ITSS SOFTWARE DEVELOPMENT

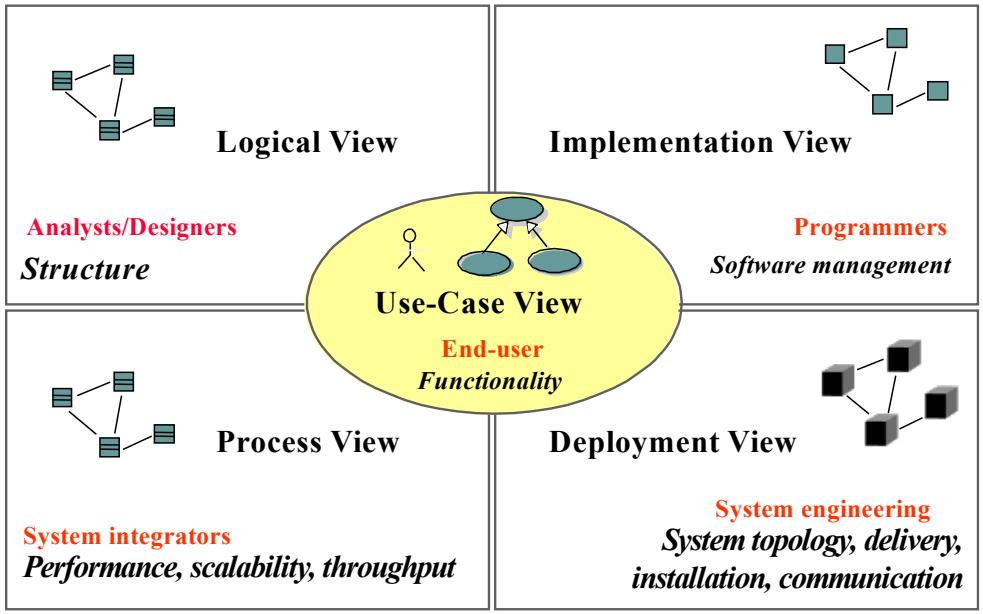
2. REQUIREMENT MODELING WITH UC

Nguyen Thi Thu Trang
trangntt@soict.hust.edu.vn

Some slides extracted from IBM coursewares

No Single Model Is Sufficient

- No single model is sufficient. Every non-trivial system is best approached through a small set of nearly independent models.
  - Create models that can be built and studied separately, but are still interrelated.



Content

1. Requirements
2. Use case diagram
3. Use case specification/scenario
4. Glossary
5. Supplementary Specification

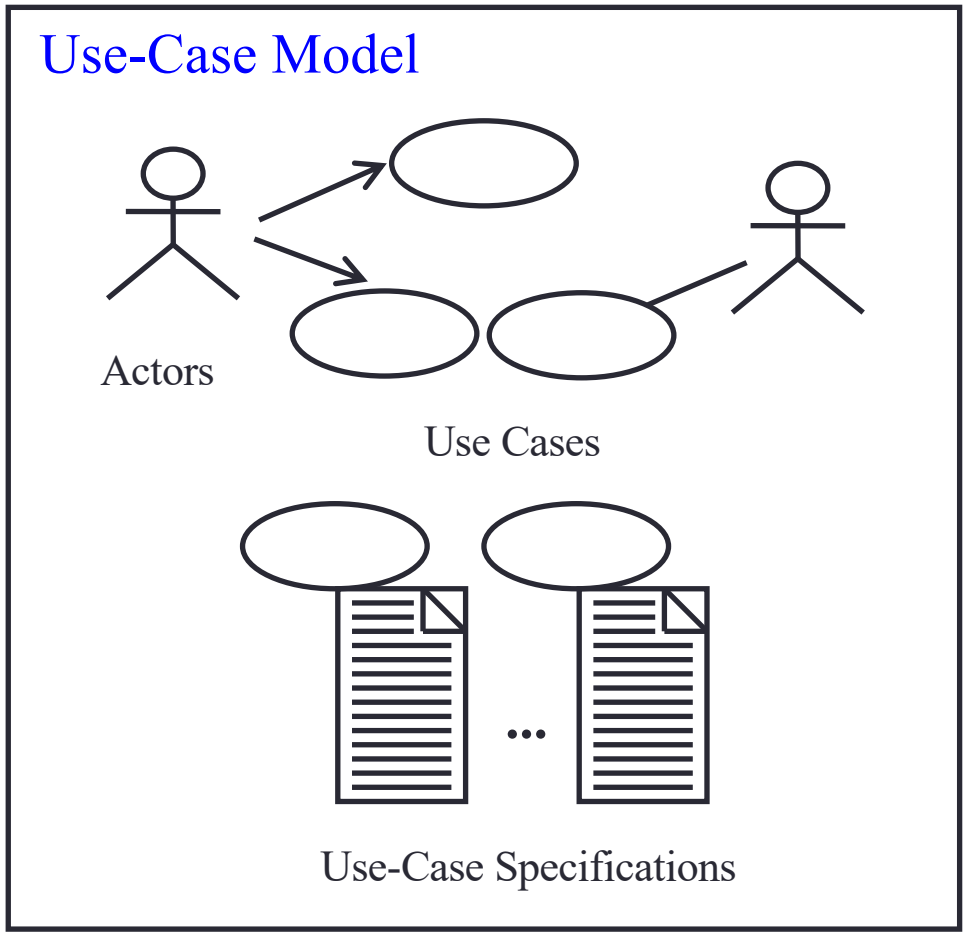
Review: Software Requirements Analysis process

- Purpose: "to establish the requirements of the software elements of the system"
- Main items written on the brief requirement description
  - **System environmental conditions** under which the software is to perform.
  - The **functional requirements** and the **interface requirements.**
  - **Data definition and database requirements**.
  - Some **non-functional requirement** items such as reliability, usability, time efficiency
  - **Qualification requirements**: The requirements are used as criteria or conditions to qualify a software product as complying with its specifications.

Purpose of Requirement

- Establish and maintain agreement with the customers and other stakeholders on what the system should do.
- Give system developers a better understanding of the requirements of the system.
- Delimit the system.
- Provide a basis for planning the technical contents of the iterations.
- Provide a basis for estimating cost and time to develop the system.
- Define a user interface of the system.

Relevant Requirements Artifacts



Glossary

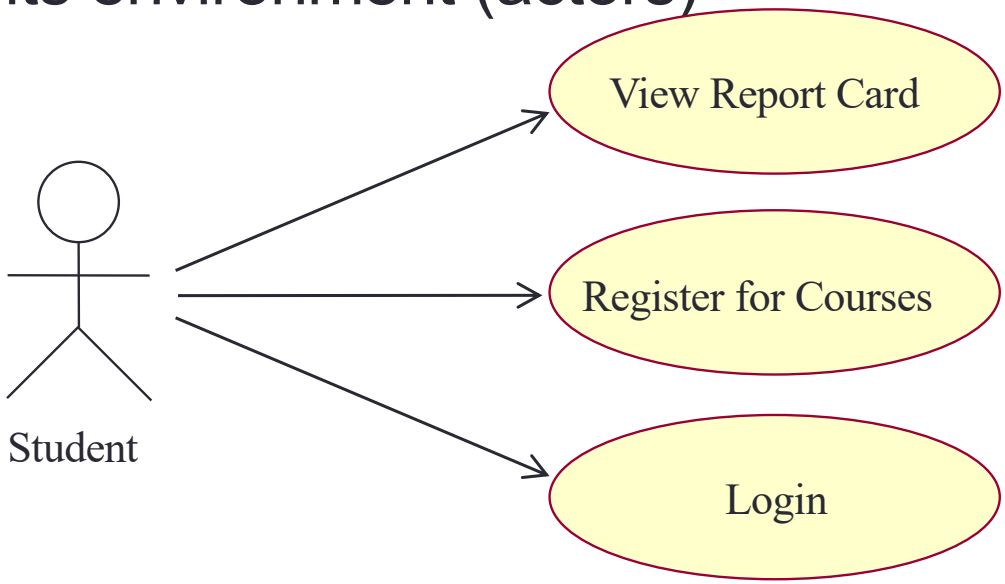
Supplementary Specification

Content

1. Requirements
2. Use case diagram
3. Use case specification/scenario
4. Glossary
5. Supplementary Specification

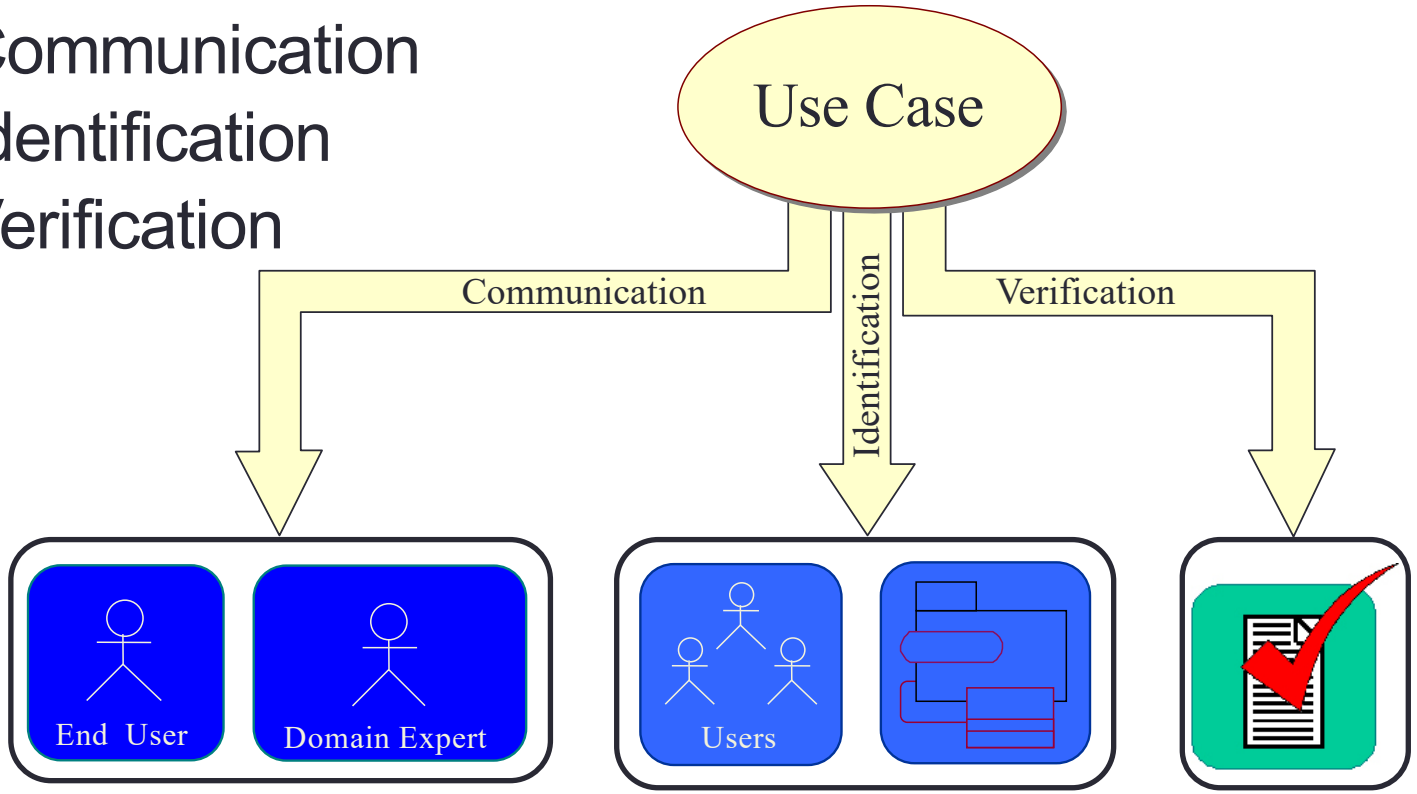
2.1. Overview of Use-Case Diagram

- A diagram modeling the dynamic aspects of systems that describes a software's functional requirements in terms of use cases.
- A model of the software's intended functions (use cases) and its environment (actors)



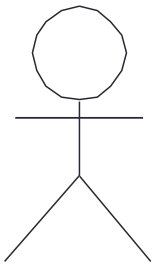
Benefits of a Use-Case Model

- Communication
- Identification
- Verification



Major Concepts in Use-Case Modeling

- An actor represents anything that interacts with the software.

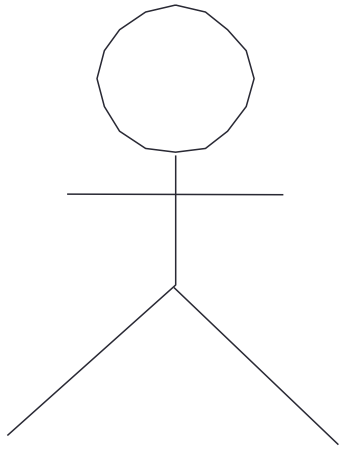


Actor

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

2.2. Actors

- Actors represent roles a user of the system can play
  - They can represent a human, a machine, or another system
  - They can be a peripheral device or even database
- 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

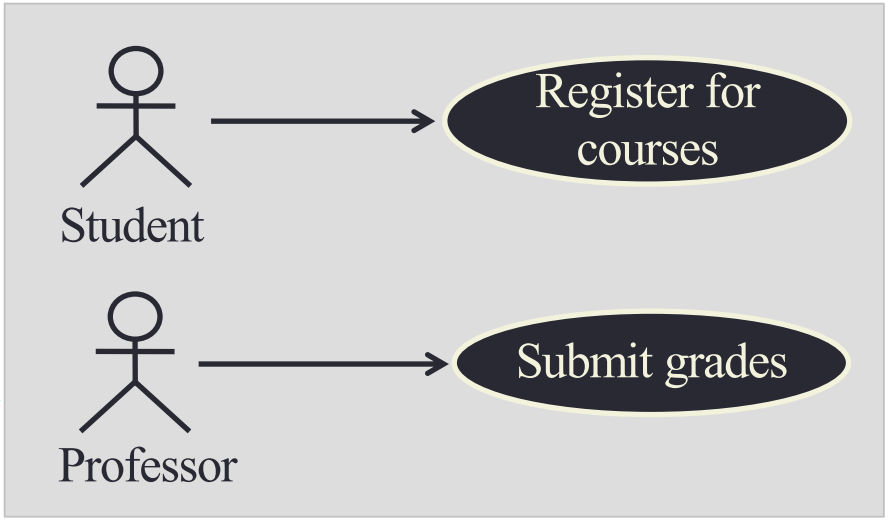
Actors and Roles

Charlie: Is employed as a math professor and is an economics undergraduate.

Jodie: Is a science undergraduate.

Charlie and Jodie both act as a Student.

Charlie also acts as a Professor.



Some guideline to extract actors

- Pay attention to a noun in the problem description, and then extract a subject of action as a Actor.
- Ensure that there are no any excesses and deficiencies between the problem description and Actors extracted.
- Actor names
  - should clearly convey the actor's role
  - good actor names describe their responsibilities

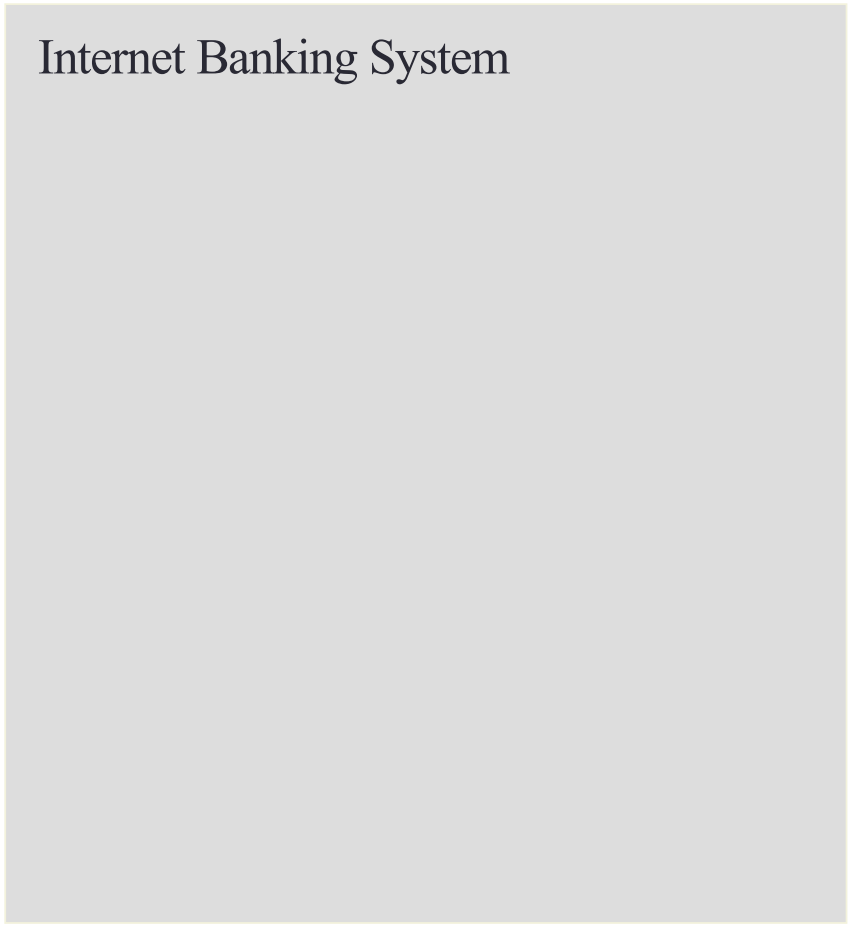
Put some questions to find actors

- Who or what uses the system?
- Who or what gets information from this system?
- Who or what provides information to the system?
- Where in the company is the system used?
- Who or what supports and maintains the system?
- What other systems use this system?

Internet banking system

- The internet banking system, allowing interbank network, communicates with bank customers via a web application. To perform transactions, customers have to log in the software. Customers may change password or view personal information.
- Customers can select any of transaction types: transfer (internal and in interbank network), balance inquiries, transaction history inquiries, electric receipt payment (via EVN software), online saving.
- In the transfer transaction, after receiving enough information from the customer, the software asks the bank consortium to process the request. The bank consortium forwards the request to the appropriate bank. The bank then processes and responses to the bank consortium which in turn notifies the result to the software.
- The bank officers may create new account for a customer, reset password, view transaction history of a customer.

Find actors in the Internet banking system



2.3. Use Cases

- Define 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

Some guidelines to extract use cases

- Pay attention to a verb in the problem description, and then extract a series of Actions as a UC.
- Ensure that there are no any excesses and deficiencies between the problem description and Use cases extracted.
- Check the consistency between Use Cases and related Actors.
- Conduct a survey to learn whether customers, business representatives, analysts, and developers all understand the names and descriptions of the use cases

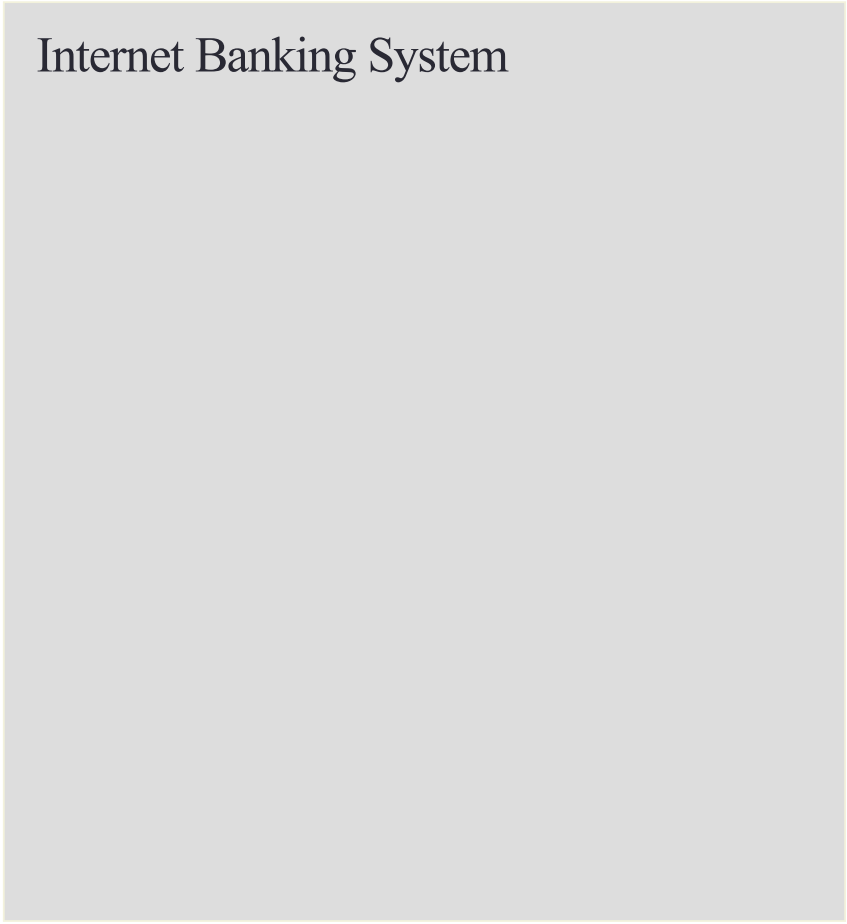
Use case name

- Be unique, intuitive, and self-explanatory
- Define clearly and unambiguously the observable result of value gained from the use case
- Be from the perspective of the actor that triggers the use case
- Describe the behavior that the use case supports
- Start with a verb and use a simple verb-noun combination

Put some questions to find use cases

- What are the goals of each actor?
- Why does the actor want to use the system?
- Will the actor create, store, change, remove, or read data in the system? If so, why?
- Will the actor need to inform the system about external events or changes?
- Will the actor need to be informed about certain occurrences in the system?

Find use cases in the Internet Banking system

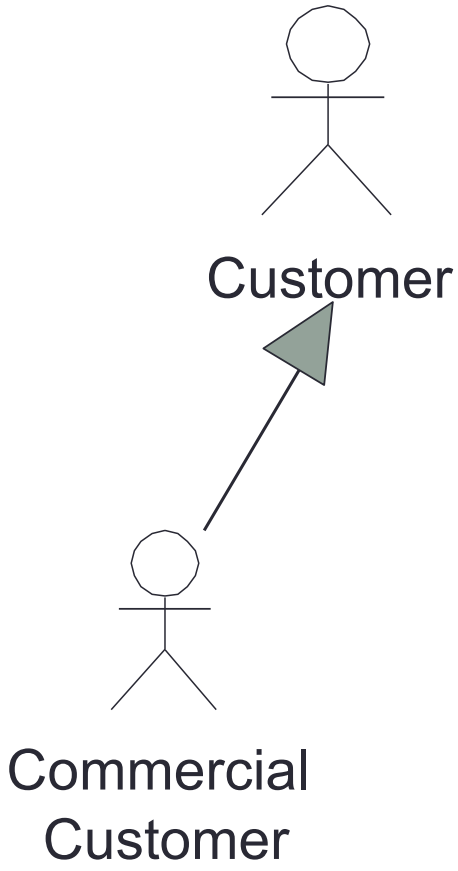


2.4. Relationships

- Not recommended to use many times
- Three kinds
  - Between actors: generalization, association
  - Between actor and use cases: association
  - Between use cases: generalization, include, extend

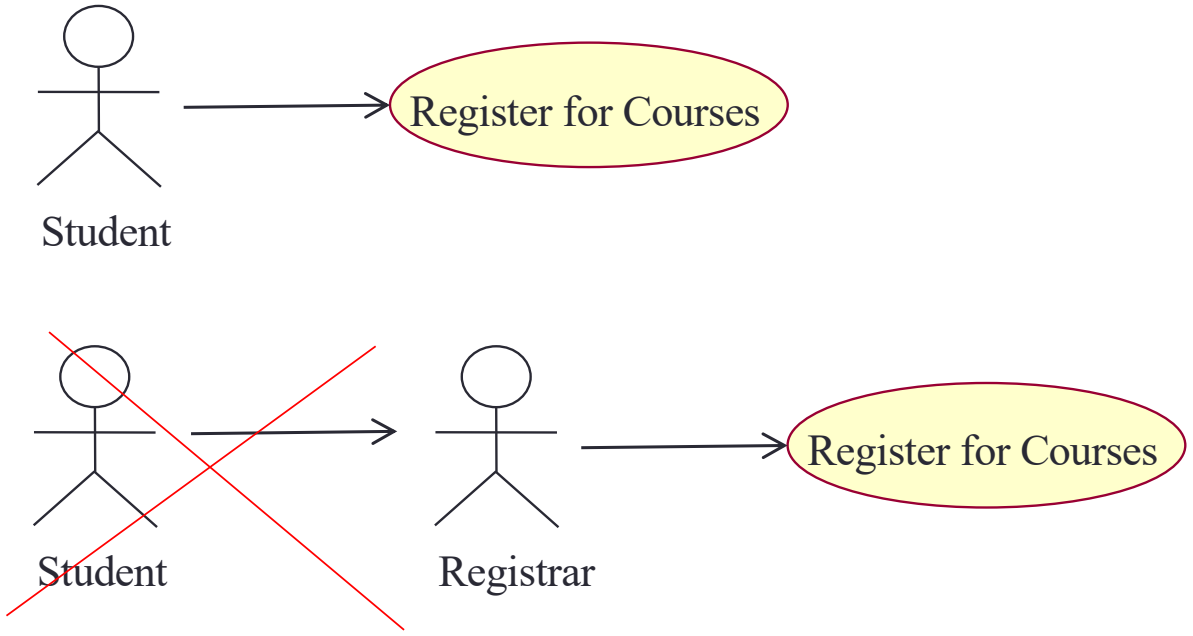
2.4.1. Between actors

- Generalization
  - The child actor inherits parent's characteristics and behaviors.



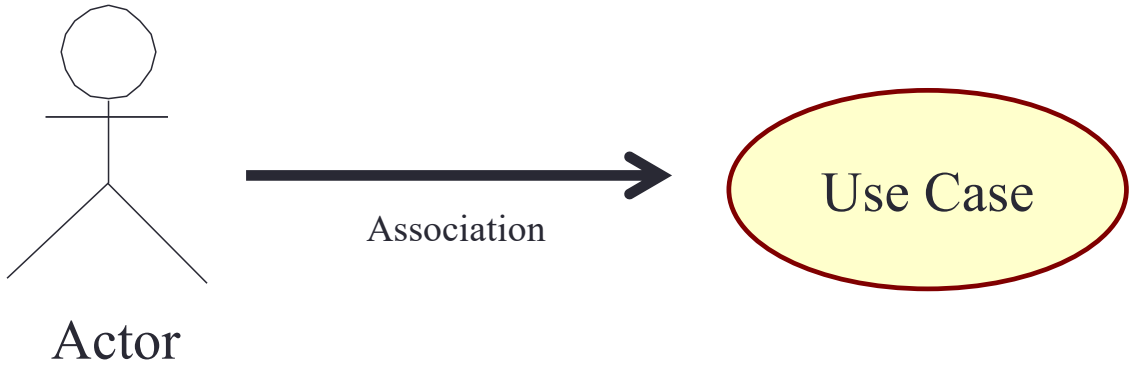
2.4.1. Between actors (2)

- Association
  - Consider the difference between two diagrams

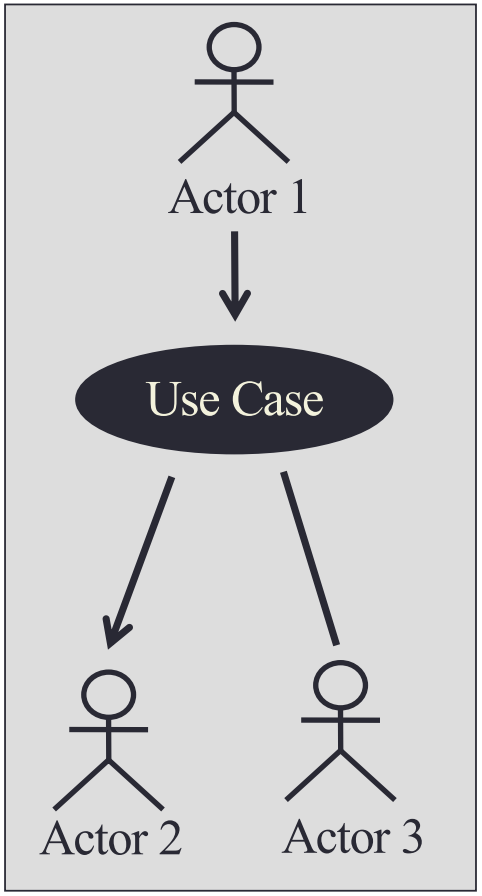


2.4.2. Between actor and use case

- Establish the actors that interact with related use cases → Use associations
  - Associations clarify the communication between the actor and use case.
  - Association indicate that the actor and the use case instance of the system communicate with one another, each one able to send and receive messages.
- The arrow head is optional but it's commonly used to denote the initiator.

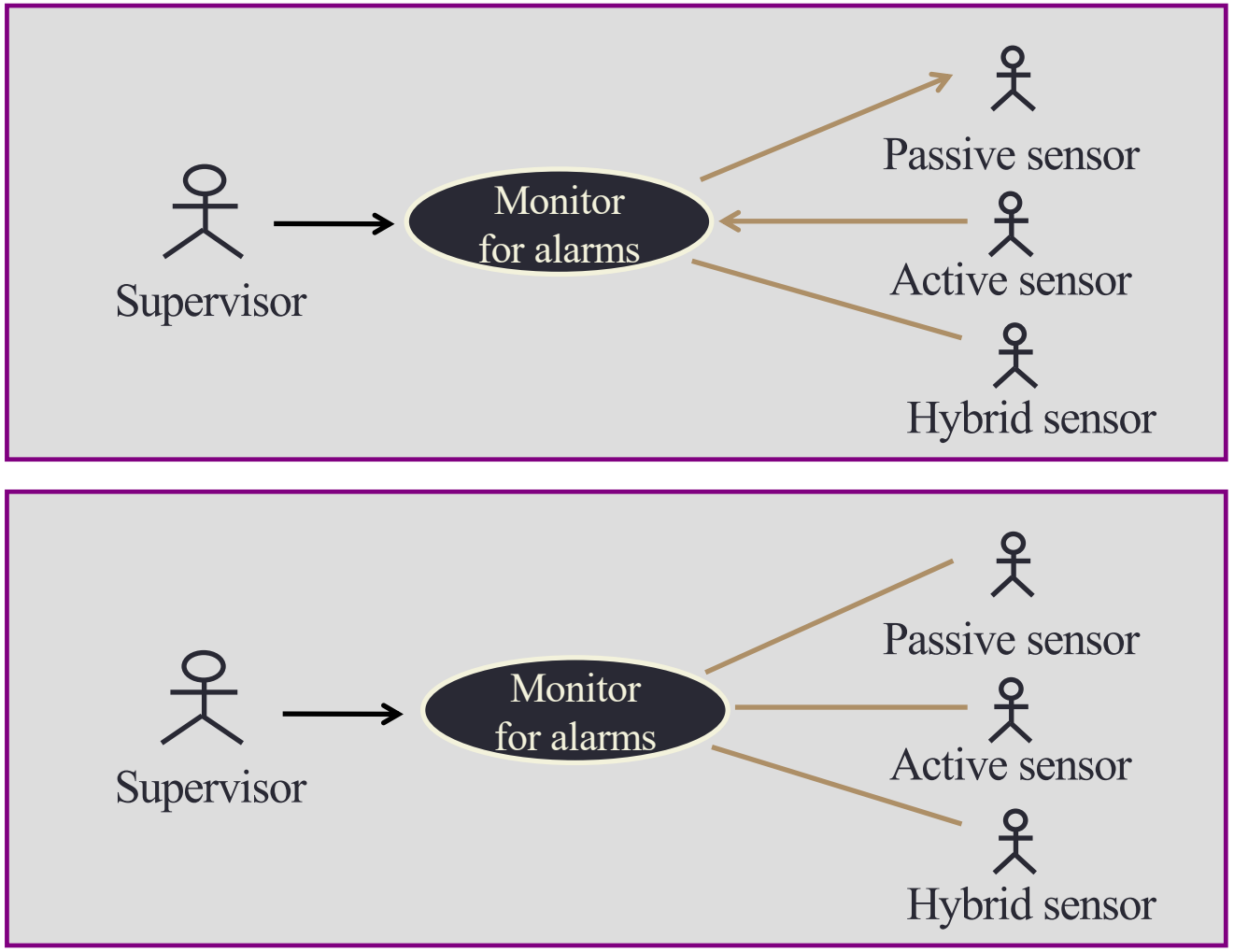


Communicates-Association



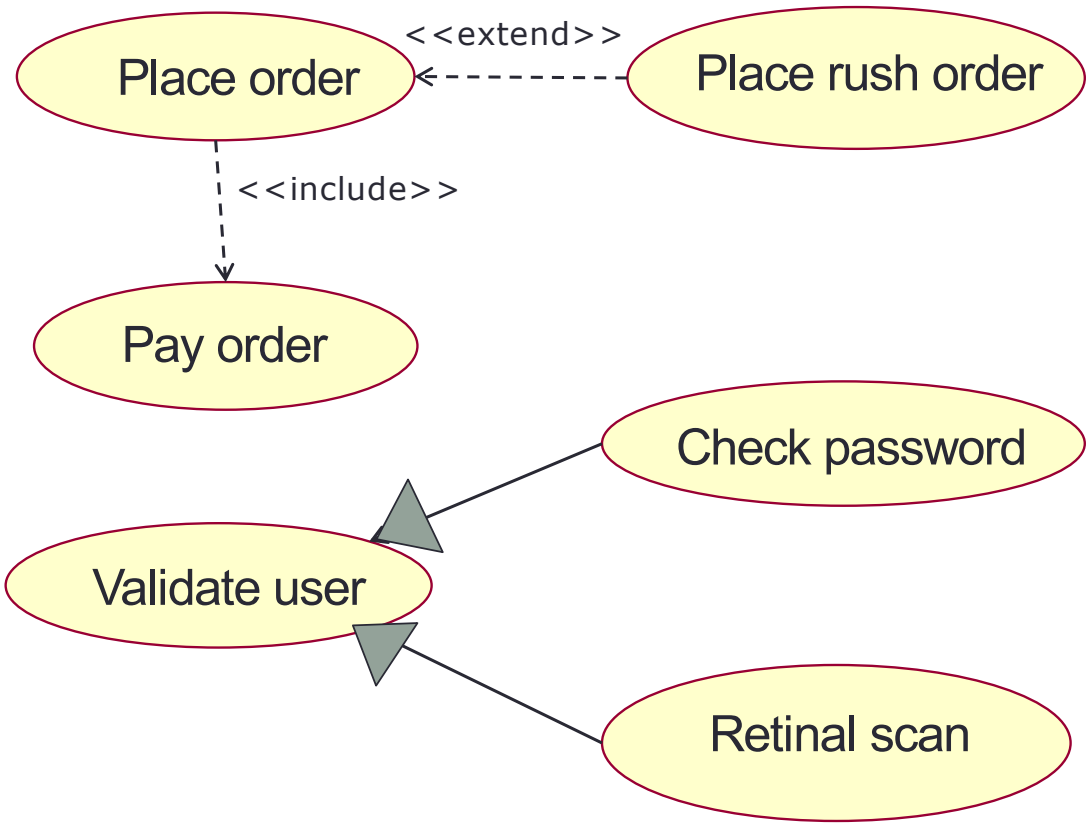
- A channel of communication between an actor and a use case.
- A line is used to represent a communicates-association.
  - An arrowhead indicates who initiates each interaction.
  - No arrowhead indicates either end **can** initiate each interaction.

Arrowhead Conventions



2.4.3. Between use cases

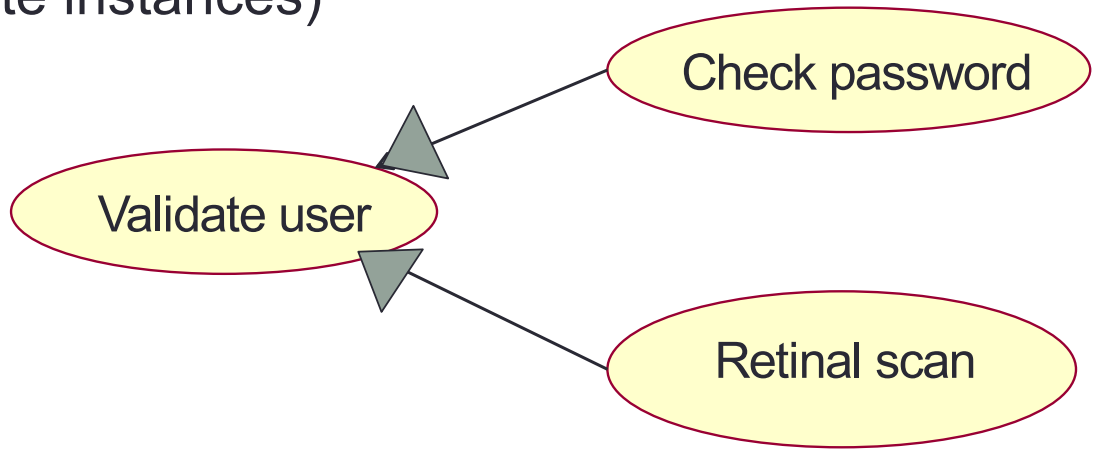
- Generalization
  - parent use case
- <<include>>
  - always use
- <<extend>>
  - sometimes use



28

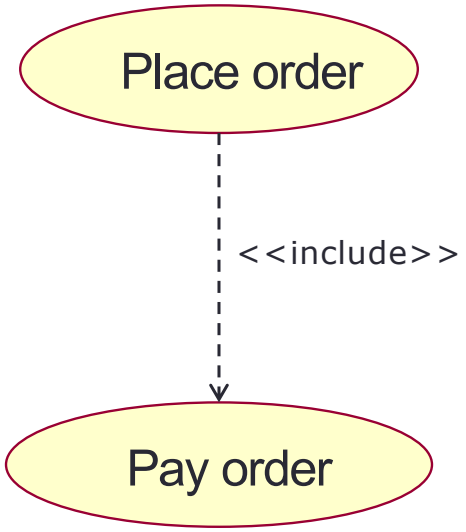
Between use cases - Generalization

- The child use case inherits the behavior and meaning of the parent use case;
  - the child may add to or override the behavior of its parent;
  - the child may be substituted any place the parent appears (both the parent and the child may have concrete instances)



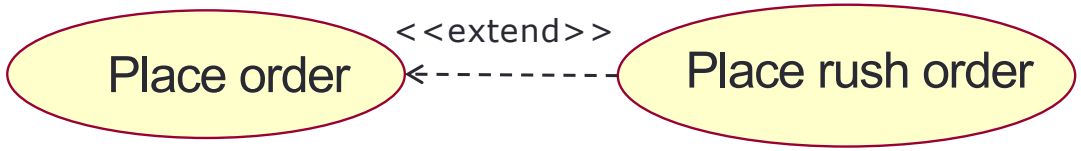
Between use cases - Include

- The base use case explicitly incorporates the behavior of another use case at a location specified in the base.
- The included use case never stands alone, but is only instantiated as part of some larger base that includes it



Between use cases - Extend

- The base use case implicitly incorporates the behavior of another use case at a location specified indirectly by the extending use case.
- The base use case may stand alone, but under certain conditions its behavior may be extended by the behavior of another use case.



2.5. Use case diagram

- The Use case diagram shows a set of use cases and actors and their relationships.
- The Use case diagram serves as a contract between the customer and the developers.
- Because it is a very powerful planning instrument, the Use case diagram is generally used in all phases of the development cycle

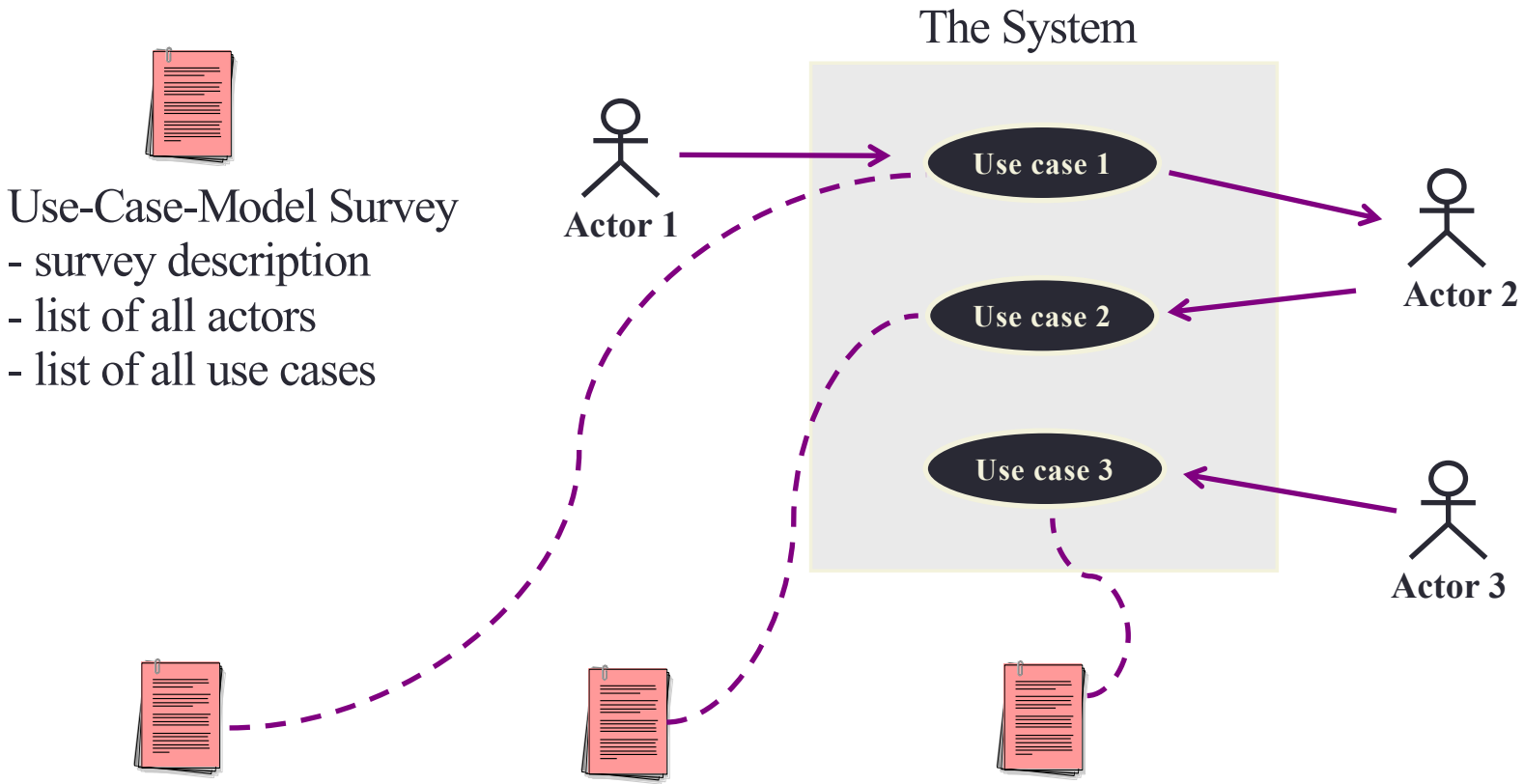
Notes

- Should not use two many relationships between use case in the Use case diagram
  - Tangle and make the diagram difficult to observe
  - Only use the relationship if necessary
  - In the Use case diagram, the sequence do use cases are not specified

Content

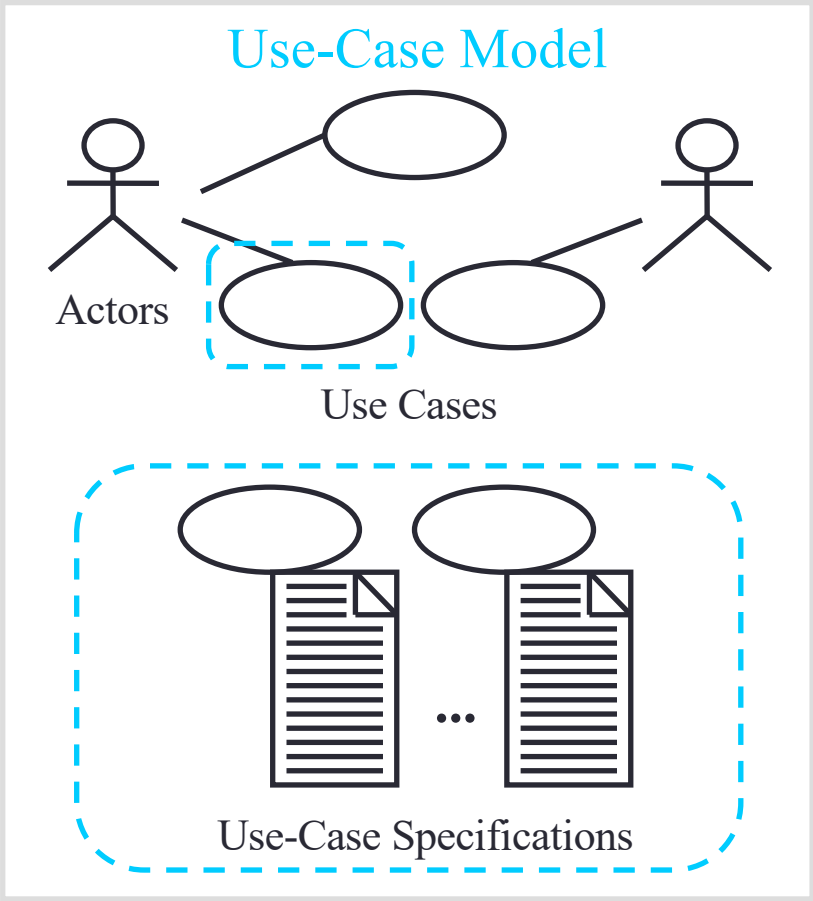
1. Requirements
2. Use case diagram
3. Use case specification/scenario
4. Glossary
5. Supplementary Specification

Use-Case Specification



Use-Case Specification

- Code
- Name
- Brief description
- Flow of Events
- Relationships
- Activity diagrams
- Use-Case diagrams
- Special requirements
- Pre-conditions
- Post-conditions
- Other diagrams



Some guidelines to make UC specification

- UC Scenario description for each UC:
  - External Interface
  - Permanent data
- Excess and deficiency check between between the problem description and Requirements
- Consistency in the Requirements
- Feasibility of later phase

Brief description of UC

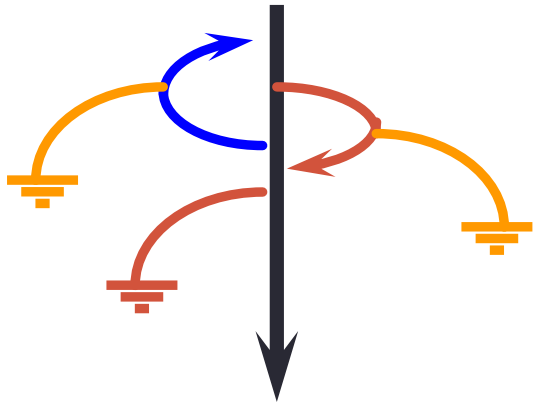
- *Describe briefly the purpose of UC*

- Example: Use case “Log in” in the ATM system:

  “This use case describes the interaction between bank customers and the ATM machine when the customer wishes to log in to the system to perform transactions”

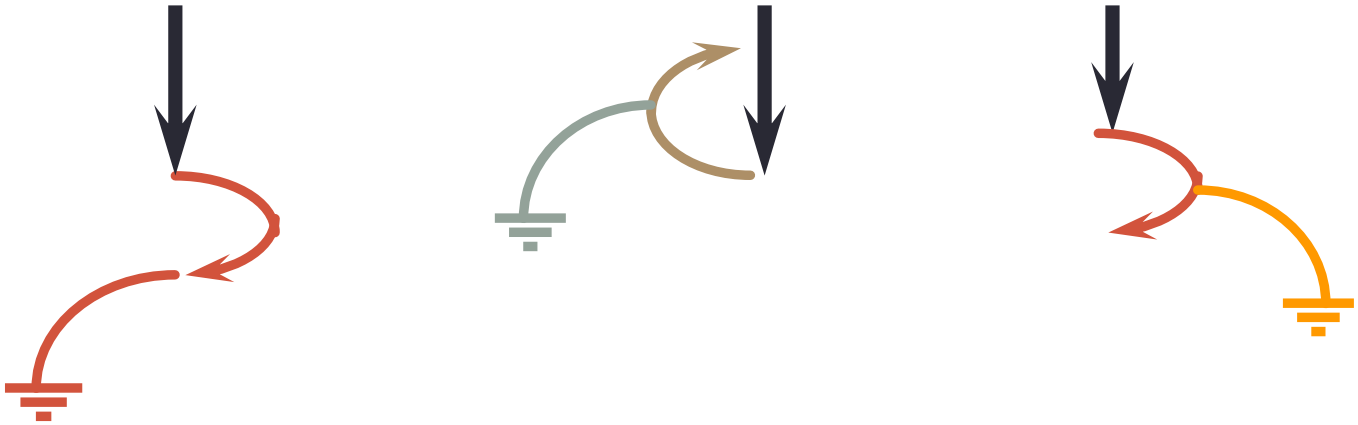
Use-Case Flow of Events

- Has one normal, basic flow
- Several alternative flows
  - Regular variants
  - Odd cases
  - Exceptional flows for handling error situations



What Is a Scenario?

- A scenario is an instance of a use case.



UC Login in the ATM system

- Main flows of events: The use case starts when system prompts the Customer for a PIN Number. The Customer can now enter a PIN number. The Customer commits the entry. The system then checks this PIN number to see if it is valid. If valid, the system acknowledges the entry, thus ending the use case
- Regular variants: The Customer cancel a transaction at any time, thus restart the UC. No changes are made to the Customer's account.
- Odd case: The Customer clear a PIN number anytime before committing it and re-enter a new PIN number
- Exceptional flow of events: If the Customer enter an invalid PIN number, the UC restarts. If this happens 3 times in a row, the system cancel the entire transaction, and keep the ATM card.

Flow of events

Success / Main flow of event

#	Doer	Action
1	Customer	requests to log in
2	software	prompts a Log in screen
3	Customer	enters a PIN number
4	Customer	submit to login
5	software	checks if the PIN number is valid
6	software	displays the main menu if the PIN number is valid

Alternative flow of event

#	Doer	Action
	Customer	cancels a transation **at any time**
4a	Customer	clears PIN number **before submitting**
6a	software	notifies Invalid PIN number, goes to Step 2 **if the PIN number is not valid less than 3 times**
6b	software	notifies invalid PIN number 3 times, keep the ATM card **if the PIN number is not valid 3 times**

Detail of Alternative Flows

Alternative Flows

2.8 Unidentified Student.

In the Log On step of the Basic Flow, ***if the system determines that the student identification information is not valid***, an error message is displayed, and the use case ends.

2.9 Quit and Save.

At any time, the system will allow the Student to quit. The student chooses to quit and save a partial schedule before quitting. The system saves the schedule, and the use case ends.

2.10 Waiting List

In the Select Courses step of the Basic Flow, ***if a course the Student wants to take is full***, the systems allows the student to be added to a waiting list for the course. The use case resumes at the Select Courses step in the Basic Flow.

Describe what happens

Location

Condition

Actions

Resume location

How to do use case specification for:

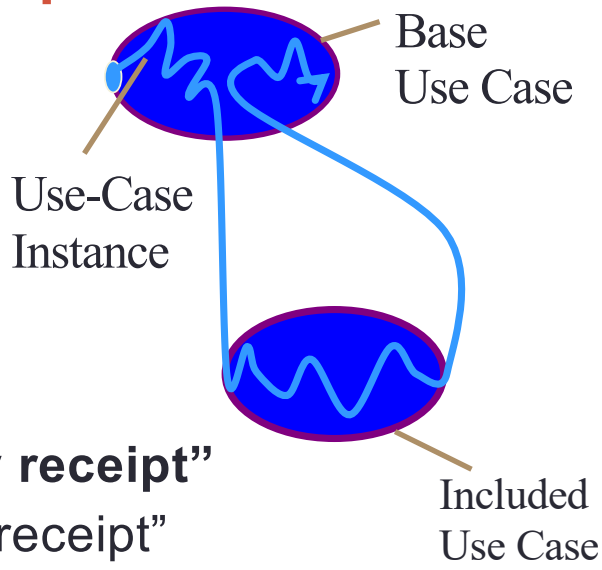
- Included use cases
  - **Explicitly call** the included use case in **a step (i.e. the inclusion point**) in the basic flow of the base use case
- Extended use cases
  - **Insert** extension use case's behavior into base use case at the **extension point** if **the extending condition** is true
- Generalized use cases
  - Use placeholders in parent use cases

E.g. Include use case in specification

- **Use case "Place order"**

  Basic flow

  1. …
  2. …
  3. Passenger confirm to buy tickets
  4. Software **calls the use case "Pay receipt"**
  5. Software calls the use case "Print receipt"
  6. …

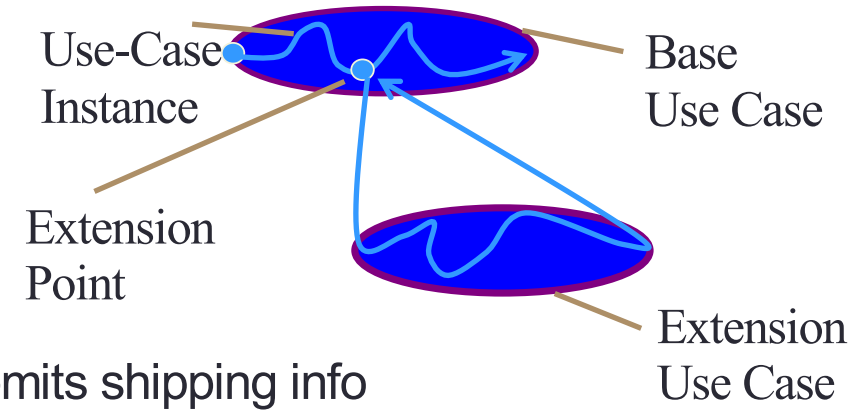


- **Use case "Pay receipt"**

  Basic flow

  1. Software asks passenger to insert a credit card
  2. Passenger inserts a credit card to the credit card slot
  3. …

E.g. Extend use case in specification



- **Use case "Place order"**

  Basic flow

  1. …
  2. Customer enters and submits shipping info
  3. Software displays receipt
  4. …

  Alternative flow

  A1. In the step 2, if customer chooses to place a rush order, insert use case "Place rush order", then resume at the step 3.

  …

- **Use case "Place rush order"**

  1. Software displays rush order form
  2. Customer
  3. …

How do you handle repetitive behavior?

Simple, repetitive behavior can be captured within the basic flow.

Register for Courses

Flow of Events

1. Basic Flow

1.1. LOG ON.
This use case starts when a student accesses the Course Registration System. The student enters a student ID and password and the system validates the student.

.2. CREATE SCHEDULE.
The system displays the functions available to the student. These functions are: Create A Schedule, Modify a Schedule and Delete a Schedule. The student selects 'Create a Schedule'.

.3. SELECT COURSES
The system retrieves a list of available course offerings from the Course Catalog System and displays the list to the student. The Student selects up to 4 primary course offerings and 2 alternative course offerings from the list of available offerings. The student can add and delete courses as desired until choosing to submit the schedule.

1.4. SUBMIT SCHEDULE.
The student indicates that the schedule is complete. The system validates the courses selected and displays the schedule to the student. The system calculates the confirmation number using a hashing algorithm based on the student ID and adds it to the student record. The system displays the confirmation number for the schedule. The system saves the student's schedule information in the Student Information Database. The use case ends.

How do you handle repetitive behavior? (cont.)

Repetitive flow of events can be captured using a subflow.

Register for Courses

Basic Flow

1. Log On.

…

2. Create Schedule.

1.2. The system displays the functions available to the student. These functions are Create A Schedule, Modify a Schedule and Delete a Schedule. The student selects 'Create a Schedule'.

3. Perform Subflow ***Select Courses***

4. Submit Schedule

…

Alternative Flows

1. Modify Schedule.

1.1 In the Create Schedule step of the Basic Flow, if the student already has a schedule that has been saved; the system retrieves and displays the Student's current schedule (e.g., the schedule for the current semester) and allows him/her to use it as a starting point.

1.2 Perform Subflow ***Select Courses***.

1.3 The use case resumes at the Submit Schedule step of the Basic Flow.

…

Subflows

1. Select Courses.

1.1 The system retrieves a list of available course offerings from the Course Catalog System and displays the list to the student.

1.2 The Student selects up to 4 primary course offerings and 2 alternative course offerings from the list of available offerings.

1.3 The student can add and delete courses as desired until choosing to submit the schedule.

Example subflow

1. Use Case Name: Register for Courses

1.1 Brief Description

...

2. Flow of Events

2.1 Basic Flow

1. Log On

...

2. Select 'Create a Schedule'

...

3. Obtain Course Information

Perform subflow S1: Obtain Course Information

4. Select Courses

...

5. Submit Schedule

...

6. Accept Completed Schedule

...

2.2 Subflows

2.2.1 S1: Obtain Course Information

The student requests a list of course offerings. The student can search the list by department, professor or topic to obtain desired course information. The system retrieves a list of available course offerings from the Course Catalog System and displays the list to the student.

Visualize behavior

- Visual modeling tools
  - Activity diagrams or flow charts
  - Business process models
- Should you illustrate behavior?
  - Pro
    - Great tool to identify alternative flows, especially for visually oriented people
    - Succinctly conveys information about use case flows
  - Con
    - Costly to keep diagrams and use-case specifications synchronized

What Is an Activity Diagram?

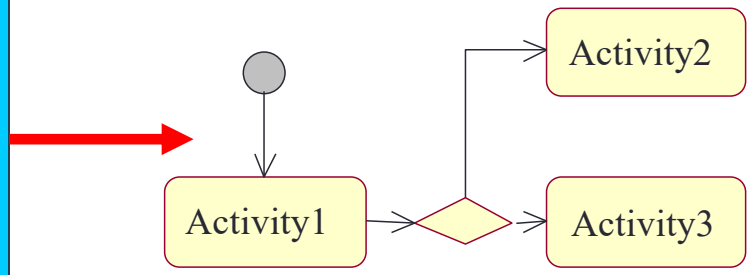
- An activity diagram in the Use-Case Model can be used to capture the activities in a use case.
- It is essentially a flow chart, showing flow of control from one activity or action to another.

Flow of Events

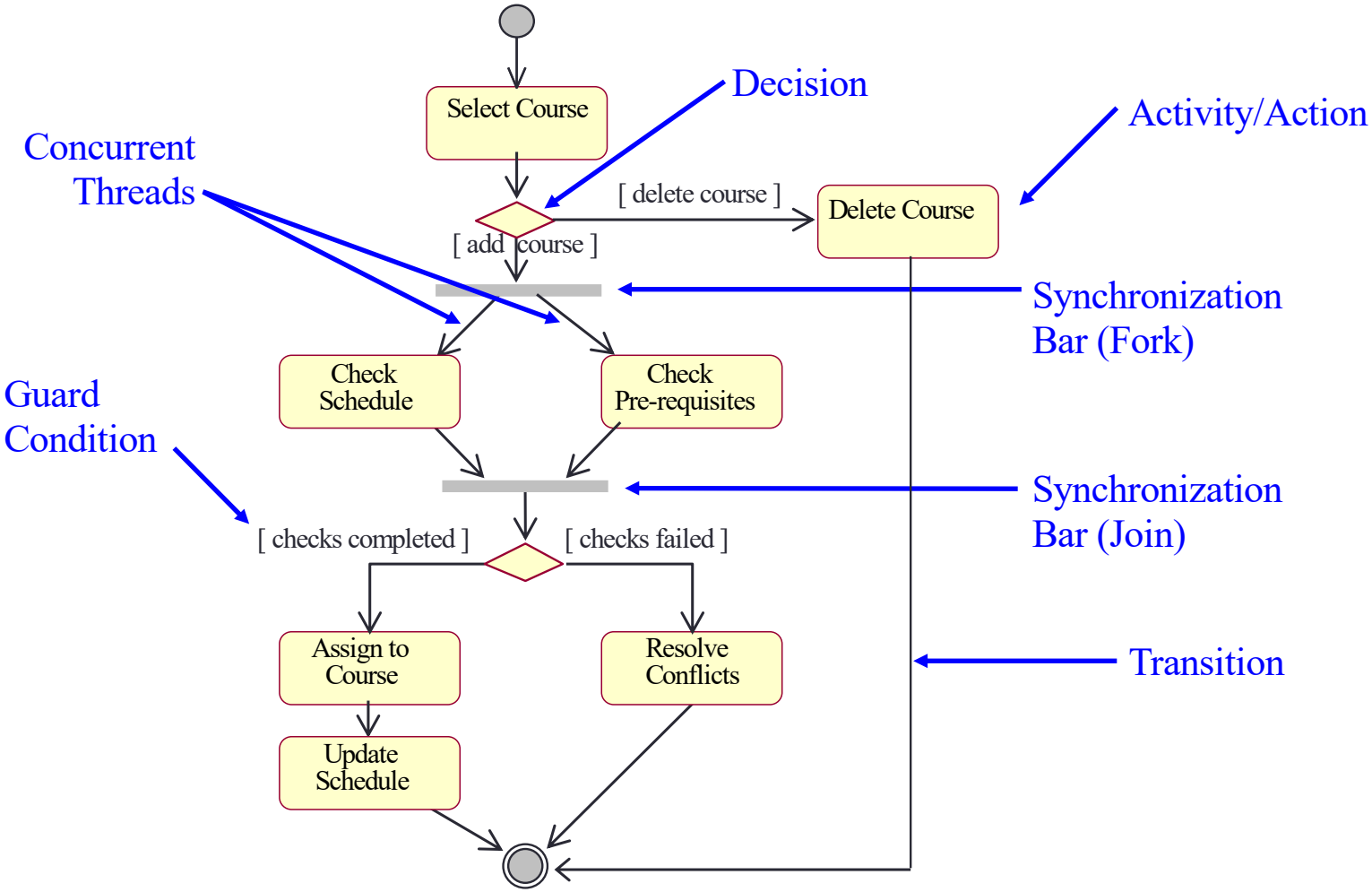
This use case starts when the Registrar requests that the system close registration.

1. The system checks to see if registration is in progress. If it is, then a message is displayed to the Registrar and the use case terminates. The Close Registration processing cannot be performed if registration is in progress.

2. For each course offering, the system checks if a professor has signed up to teach the course offering and at least three students have registered. If so, the system commits the course offering for each schedule that contains it.

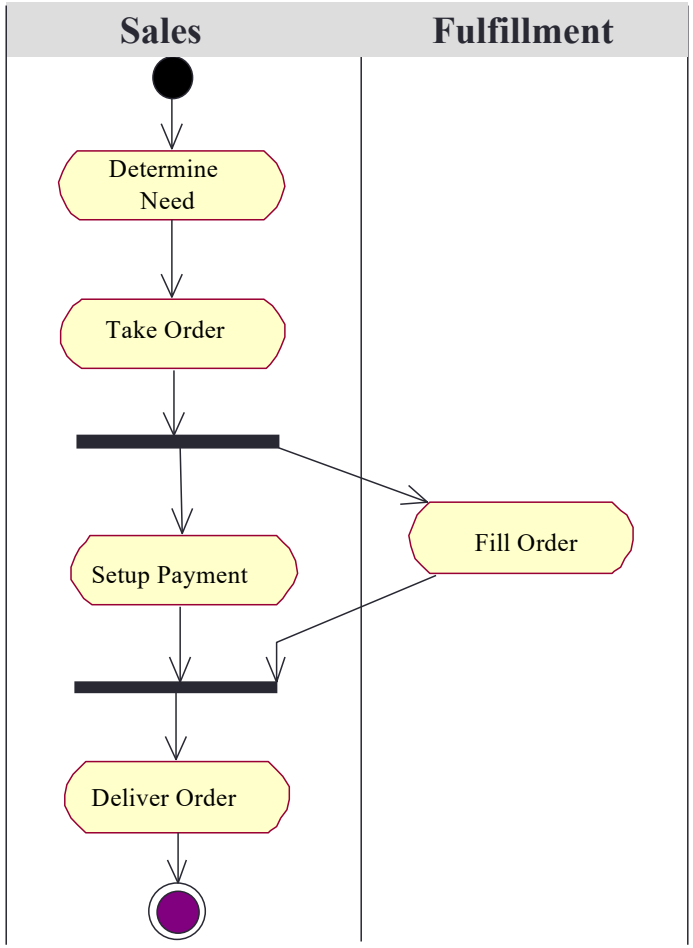


Example: Activity Diagram



Partitions

- Each partition should represent a responsibility for part of the overall workflow, carried by a part of the organization.
- A partition may eventually be implemented by an organization unit or a set of classes in the business object model.



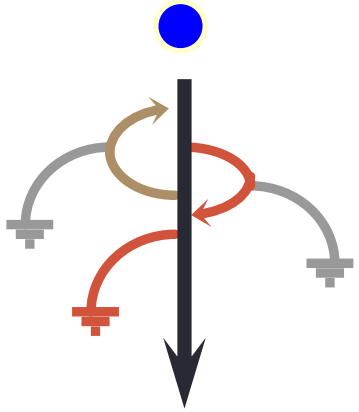
Preconditions

- Describe the state that the system must be in before the use case can start
  - Simple statements that define the state of the system, expressed as conditions that must be true
  - Should never refer to other use cases that need to be performed prior to this use case
  - Should be stated clearly and should be easily verifiable
- Optional: Use only if needed for clarification
- Example

  Register for Courses use case

  Precondition:

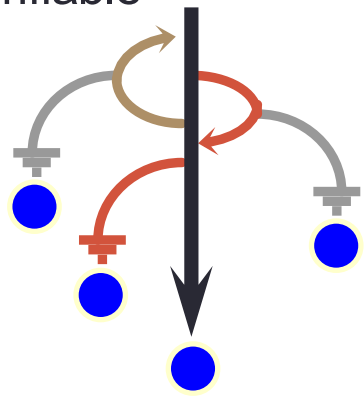
  - The list of course offerings for the semester has been created and is available to the Course Registration System
  - The registration is open for student
  - Student has logged into the Course Registration System



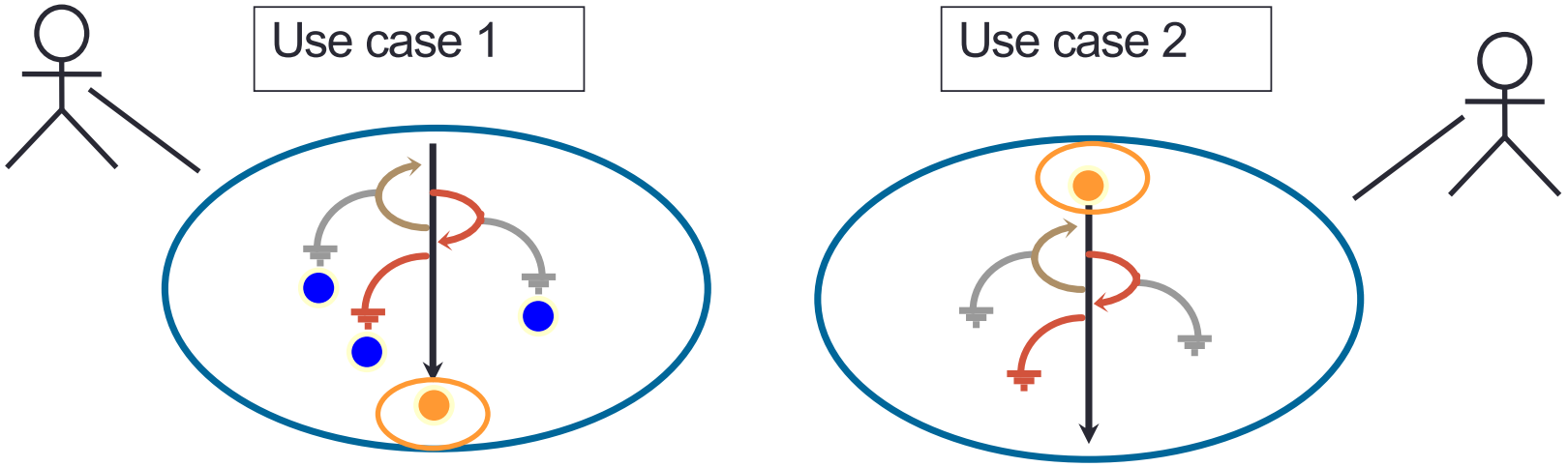
Postconditions

- Describe the state of the system at the end of the use case
  - Use when the system state is a precondition to another use case, or when the possible use case outcomes are not obvious to use case readers
  - Should never refer to other, subsequent use cases
  - Should be stated clearly and should be easily verifiable
- Optional: Use only if needed for clarification
- Example:

  Register for Courses use case
  Postcondition: At the end of this use case either the student has been enrolled in courses, or registering was unsuccessful and no changes have been made to the student schedules or course enrollments



Sequence use cases with pre- and postconditions



Use cases do **not** interact with each other.
However, a postcondition for one use case
can be the same as the precondition for another.

Other use case properties

- Special requirements
  - Related to this use case, not covered in flow of events
  - Usually nonfunctional requirements, data, and business rules
- Extension points
  - Name a set of places in the flow of events where extending behavior can be inserted
- Additional information
  - Any additional information required to clarify the use case

Content

1. Requirements
2. Use case diagram
3. Use case specification/scenario
4. Glossary
5. Supplementary Specification

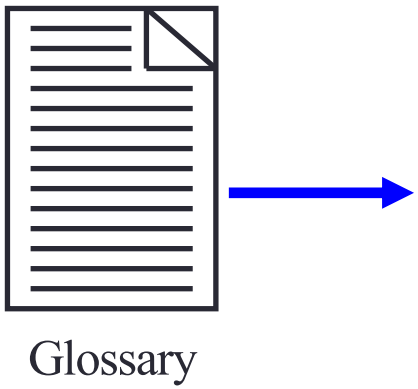
4. Glossary

- The **Glossary** defines important terms used in the project for all models.
- There is only one Glossary for the system.
- This document is important to many developers, especially when they need to understand and use the terms that are specific to the project.
- The **Glossary** is used to facilitate communications between domain experts and developers

4. Glossary (2)

- **Introduction**: Provides a brief description of the Glossary and its purpose.
- **Terms**: Define the term in as much detail as necessary to completely and unambiguously characterize it.

4. Glossary (3)



Course Registration System Glossary

1. Introduction

This document is used to define terminology specific to the problem domain, explaining terms, which may be unfamiliar to the reader of the use-case descriptions or other project documents. Often, this document can be used as an informal *data dictionary*, capturing data definitions so that use-case descriptions and other project documents can focus on what the system must do with the information.

2. Definitions

The glossary contains the working definitions for the key concepts in the Course Registration System.

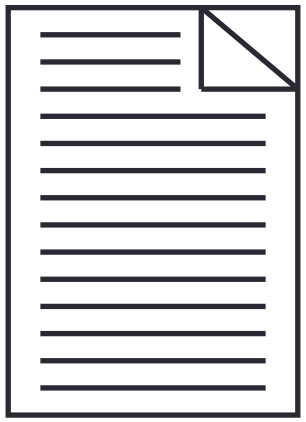
2.1 Course: A class offered by the university.

2.2 Course Offering: A specific delivery of the course for a specific semester – you could run the same course in parallel sessions in the semester. Includes the days of the week and times it is offered.

2.3 Course Catalog: The unabridged catalog of all courses offered by the university.

Case Study: Glossary

- Make the Glossary of the Course Registration System



Glossary

Content

1. Requirements
2. Use case diagram
3. Use case specification/scenario
4. Glossary
5. Supplementary Specification

5. Supplementary Specification

- Includes the nonfunctional requirements and functional requirements not captured by the use cases
- Contains those requirements that do not map to a specific use case: Functionality, Usability, Reliability, Performance, Supportability

Supplementary Specification

5. Supplementary Specification (2)

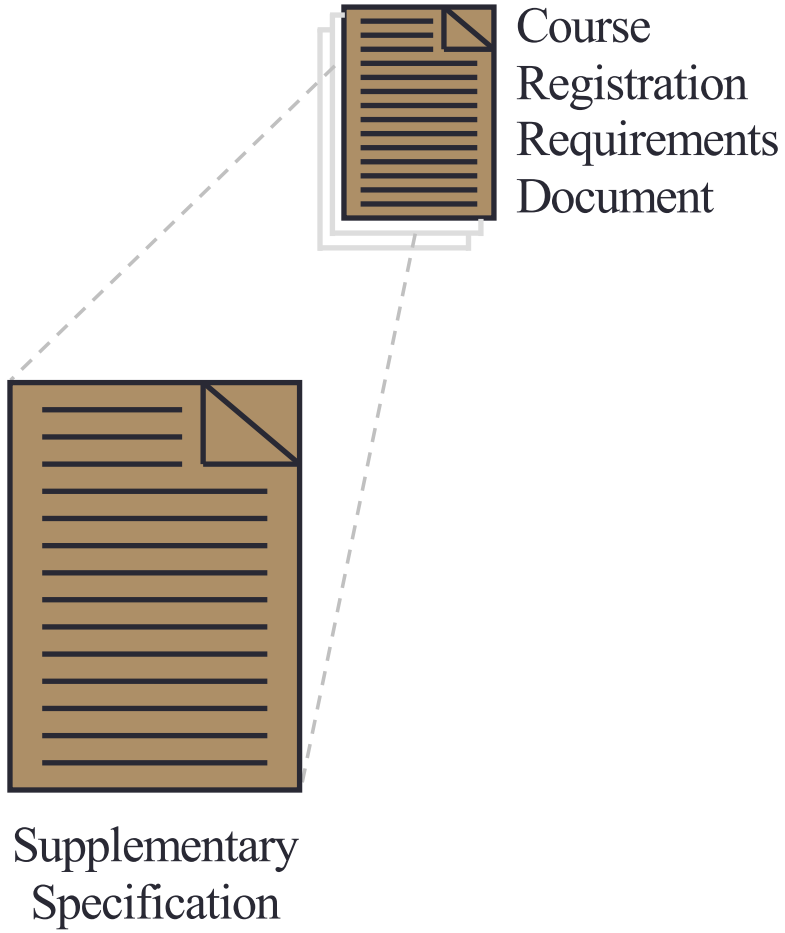
- **Functionality**: List of the functional requirements that are general to many use cases.
- **Usability**: Requirements that relate to, or affect, the usability of the system. Examples include ease-of-use requirements or training requirements that specify how readily the system can be used by its actors.

5. Supplementary Specification (3)

- **Reliability**: Any requirements concerning the reliability of the system. Quantitative measures such as mean time between failure or defects per thousand lines of code should be stated.
- **Performance**: The performance characteristics of the system. Include specific response times. Reference related use cases by name.
- **Supportability**: Any requirements that will enhance the supportability or maintainability of the system being built.

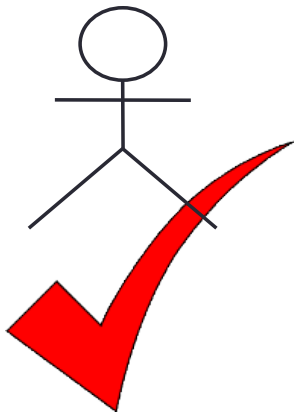
Case study: Supplementary Specification

- Make the Supplementary Specification for the Course Registration System



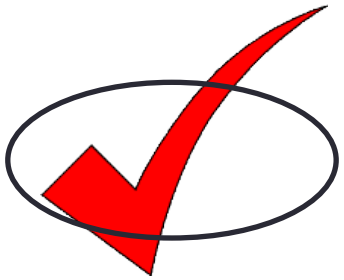
Checkpoints: Actors

- Have all the actors been identified?
- Is each actor involved with at least one use case?
- Is each actor really a role? Should any be merged or split?
- Do two actors play the same role in relation to a use case?
- Do the actors have intuitive and descriptive names? Can both users and customers understand the names?



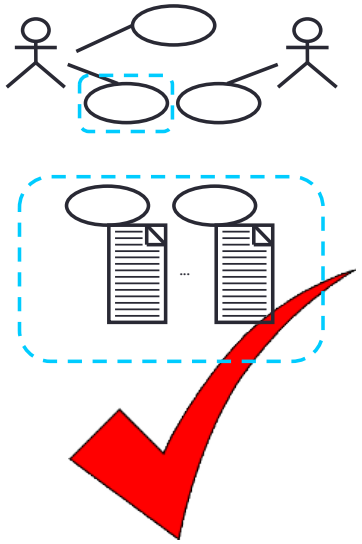
Checkpoints: Use-Cases

- Is each use case involved with at least one actor?
- Is each use case independent of the others?
- Do any use cases have very similar behaviors or flows of events?
- Do the use cases have unique, intuitive, and explanatory names so that they cannot be mixed up at a later stage?
- Do customers and users alike understand the names and descriptions of the use cases?



Checkpoints: Use-Case Model

- Is the Use-Case Model understandable?
- By studying the Use-Case Model, can you form a clear idea of the system's functions and how they are related?
- Have all functional requirements been met?
- Does the Use-Case Model contain any superfluous behavior?
- Is the division of the model into use-case packages appropriate?



Checkpoints: Use-Case Specifications

- Is it clear who wants to perform a use case?
- Is the purpose of the use case also clear?
- Does the brief description give a true picture of the use case?
- Is it clear how and when the use case's flow of events starts and ends?
- Are the actor interactions and exchanged information clear?
- Are any use cases overly complex?

Checkpoints: Glossary

- Does each term have a clear and concise definition?
- Is each glossary term included somewhere in the use-case descriptions?
- Are terms used consistently in the brief descriptions of actors and use cases?

Review

- What are the main artifacts of Requirements?
- What are the Requirements artifacts used for?
- What is a Use-Case Model?
- What is an actor?
- What is a use case? List examples of use case properties.
- What is the difference between a use case and a scenario?
- What is a Supplementary Specification and what does it include?
- What is a Glossary and what does it include?

Question?