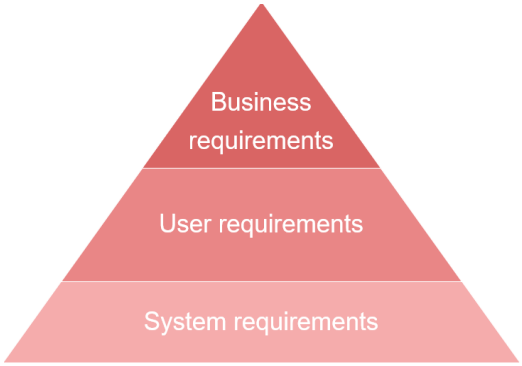


Week-4-Lesson-1
Understanding Requirements, Use Case and Use case Description

Abdus Sattar
Assistant Professor
Department of Computer Science and Engineering
Daffodil International University
Email: abdus.cse@diu.edu.bd

Daffodil International University

Topics Covered

- Requirements Engineering
- Requirements analysis
- Elements of Requirements Engineering
- Classification of Requirements
- Functional Requirements
- Non-Functional Requirements
- Use Case Diagram
- Use Case Description

Requirements Engineering

- **Requirements Engineering**
  - Requirements are statements of what the system must do, how it must behave, the properties it must exhibit, the qualities it must possess, and the constraints that the system and its development must satisfy.
- **Requirements analysis**
  - specifies software's operational characteristics
  - indicates software's interface with other system elements
  - establishes constraints that software must meet

Requirements Engineering

1. Inception—ask a set of questions that establish …
   - basic understanding of the problem
   - the people who want a solution
   - the nature of the solution that is desired, and
   - the effectiveness of preliminary communication and collaboration between the customer and the developer
2. Elicitation—elicit requirements from all stakeholders
3. Elaboration—create an analysis model that identifies data, function and behavioral requirements
4. Negotiation—agree on a deliverable system that is realistic for developers and customers

Requirements Engineering

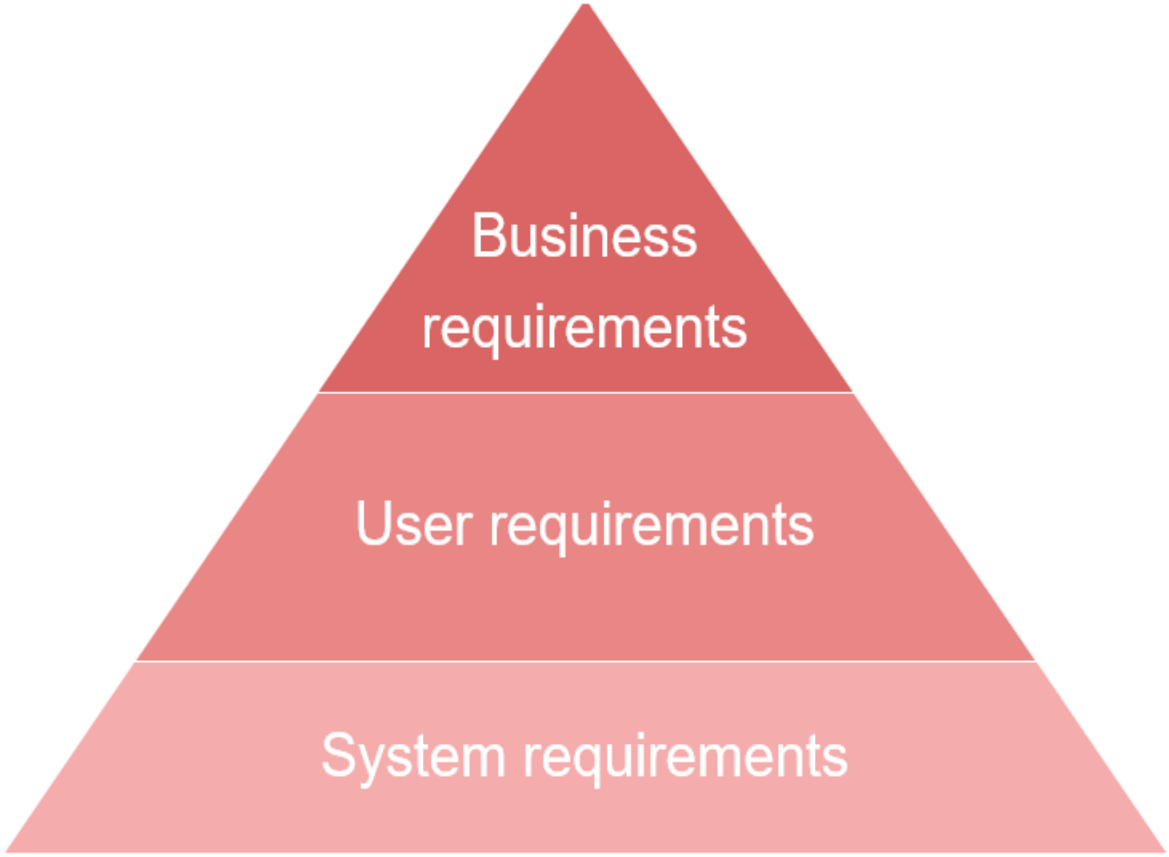
5. Specification—can be any one (or more) of the following:
   - A written document
   - A set of models
   - A formal mathematical
   - A collection of user scenarios (use-cases)
   - A prototype
6. Validation—a review mechanism that looks for
   - errors in content or interpretation
   - areas where clarification may be required
   - missing information
   - inconsistencies (a major problem when large products or systems are engineered)
   - conflicting or unrealistic (unachievable) requirements.
7. Requirements management

Classification of Requirements

Requirements classification

High level

Detailed



Classification of requirements

- **Business requirements.** These include high-level statements of goals, objectives, and needs.
- **Stakeholder requirements.** The needs of discrete stakeholder groups are also specified to define what they expect from a particular solution.
- **Solution requirements.** Solution requirements describe the characteristics that a product must have to meet the needs of the stakeholders and the business itself.
  - **Nonfunctional** requirements describe the general characteristics of a system. They are also known as *quality attributes*.
  - **Functional** requirements describe how a product must behave, what its features and functions.
- **Transition requirements.** An additional group of requirements defines what is needed from an organization to successfully move from its current state to its desired state with the new product.

Functional Requirements

- Functional requirements describe system behavior under specific conditions and include the product features and functions which web & app developers must add to the solution. Such requirements should be precise both for the development team and stakeholders.
- The list of examples of functional requirements includes:
  - Business Rules
  - Transaction corrections, adjustments, and cancellations
  - Administrative functions
  - Authentication
  - Authorization levels
  - Audit Tracking
  - External Interfaces
  - Certification Requirements
  - Reporting Requirements
  - Historical Data

Example of Functional Requirements

Here, are some examples of non-functional requirement:

1. The software automatically validates customers against the ABC Contact Management System
2. The Sales system should allow users to record customers sales
3. The background color for all windows in the application will be blue and have a hexadecimal RGB color value of 0x0000FF.
4. Only Managerial level employees have the right to view revenue data.
5. The software system should be integrated with banking API
6. The software system should pass Section 508 accessibility requirement.

Non-Functional Requirements

- A non-functional requirement defines the quality attribute of a software system. They represent a set of standards used to judge the specific operation of a system. Example, how fast does the website load?
- Some typical non-functional requirements are:
  - Performance – for example Response Time, Throughput, Utilization, Static Volumetric
  - Capacity
  - Availability
  - Reliability
  - Recoverability
  - Maintainability
  - Serviceability
  - Security
  - Regulatory
  - Manageability
  - Environmental
  - Data Integrity
  - Usability

Example of Non-Functional Requirements

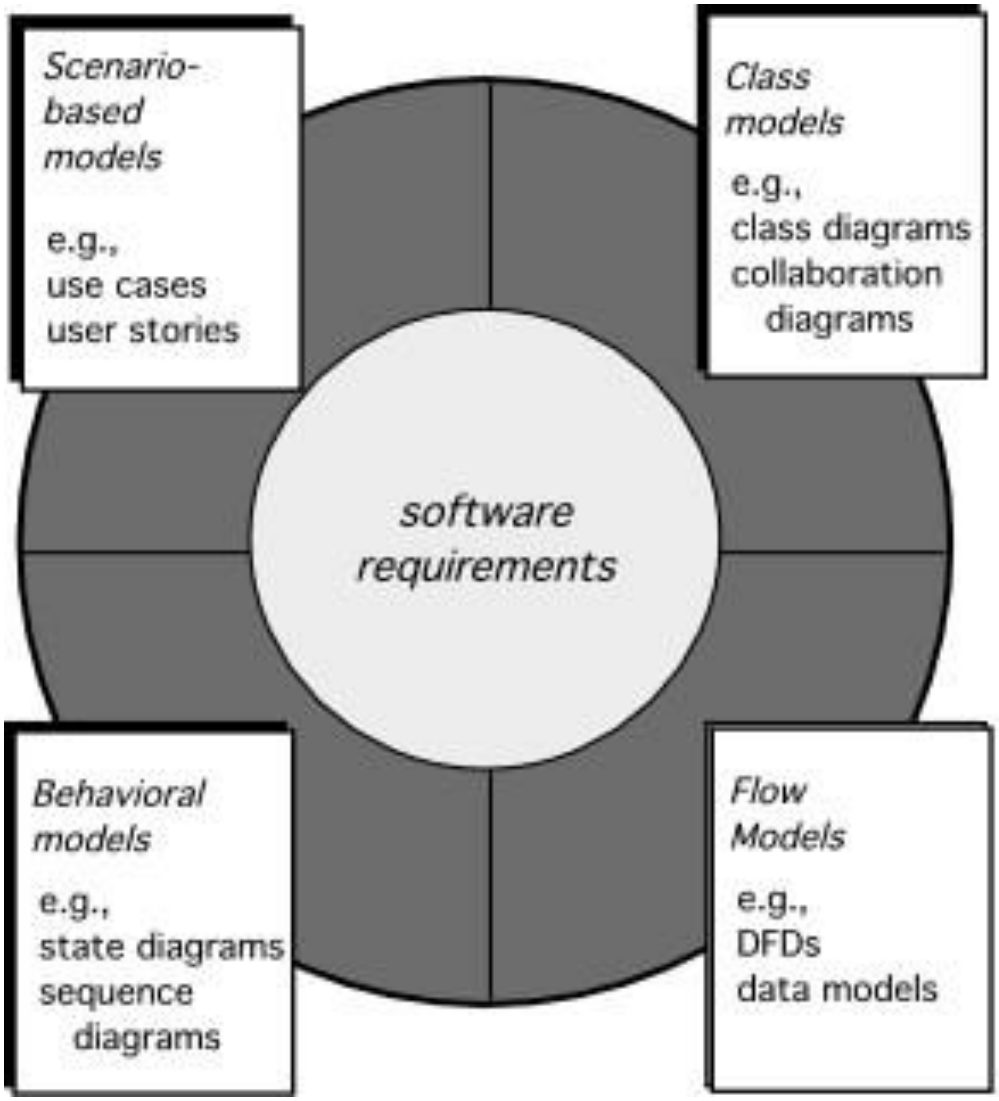
Here, are some examples of non-functional requirement:

1. Users must change the initially assigned login password immediately after the first successful login. Moreover, the initial should never be reused.
2. Employees never allowed to update their salary information. Such attempt should be reported to the security administrator.
3. Every unsuccessful attempt by a user to access an item of data shall be recorded on an audit trail.
4. A website should be capable enough to handle 20 million users with affecting its performance
5. The software should be portable. So moving from one OS to other OS does not create any problem.
6. Privacy of information, the export of restricted technologies, intellectual property rights, etc. should be audited.

Elements of the analysis model

- Elements of the analysis model
  - Scenario-based elements
    - **Functional**—processing narratives for software functions
    - **Use-case**—descriptions of the interaction between an "actor" and the system
  - Class-based elements
    - Implied by scenarios
  - Behavioral elements
    - State diagram
  - Flow-oriented elements
    - Data flow diagram

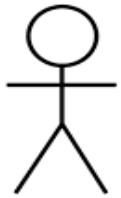
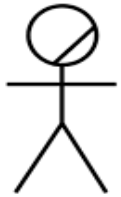
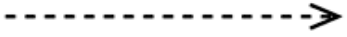
Elements of Requirements Analysis



Use-Cases

- Use cases describe the interaction between the system and external users that leads to achieving particular goals.
- Each use case includes three main elements:
  - **Actors.** These are the users outside the system that interact with the system.
  - **System**. The system is described by functional requirements that define an intended behavior of the product.
  - **Goals**. The purposes of the interaction between the users and the system are outlined as goals.
- There are two formats to represent use cases:
  - Use case specification/description
  - Use case diagram

Use-Cases Elements

Symbol Name	Symbol
Actor	
Business Actor	
Use Case	
Business Use Case	
Association	
Dependency	
Generalization	

Use-Cases Elements

Connection between Actor and Use Case

Boundary of system

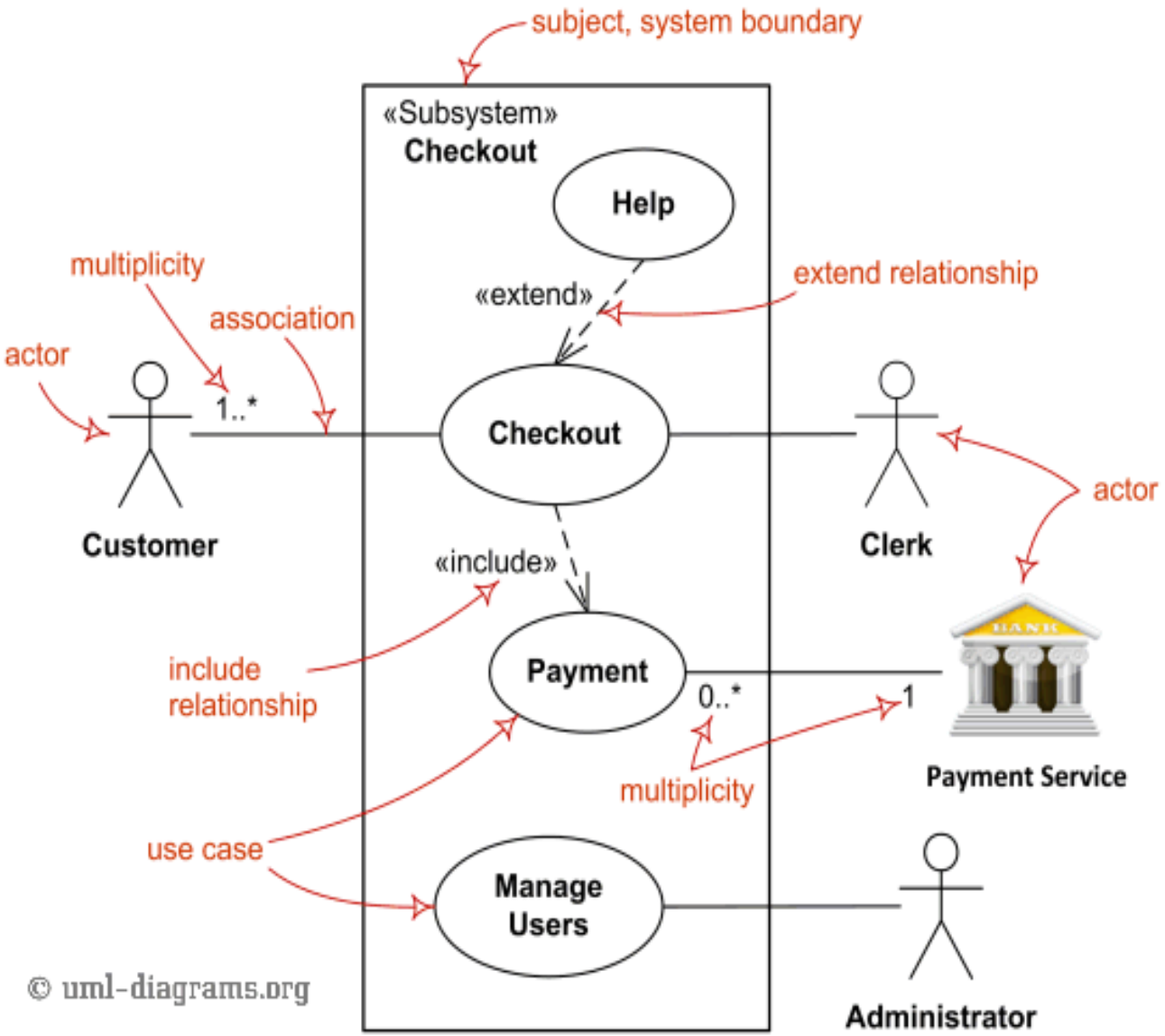
<<include>>

Include relationship between Use Cases (one UC must call another; e.g., Login UC includes User Authentication UC)

<<extend>>

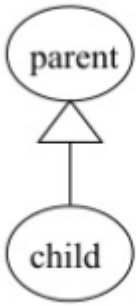
Extend relationship between Use Cases (one UC calls Another under certain condition; think of if-then decision points)

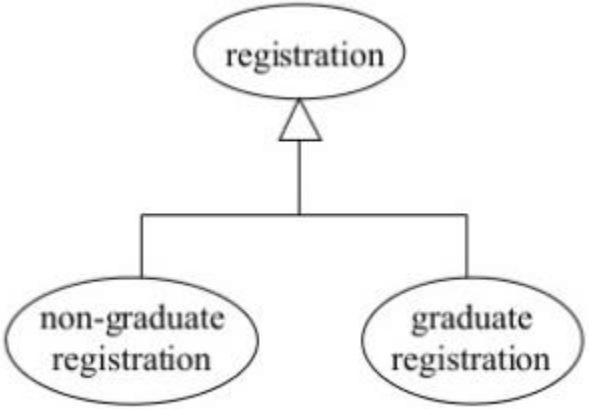
Use Cases Elements

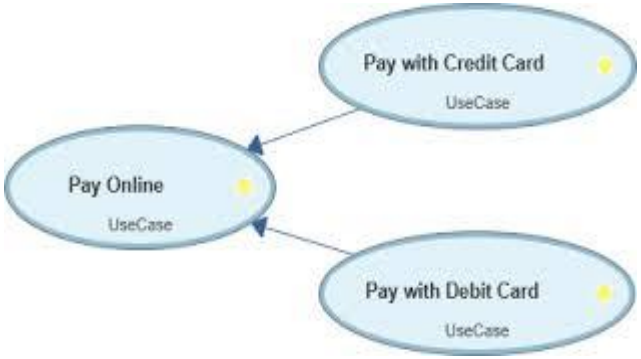


Use-Cases Generalization

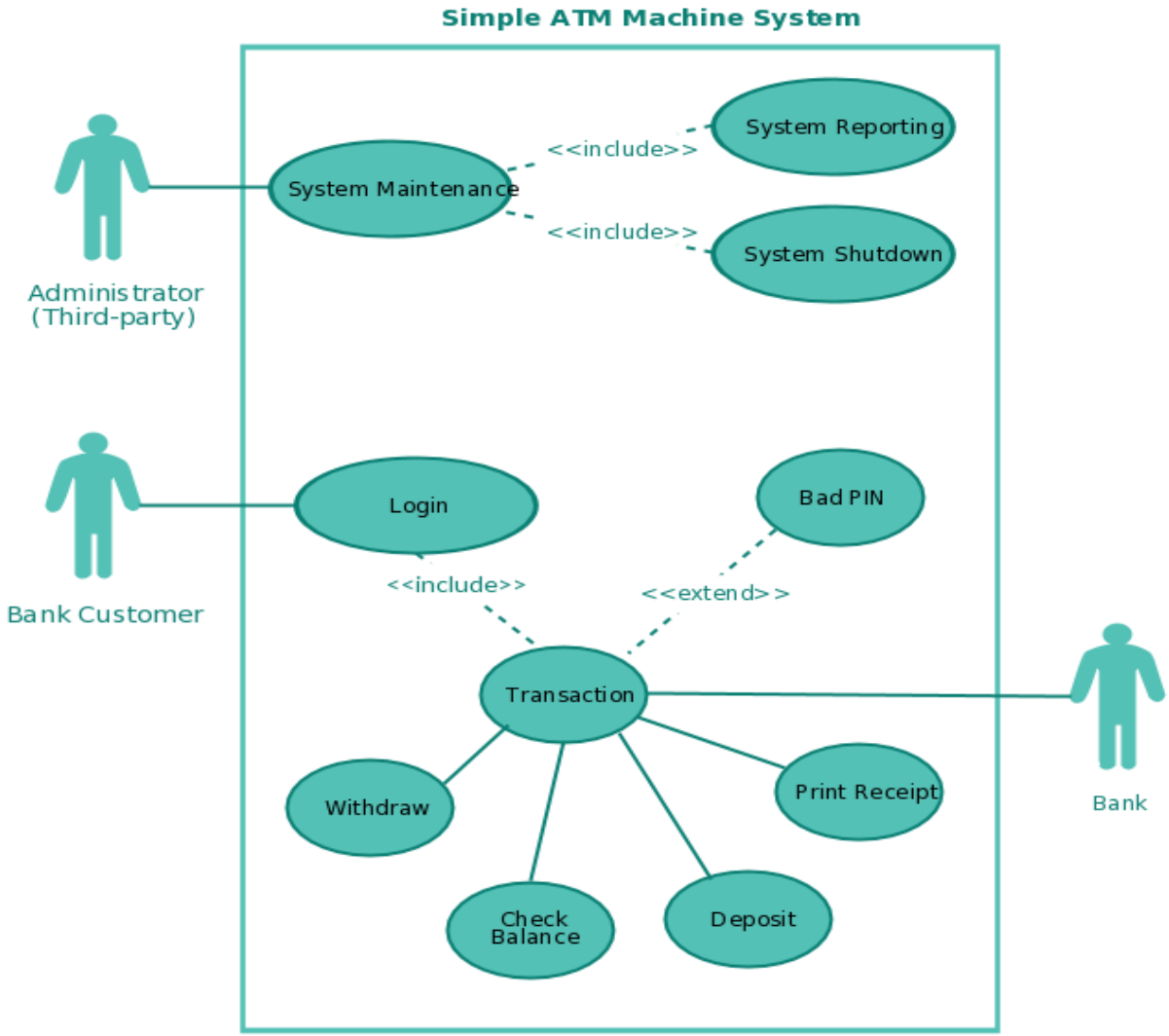
- The child use case inherits the behavior meaning of the parent use case
- The Child may add to or override the behavior of its parent.



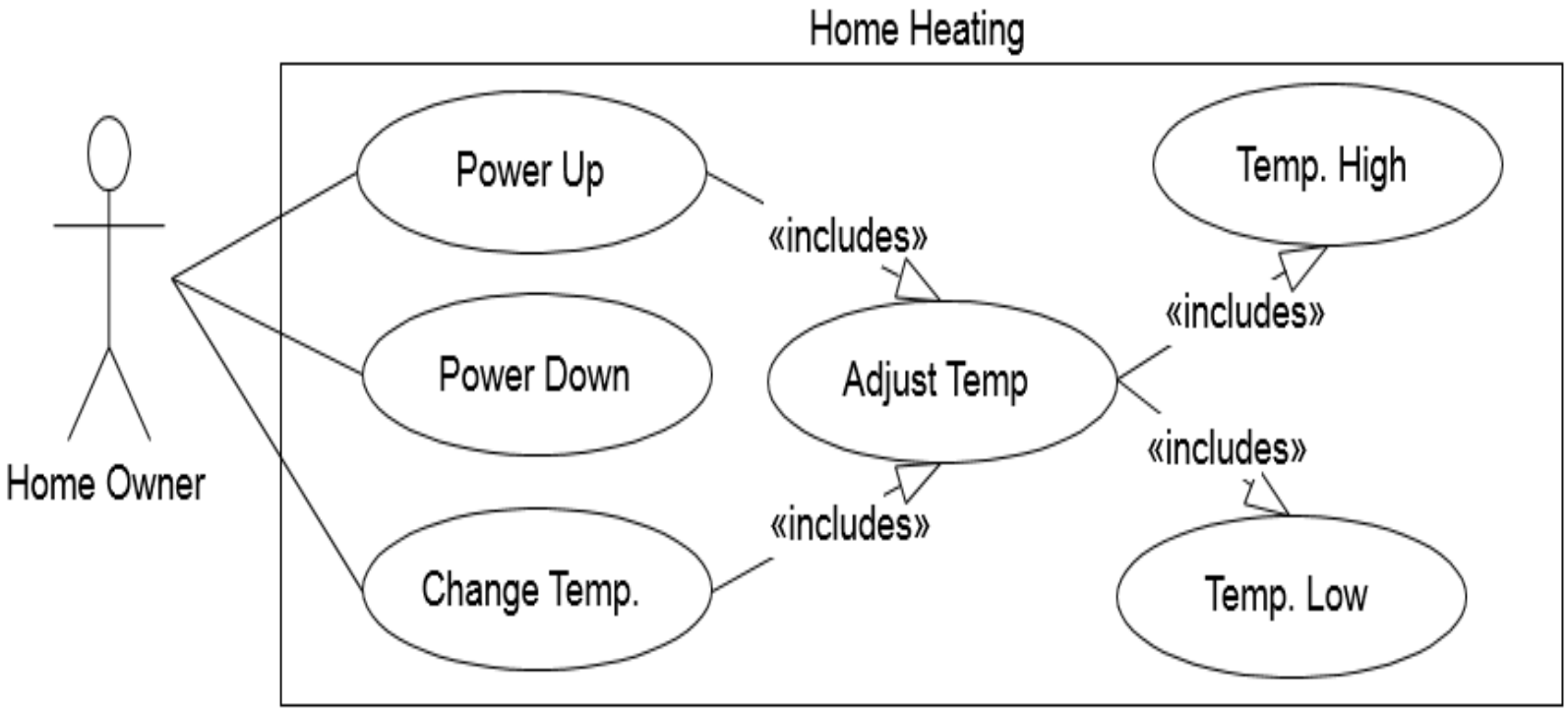




Use-Cases Diagram



Home Heating System



Home Heating System

Use Case Description

Use case:	**Power Up**
Actors:	**Home Owner (initiator)**
Type:	Primary and essential
Description:	The Home Owner turns the power on. **Perform Adjust Temp**. If the temperature in all rooms is above the desired temperature, no actions are taken.
Cross Ref.:	Requirements XX, YY, and ZZ
Use-Cases:	Perform Adjust Temp

Home Heating System

Use case: **Adjust Temp**

Actors: **System (initiator)**

Type: Secondary and essential

Description: Check the temperature in each room. For each room:

Below target: **Perform Temp Low**

Above target: **Perform Temp High**

Cross Ref.: Requirements XX, YY, and ZZ

Use-Cases: Temp Low, Temp High

Home Heating System

Use case: **Temp Low**

Actors: **System (initiator)**

Type: Secondary and essential

Description: Open room valve, start pump if not started.
If water temp falls below threshold,
open fuel value and ignite burner.

Cross Ref.: Requirements XX, YY, and ZZ

Use-Cases: None

Use Case Scenario

Home Assignment distribution and Collection System(HACS)

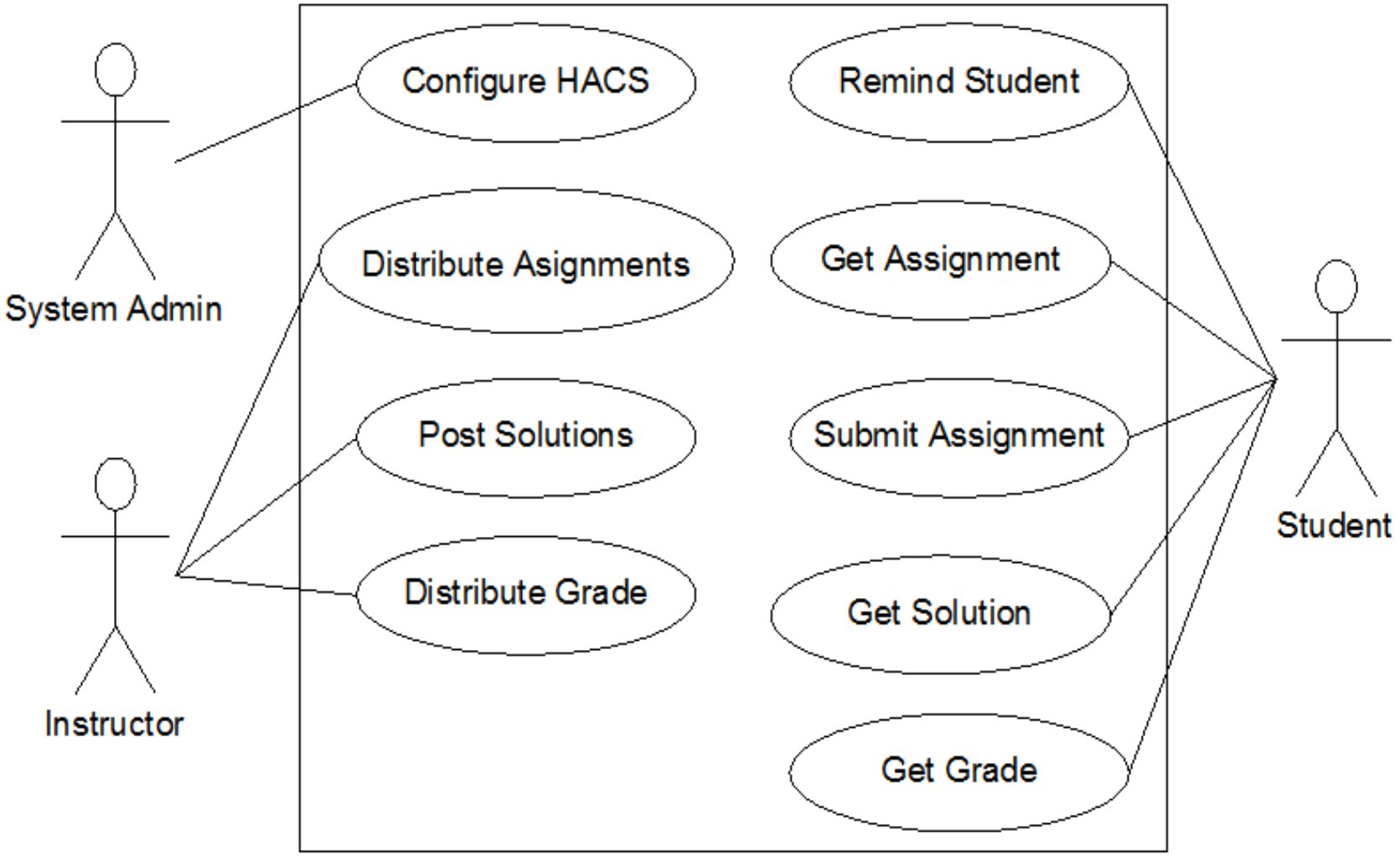
Homework assignment and collection are an integral part of any educational system. Today, this task is performed manually. What we want the homework assignment distribution and collection system to do is to automate this process.

The system will be used by the Instructor/Teacher to distribute the homework assignments, review the students' solutions, distribute suggested solution, and distribute student grades on each assignment.

This system will also help the students by automatically distributing the assignments to the students, provide a facility where the students can submit their solutions, remind the students when an assignment is almost due, remind the students when an assignment is overdue.

Home Assignment distribution and Collection System

Use Case Diagram

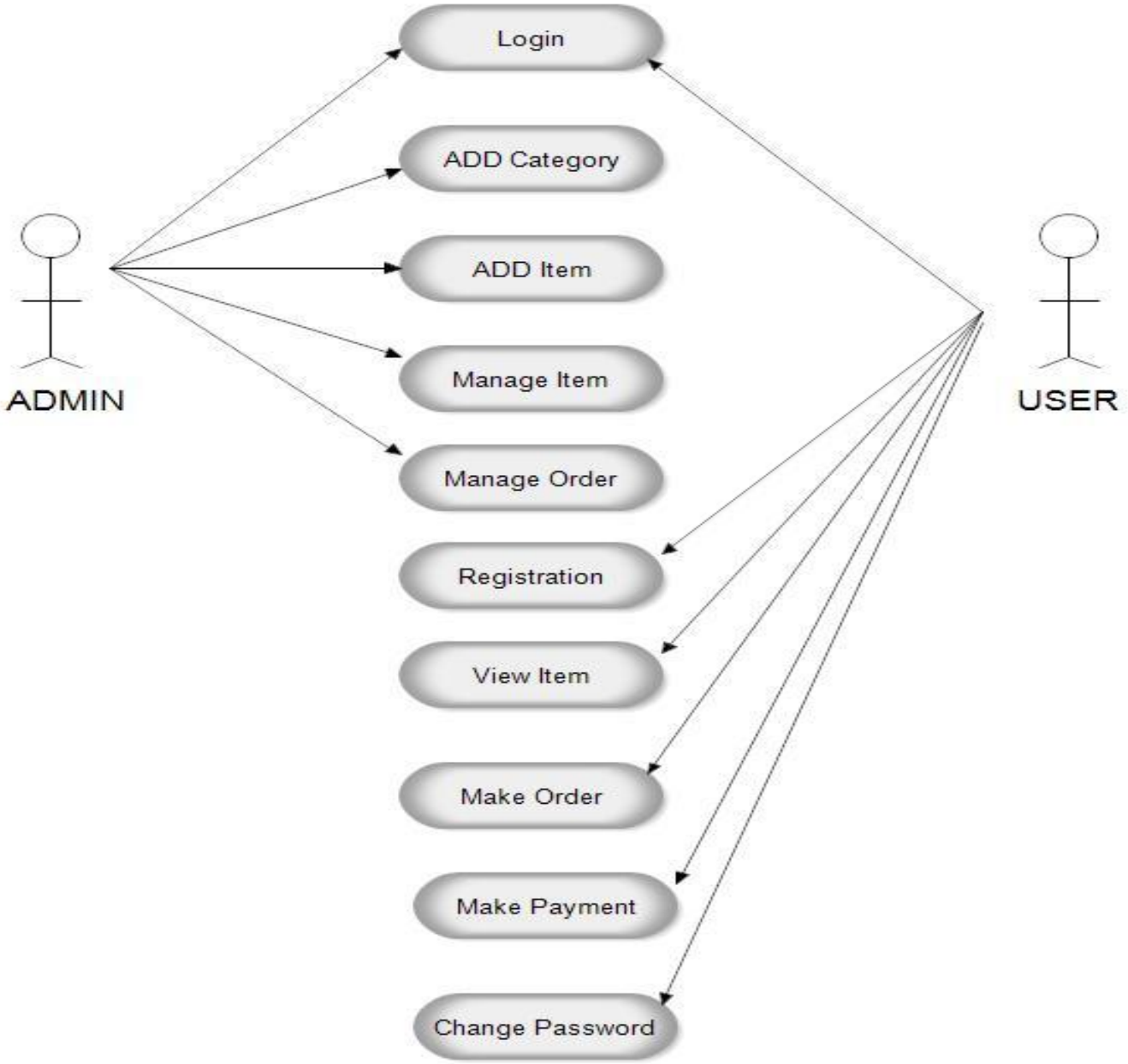


HACS Use Cases Description

Use case: **Distribute Assignments**
Actors: Instructor (initiator)
Type: Primary and essential
Description: The Instructor completes an assignment and submits it to the system. The instructor will also submit the due date and the class the assignment is assigned for.

Cross Ref.: Requirements XX, YY, and ZZ
Use-Cases: *Configure HACS* must be done before any user (Instructor or Student) can use HACS

Use Case Diagram for Online Shopping Website



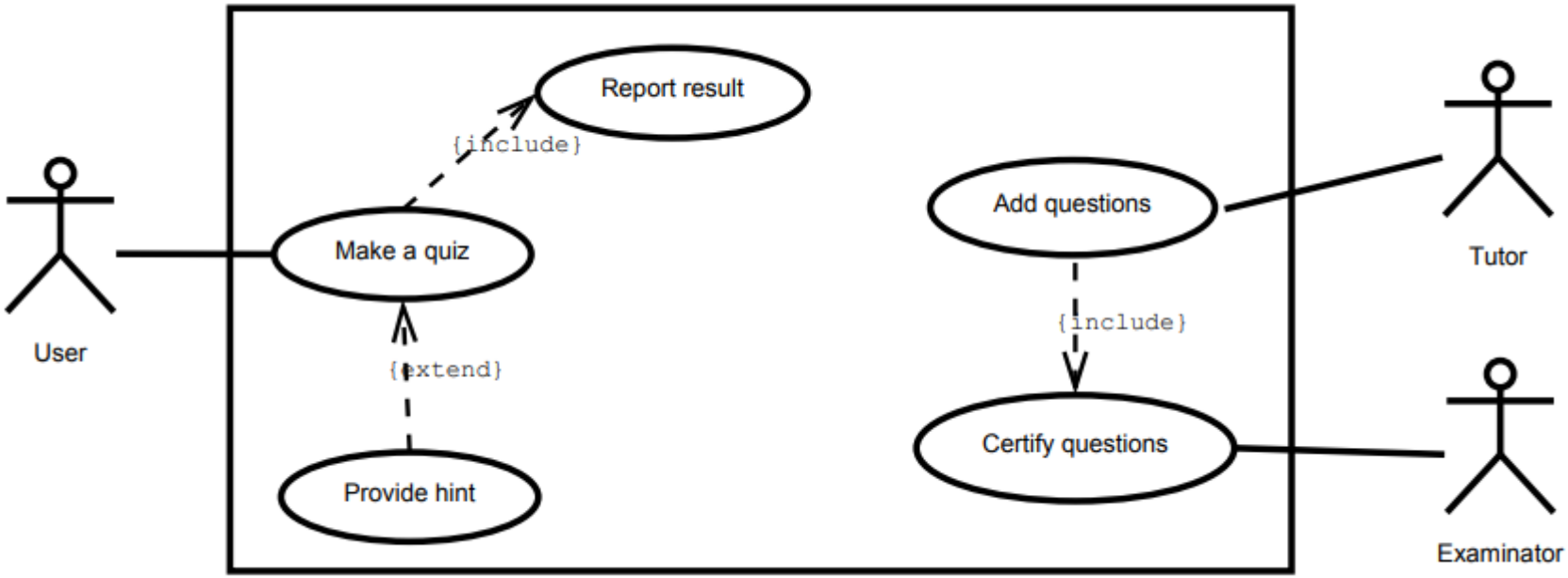
Example: Use Case Scenario

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 examinations who must certify questions to make sure they are not too trivial, and that they are sensual.

Make a use case diagram to model this system. Work out some of your use cases. Since we don't have real stake holders here, you are free to fill in details you think is sensual for this example.

Use Case Diagram



Use Case Description

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'].

References:

1. **Software Engineering A practitioner's Approach** by Roger S. Pressman, 7th edition, McGraw Hill, 2010.
2. **Software Engineering by Ian Sommerville,** 9th edition, Addison-Wesley, 2011
3. **FUNCTIONAL VS NON-FUNCTIONAL REQUIREMENTS: MAIN DIFFERENCES & EXAMPLES** https://theappsolutions.com/blog/development/functional-vs-non-functional-requirements/