

3. REQUIREMENT MODELING WITH UC

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



Some slides extracted from IBM coursewares

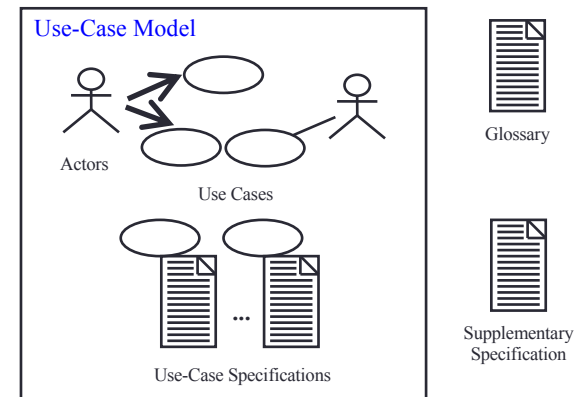
Content

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

1.1. Purpose of Requirement

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

1.2. Relevant Requirements Artifacts



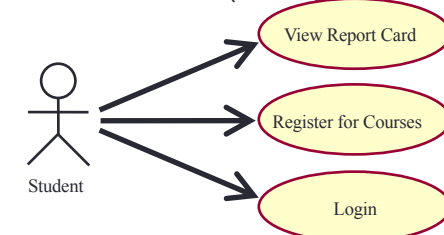
Content

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

5

2.1. Overview of Use-Case Diagram

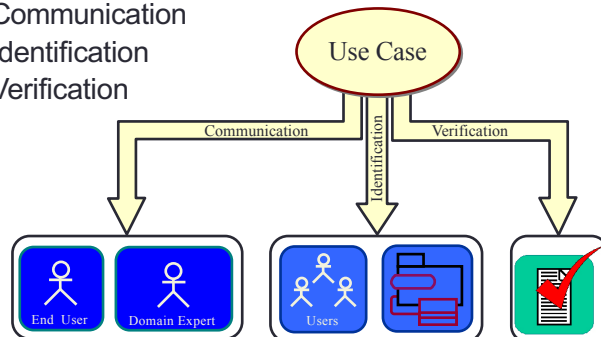
- A diagram modeling the dynamic aspects of softwares that describes a software's functional requirements in terms of use cases.
- A model of the software's intended functions (use cases) and its environment (actors).



6

Benefits of a Use-Case Model

- Communication
- Identification
- Verification



7

Major Concepts in Use-Case Modeling

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



Actor

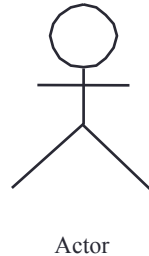


Use Case

8

2.2. Actors

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



Actors and Roles



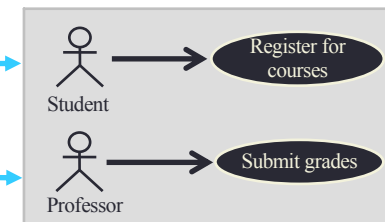
Charlie: Is employed as a math professor and is an economics undergraduate.



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

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.

2.3. Use Cases

- Define a set of use-case instances, where each instance is a sequence of actions a software 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 software
 - A use case describes the actions the software takes to deliver something of value to the actor



Use Case

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

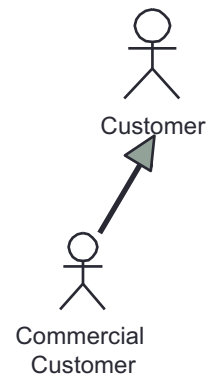
- What are the goals of each actor?
- Why does the actor want to use the software?
- Will the actor create, store, change, remove, or read data in the software? If so, why?
- Will the actor need to inform the software about external events or changes?
- Will the actor need to be informed about certain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

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

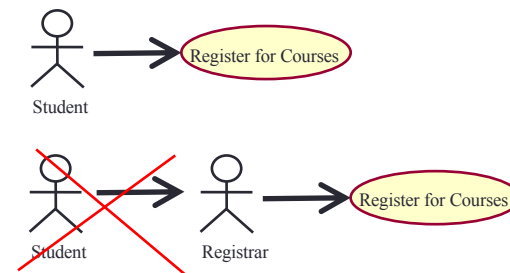
2.4.1. Between actors

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



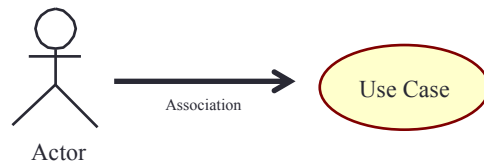
2.4.1. Between actors (2)

- Association
 - Consider the difference between two diagrams

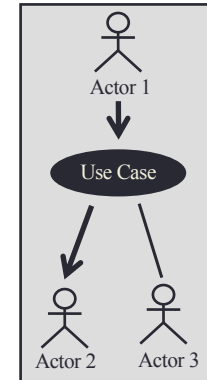


2.4.2. Between actor and use case

- 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 software 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.

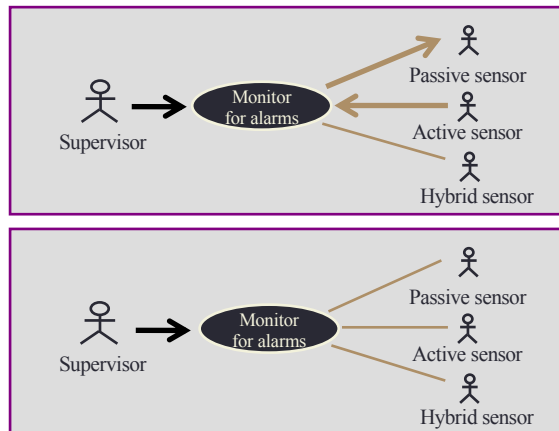


Communicates-Association



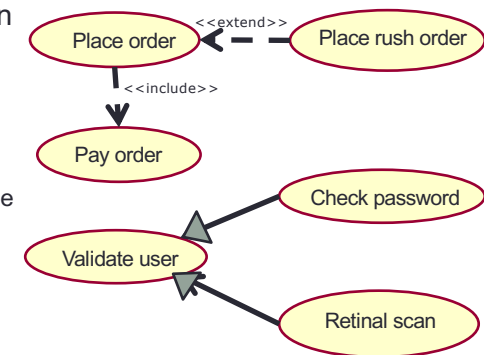
- A channel of communication between an actor and a use case.
- A line is used to represent a communicates-association.
 - An arrowhead indicates who initiates each interaction.
 - No arrowhead indicates either end **can** initiate each interaction.

Arrowhead Conventions



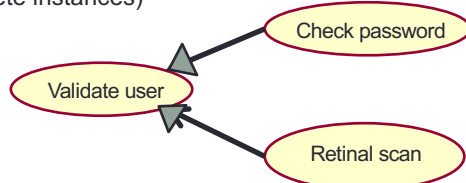
2.4.3. Between use cases

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



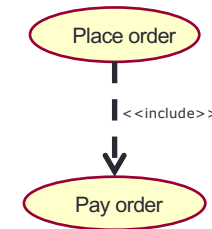
Between use cases - Generalization

- 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

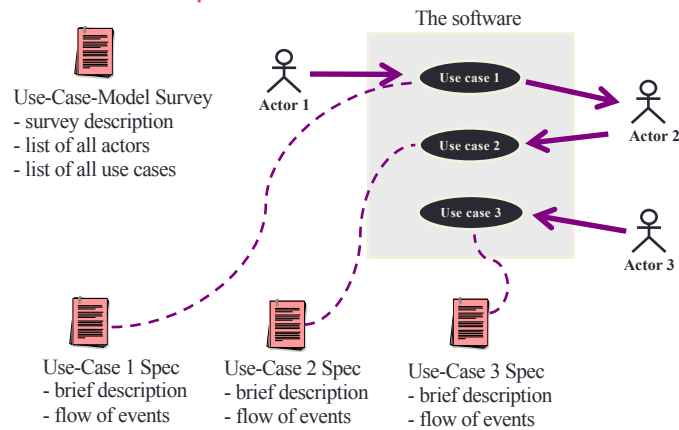
Notes

- Should not use too 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 of use cases are not specified

Content

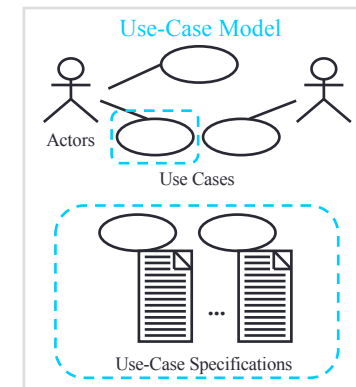
1. Requirements
2. Use case diagram
- ➔ 3. Use case specification/scenario
4. Glossary
5. Supplementary Specification

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

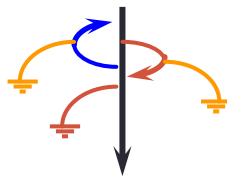
Brief description of UC

- *Describe briefly the purpose of UC*
- Example: Use case “Log in” in the ATM software:

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

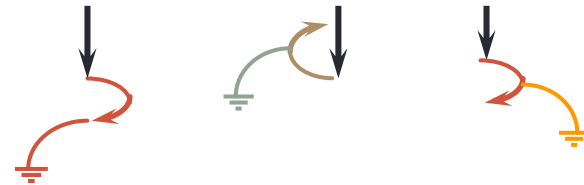
Use-Case Flow of Events

- ♦ 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.



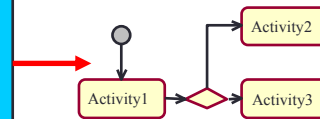
What Is an Activity Diagram?

- ◆ 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.

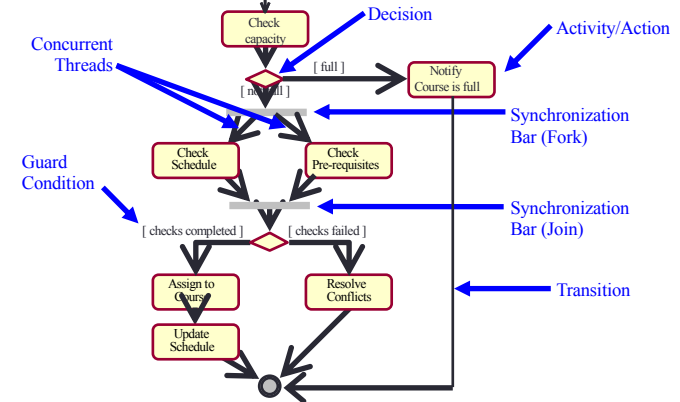
Flow of Events

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

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

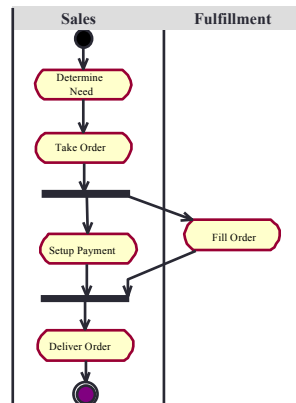


Example: Activity Diagram



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.



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 software.
- 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)



Glossary



Course Registration software 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 software must do with the information.

2. Definitions
The glossary contains the working definitions for the key concepts in the Course Registration software.

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 software



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)

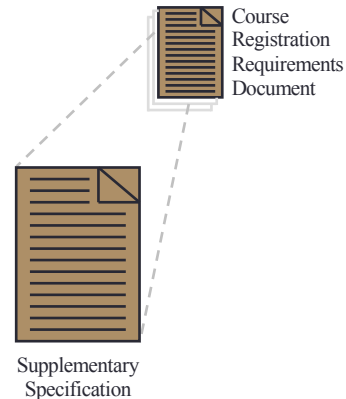
- **Functionality:** List of the functional requirements that are general to many use cases.
- **Usability:** Requirements that relate to, or affect, the usability of the software. Examples include ease-of-use requirements or training requirements that specify how readily the software can be used by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

5. Supplementary Specification (3)

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

Case study: Supplementary Specification

- Make the Supplementary Specification for the Course Registration software



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 software'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?

