204362 – Object-Oriented Design

Use Case Models

Adapted for 204362
by Areerat Trongratsameethong

Bennett, McRobb and Farmer:
Object Oriented Systems Analysis and Design Using UML, (4th Edition), McGraw Hill, 2010

Chapter Overview

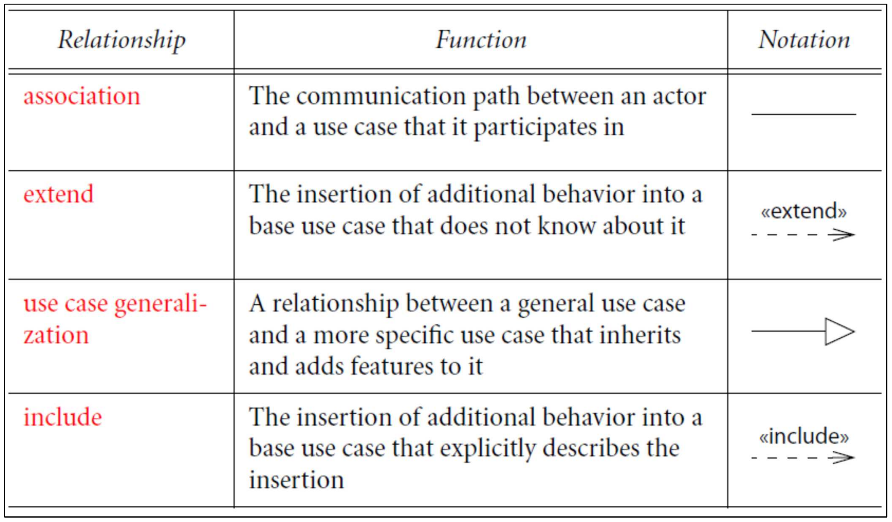
- Use Case Modelling
- Prototyping
- Advance Use Case

Use Cases

- **Use Cases** – descriptions of the **functionality** of the system from the users' perspective.
  - Use Case diagrams
    - show which users will communicate with the system
    - define the scope of the system
  - Use Case descriptions
    - specify the interaction between the users (actors) and the system for each use case as the users see it
    - could be further elaborated by communication/sequence diagrams.

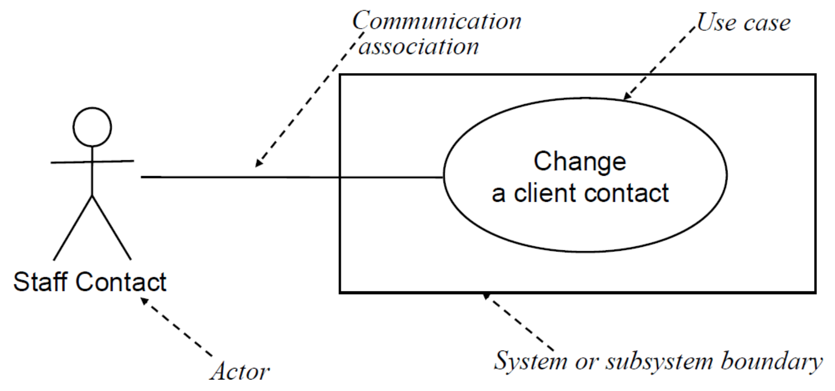
What are the key activities that make this business work?

Use Case Relationships



Relationship	*Function*	*Notation*
association	The communication path between an actor and a use case that it participates in	———
extend	The insertion of additional behavior into a base use case that does not know about it	«extend» - - - ->
use case generalization	A relationship between a general use case and a more specific use case that inherits and adds features to it	——▷
include	The insertion of additional behavior into a base use case that explicitly describes the insertion	«include» - - - ->

Use Case Diagrams



Use Case Diagrams: Actor

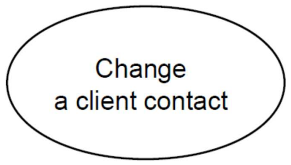
- describe the role that people, other systems or devices take when communicating with a particular use case or use cases
  - not the same as job title or specific person
    - one job title may play the roles of several actors
    - one actor may represent several job titles
  - drawn as a stick figure with a name

Staff Contact

Lecturer Dell Zhang	*Role: Admission Tutor*
Professor Mark Levene	*Role: Research Tutor*

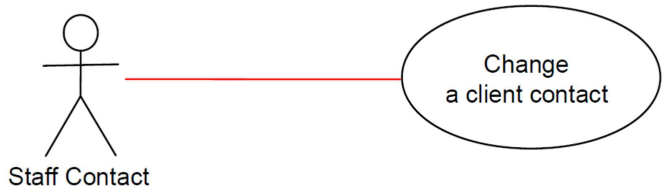
Use Case Diagrams: Use Case

- describe a sequence of actions that the system performs to achieve an observable result of value to an actor
  - drawn as a bubble (ellipse) with a name in or below
    - the name is usually an active verb and a noun phrase

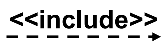


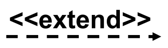
Use Case Diagrams: Communication Association

- describes the communication link between an instance of the use case and an instance of the actor
  - drawn as a line between the actor and the use case



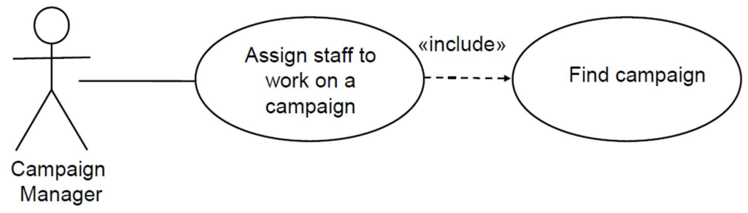
Use Case Diagrams: Extend and Include relationships

- The extend and include relationships between Use Cases are shown as stereotyped dependencies (text strings in guillemets) :
  - «include» 

  - «extend» 

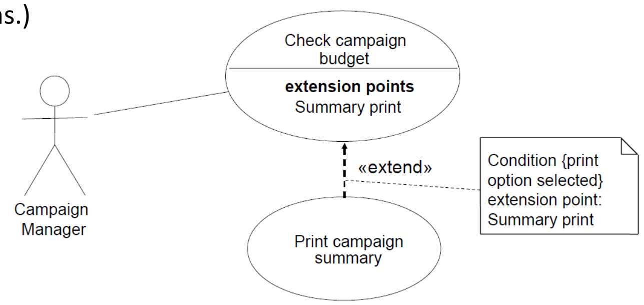
Use Case Diagrams: «include»

- One use case always includes the functionality of another use case
- A use case may include more than one other
- Can be used to separate out a sequence of behavior that is used in many use cases
- Should not be used to create a hierarchical functional decomposition of the system



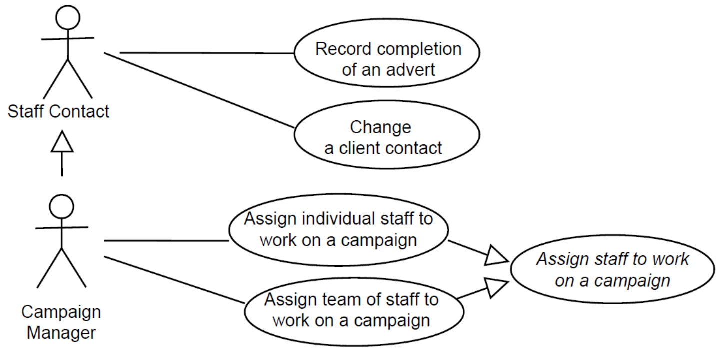
Use Case Diagrams: «include»

- One use case provides additional functionality that may be required in another use case
- There may be multiple ways of extending a use case, which represent variations in the way that actors interact with the use case
- The extension points show when the extension occurs
- A condition can be placed in a note joined to the dependency arrow (Note that it is not put in square brackets, unlike conditions in other diagrams.)



Use Case Diagrams: Generalization

- Between use cases: shows that one use case provides all the functionality of the more specific use case and some additional functionality
- Between actors: shows that one actor can participate in all the associations with use cases that the more specific actor can plus some additional use cases


Staff Contact
Record completion of an advert
Change a client contact
Campaign Manager
Assign individual staff to work on a campaign
Assign team of staff to work on a campaign
Assign staff to work on a campaign

Use Case Descriptions

- Using a simple paragraph
  - *Assign staff to work on a campaign*

    The campaign manager wishes to record which staff are working on a particular campaign. This information is used to validate timesheets and to calculate staff year-end bonuses.
- Using a step-by-step breakdown of interaction between actor and system

Assign staff to work on a campaign

Actor Action	**System Response**
1. The actor enters the client name.	2. Lists all campaigns for that client.
3. Selects the relevant campaign.	4. Displays a list of all staff members not already allocated to this campaign.
5. Highlights the staff members to be assigned to this campaign.	6.Presents a message confirming that staff have been allocated.

Alternative Courses
Steps 1–3. The actor knows the campaign name and enters it directly.

Use Case Descriptions

- Using a template
  - name of use case
  - pre-conditions
  - post-conditions
  - Purpose
  - Description
  - alternative courses (routes)
  - errors

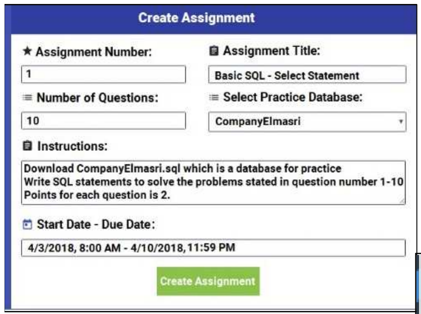
Use Case Description Example

Use case ID	UC01
Use case Name	Create Assignment
Actor/User	Instructor
Description	Create assignment and their questions for each subject in each semester.
Pre-condition	-
Post-condition	Assignment and question information are stored in the Assignment database.
Flow of events	1 User inputs assignment data: assignment number, assignment title, number of questions, practice database name, assignment start date, and assignment due date. 2 The assignment data are validated as follows: • Length of assignment title must not exceed 80 characters. • Length of practice database name must not exceed 30 characters. • Assignment number and number of questions must be positive integers. 3 User adds question data until all question data of the assignment are added to the Assignment database as follows: 3.1 Question number is automatic generated by the system starting from 1. 3.2 User inputs other question data: question description, complication level, SQL statement solution, and question score. 3.3 The question data are validated as follows: • Length of question description must not exceed 256 characters. • Question score must be positive integer. 3.4 Increasing question number by 1. …
Alternate Flow	2.1 If there are any invalid data, the system displays error message and forces user to re-enter assignment data. 3.3.1 If there are any invalid data, the system displays error message and forces user to re-enter question data.

Prototyping

- A prototype is a system or a partially complete system that is built quickly to explore some aspects of the system requirements and that is not intended as the final working system.
- Prototyping can be used to support use case modelling
- Help elicit functional requirements
- Test out system architectures based on the use cases in order to meet the non-functional requirements

Prototyping Example


Create Assignment

★ Assignment Number: 1

Assignment Title: Basic SQL - Select Statement

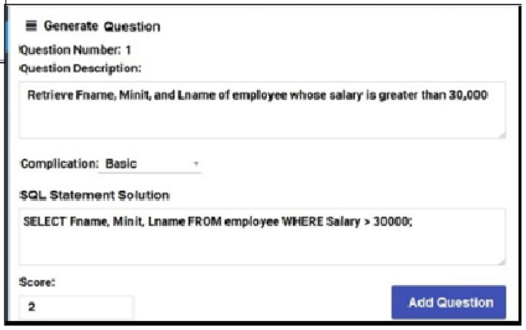
Number of Questions: 10

Select Practice Database: CompanyElmasri

Instructions:

Download CompanyElmasri.sql which is a database for practice
Write SQL statements to solve the problems stated in question number 1-10
Points for each question is 2.

Start Date - Due Date: 4/3/2018, 8:00 AM - 4/10/2018, 11:59 PM


Generate Question

Question Number: 1

Question Description:

Retrieve Fname, Minit, and Lname of employee whose salary is greater than 30,000

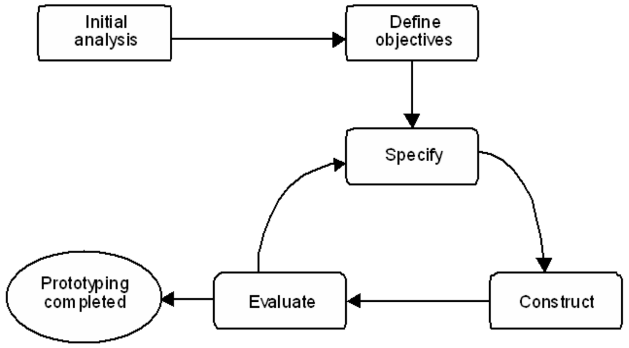
Complication: Basic

SQL Statement Solution

```
SELECT Fname, Minit, Lname FROM employee WHERE Salary > 30000;
```

Score: 2

Prototyping



Advantages of prototyping

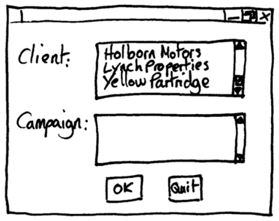
- Early demonstrations of system functionality help identify any misunderstandings between developer and client
- Client requirements that have been missed can be identified
- Difficulties in the interface can be identified
- The feasibility and usefulness of the system can be tested, even though, by its very nature, the prototype is incomplete

Disadvantages of prototyping

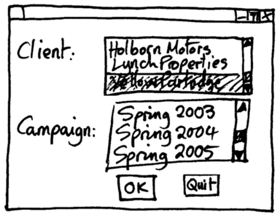
- The client may perceive the prototype as part of the final system
- The prototype may divert attention from functional to solely interface issues
- Prototyping requires significant user involvement
- Managing the prototyping life cycle requires careful decision making

User Interface prototypes

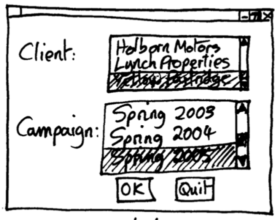
- For user interface prototypes, storyboarding can be used with hand-drawn designs



Dialogue initialized.



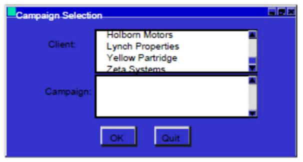
User selects Client. Campaigns listed.



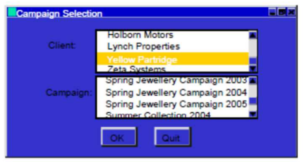
User selects Campaign.

User Interface prototypes

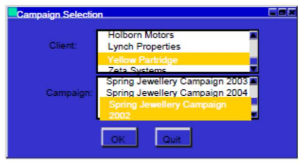
- User interface prototypes can be implemented using languages other than the one that the system will be developed in, for example, Visual Basic.



Dialogue initialized.



User selects Client. Campaigns listed.



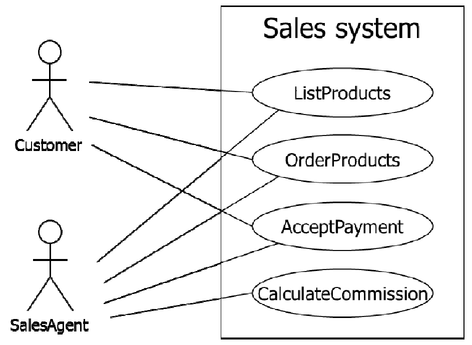
User selects Campaign.

Advanced Use Case Modelling

- Actor generalisation
- Use case generalisation
- «include» – between use cases
- «extend» – between use cases

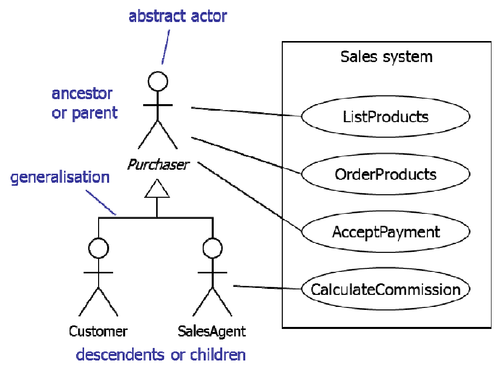
Actor generalization - example

- The Customer and the Sales Agent actors are *very* similar
- They both interact with List products, Order products, Accept payment
- Additionally, the Sales Agent interacts with Calculate commission
- Our diagram is a *mess* – can we simplify it?



Actor generalization

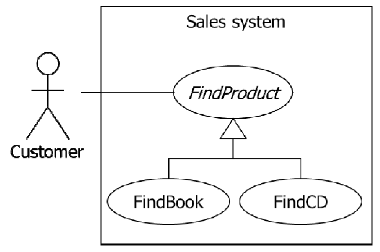
- If two actors communicate with the same set of use cases in the same way, then we can express this as a generalisation to another (possibly abstract) actor
- The descendent actors inherit the roles and relationships to use cases held by the ancestor actor
- We can substitute a descendent actor anywhere the ancestor actor is expected. This is the *substitutability principle*



Use actor generalization when it simplifies the model

Use case generalisation

- The ancestor use case must be a more general case of one or more descendant use cases
- Child use cases are more specific forms of their parent
- They can inherit, add and override features of their parent

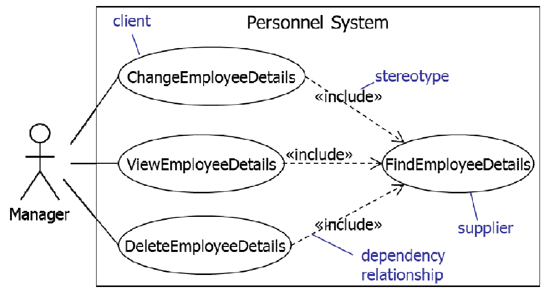


Example of Use Case Description

- Find Product
  - Input searching criteria to be search
  - Find products according to searching criteria
  - Display products matching to search criteria
- Find Book
  - Input one of followings: ISBN, author name, book title, or publisher
  - Find books according to searching criteria specified in previous step
  - Display books matching to search criteria
- Find CD
  - Input one of followings: CD Id or CD title
  - Find CDs according to searching criteria specified in previous step
  - Display CDs matching to search criteria

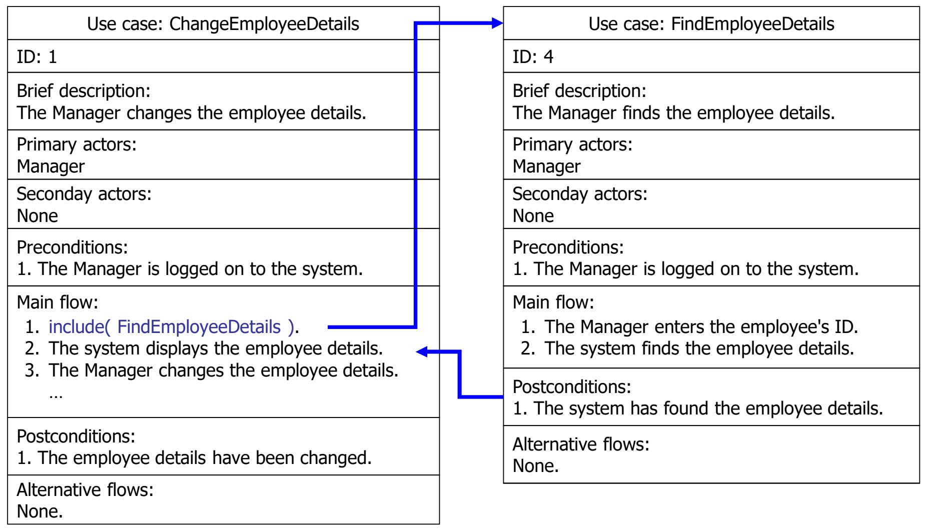
«include»

- The client use case executes until the point of inclusion: include(SupplierUseCase)
  - Control passes to the supplier use case which executes
  - When the supplier is finished, control passes back to the client use case which finishes execution
- Note:
  - Client use cases are *not complete* without the included supplier use cases
  - Supplier use cases may be complete use cases, or they may just specify a fragment of behaviour for inclusion elsewhere



When use cases share common behaviour we can factor this out into a separate supplier use case and «include» it in the clients

«include» Example

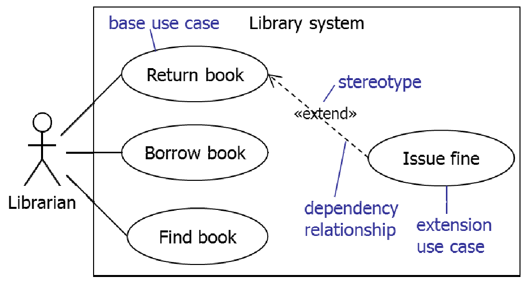


Use case: ChangeEmployeeDetails
ID: 1
Brief description: The Manager changes the employee details.
Primary actors: Manager
Seconday actors: None
Preconditions: 1. The Manager is logged on to the system.
Main flow: 1. include(FindEmployeeDetails). 2. The system displays the employee details. 3. The Manager changes the employee details. …
Postconditions: 1. The employee details have been changed.
Alternative flows: None.

Use case: FindEmployeeDetails
ID: 4
Brief description: The Manager finds the employee details.
Primary actors: Manager
Seconday actors: None
Preconditions: 1. The Manager is logged on to the system.
Main flow: 1. The Manager enters the employee's ID. 2. The system finds the employee details.
Postconditions: 1. The system has found the employee details.
Alternative flows: None.

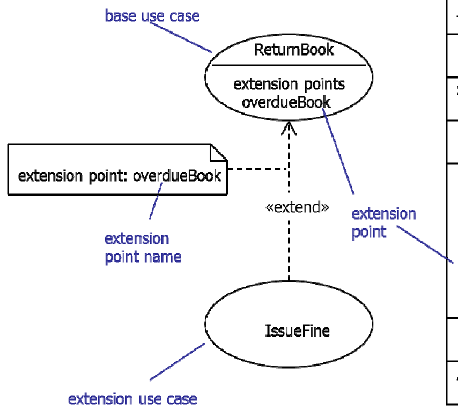
«extend»

- «extend» is a way of adding new behaviour into the base use case by inserting behaviour from one or more extension use cases
  - The base use case specifies one or more extension points in its flow of events
- The extension use case may contain several insertion segments
- The «extend» relationship may specify *which* of the base use case extension points it is extending



The client use case inserts behaviour into the base use case. The base use case provides extension points, but *does not know* about the extensions.

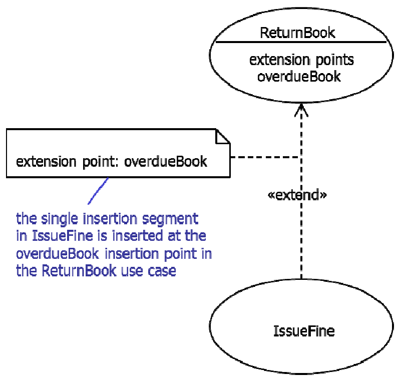
Base use case



Use case: ReturnBook
ID: 9
Brief description: The Librarian returns a borrowed book.
Primary actors: Librarian
Secondary actors: None.
Preconditions: 1. The Librarian is logged on to the system.
Main flow: 1. The Librarian enters the borrower's ID number. 2. The system displays the borrower's details including the list of borrowed books. 3. The Librarian finds the book to be returned in the list of books. extension point: overdueBook 4. The Librarian returns the book. ...
Postconditions: 1. The book has been returned.
Alternative flows: None.

- There is an extension point overdueBook just before step 4 of the flow of events
- Extension points are *not* numbered, as they are *not* part of the flow

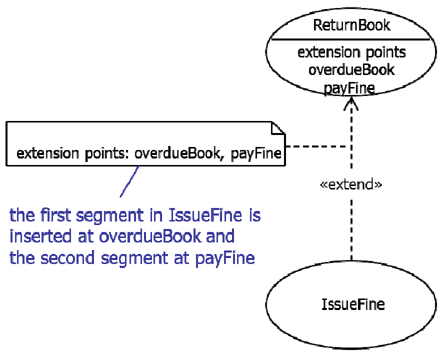
Extension use case


ReturnBook
extension points
overdueBook
extension point: overdueBook
«extend»
the single insertion segment in IssueFine is inserted at the overdueBook insertion point in the ReturnBook use case
IssueFine

Extension Use case: IssueFine
ID: 10
Brief description: Segment 1: The Librarian records and prints out a fine.
Primary actors: Librarian
Secondary actors: None.
Segment 1 preconditions: 1. The returned book is overdue.
Segment 1 flow: 1. The Librarian enters details of the fine into the system. 2. The system prints out the fine.
Segment 1 postconditions: 1. The fine has been recorded in the system. 2. The system has printed out the fine.

- Extension use cases have one or more *insertion segments* which are behaviour fragments that will be inserted at the specified extension points in the base use case

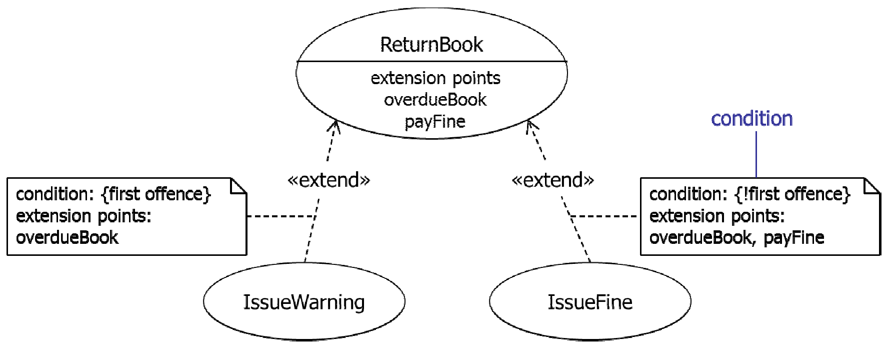
Multiple insertion points


ReturnBook
extension points
overdueBook
payFine
extension points: overdueBook, payFine
«extend»
the first segment in IssueFine is inserted at overdueBook and the second segment at payFine
IssueFine

Extension Use case: IssueFine
ID: 10
Brief description: Segment 1: The Librarian records and prints out a fine. Segment 2: The Librarian accepts payment for a fine.
Primary actors: Librarian
Secondary actors: None.
Segment 1 preconditions: 1. The returned book is overdue.
Segment 1 flow: 1. The Librarian enters details of the fine into the system. 2. The system prints out the fine.
Segment 1 postconditions: 1. The fine has been recorded in the system. 2. The system has printed out the fine.
Segment 2 preconditions: 1. A fine is due from the borrower.
Segment 2 flow: 1. The Librarian accepts payment for the fine from the borrower. 2. The Librarian enters the paid fine in the system. 3. The system prints out a receipt for the paid fine.
Segment 2 postconditions: 1. The fine is recorded as paid. 2. The system has printed a receipt for the fine.

- If more than one extension point is specified in the «extend» relationship then the extension use case must have the *same number* of insertion segments

Conditional extensions



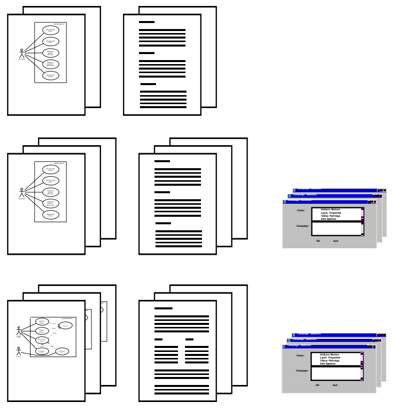
- We can specify conditions on «extend» relationships
  - Conditions are Boolean expressions
  - The insertion is made if and only if the condition evaluates to true

Development of the Use Case Model

Iteration 1
Obvious use cases.
Simple use case descriptions.

Iteration 2
Additional use cases.
Simple use case descriptions.
Prototypes.

Iteration 3
Structured use cases.
Structured use case descriptions.
Prototypes.



Summary

- We have learned about techniques for:
- Use case Modelling
- Advanced use case modelling:
  - Actor generalisation
  - Use case generalisation
  - «include»
  - «extend»
- Use advanced features with discretion only where they simplify the model!