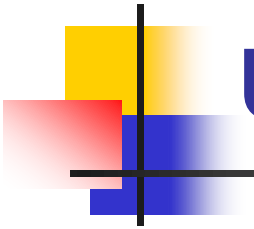
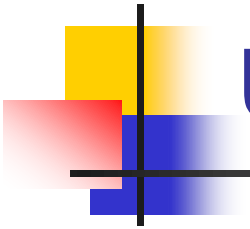
8. Dynamic Modeling using the Unified Modeling Language (UML) - **Use case**



Use case

"A *use case* specifies the behavior of a system or a part of a system, and is a description of a set of sequences of actions, that a system performs to yield an observable result of value to an actor."

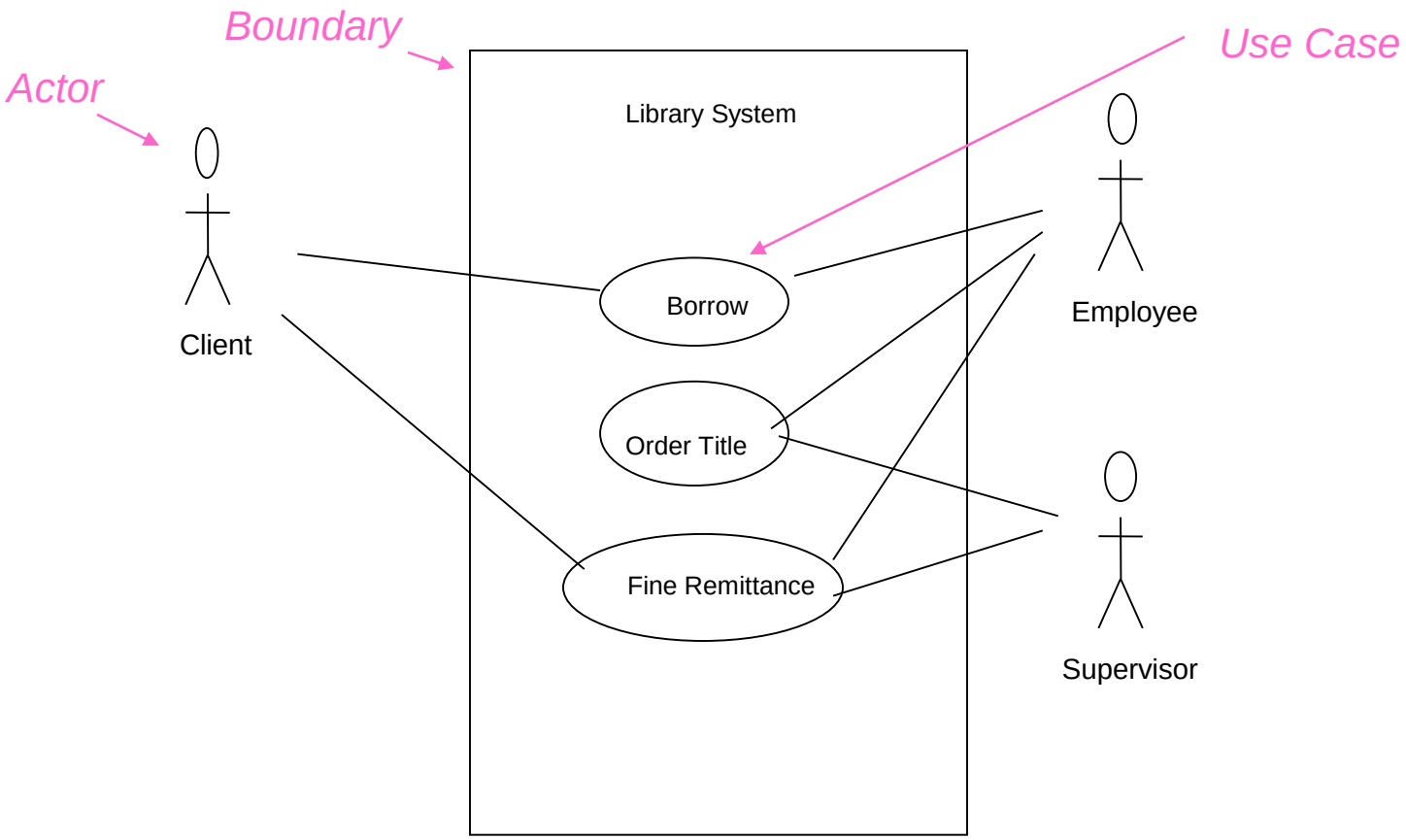
- *The UML User Guide, [Booch,99]*



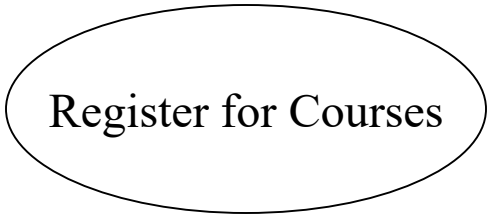
Use-Case Diagrams

- A use-case diagram is a set of use cases
- A use case is a model of the interaction between
  - External users of a software product (actors) and
  - The software product itself
  - More precisely, an actor is a user playing a specific role
- describing a set of user **scenarios**
- capturing user requirements

Use-Case Diagrams

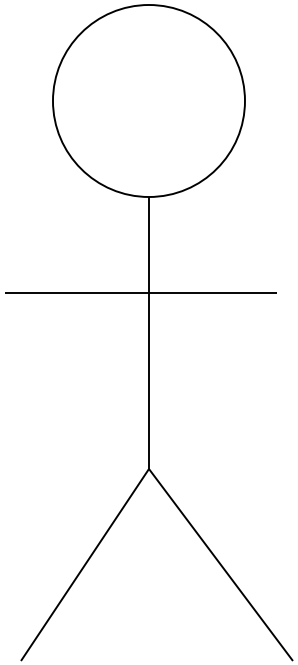


Use case



A use case is rendered as an eclipse in a use case diagram. A use case is always labeled with its name.

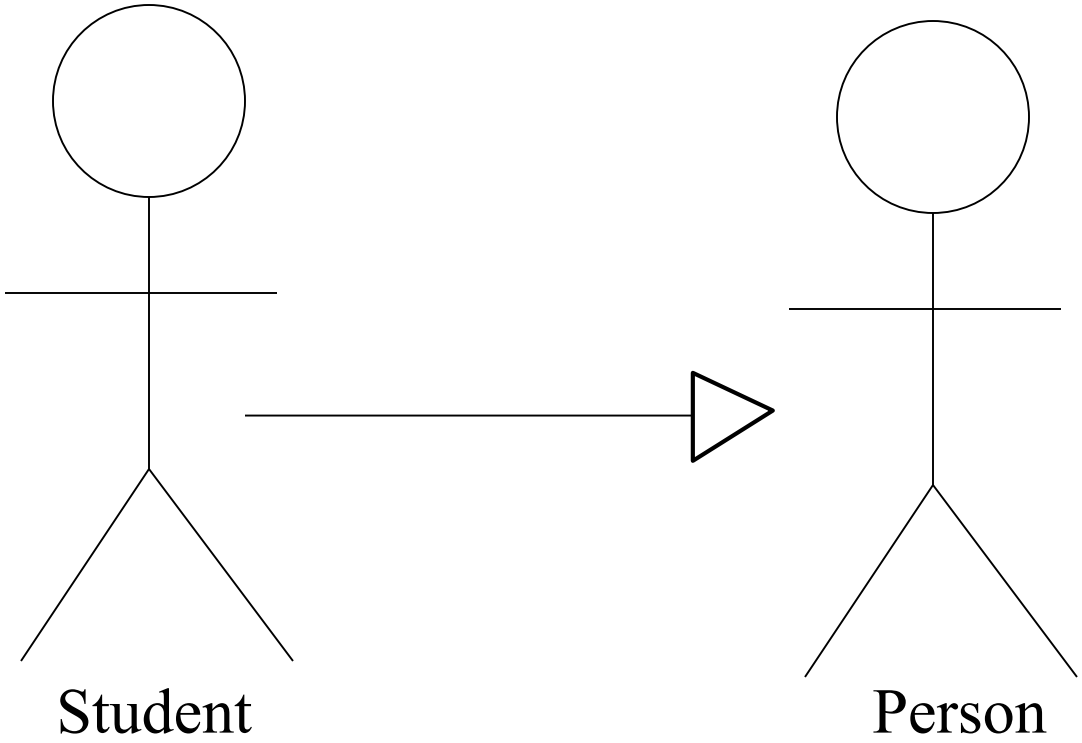
Use case



An actor is rendered as a stick figure in a use case diagram. Each actor participates in one or more use cases.

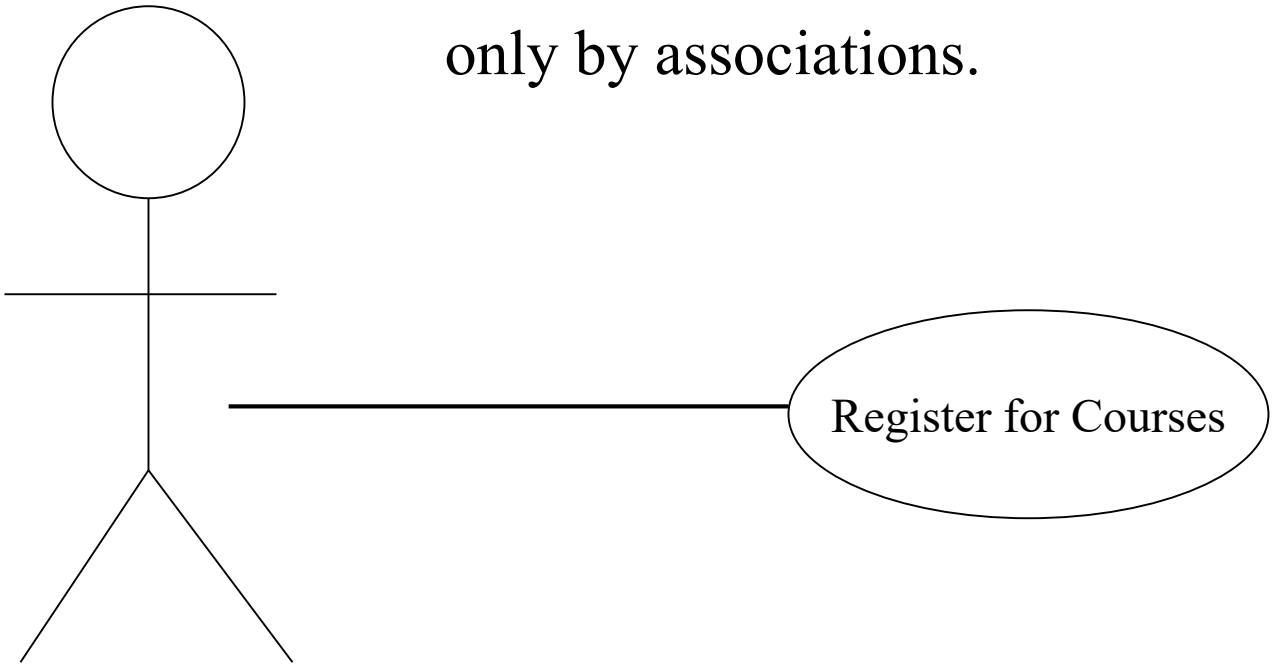
Use case

Actors can participate in a generalization relation with other actors.



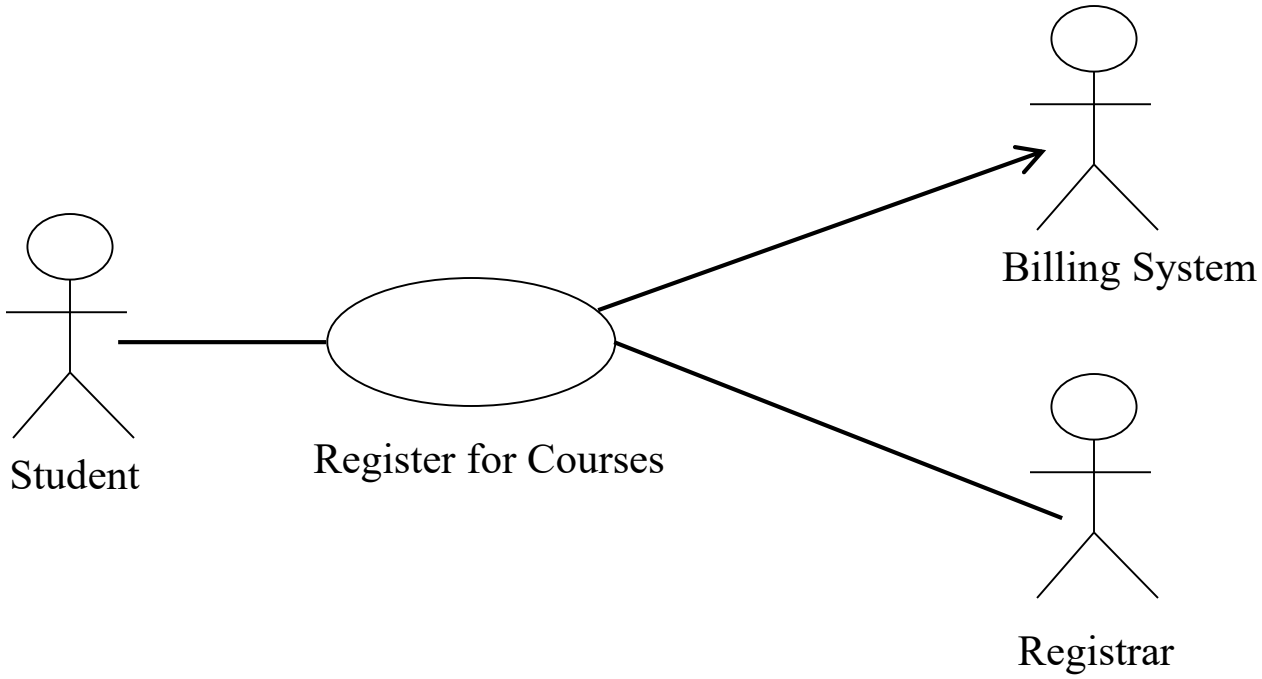
Use case

Actors may be connected to use cases only by associations.

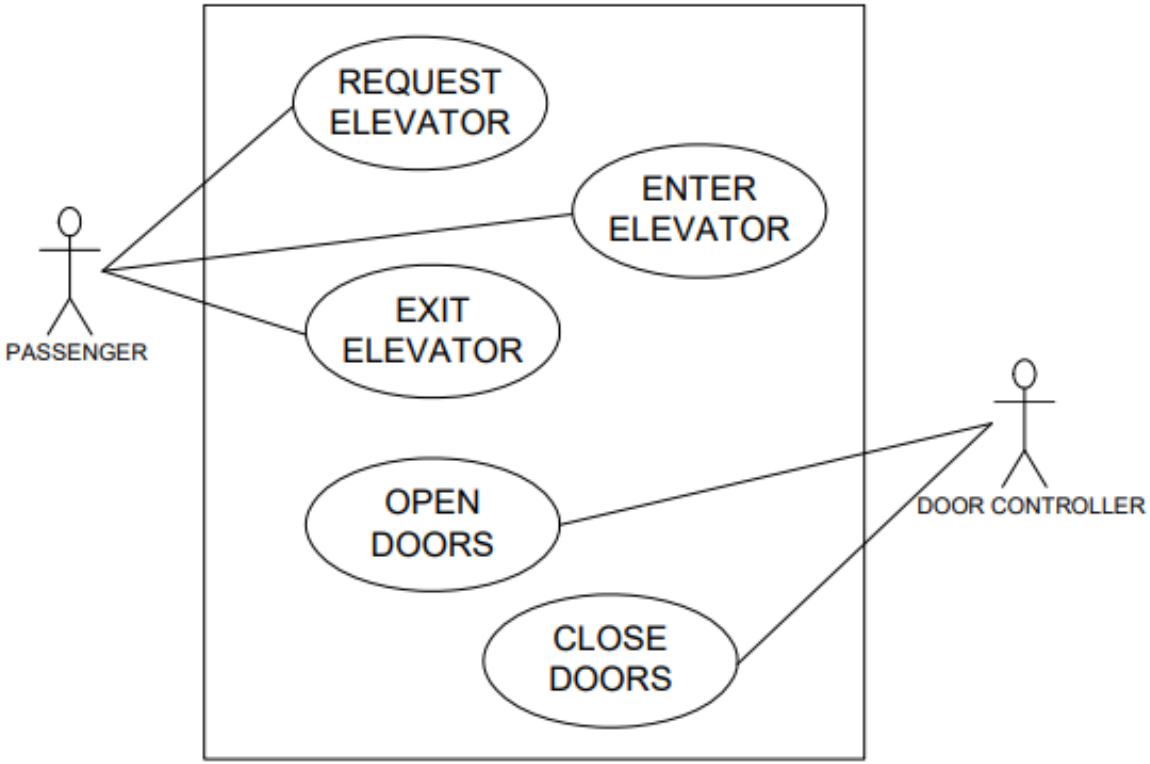


Use case

Here we have a *Student* interacting with the *Registrar* and the *Billing System* via a "*Register for Courses*" use case.



Use case - Example

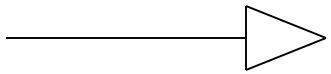


Use-Case Diagrams

- **Association:**

communication between an actor and a use case; Represented by a solid line.

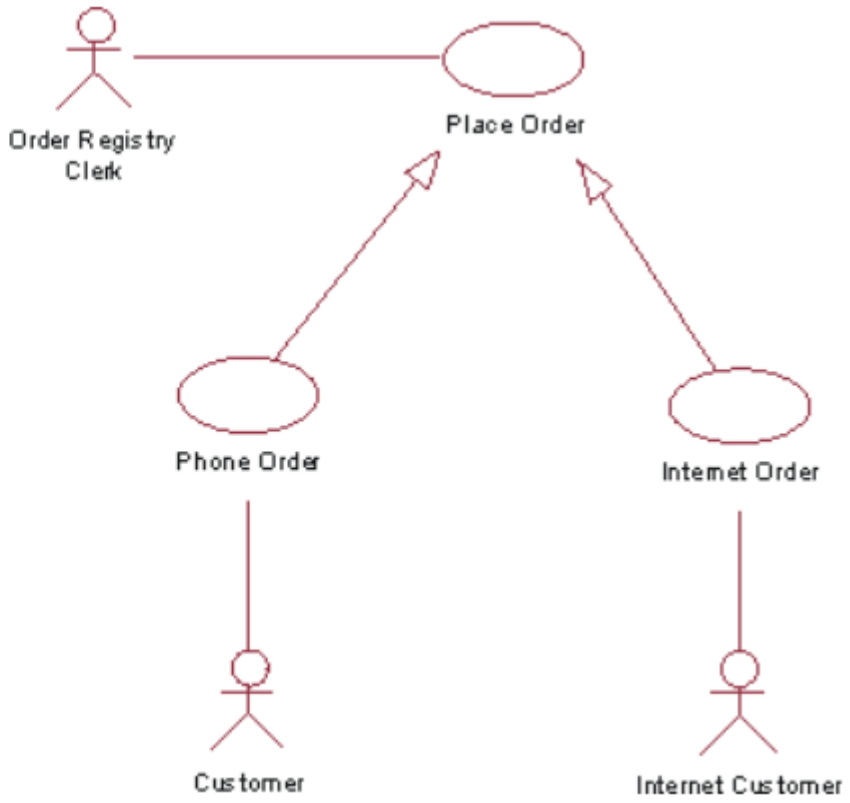
- **Generalization:** relationship between one general use case and a special use case (used for defining special alternatives) Represented by a line with a triangular arrow head toward the parent use case.



Use-Case Diagrams

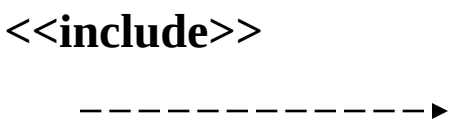
The actor Order Registry Clerk can instantiate the general use case Place Order.

Place Order can also be specialized by the use cases Phone Order or Internet Order.



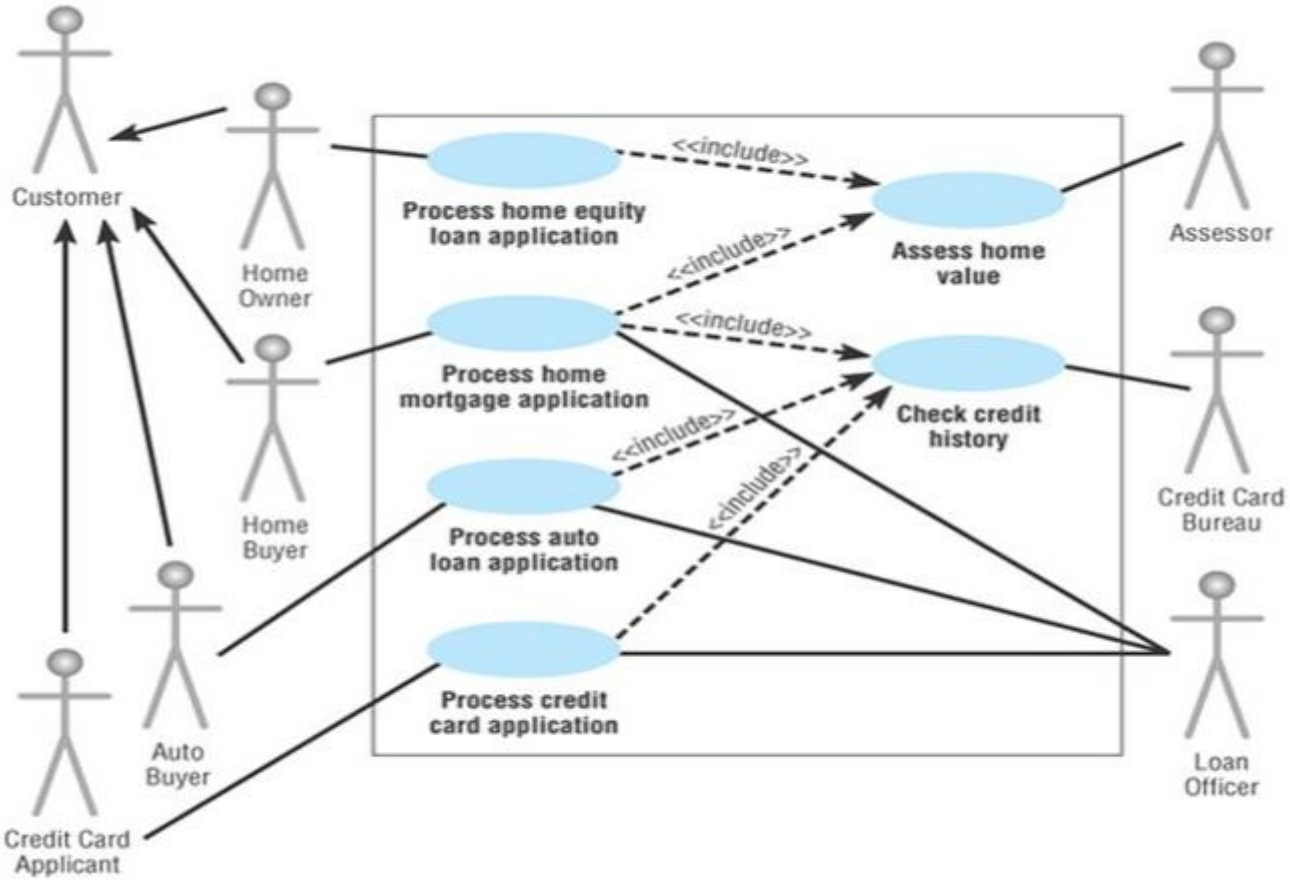
Use-Case Diagrams

Include: a dotted line labeled <<include>> beginning at base use case and ending with an arrows pointing to the include use case. The include relationship occurs when a chunk of behavior is similar across more than one use case. Use “include” in stead of copying the description of that behavior.



Extend: a dotted line labeled <<extend>> with an arrow toward the base case. The extending use case may add behavior to the base use case. The base class declares “extension points”.

Use-Case Diagrams



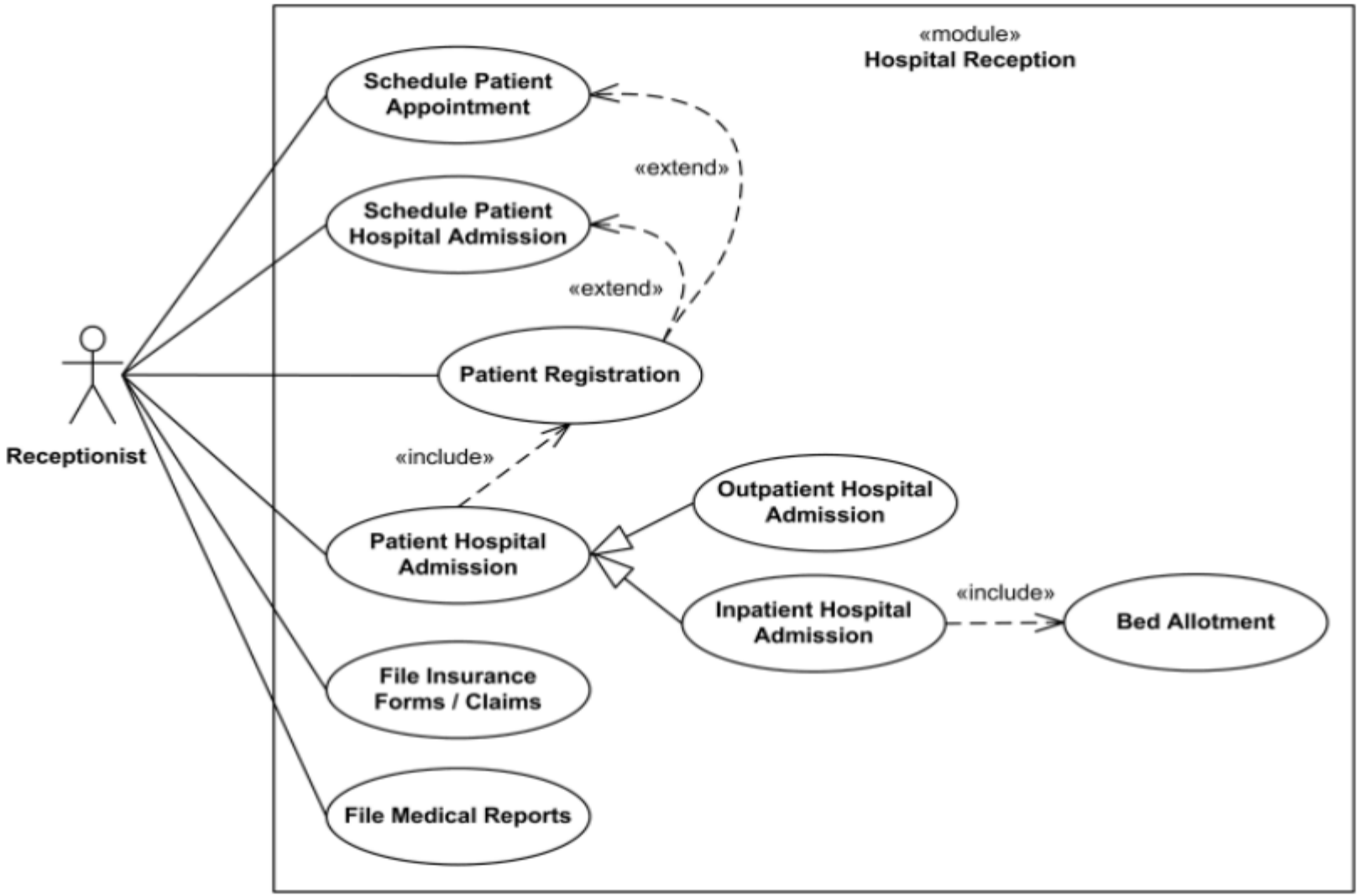
Use-Case Diagrams

Product Page

Shopping Cart

Review Writing

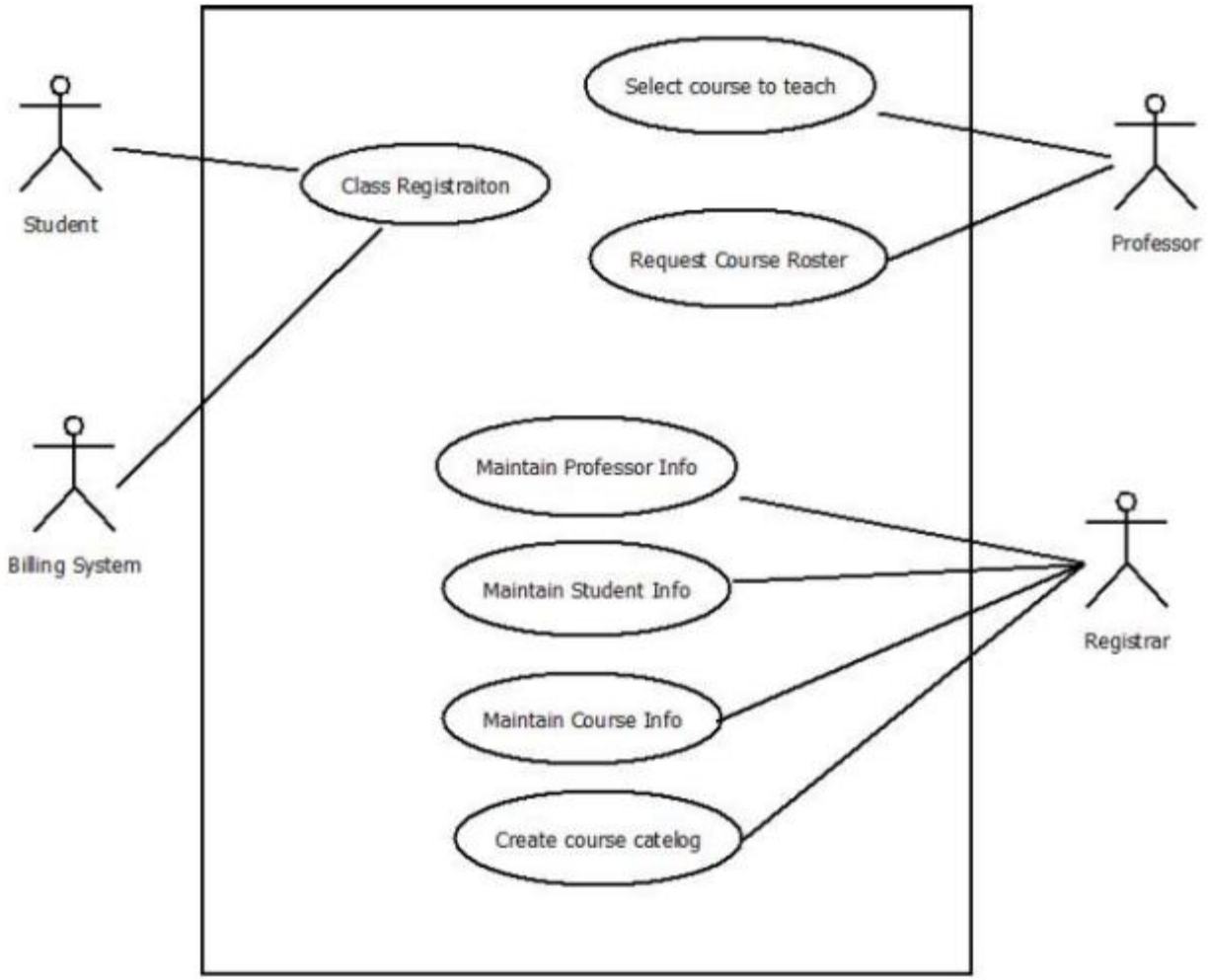
Use-Case Diagrams



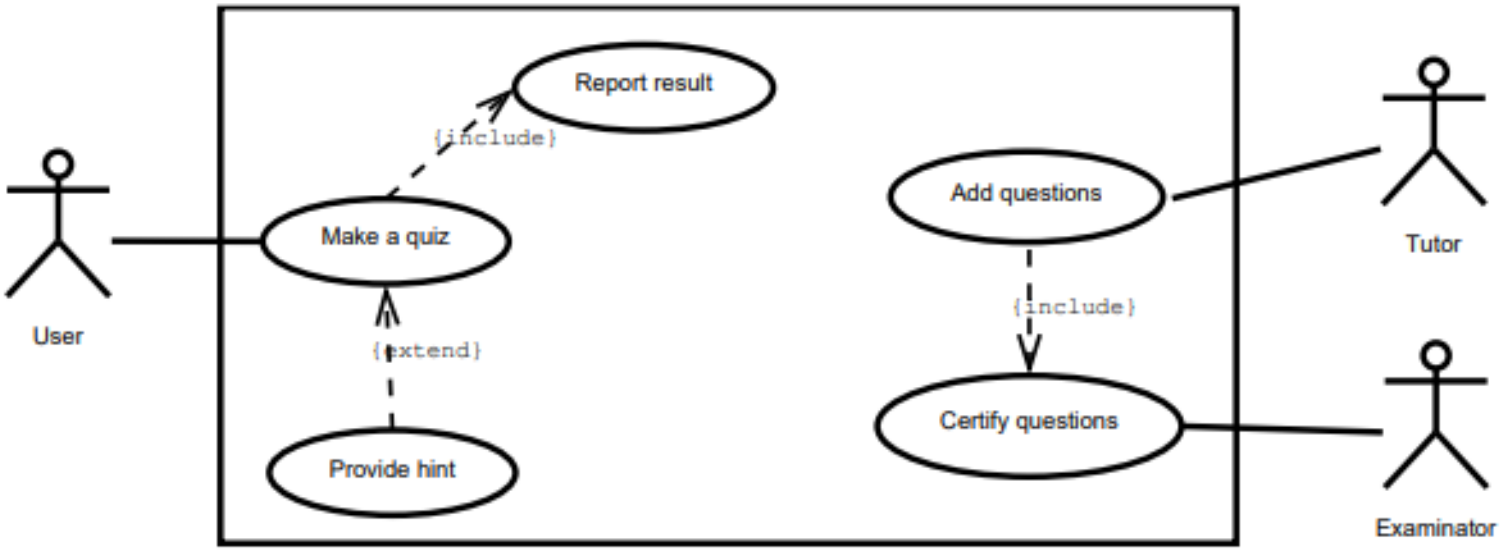
Use-Case Diagrams – Example #1

1. Professors indicate which courses they will teach on-line.
2. A course catalog can be printed
3. Allow students to select on-line four courses for upcoming semester.
4. No course may have more than 10 students or less than 3 students.
5. When the registration is completed, the system sends information to the billing system.
6. Professors can obtain course rosters on-line.
7. Students can add or drop classes on-line.

Use-Case Diagrams – Example #1



Use-Case Diagrams – Example #2



Use-Case Diagrams – Example #2

Use case: Make quiz.
Primary actor: User
Secondary actors: -
Pre-condition: The system has at least 10 questions.
Post-condition: -
Main flow:
1. The use-case is activated when the user requests it.
2. The user specifies the difficulty level.
3. The system selects 10 questions, and offers them as a quiz to the user.
4. The system starts a timer.
5. For every question:
   5a. The user selects an answer, or skip. [Extension point]
6. If the user is done with the quiz, or the timer runs out, the quiz is concluded, and [include use case 'Report result'].

Use-Case Diagrams – Example #2

Use case: Provide hint

Primary actor: User

Secondary actors: -

Pre-condition: The user requests for a hint.

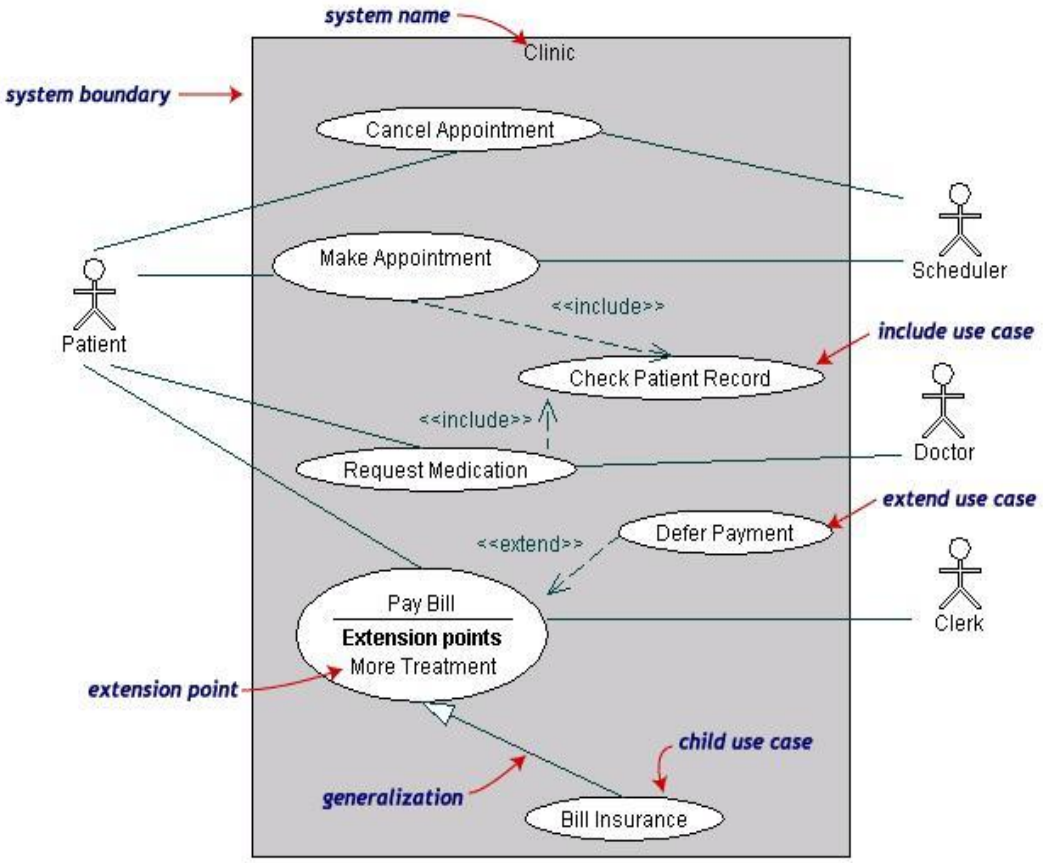
Post-condition: -

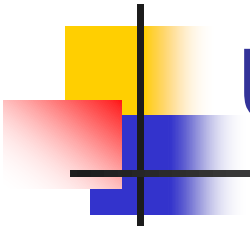
Main flow:

1. The system provides a hint. The verbosity of the hint is determined by the difficulty level set previously by the user.
2. Return to to Make quiz' main flow.

Use-Case Diagrams – Example #3

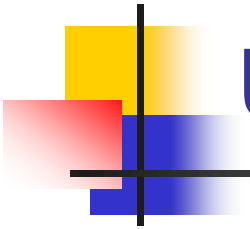
- Both **Make Appointment** and **Request Medication** include **Check Patient Record** as a subtask (include)
- The **extension point** is written inside the base case **Pay bill**; the extending class **Defer payment** adds the behavior of this extension point. (extend)
- **Pay Bill** is a parent use case and **Bill Insurance** is the child use case. (generalization)





Use-Case Diagrams

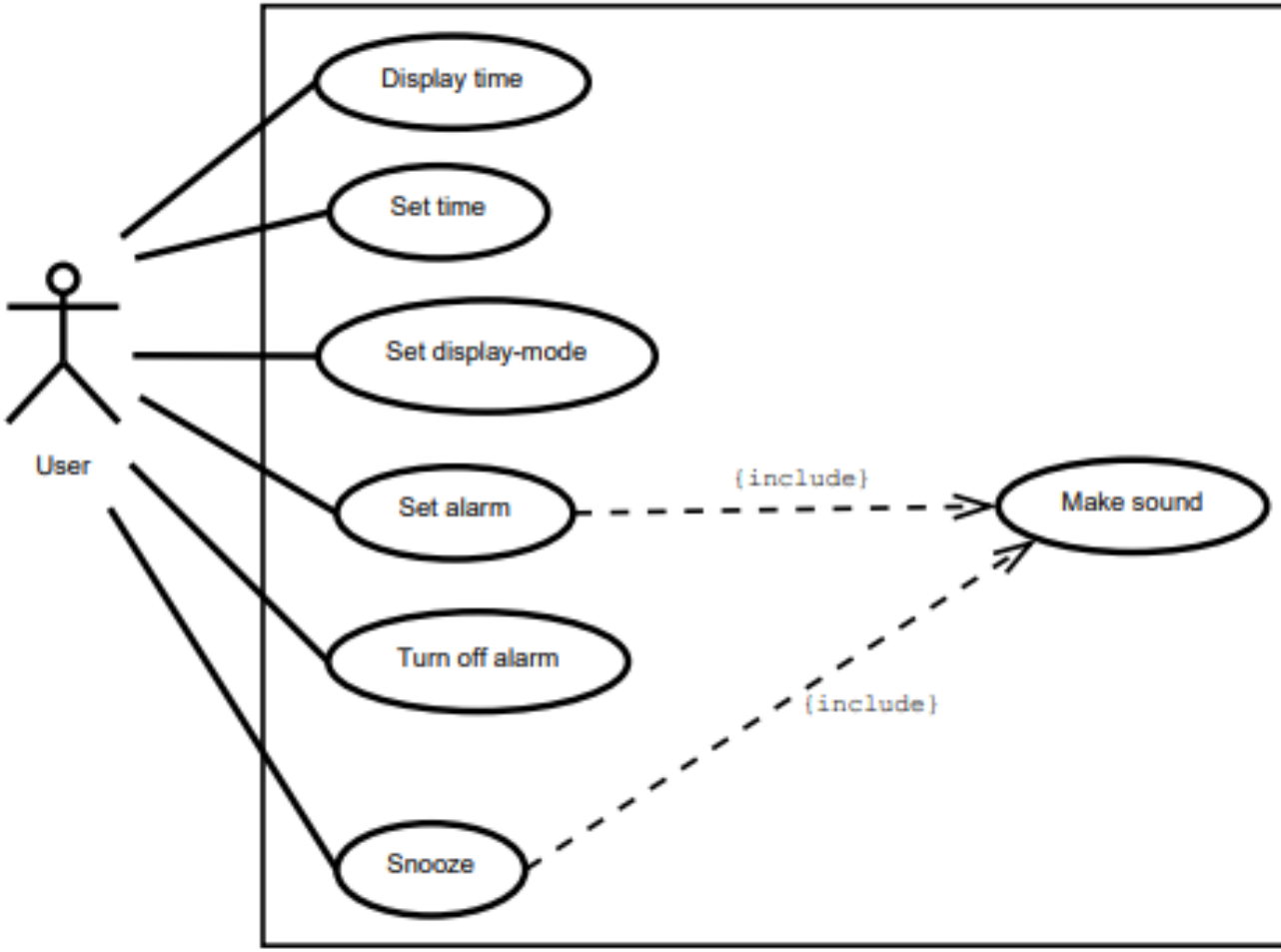
- The purposes of use case diagrams can be said to be as follows
  - Used to gather the requirements of a system.
  - Used to get an outside view of a system.
  - Identify the external and internal factors influencing the system.
  - Show the interaction among the requirements are actors.

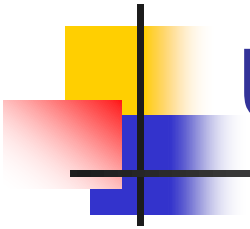


Use-Case Diagrams

- Suppose we want to develop software for an alarm clock.
  - The clock shows the time of day. Using buttons, the user can set the hours and minutes fields individually, and choose between 12 and 24-hour display.
  - It is possible to set one or two alarms. When an alarm fires, it will sound some noise. The user can turn it off, or choose to 'snooze'. If the user does not respond at all, the alarm will turn off itself after 2 minutes. 'Snoozing' means to turn off the sound, but the alarm will fire again after some minutes of delay. This 'snoozing time' is pre-adjustable.
- Identify the top-level functional requirement for the clock, and model it with a **use case diagram**

Use-Case Diagrams





Use-Case Diagrams

Use case: Snooze.
Primary actor: User
Secondary actors: -
Pre-condition: An alarm is firing.
Post-condition: -
Main flow:
1. The use-case is activated when the user hits the snooze button.
2. The alarm is turned off.
3. Wait for snooze time.
4. Include use case 'Make sound'