# Object-Oriented Software Design

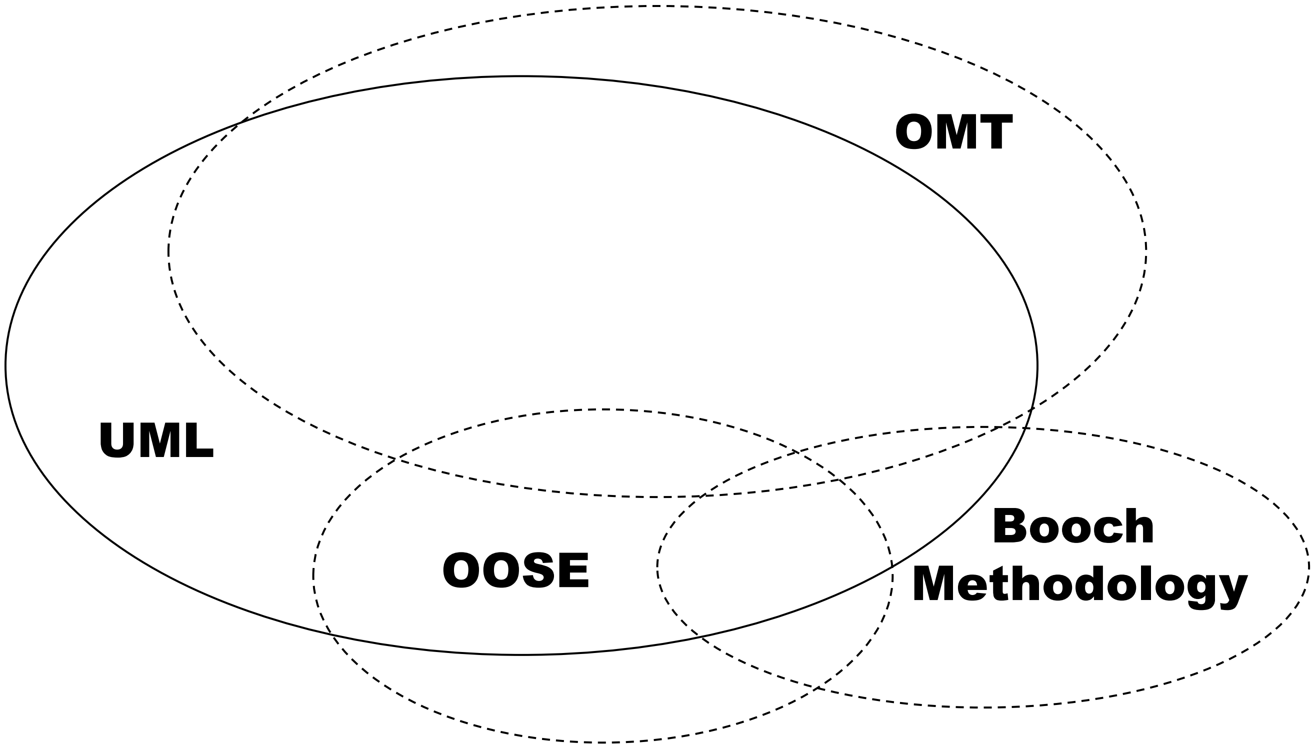
# Unified Modelling Language (UML)

- Origin
  - In late 1980s and early 1990s different software development houses were using different notations
  - Developed in early 1990s to standardize the large number of object-oriented modelling notations

# UML

- Based Principally on
  - OMT [Rumbaugh 1991]
  - Booch’s methodology[Booch 1991]
  - OOSE [Jacobson 1992]
  - Odell’s methodology[Odell 1992]
  - Shlaer and Mellor [Shlaer 1992]

# UML



**Different object modelling techniques in UML**

# UML

- As a Standard
  - Adopted by Object Management Group (OMG) in 1997
  - OMG an association of industries
  - Promote consensus notations and techniques
  - Used outside software development, example car manufacturing

# Why UML is required?

- Model is required to capture only important aspects
- UML a graphical modelling tool, easy to understand and construct
- Helps in managing complexity

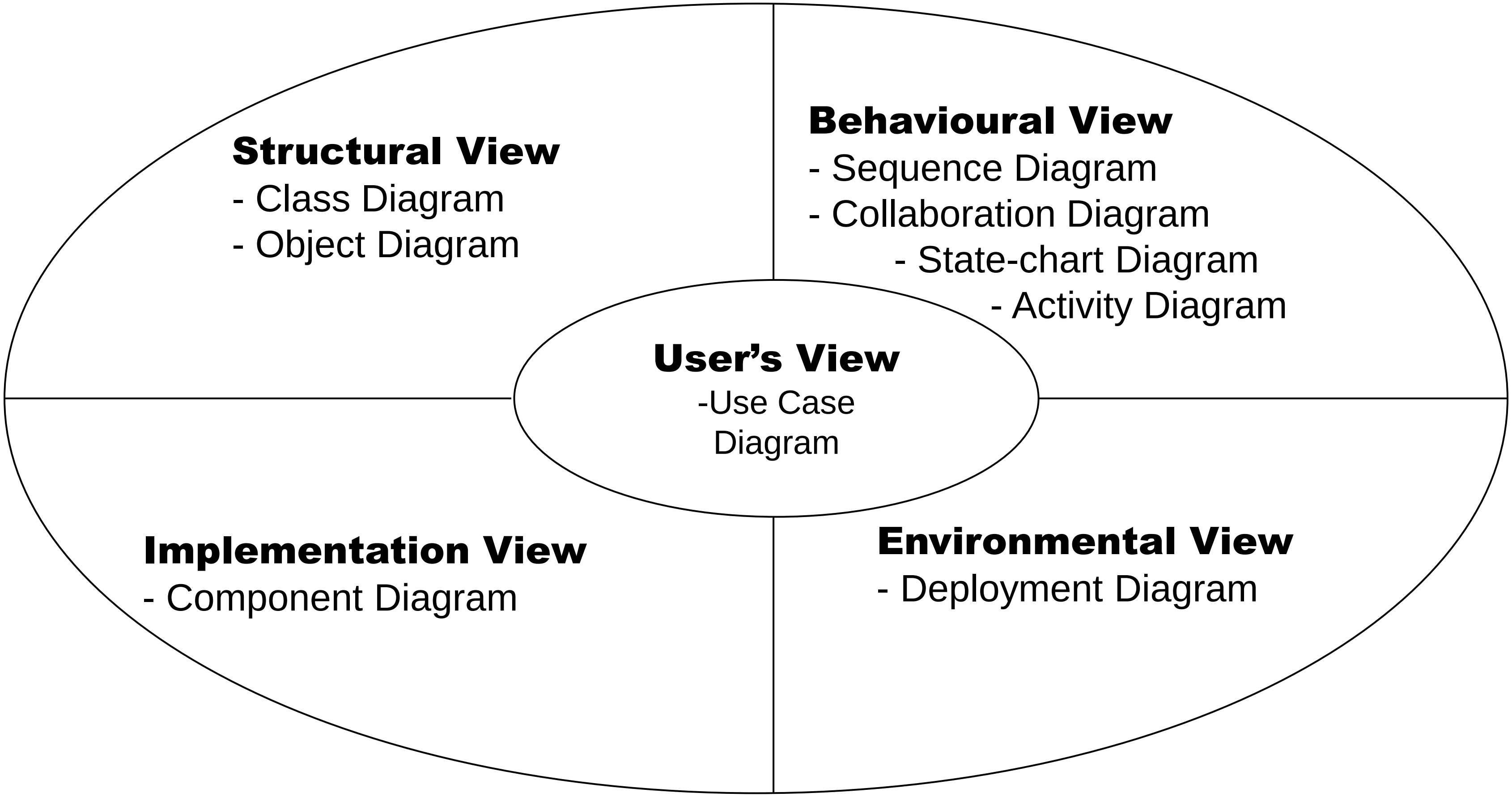
# UML diagrams

- Nine diagrams to capture different views of a system
- Provide different perspectives of the software system
- Diagrams can be refined to get the actual implementation of the system

# UML diagrams

- Views of a system
  - User’s view
  - Structural view
  - Behavioral view
  - Implementation view
  - Environmental view

# UML diagrams



Diagrams and views in UML

# UML diagrams

- User's view
  - It captures the view of the system in terms of the functionalities offered by the system to its user
  - It is a black-box view of the system
  - Dynamic behavior of the components, the implementation etc are not captured.

# UML diagrams

- Structural view
  - Defines the structure of the problem (or the solution) in terms of the objects (or classes) important to the understanding of the working of a system and its implementation.
  - It captures the relationship among the classes ( or objects)
  - Called the static model, since the structure of a system does not change with time.

# UML diagrams

- Behavioural view
  - Captures how objects interact with each other in time to realize the system behavior.
  - System behavior captures the time-dependent (dynamic) behavior.
  - It constitutes the dynamic model of the system

# UML diagrams

- Implementation view
  - Captures important components of the system and their interdependencies.
  - Example, implementation view might show the GUI part, the middleware, and the database parts and would capture their interdependencies.

# UML diagrams

- Environmental view
  - This view models how the different components are implemented on different pieces of hardware.

# Are all views required?

- NO
- Use case model, class diagram and one of the interaction diagram for a simple system
- State chart diagram in case of many state changes
- Deployment diagram in case of large number of hardware components

# Use Case model

- Consists of set of “use cases”
- An important analysis and design artifact
- Other models must confirm to this model
- Not really an object-oriented model
- Represents a functional or process model

# Use Cases

- Different ways in which system can be used by the users
- Corresponds to the high-level requirements
- Represents transaction between the user and the system
- Define behavior without revealing internal structure of system
- Set of related scenarios tied together by a common goal

# Use Cases

- Normally, use cases are independent of each other
- Implicit dependencies may exist
- Example: In Library Automation System, renew-book & reserve-book are independent use cases. But in actual implementation of renew-book, a check is made to see if any book has been reserved using reserve-book

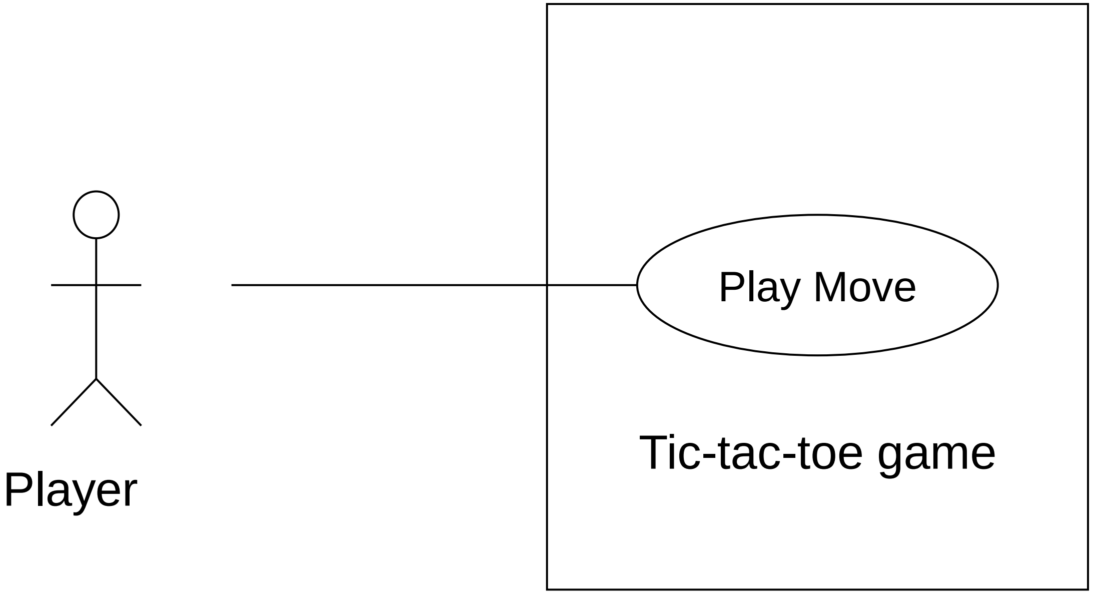
# Example of Use Cases

- For library information system
  - issue-book
  - Query-book
  - Return-book
  - Create-member
  - Add-book, etc.

# Representation of Use Cases

- Represented by use case diagram
- Use case is represented by ellipse
- System boundary is represented by rectangle
- Users are represented by stick person icon (actor)
- Communication relationship between actor and use case by line
- External system by stereotype

# Example of Use Cases



Use case model

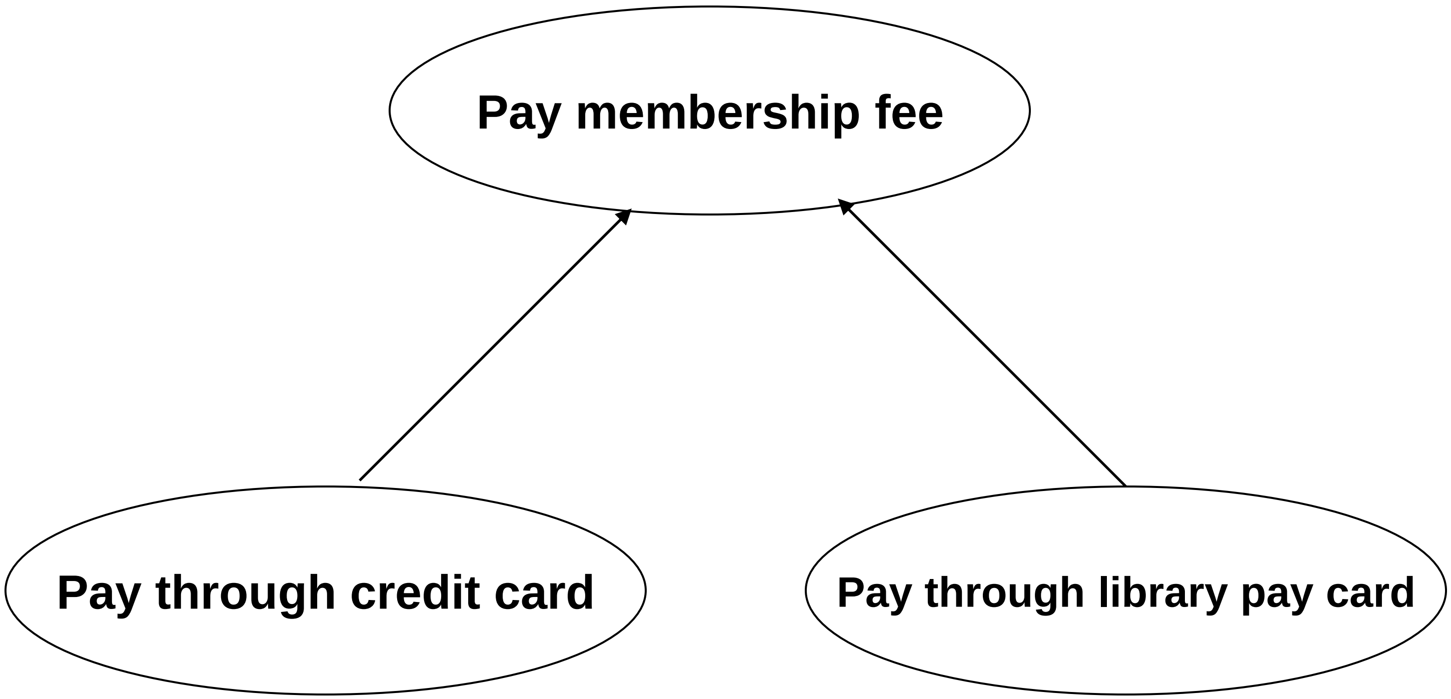
# Why develop Use Case diagram?

- Serves as requirements specification
- Users identification helps in implementing security mechanism through login system
- Another use in preparing the documents (e.g. user’s manual)

# Factoring Use Cases

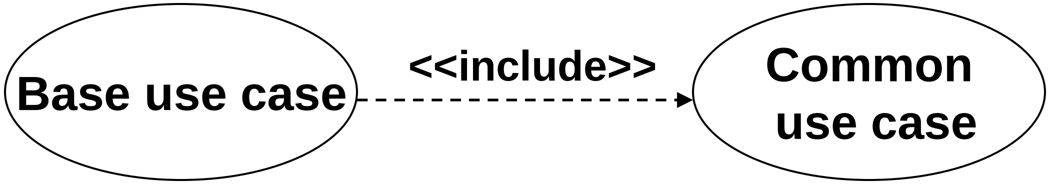
- Complex use cases need to be factored into simpler use cases
- Represent common behavior across different use cases
- Three ways of factoring
  - Generalization
  - Includes
  - Extends

# Factoring Using Generalization



**Use case generalization**

# Factoring Using Includes

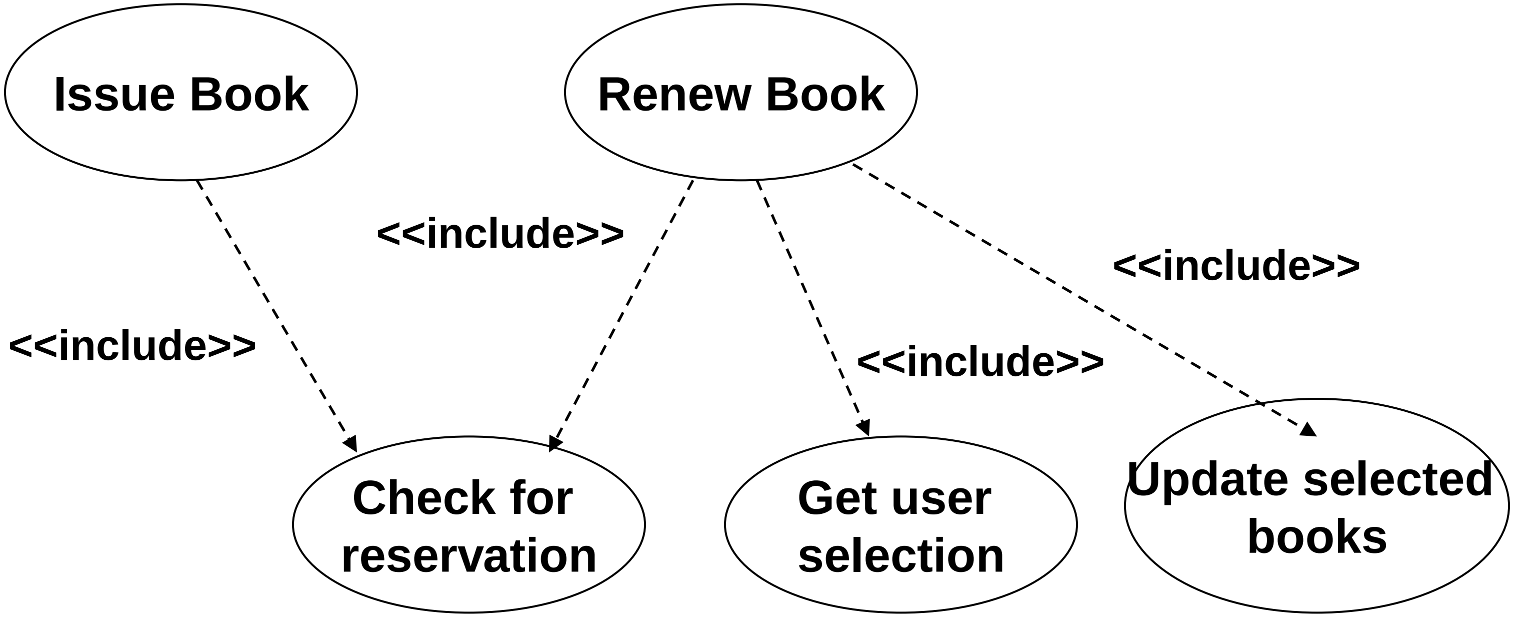


Base use case

<<include>>

Common use case

**Use case inclusion**



Issue Book

Renew Book

<<include>>

<<include>>

<<include>>

<<include>>

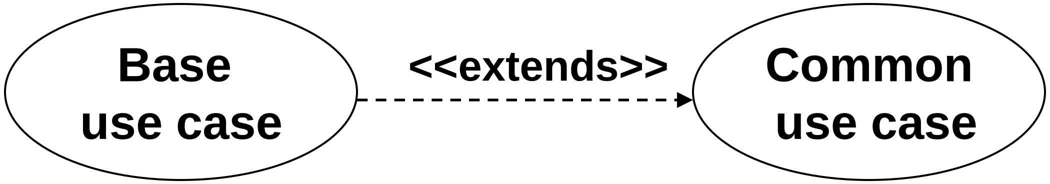
Check for reservation

Get user selection

Update selected books

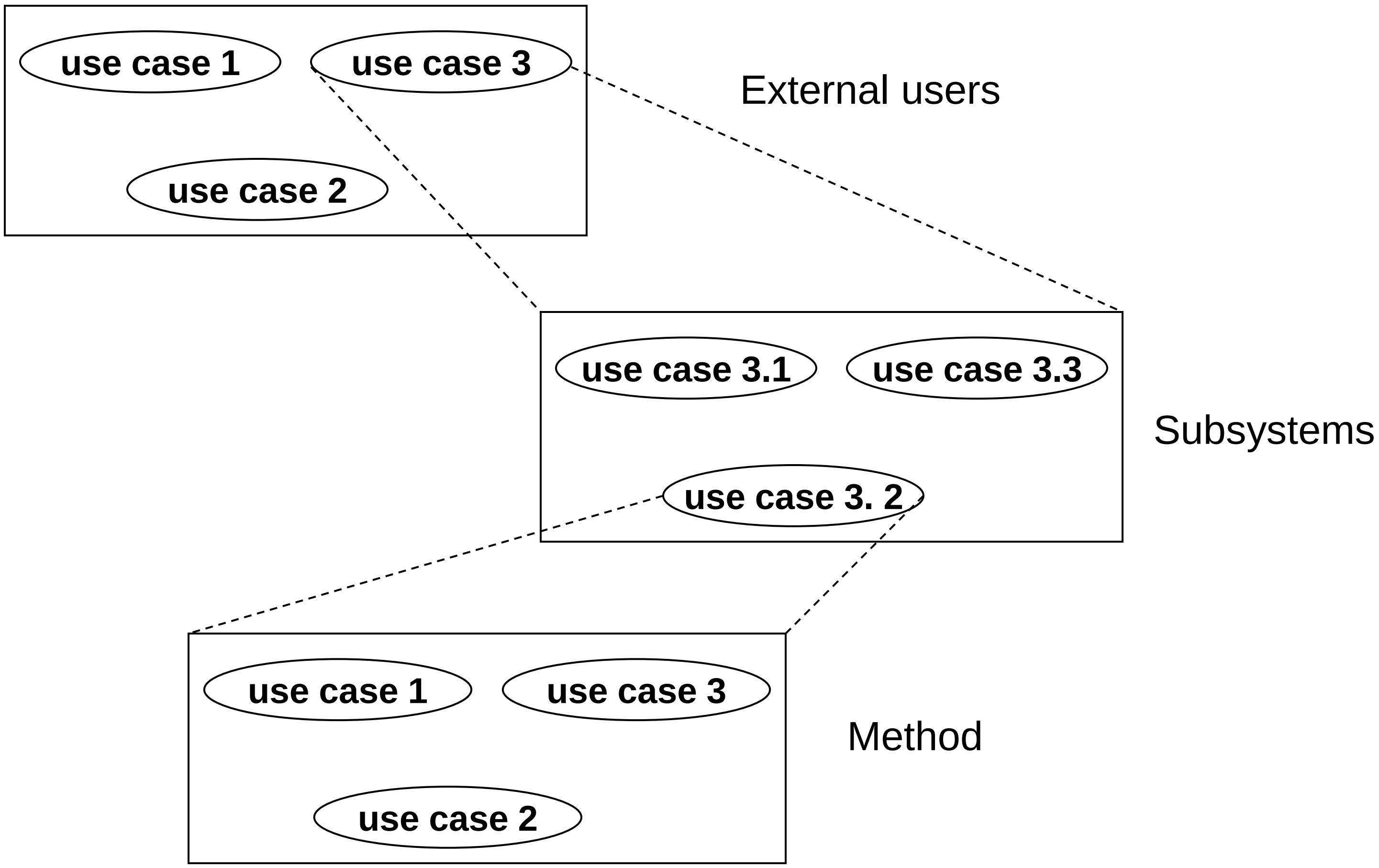
**example**

# Factoring Using Extends



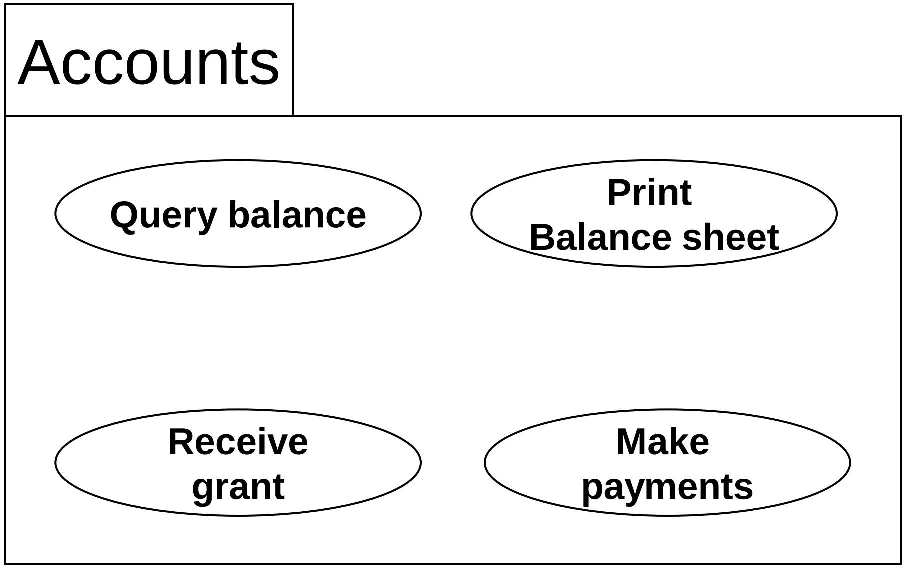
**Use case extension**

# Hierarchical Organization of Use Cases



**Hierarchical organization of use cases**

# Use Case Packaging



**Use case packaging**