CS5030

Modelling Use Cases in UML

Learning objectives

- On completing this lecture and associated reading, you should
  - Know the purpose of use case diagrams
  - Be aware of the fundamental constructs of use case diagrams
  - Be able to construct use case diagrams for a given specification

Use case diagram

- High level view of the functionality of the system
  - *What* the system does and *who* uses it
- Use case modelling is a form of requirement engineering
  - Eliciting and documenting functional requirements
- Best used when
  - The system is dominated by functional requirements
  - The system has many types of users to whom it delivers different functionality

Use case modelling process

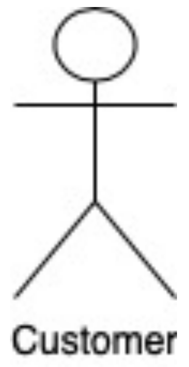
- Identifying a system boundary
- Identifying actors
- Identifying the use cases

- Iterating these steps until use cases and system boundary are stable

Actor

- Someone or something that interacts with the system
  - External to the system

- UML notation



Actors

- An actor specifies a role that some external entity adopts when interacting with the system directly
  - Entities may have many roles simultaneously and over time
  - A given role may be played by different entities simultaneously and over time

- Dinstinction between a role and an entity
  - What role does this entity play with respect to the system?
  - Can find common behaviours among many different entities and thus simplify the use case model

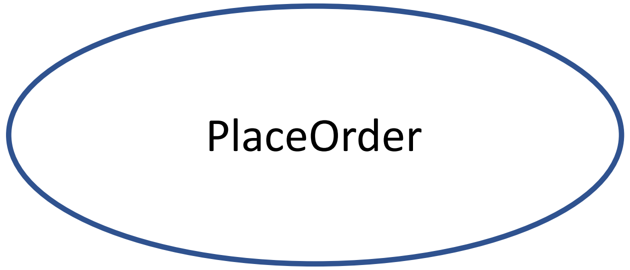
Use case

- Formally
  - A specification of sequence of actions, including variant sequences and error sequences, that a system, subsystem or class can perform by interacting with outside actors

- Informally
  - A use case is something an actor wants the system to do
  - Use cases are always started by an actor
  - Use cases are always written from the point of view of the actor

Use case

- UML notation



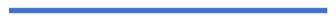
Identifying use cases

- One obvious way is to
  - start with the list of actors
  - and then consider how each actor is going to use the system
- You may find new actors when identifying use cases
  - Use case modelling is iterative and proceeds via a process of stepwise refinement

Communication link

- Participation of an actor in a use case

- UML notation



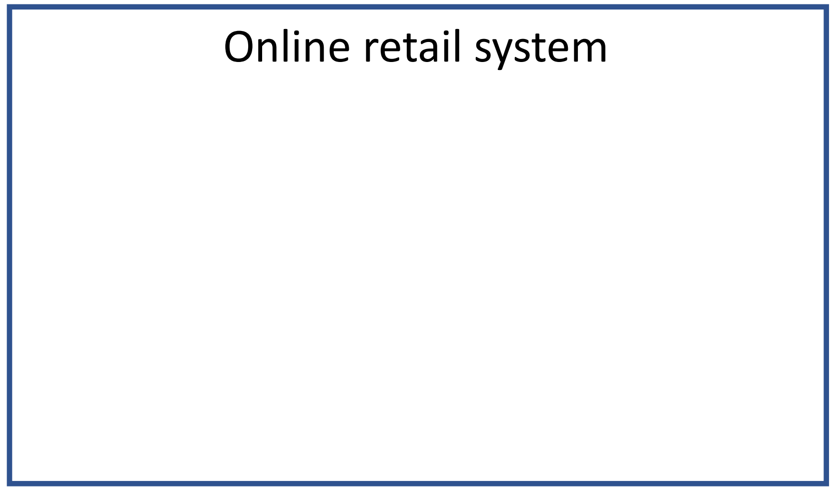
System boundary

- System boundary is used to define
  - What is part of the system (inside the boundary)
  - What is external to the system (outside the boundary)

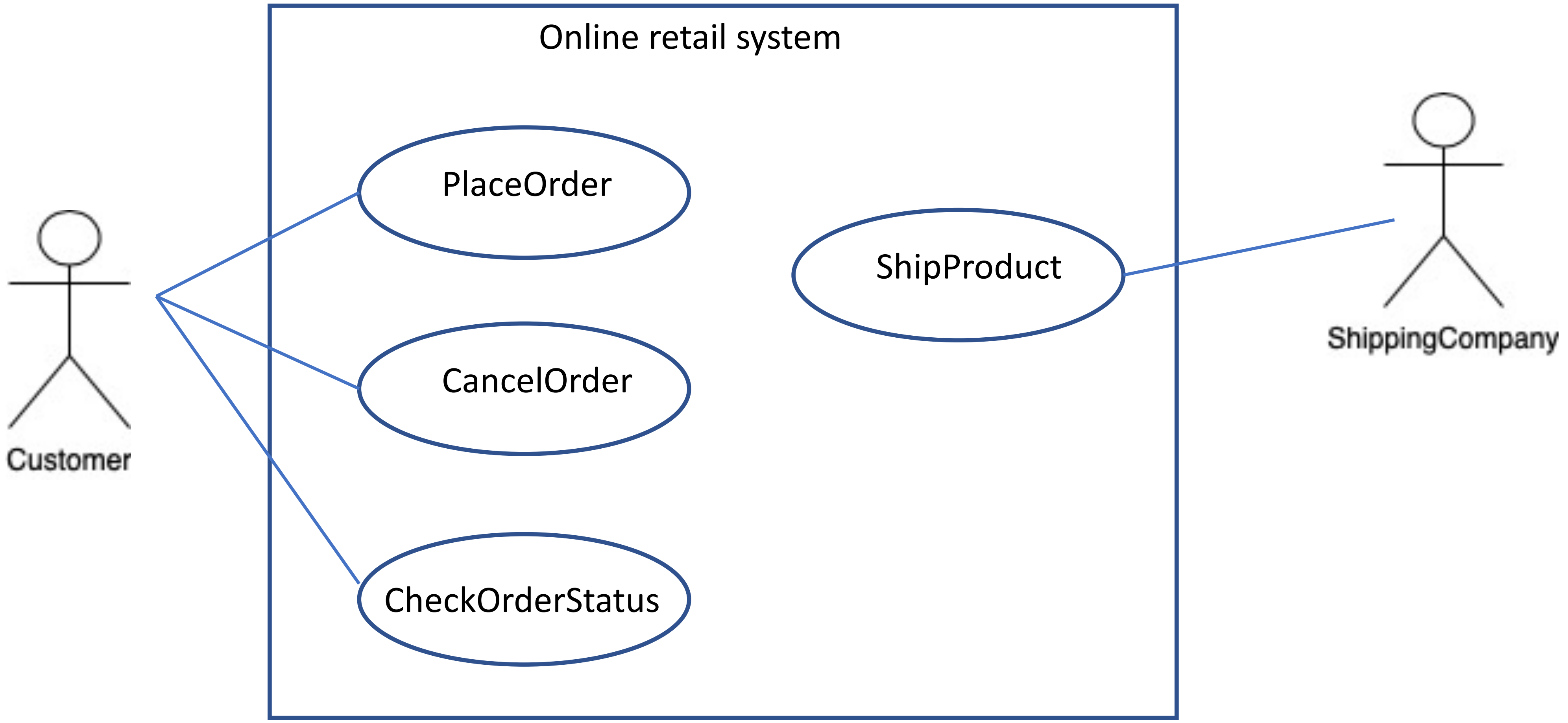
- A system boundary is represented as a box
  - labelled with the name of the system
  - with actors drawn outside the box
  - with use cases inside the box

System boundary

- UML notation



Example - an online retail system



Use case specification

- A document for each use case
- No UML standard
- Typical content
  - How the use case starts and ends
  - Normal flow of events
  - Alternate flow of events

General use case template

- Use case name
- Use case ID
- Brief description
- Actors
- Precondition
  - What must be true before the use case starts
- Main flow
  - Steps in the use case
- Postcondition
  - What must be true at the end of the use case
- Alternative flows

Use case spec - example

- Use case name: CreateNewCustomerAccount
- Use case ID: 5
- Brief description: System creates a new account for the Customer
- Actors: Customer
- Precondition: None
- Main flow: 1. The use case starts when the Customer selects ….
- Postcondition: A new account has been created for the Customer
- Alternative flows
  - InvalidEmailAddress
  - InvalidPassword
  - Cancel

Main flow vs alternative flow

- Main flow
  - Lists the steps that capture the situation where everything goes as expected and desired
  - There are no errors, deviation, interrupts or branches
- If deviation exists
  - If it is small, we create branches in the main flow
  - Otherwise, we create alternative flows

Main flow template

- Starting a flow of events

  The use case starts when an <actor> does <function>

- The flow of events consist of a sequence of short steps that are declarative, numbered, and time-ordered

- Each step should be in the form:

  <number> The <something> does <some action>

- An example of ill-formed step description:

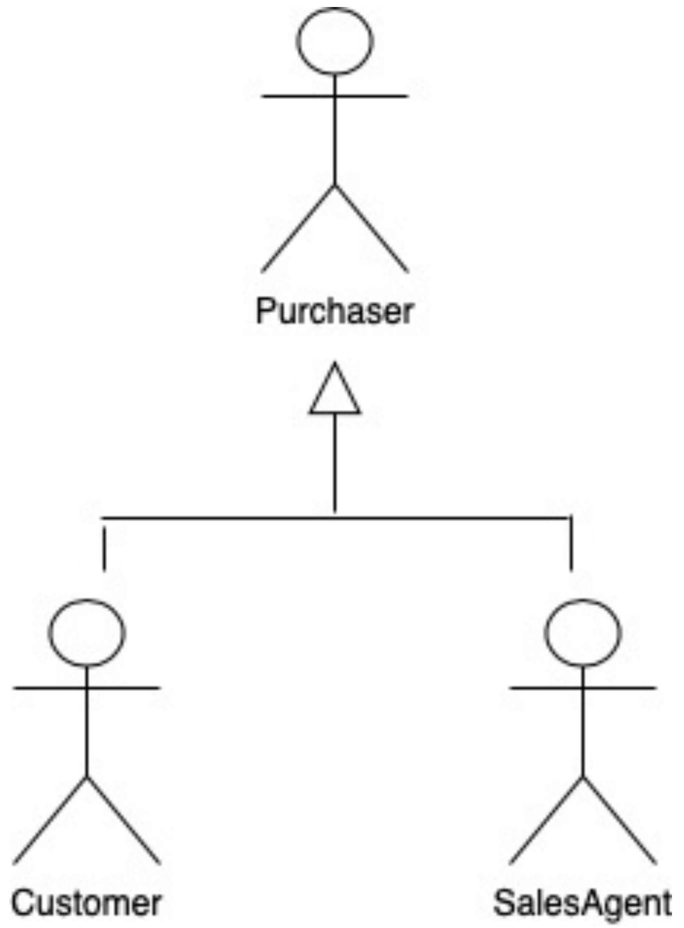
  2. Customer details are entered

  - Any step written in the passive voice is usually ill-formed

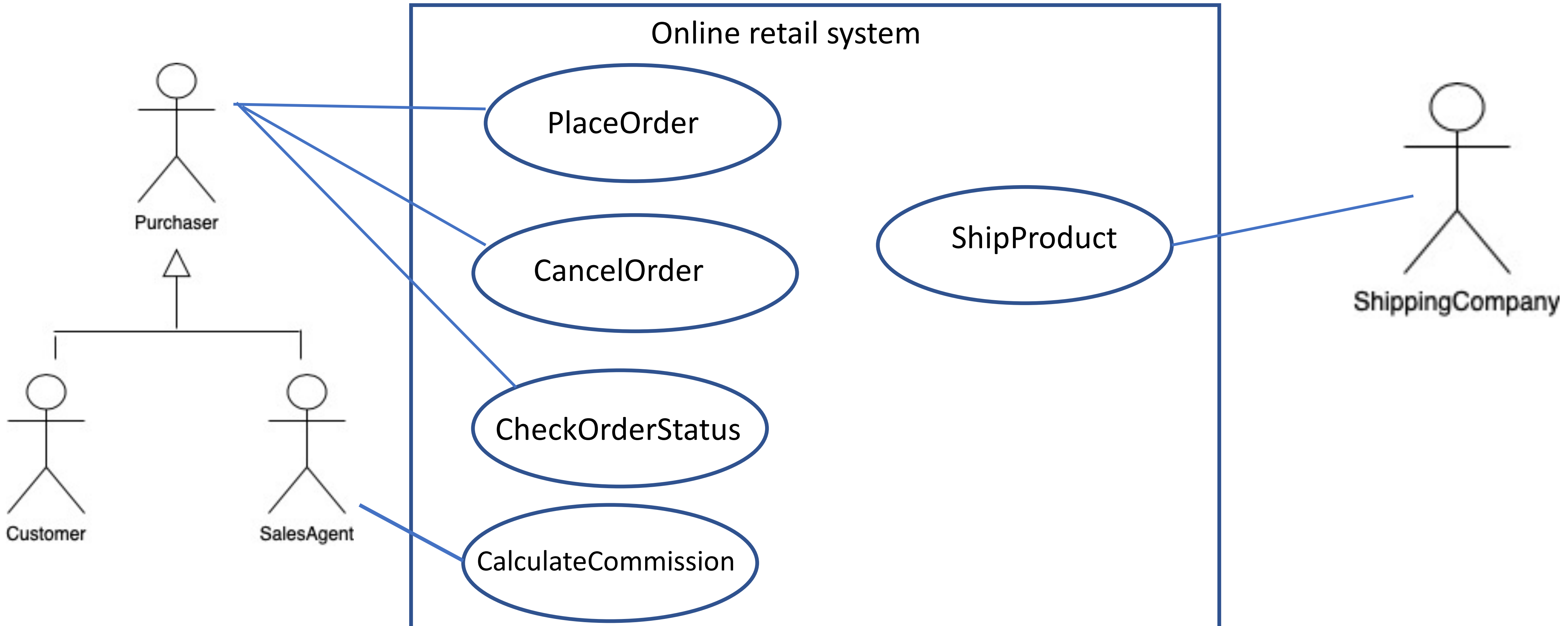
Relationships in use case diagrams

- Actor generalisation
  - A relationship between a general actor and a more specific actor
- Use case generalisation
  - A relationship between a general use case and a more specific use case
- <<include>>
  - A relationship between use cases that lets one use case include behaviours from others
- <<extend>>
  - A relationship between use cases that lets one use case extend its behaviour with one or more behaviour fragments from another

Actor generalisation - notation



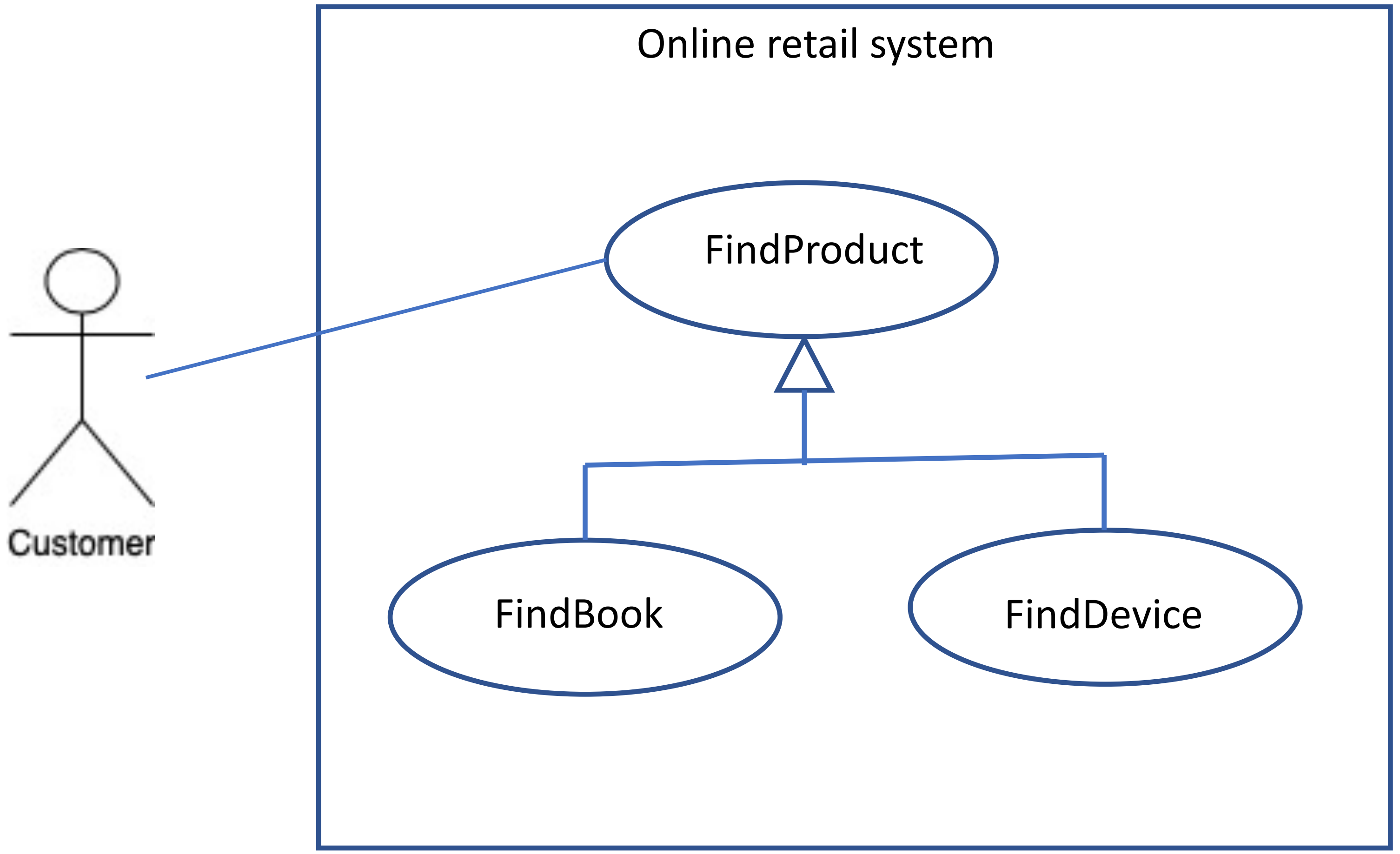
Actor generalisation - example



Use case generalisation

- Factors out behaviour common to one or more use cases into a parent use case
- Child use cases represent more specific forms of the parent
- Children may
    - Inherit features from their parent use case
    - Add new features
    - Override (change) inherited features

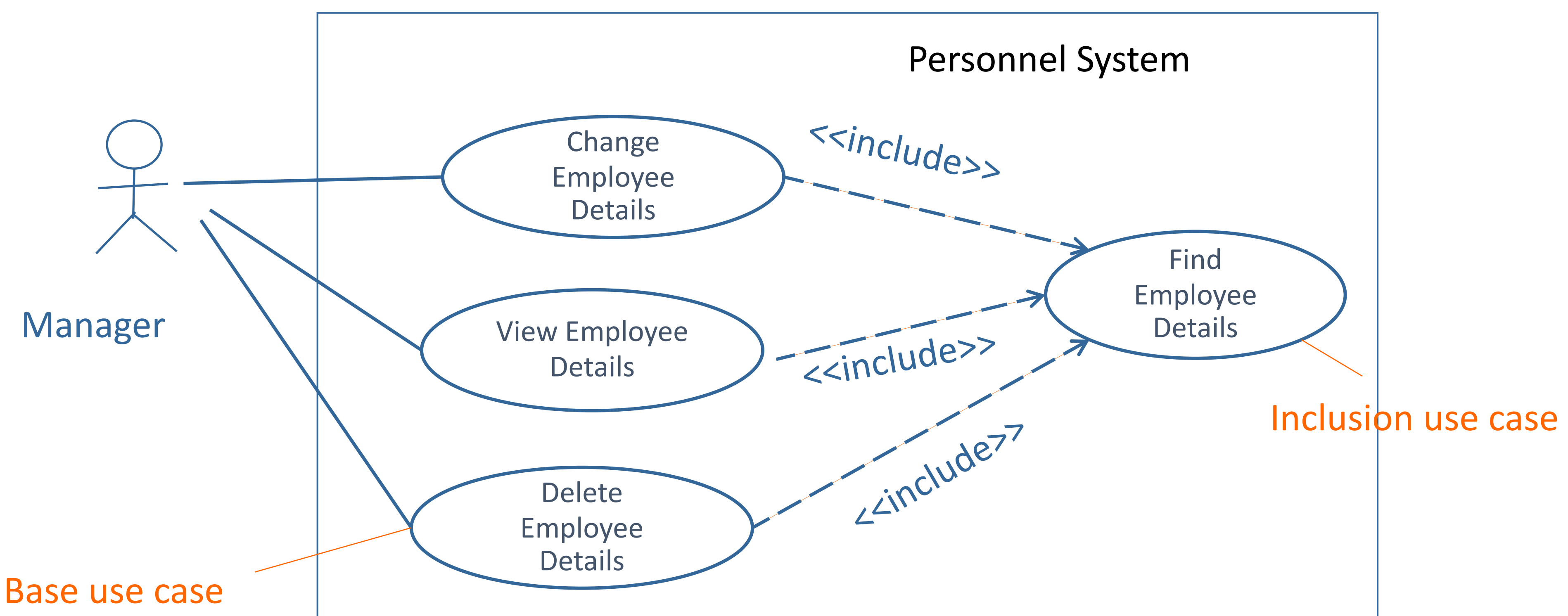
Use case generalisation - example



<<include>> relationship

- The base use case explicitly incorporates the behaviour of another use case at a location specified in the base
- Uses
  - To avoid repetition of a use case in multiple base use cases
  - To show how the system can reuse a pre-existing component
  - To show common functionalities between use cases
  - To document the fact that the project has developed a new reusable component

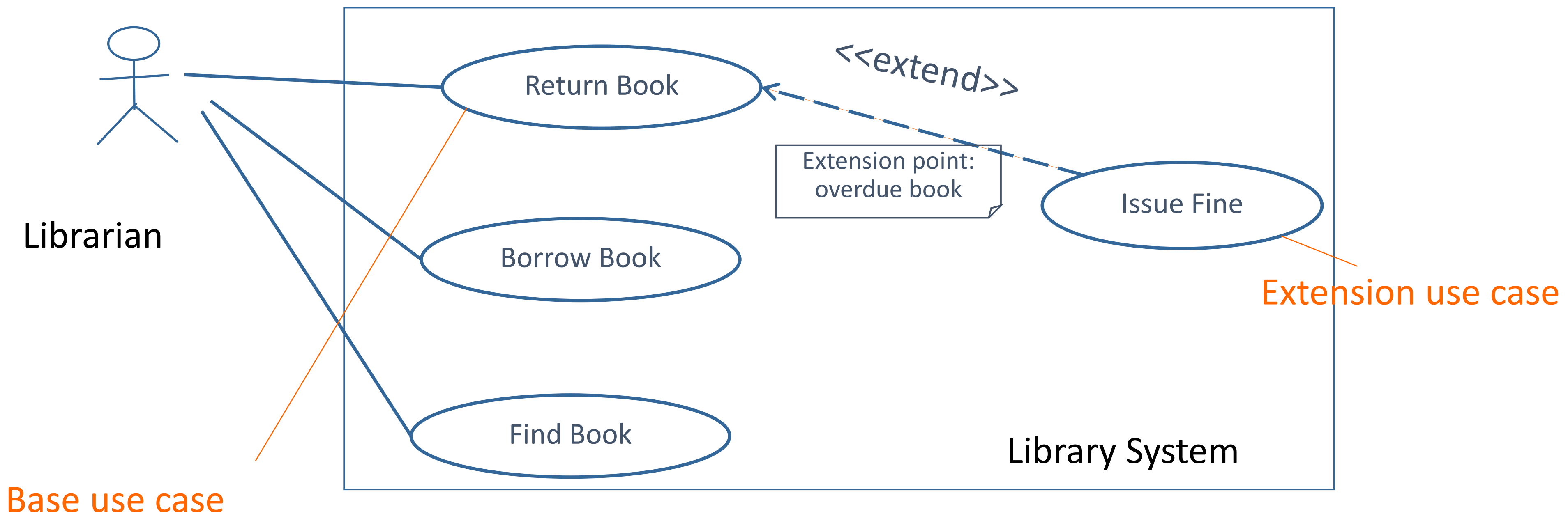
<<include>> example


Personnel System
Manager
Change Employee Details
View Employee Details
Delete Employee Details
Find Employee Details
<<include>>
<<include>>
<<include>>
Inclusion use case
Base use case

<<extend>> relationship

- A way to insert new behaviour into an existing use case
  - The base use case provides a set of extension points where new behaviour may be added
  - The extension use case provides a set of insertion segments that can be inserted into the base use case
- The extend relationship contains a condition and references a sequence of extension points in the target use case
  - The condition must be satisfied if the extension is to take place,
  - References to extension points define the locations in the base use case where additions are to be made

<<extend>> relationship - example



Key points

- Use case modelling is part of requirements engineering
- Use case diagram defines the system boundary, the actors who interact directly with the system and the functionality (use cases) provided by the system to actors
- More advanced features can be used in use case diagrams
- General guidance: keep the model simple
    - If in doubt, leave it out