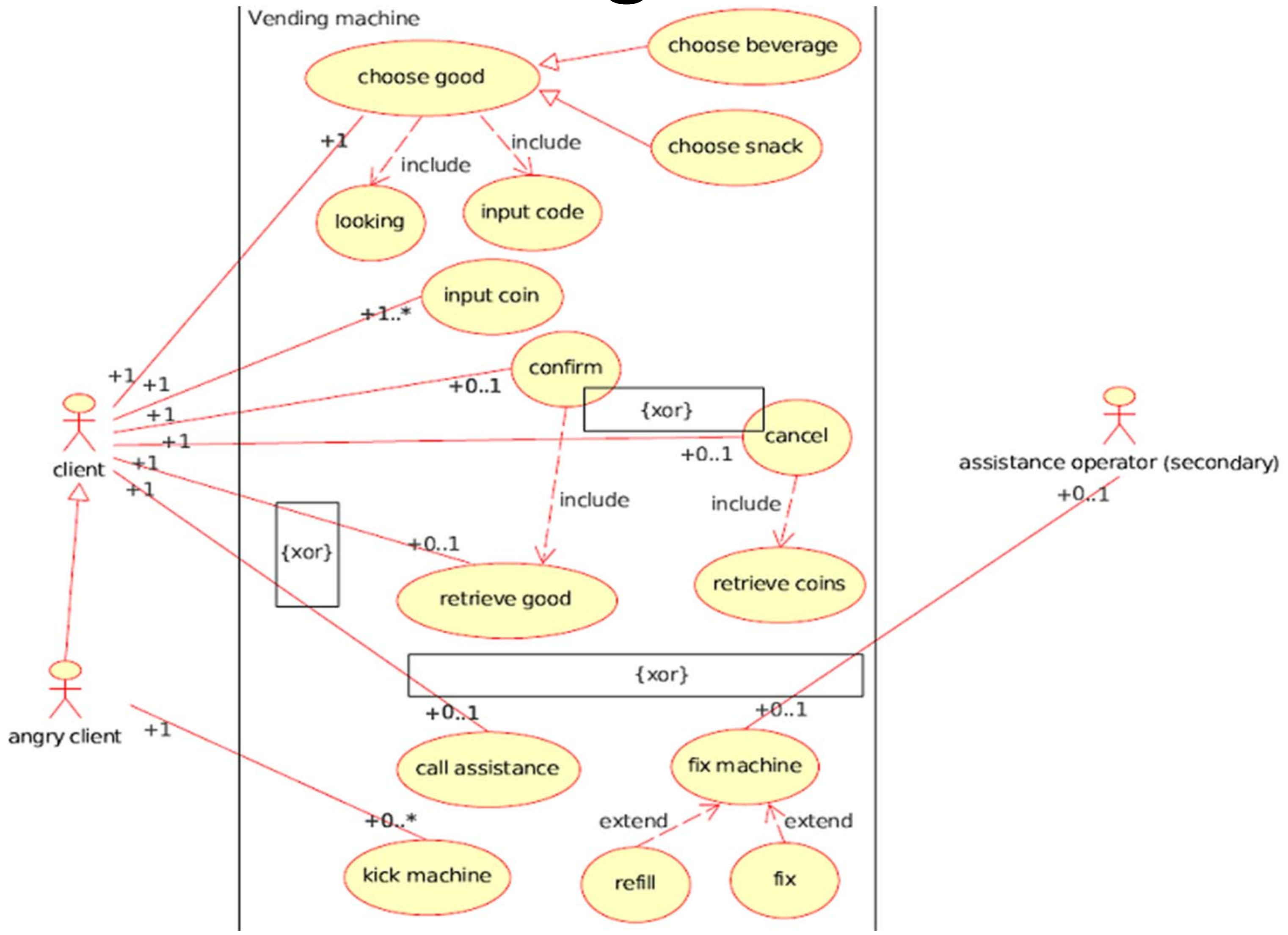
Simplified ATM machine



Vending machine

- Propose a use case diagram for a vending machine that sells beverages and snacks. Make use of inclusion and extension associations, mark multiplicities and remember that a vending machine may need technical assistance from time to time.

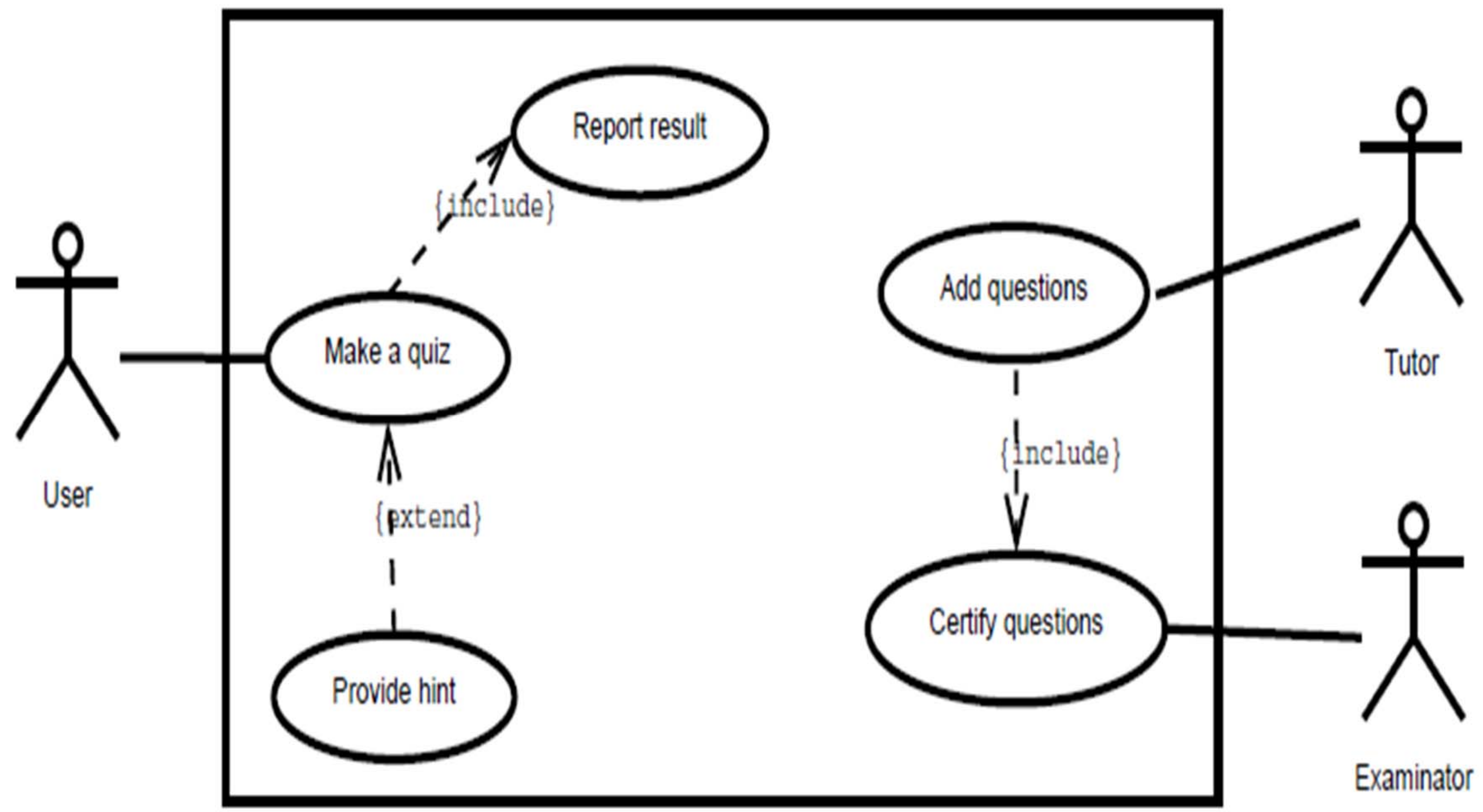
Vending machine



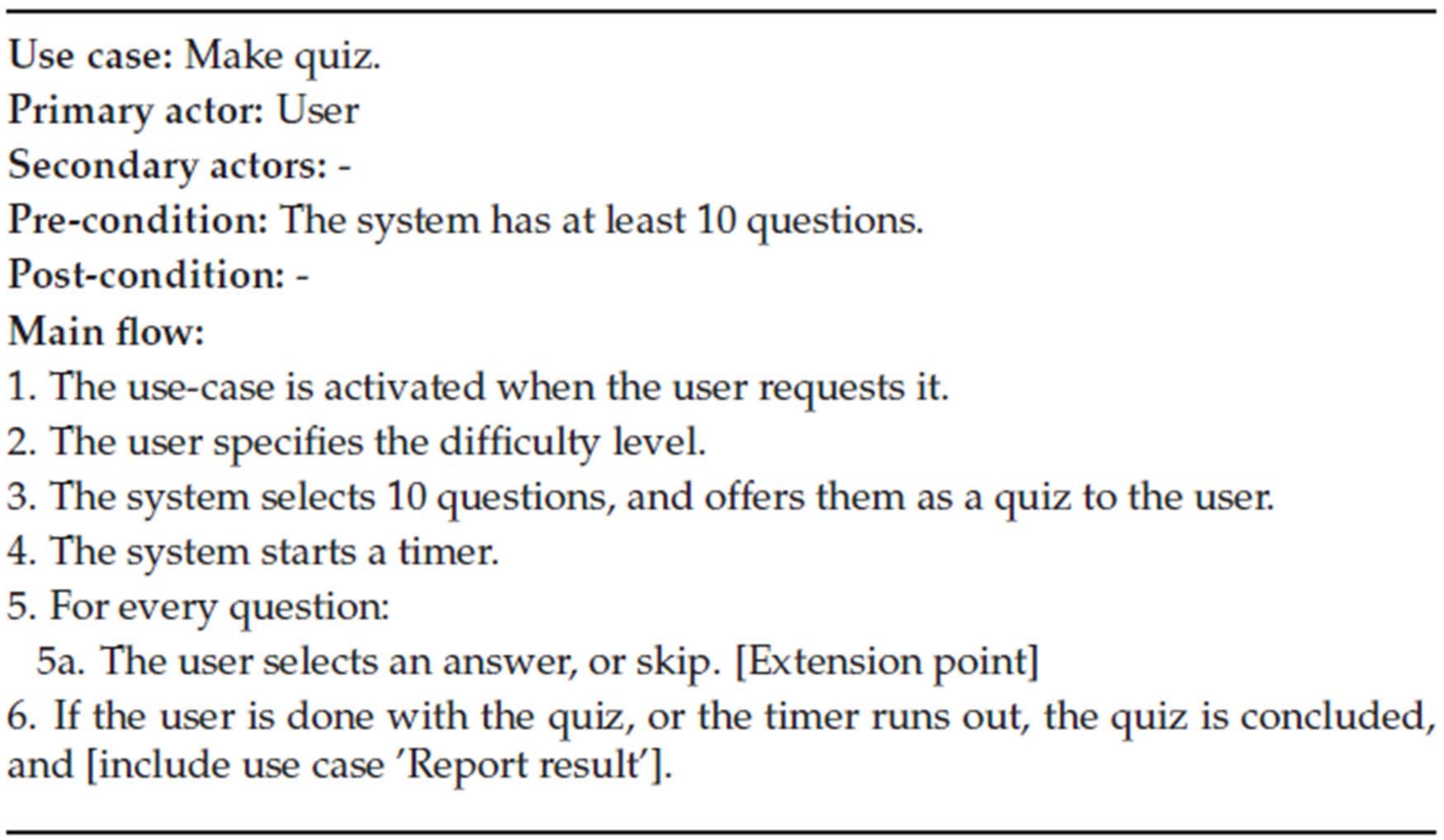
Exercise: Use Cases

- Do you know that it costs a lot of money to get a ’Certified Java Programmer’ certificate? It could cost you thousands of euros. Let’s imagine we will develop a browser-based training system to help people prepare for such a certification exam.
  - A user can request a quiz for the system. The system picks a set of questions from its database, and compose them together to make a quiz. It rates the user’s answers, and gives hints if the user requests it.
  - In addition to users, we also have tutors who provide questions and hints. And also examinators who must certify questions to make sure they are not too trivial, and that they are sensical.

Solution


Report result
{include}
Make a quiz
{extend}
Provide hint
Add questions
{include}
Certify questions
User
Tutor
Examinator

Solution



Use case: Make quiz.
Primary actor: User
Secondary actors: -
Pre-condition: The system has at least 10 questions.
Post-condition: -
Main flow:
1. The use-case is activated when the user requests it.
2. The user specifies the difficulty level.
3. The system selects 10 questions, and offers them as a quiz to the user.
4. The system starts a timer.
5. For every question:
   5a. The user selects an answer, or skip. [Extension point]
6. If the user is done with the quiz, or the timer runs out, the quiz is concluded, and [include use case 'Report result'].