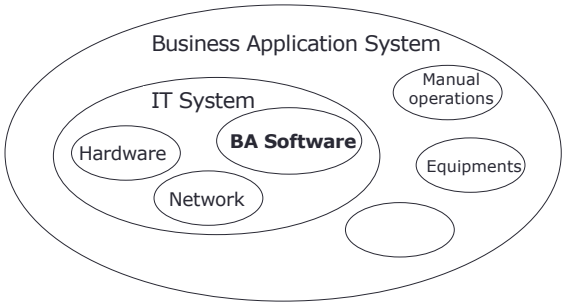
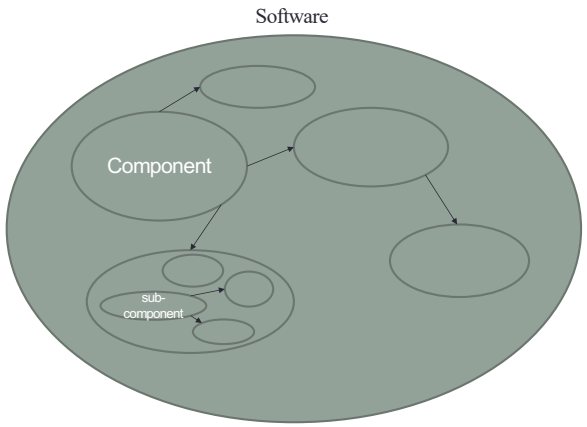
A big picture of Business Application System



1



2

Software Implementation Process

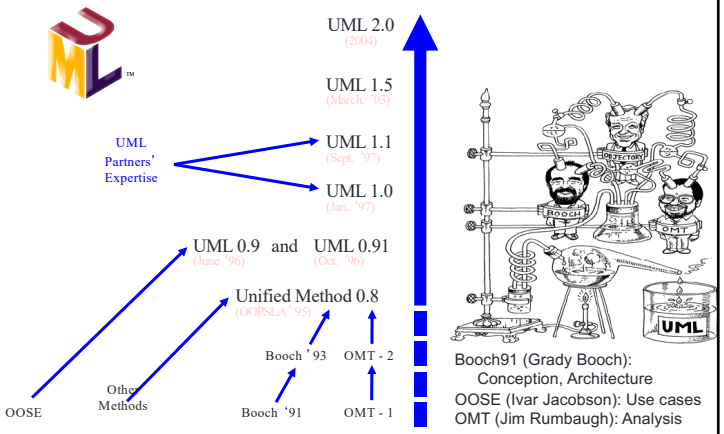
System Requirements Analysis Process and System Architectural Design Process are achieved just before Software Implementation Process .

Software Implementation Process includes the following lower-level processes:

1. **Software Requirements Analysis Process**
2. **Software Architecture Design Process**
3. **Software Detailed Design Process**
4. **Software Construction Process = Code + Unit Test**
5. **Software Integration Process -> git**
6. **Software Qualification Testing Process -> Tester**

3

History of the UML

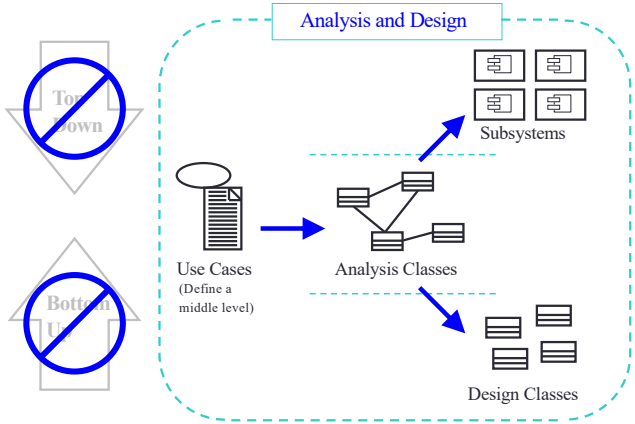


4

- Object-Oriented Analysis and Design
  - Bottom-up? x
  - Top-down? x
- Use case approach --- middle approach

5

Analysis and Design Are Not Top-Down or Bottom-Up



6

ITSS SOFTWARE DEVELOPMENT/SOFTWARE DESIGN AND CONSTRUCTION

2. REQUIREMENT MODELING WITH UC

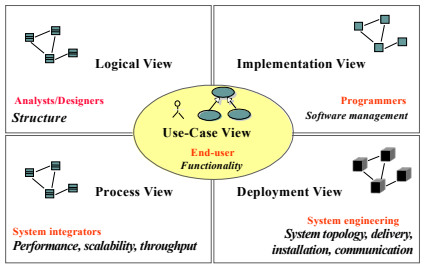
Nguyen Thi Thu Trang
trangntt@soict.hust.edu.vn

Some slides extracted from IBM coursewares

7

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.



8

Content

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

9

9

10

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.

10

11

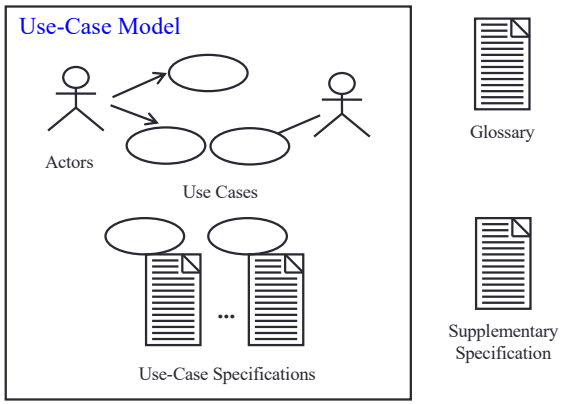
Purpose of Requirement

- 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.

11

12

Relevant Requirements Artifacts
SRS – Software Requirement Specification



12

Content

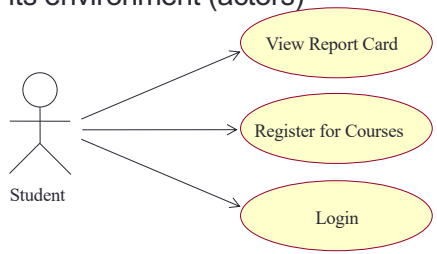
1. Requirements
2. Use case diagram
3. Use case specification/scenario
4. Glossary
5. Supplementary Specification

13

13

2.1. Overview of Use-Case Diagram

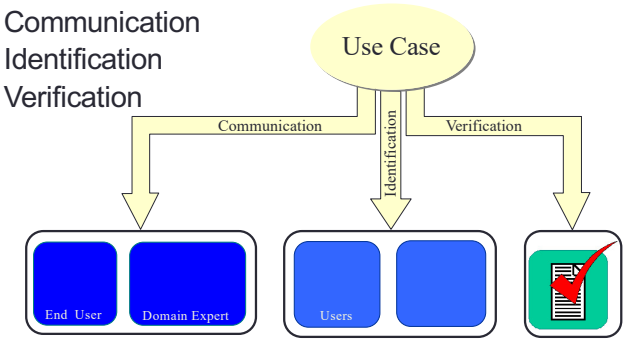
- A diagram modeling the dynamic aspects of systems that describes a software's functional requirements in terms of use cases.
- A model of the software's intended functions (use cases) and its environment (actors)



14

Benefits of a Use-Case Model

- Communication
- Identification
- Verification



15

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.

16

2.2. Actors

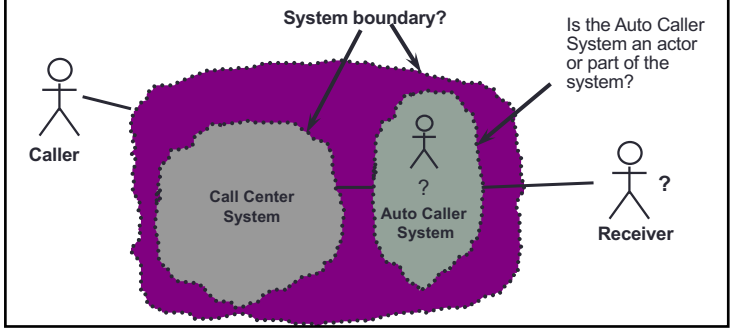
- Actors represent roles a user of the system can play
  - They can represent a human, a machine, or another system
  - They can be a peripheral device or even database
- They can actively interchange information with the system
  - They can be a giver of information
  - They can be a passive recipient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

Actor

17

Actors and the system boundary

- Determine what the system boundary is
- Everything beyond the boundary that interacts with the system is an instance of an actor



18

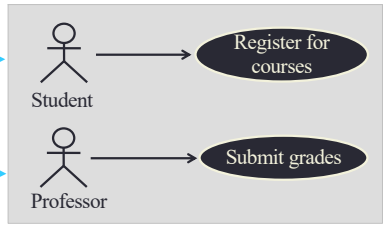
Actors and Roles

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.



19

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**

20

Put some questions to find actors

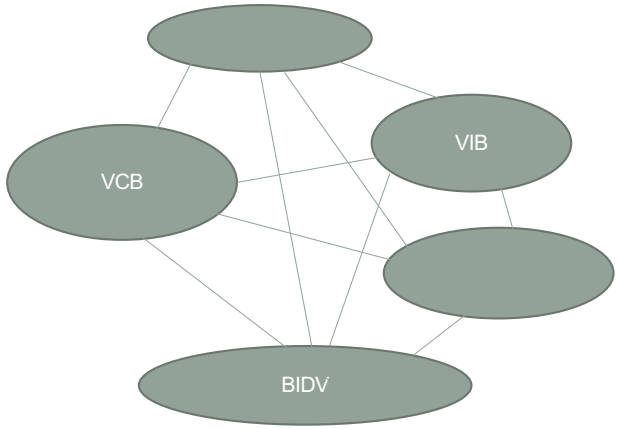
- Who or what uses the system?
- Who or what gets information from this system?
- Who or what provides information to the system?
- Where in the company is the system used?
- Who or what supports and maintains the system?
- What other systems use this system?

21

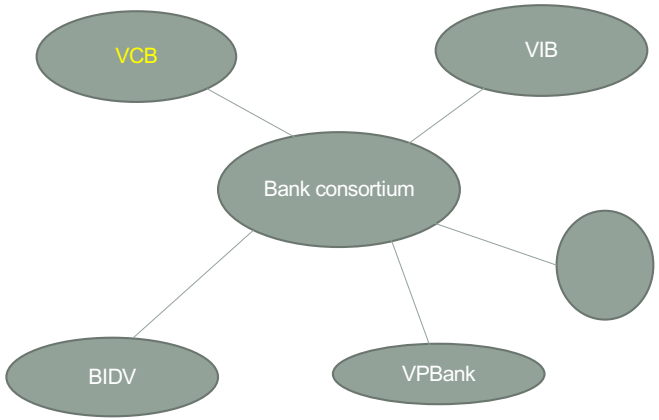
Internet banking software of VCB

- The internet banking software, 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.

22



23



24

Find actors in the Internet banking system

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

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

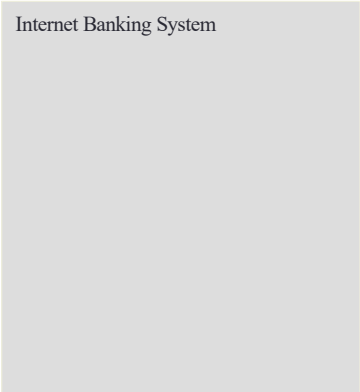
- Be unique, intuitive, and self-explanatory
- Define clearly and unambiguously the observable result of value gained from the use case
- Be from the perspective of the actor that triggers the use case
- Describe the behavior that the use case supports
- Start with a verb and use a simple verb-noun combination

Put some questions to find use cases

- What are the goals of each actor?
- Why does the actor want to use the 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?

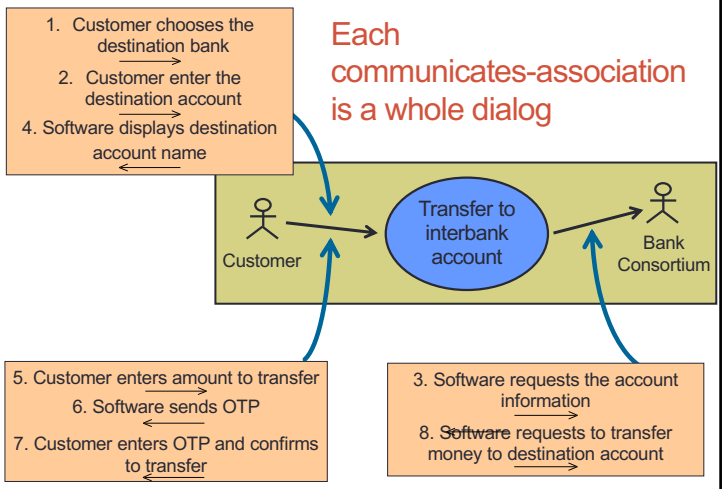
29

Find use cases in the Internet Banking system



30

Each communicates-association is a whole dialog



31

What is NOT a use case

- Functional decomposition
- System design specifications
- User interface specifications

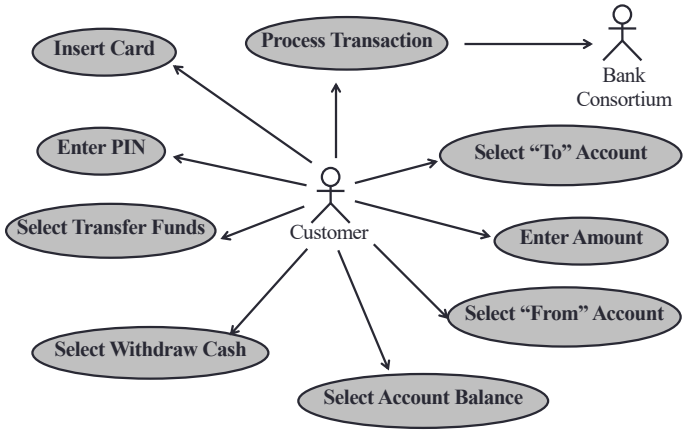
32

Functional Decomposition

- Functional Decomposition is breaking down a problem into small, isolated parts
  - The parts work together to provide the functionality of the system
    - Often do not make sense in isolation
- Use cases:
  - Are NOT functional decomposition
  - Keep the functionality together to describe a complete use of the system
  - Provide context.

33

Functional Decomposition: An Example



34

Avoid Functional Decomposition

Symptoms
- Very small use cases
- Too many use cases
- Uses cases with no result of value
- Names with low-level operations
  - "Operation" + "object"
  - "Function" + "data"
  - Example: "Insert Card"
- Difficulty understanding the overall model

Corrective Actions
- Search for larger context

  "Why are you building this system?"
- Put yourself in user's role

  "What does the user want to achieve?"

  "Whose goal does this use case satisfy?"

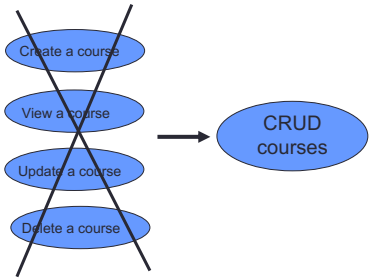
  "What value does this use case add?"

  "What is the story behind this use case?"

35

CRUD Use Cases

- A CRUD use case is a Create, Read, Update, or Delete use case
- Remove CRUD use cases if they are data-management use cases that do not provide results that are of value to actors



- **Do not confuse use cases with functions**
- **Focus on value**

36

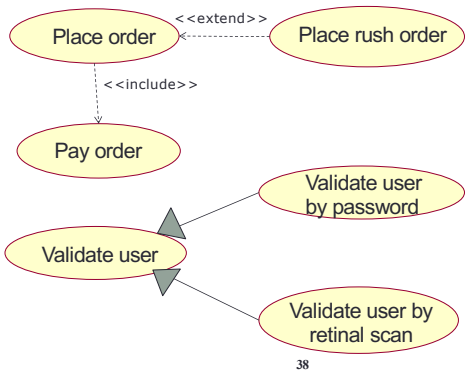
Checkpoints for use case diagram

- ✓The brief description of each use case gives a true picture of the use case
- ✓No use cases have very similar behaviors or flows of events
- ✓ The use-case model contains no superfluous behavior; all use cases can be justified by tracing them back to a functional requirement.
- ✓ All **CRUD** use cases have been removed.
  - **C**reate, **R**etrieve, **U**pdate, **D**elete

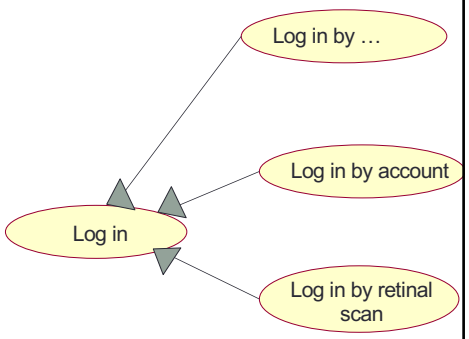
37

2.4.3. Between use cases

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



38



39

Origin "Place order"

- Basic flow
  - Customer views cart
  - Customer requests to place order
  - Software checks availability of the items
  - Customer confirms cart
  - …enter shipping info…
  - …pay order…
  - … create order …
  - … update product availability …
  - … confirm order to customer
- Alternative flow
  - Exception: invalid shipping info, invalid credit card info, not enough balance…
  - Optional case: place rush order: …

40

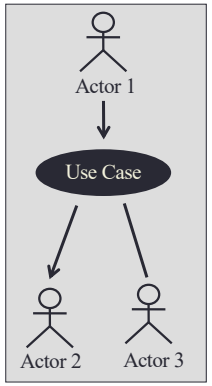
"Place order" (base use case)

- Basic flow
  - Confirm cart
  - …enter shipping info…
  - *call "Pay order" use case*
  - … create order …
  - … update product availability …
  - … confirm order to customer
- Alternative flow
  - Exception: invalid shipping info…
  - Optional case: place rush order: …

Association between actor and use case

- Establish the actors that interact with related use cases
  - 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.

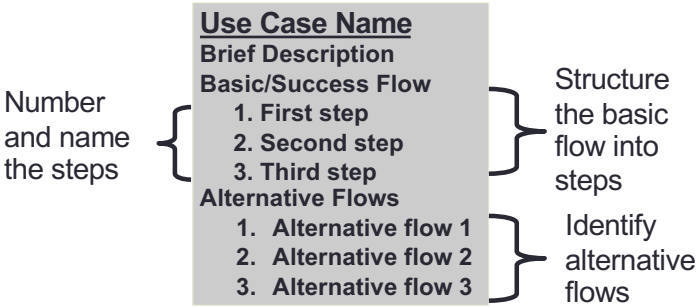
Communicates-Association



- 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

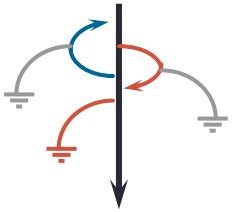
Outline each use case

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

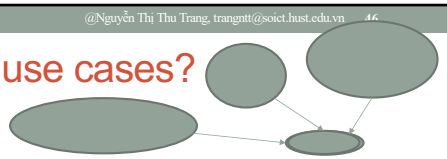


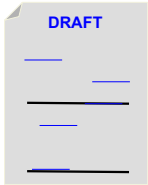
Flows of events (basic and alternative)

- A flow is a sequential set of steps
- One basic flow
  - Successful scenario from start to finish
- Many alternative flows
  - Regular variants
  - Odd cases
  - Exceptional (error) flows



Why outline use cases?

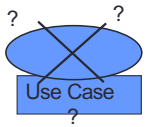




Use-Case Size

Too Small? Too Big?

Is it more than one use case?



Outlining helps find alternative flows

Outline the flows of events

- Basic flow
  - What event starts the use case?
  - How does the use case end?
  - How does the use case repeat some behavior?
- Alternative flows
  - Are there optional situations in the use case?
  - What odd cases might happen?
  - What variants might happen?
  - What may go wrong?
  - What may not happen?
  - What kinds of resources can be blocked?

View cart

- Success/Basic flow
  - Customer asks to view cart
  - Software gets all items and its availability
  - Software displays in the cart including item info. and its availability
    - Item name, quantity, price, amount <availability info>
- Alternative flow
  - error of unavailable items

- Use case: Only extend/include
  - The use case is very big → Extract for team working or ease of management
  - The duplication of a part of flow of events in several use cases → Extract for reuse or ease of maintenance
- -> Screen transition: Not captured in the use case diagram
- Interface design
  - UI design: Screen transition diagram

Step-by-step outline: Transfer to interbank account

Basic Flow / Success Flow

1. Customer chooses to the destination bank
2. Customer enters destination account
3. Software asks bank consortium to get destination name
4. Bank consortium returns destination name
5. Software displays destination name
6. Customer enters and submits an amount to transfer
7. Software checks the condition to transfer and send OTP
8. Customer enters an OTP and confirms to transfer
9. Software requests bank consortium to transfer
10. …

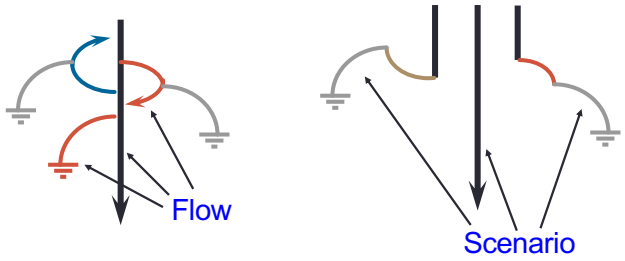
Alternative Flows

A1. Invalid destination account.
A2. The source account cannot transfer
A3. Invalid amount to transfer

What are other alternatives?

What is a use-case scenario?

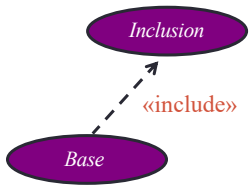
- An instance of a use case
- An ordered set of flows from the start of a use case to one of its end points



Note: This diagram illustrates only some of the possible scenarios based on the flows.

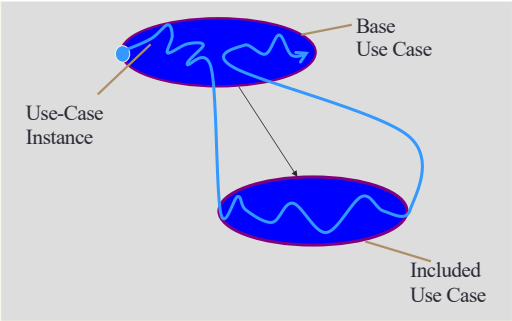
a. What Is an Include Relationship?

- A relationship from a base use case to an inclusion use case.
- The behavior defined in the inclusion use case is **explicitly included** into the base use case.

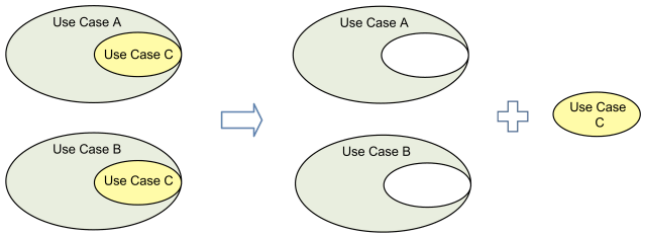


Execution of an Include Relationship

- Fully executed when the inclusion point is reached

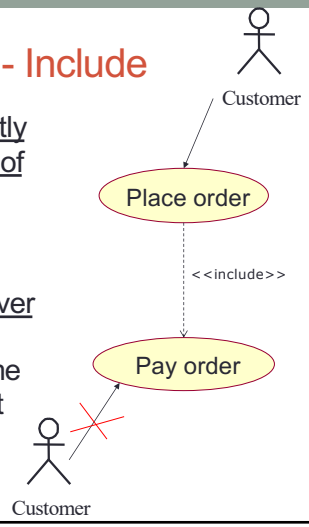


Example: include use case

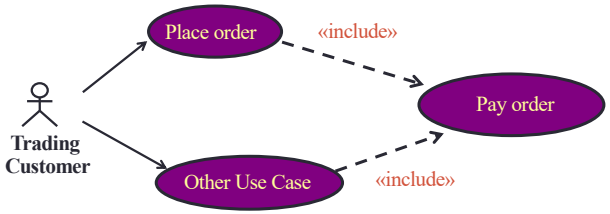


Between use cases - Include

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



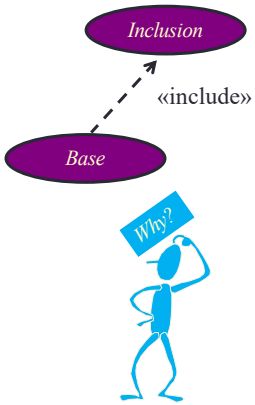
Include Relationship: RU e-st Example



<u>"Place order" Use Case</u>
1. Requests to place order
2. Displays order detail
3. Confirm to place order
4. **Call "Pay order"** to ask customer pay for the order
5. ...

<u>"Pay order" Use Case</u>
1. Choose payment method
2. Enter payment information
3. Enter OTP
4. ...

Why Use an Include Relationship?

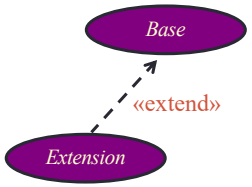


- Factor out behavior common to two or more use cases
  - Avoid describing same behavior multiple times.
  - Assure common behavior remains consistent.
- Factor out and encapsulate behavior from a base use case
  - Simplify complex flow of events.
  - Factor out behavior not part of primary purpose.

57

b. What Is an Extend Relationship?

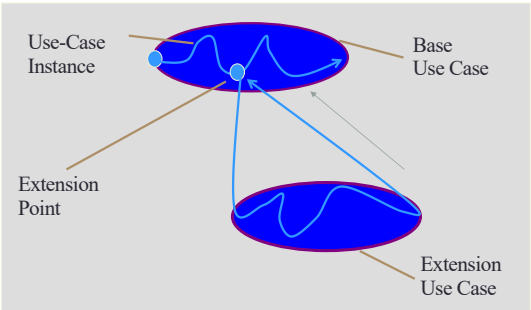
- Connects from an extension use case to a base use case.
  - Insert extension use case's behavior into base use case
  - Insert only if the extending condition is true
  - Insert into base use case at named extension point



58

Execution of an Extend

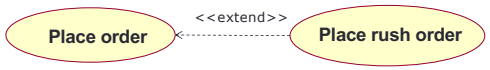
- Executed when the extension point is reached and the extending condition is true



59

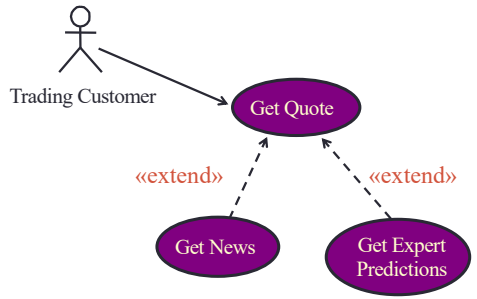
Between use cases - Extend

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



60

Extend Relationship: RU e-st Example



61

Extend Relationship: RU e-st Example (cont.)

"Get Quote" Use Case
Basic Flow:
1. Display options
2. Customer selects "Get Quote"
3. Customer gets quote
4. Customer gets other quotes
5. Customer logs off.

A1. Quote System unavailable
A2. Optional Services:
- If Customer selects "Get News", Use case "Get News" is inserted
- If Customer selects "Get Expert Predictions", Use case "Get Expert Predictions" is inserted

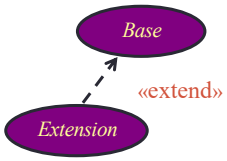
"Get News" Use Case
Basic Flow:
1. If Customer selects "Get News," the system asks customer for time period and number of news items.
2. Customer enters time period and number of items.
3. The system sends stock trading symbol to News System, receives reply, and displays the news to the customer.
4. The Get Quote Use Case continues.

A1. News System Unavailable
A2. No News About This Stock
…

This use case extends the Get Quote Use Case, at extension point "Optional Services."

62

Why Use an Extend Relationship?

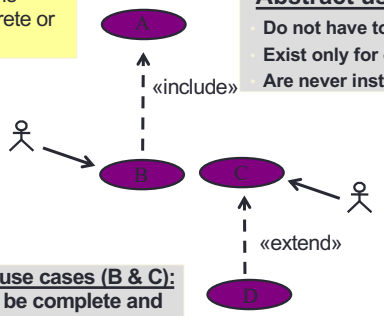


- Factor out optional or exceptional behavior
  - Executed only under certain conditions
  - Factoring out simplifies flow of events in base use case
  - Example: Triggering an alarm
- Add extending behavior
  - Behavior developed separately, possibly in later version.

63

Concrete Versus Abstract Use Cases

A use case is either concrete or abstract.



Abstract use cases (A & D):
- Do not have to be complete.
- Exist only for other use cases.
- Are never instantiated on their own.

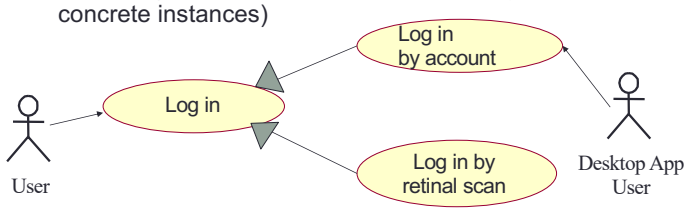
Concrete use cases (B & C):
- Have to be complete and meaningful.
- Can be instantiated on their own.

Hint:
Cover up all *abstract* use cases and you should still be able to understand the main purpose of the system.

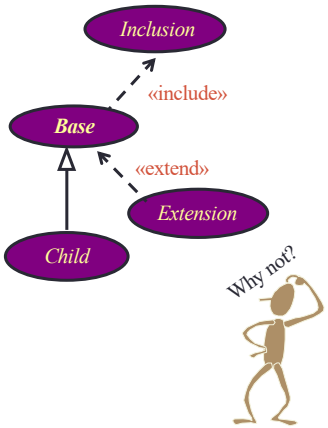
64

c. 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)



Why Wouldn't You Structure The Model?



- The solution is harder to see when the use case gets fragmented
  - Functionally decompose the requirement
  - Decrease understandability
  - Increase complexity
  - Increases effort for reviewers, implementers and testers
  - Not all stakeholders are comfortable with the format
- The use-case model looks like a design.

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

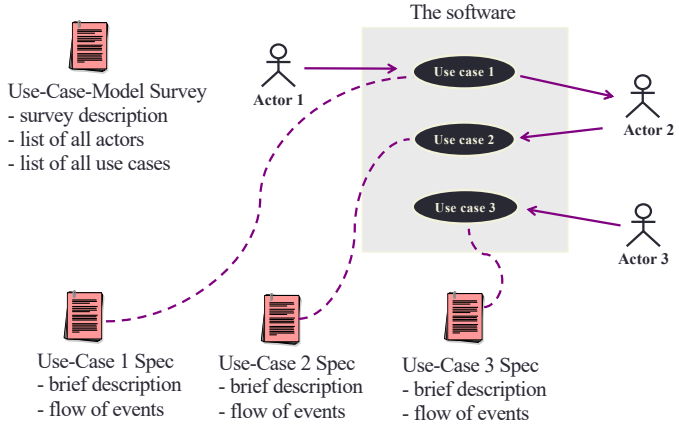
1. Requirements
2. Use case diagram
3. Use case specification/scenario
4. Glossary
5. Supplementary Specification

69

69

70

Use-Case Specification

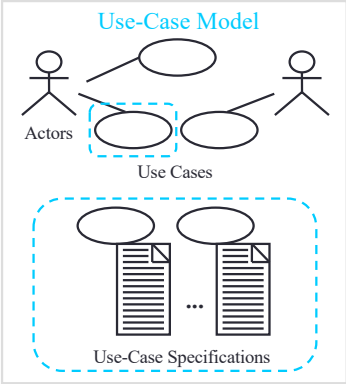


70

71

Use-Case Specification

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



71

72

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

72

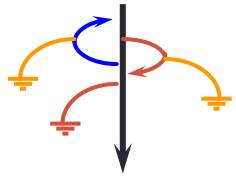
Brief description of UC

- *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"

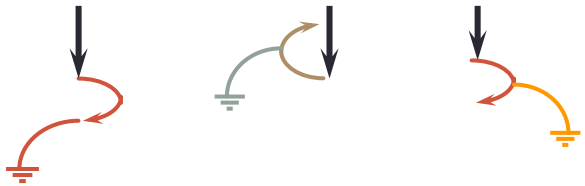
Use-Case Flow of Events

- 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.



UC Login in the ATM software

- Main flows of events: The use case starts when software prompts the Customer for a PIN Number. The Customer can now enter a PIN number. The Customer commits the entry. The software then checks this PIN number to see if it is valid. If valid, the software 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 software 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 to login**
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**

77

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

78

RUP style summary

- Basic flow
  - Steps are numbered and named
  - Steps do not reference alternative flows
  - Shows the main actor succeeding in that actor's main goal
- Alternative flows
  - Have names
  - May have steps

RUP Use-Case Specification Template

Use Case Name
1. Brief Description
2. Basic Flow of Events
3. Alternative Flows
   - 3.1 <Area of Functionality>
     - 3.1.1 < A1 First Alternative Flow >
     - 3.1.2 < A2 Second Alternative Flow >
   - 3.2 <Another Area of Functionality>
     - 3.2.1 < AN Another Alternative Flow >
4. Subflows
   - 4.1 <S1 First Subflow >
   - 4.2 < S2 Second Subflow >
5. Key Scenarios
6. Preconditions
7. Postconditions
8. Extension Points
9. Special Requirements
10. Additional Information

79

Use case specification for "Search media"

- Basic flow
  - Customer enters and submits keywords to search
  - Software queries media including keywords
  - Software displays a list of media including keywords (see Table x)
- Alternative flows

80

Input/output data specification

- Input data specification

No	Data fields	Description	Mandatory	Valid condition	Example
1.	Keyword		Yes	Not include special characters	Mắt biếc

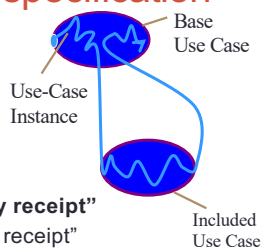
- Output data specification (Table x)

No	Data fields	Description	Display format	Example
1	Title	…	Left Alignment	Mắt biếc
2	Author	...	Left Alignment	Nguyễn Nhật Ánh
3	Price	..	Right Alignment, long, with Vietnamese dong	150.000 VNĐ
4	Published date	...	Left Alignment, dd/mm/yyyy format	30/12/1996

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"**

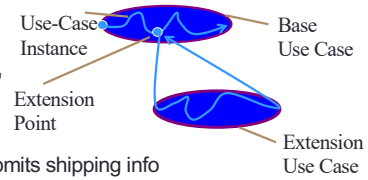
  Basic flow

  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

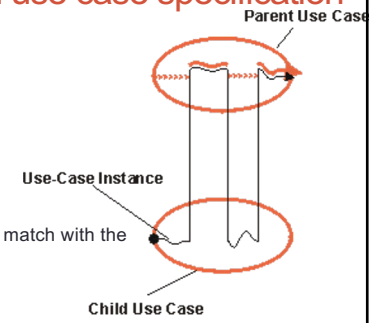
  A1. In the step 2, if customer chooses to place a rush order, insert use case "Place rush order", then resume at the step 3.

  …
- **Use case "Place rush order"**
  1. Software displays rush order form
  2. Customer
  3. …

Parent and child use case specification

- Parent use case
  - Defines common actors to all child use cases
  - Defines the steps that are common to all child use cases
  - May define default steps that apply to some but not all child use cases (*)
  - May define **empty placeholders** for steps that are to be defined by one or more of child use cases (**)
- Child use cases
  - Defines actors that is not defined by the parent use case
  - May override default steps of the parent use case's specification (*)
  - May fill in empty placeholders of the parent use case's specification (**)
  - May add its own steps, which aren't in any way anticipated in the parent use case's specification (***).

E.g. Parent and child use case specification



- **Use case "Validate user"**

  Basic flow
  1. Get user identity
  2. **<>**
  3. Match the input with user id
  4. **Return valid user**

  Alternative flow

  A1. In the step 3, if the input is not match with the user id, return invalid user
- **User "Check password"**

  Basic flow
  1. Get user password (hashed)
  2. **Hash the input (**)**
  3. Compare the hashed input and user password
  4. **Step 4->7 of the parent use case**

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.

How do you deal with the user interface?

- Leave the user interface out of the use case
  - Use cases are independent of the user interface
  - Describe user interfaces with
    - User-experience models or prototypes
    - User interface specifications

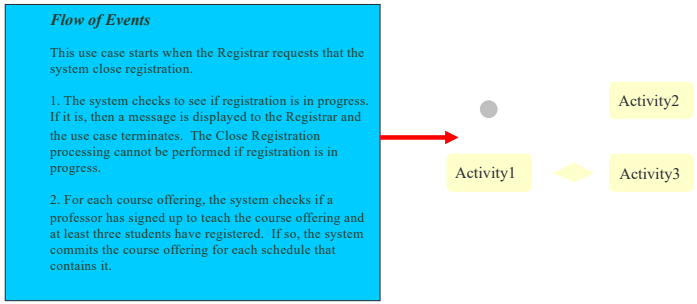
Words to Avoid			Words to Use	
Click	Drag	Form	Prompts	Chooses
Open	Close	Drop	Initiates	Specifies
Button	Field	Drop-down	Submits	Selects
Pop-up	Scroll	Browse	Starts	Displays
Record	Window		Informs	

Visualize behavior

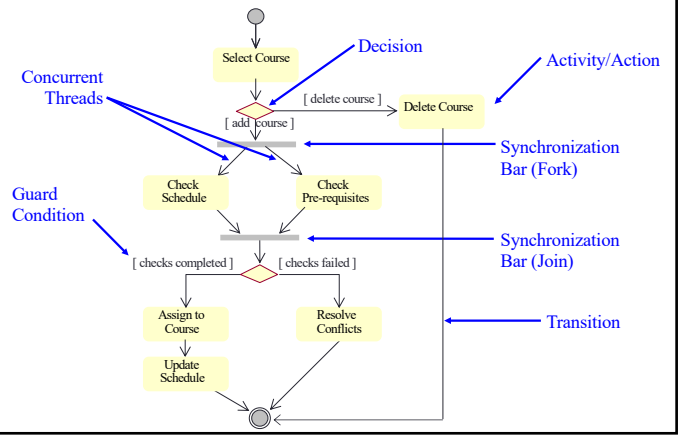
- Visual modeling tools
  - Activity diagrams or flow charts
  - Business process models
- Should you illustrate behavior?
  - Pro
    - Great tool to identify alternative flows, especially for visually oriented people
    - Succinctly conveys information about use case flows
  - Con
    - Costly to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



Example: Activity Diagram

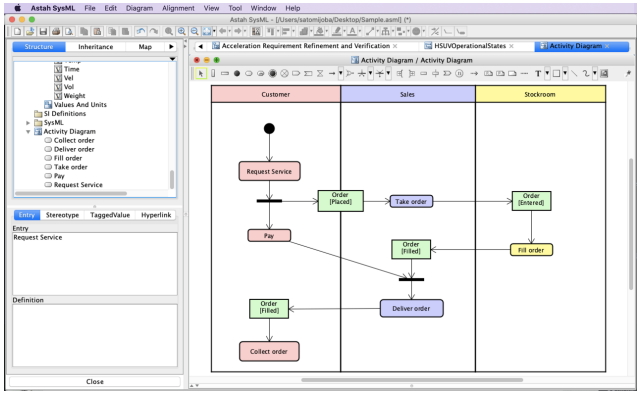


Partitions

- 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.

E.g. Activity diagram in Astah

Business Process



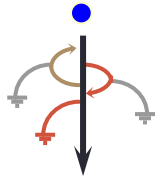
93

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



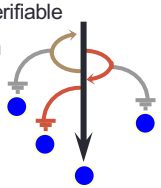
94

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:

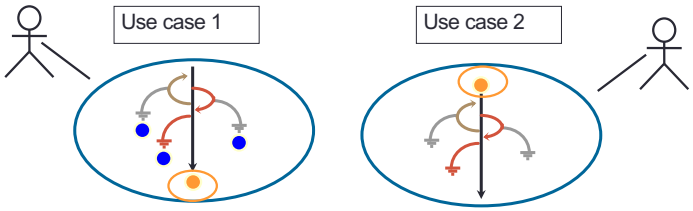
  Register for Courses use case

  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



95

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.

96

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

97

Content

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

98

98

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

99

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.

100

4. Glossary (3)

Glossary

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.

101

Case Study: Glossary

- Make the Glossary of the Course Registration System

Glossary

102

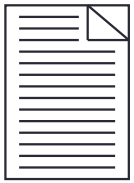
Content

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

103

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

104

5. Supplementary Specification (2)

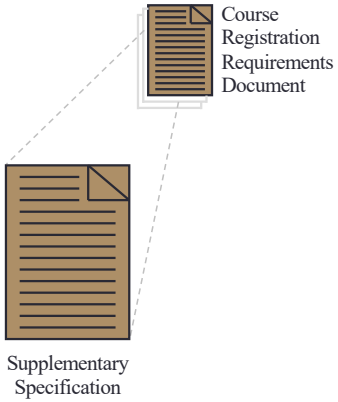
- **Functionality**: List of the functional requirements that are general to many use cases.
- **Usability**: Requirements that relate to, or affect, the usability of the system. Examples include ease-of-use requirements or training requirements that specify how readily the system can be used by its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

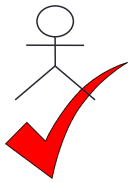
Case study: Supplementary Specification

- 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?



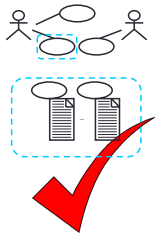
Checkpoints: Use-Cases

- 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?

109

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?



110

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?

111

112

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?

112

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?

113

Question?

114