CSC 2210
Object Oriented Analysis & Design

Dr. Akinul Islam Jony
Associate Professor
Department of Computer Science, FSIT
American International University - Bangladesh (AIUB)

akinul@aiub.edu

Use Case Diagram

>> What is Use Case?
>> Use Case Diagram
>> Elements of Use Case Diagram
>> Use Case Relationships
>> Use case Diagram Examples
>> Use case Diagram Exercises

What is Use Case?

>> Use cases specify externally visible behavior; they do not dictate how that behavior will be carried out internally.

>> For example, you can specify how an ATM system should behave by stating in **use cases** how users interact with the system; you don't need to know anything about the inside of the ATM at all.

>> A **use case** is a description of a set of **sequences of actions**, including variants, that a system/subject performs to yield an observable result of value to an actor. Each **sequence** represent an **interaction** of **actors** with the **system**.

>> A use case involves the interaction of actors and the system or other subject.

>> It represents a functional requirement of your system. It can be applied to a whole system or to part of a system, including subsystems and even individual classes and interfaces.

What is Actor?

>> An **actor** represents a coherent set of roles that users of use cases play when interacting with these use cases.

>> **Actors** can be human or they can be automated systems. **For example**, in modeling a bank, processing a loan involves, among other things, the interaction between a customer and a loan officer.

>> **Actors are not part of the system**.

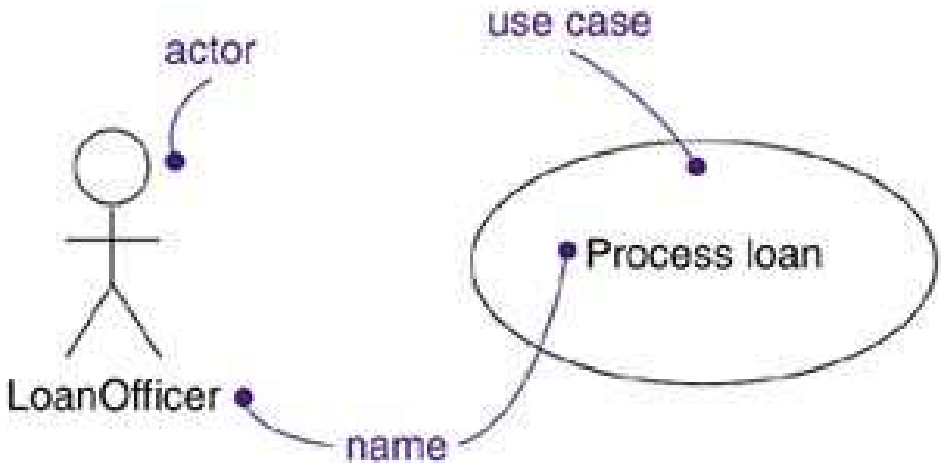


Figure 3-1. Actor and Use Case

Use Case diagram

>> **Use case diagrams** are central to **modeling the behavior** of a system, a subsystem, or a class.

>> Use case diagrams contain a set of **use cases** and **actors** and their **relationships**.

>> Use case diagrams are applied to model the **use case view of a system**.

>> **Use case diagrams** are important for **visualizing**, **specifying**, **constructing** and **documenting** the **behavior** of an element.

>> Use case diagrams are also important for **testing** executable systems through forward engineering and for comprehending executable systems through reverse engineering.

Use Case diagram

>> With the UML, you apply use case diagrams to visualize the behavior of a system, subsystem, or class so that users can comprehend how to use that element, and so that developers can implement that element.

>> For example, you can provide a use case diagram to model the behavior of a cellular phone, as shown in Figure 3-1

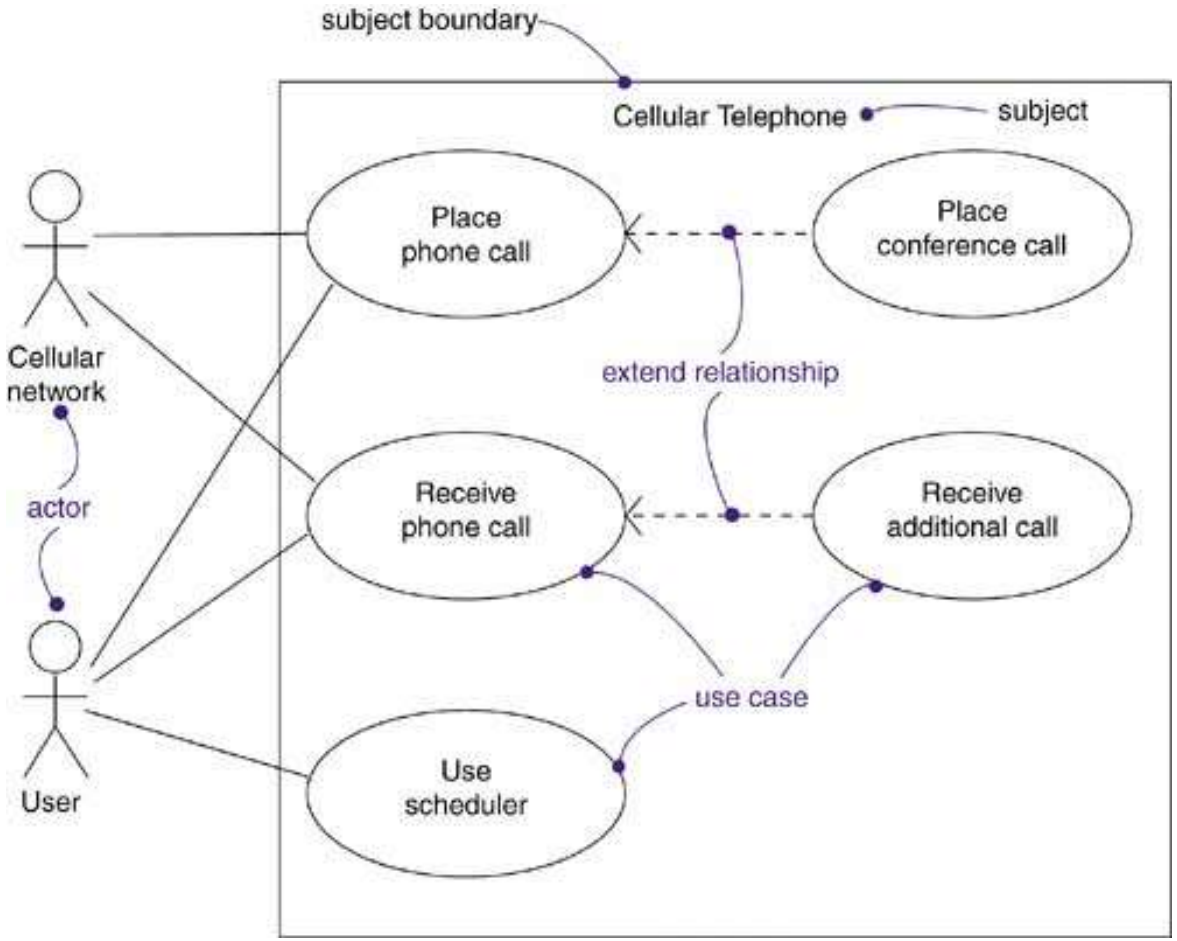
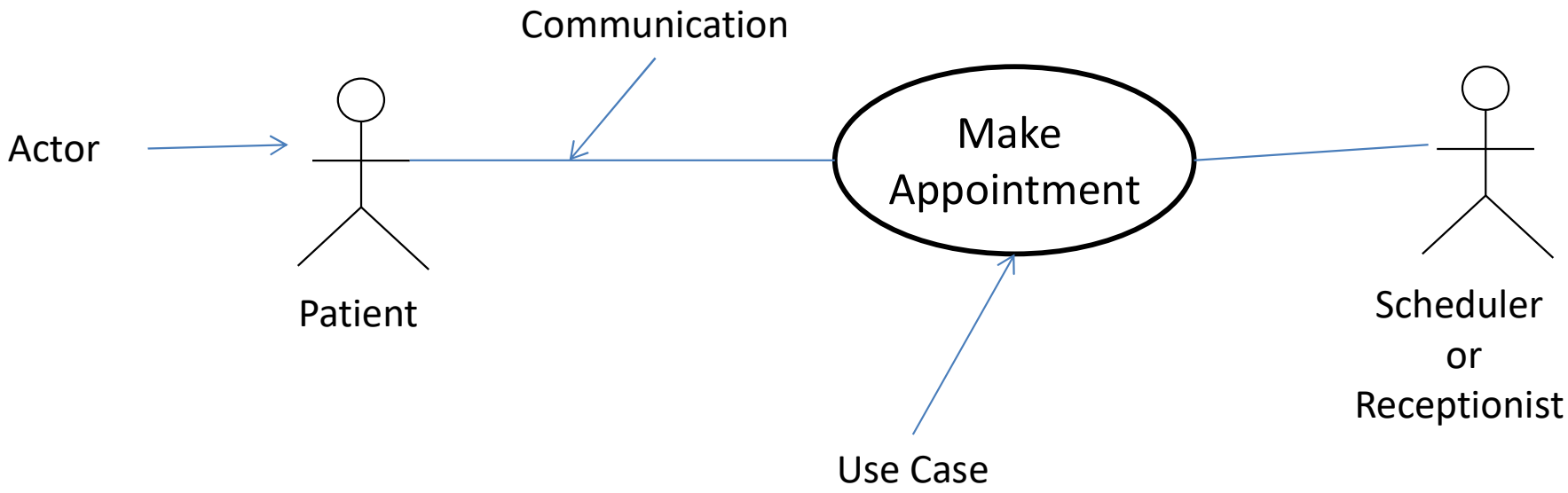


Figure 3-1. A Use Case Diagram

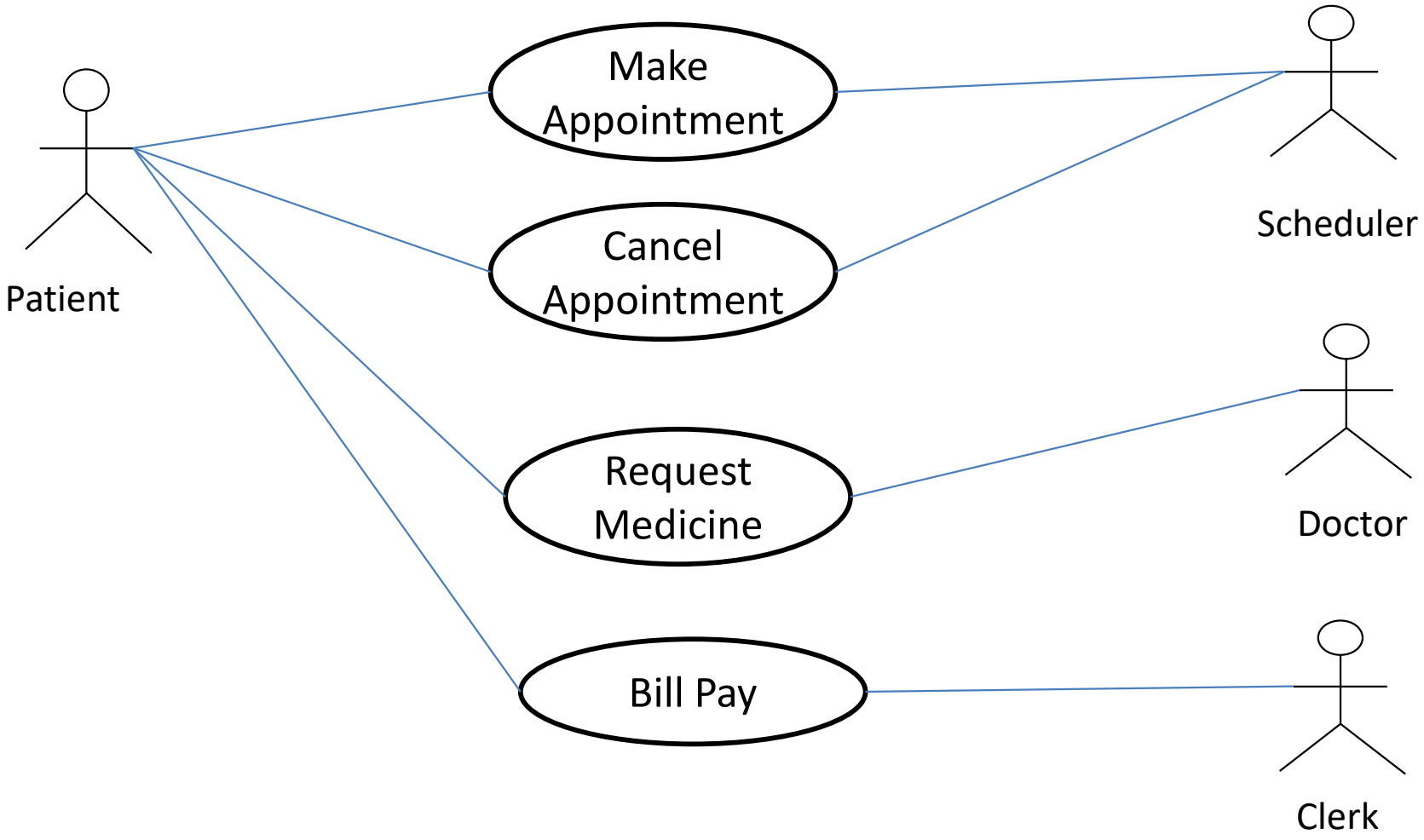
Use Case diagram

→ Here is a scenario for a medical clinic:
"A **patient** calls the clinic to make an **appointment** for a yearly checkup. The **receptionist** finds the nearest empty time slot in the appointment book and schedules the appointment for that time slot. "



Use Case diagram

>> More details of the Example:



Elements of Use Case Diagram

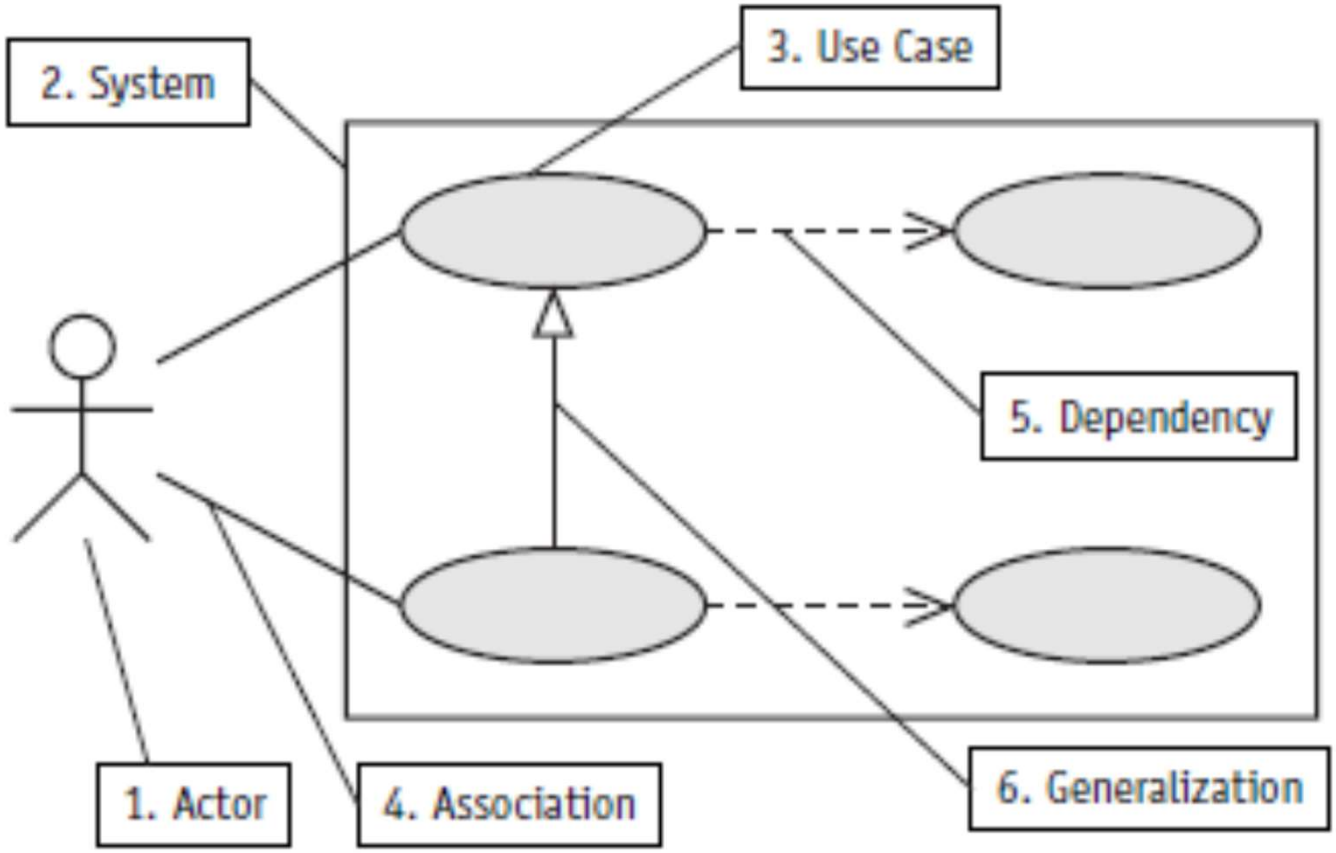


Figure 3-3 Elements of a Use Case diagram

Use Case System

>> A system is like an object, in that each has a purpose and an interface. The internal implementations of the object or system may be replaced or enhanced without affecting other entities as long as the purpose and the interfaces remain unchanged.

>> The interfaces are the channels of communication between the actors outside the system and the features of the system itself, the Use Cases.

>> System is defined as a set of requirements rather than a solution. You do not describe how the system must work. You describe what the system must be able to do.

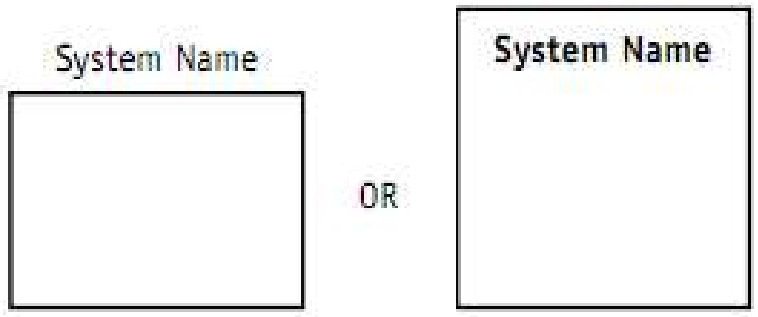


Figure 3-4 System icon for the Use Case diagram

Use Case Actors

>> Systems always have users. Users in the classic sense are people who use the system. But users can also be other systems or devices that trade information.

>> In Use Case diagrams, people, systems, and devices are all referred to as actors.

>> An actor is a role that an external entity plays in relation to the system.

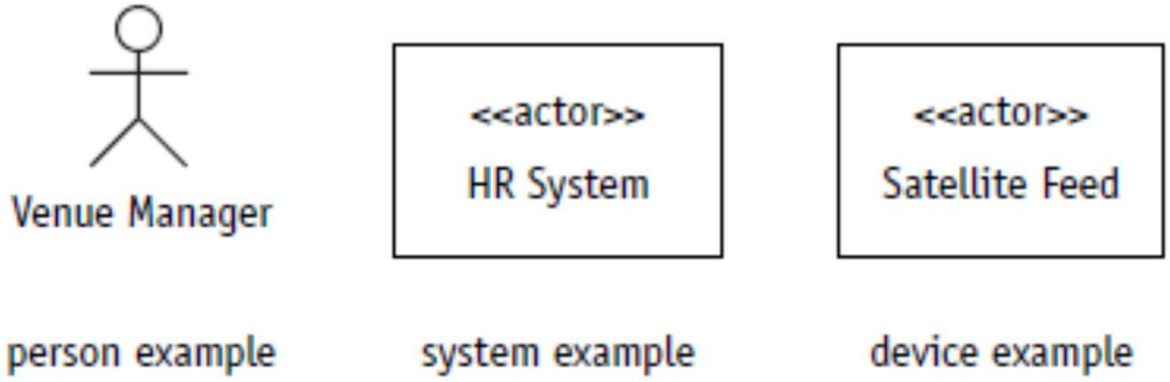


Figure 3-5 Actor icons for the Use Case diagram

Use Cases

>> Use Cases define the required features of the system. Without these features, the system cannot be used successfully.

>> Each Use Case is named using a verb phrase that expresses a goal the system must accomplish, for example, deposit money, withdraw money, and adjust account (see Figure 3-6).

>> The Use Cases describe only those features visible and meaningful to the actors who use the system

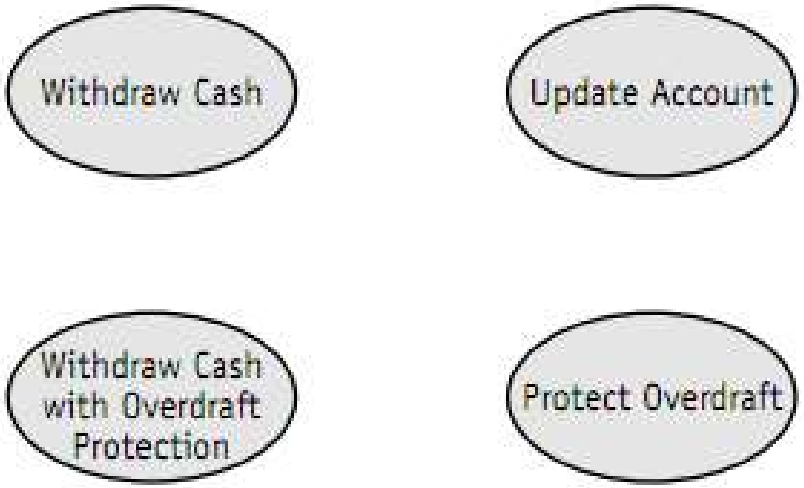


Figure 3-6 Use Case notation for the Use Case diagram

Use Case Relationships

>> Association
- –Between Actor and Use Case
- –**NOT** between Use Cases

>> Generalization
- –Between Actors
- –Between Use Cases

>> Dependency
- –Between Use Cases
- –<<include>> and <<extend>>

Association

>> A line connecting an actor to a Use Case represents an association, as shown in Figure 3-7.

>> The association represents the fact that the actor communicates with the Use Case.

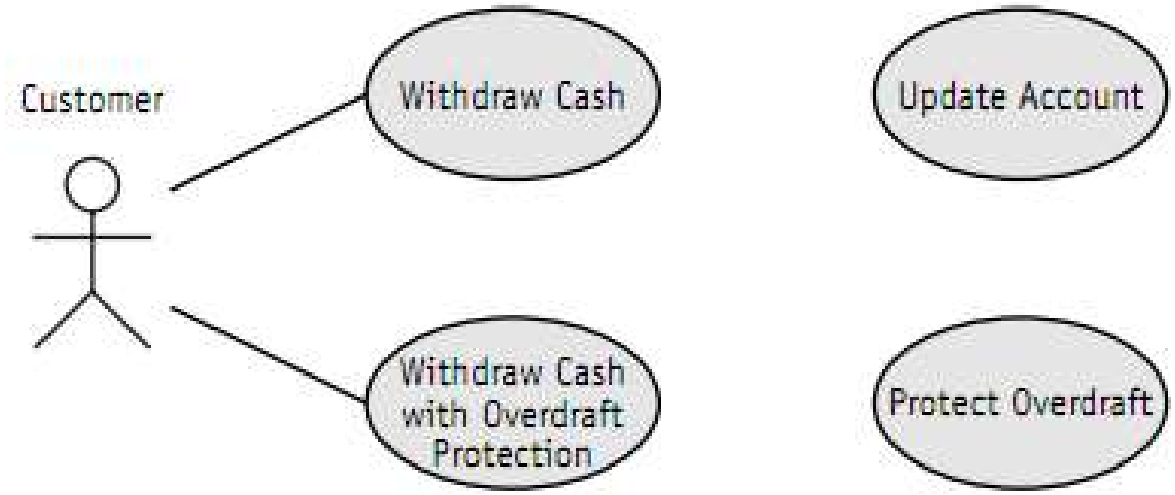


Figure 3-7 Association notation for the Use Case diagram

Dependency

→ **<<include>>**
- Mandatory dependency
- <<include>>
- Arrow towards the use case dependent on

→ **<<extend>>**
- Optional Dependency
- << extend >>
- Arrow from the use case dependent on
- Often referred as options of the use case

Dependency: **<<include>>**

>> When one Use Case delegates to another, the dependency is drawn as a dashed arrow from the “using” Use Case to the “used” Use Case and labeled with the <<include>> stereotype notation, as shown in figure 3-8.

>> Advantages: minimizes redundancy, shorter descriptions, and reuse

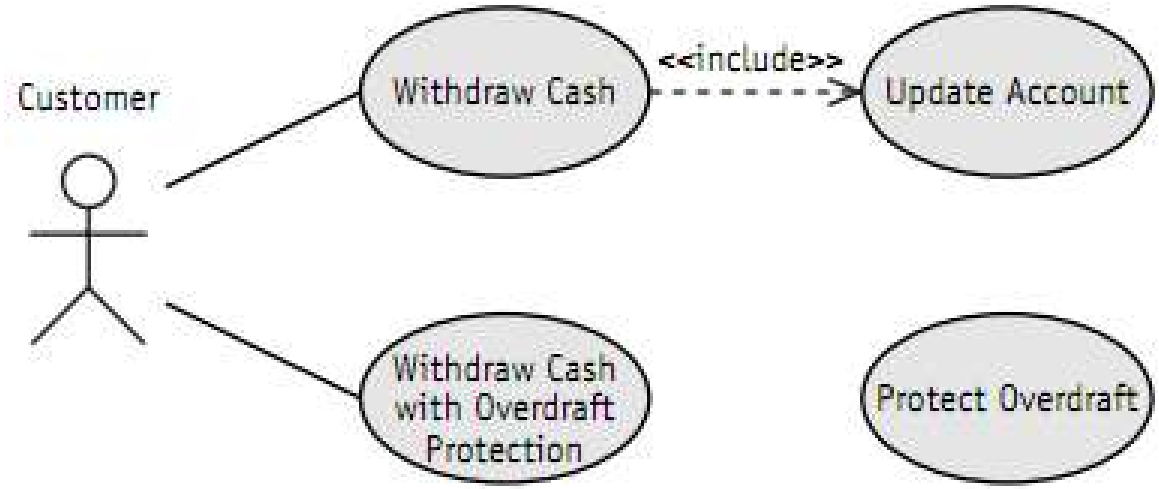
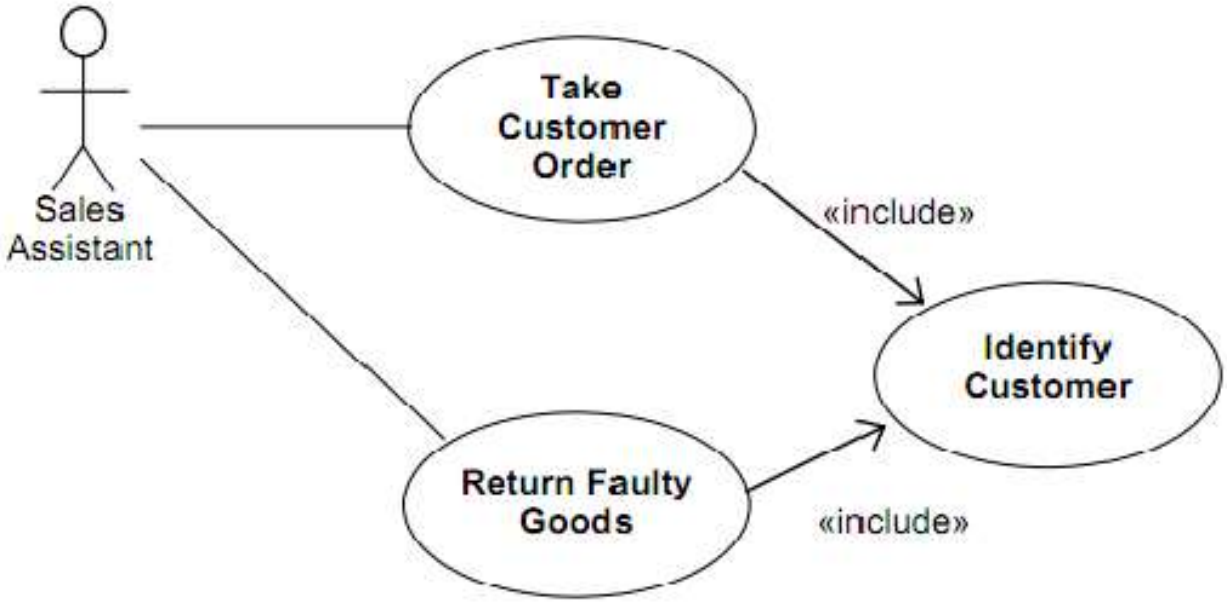


Figure 3-8 <<include>> dependency notation for the Use Case diagram

Dependency: **<<include>>**

>> Use case with **<<include>>** relationship says that one use case will always call the other use case.

>> Another example **<< include >>** based on scenario of product order:



Dependency: **<<extend>>**

>> The <<extend>> dependency stereotype says that one Use Case might need help from another Use Case.

>> The contrast between the two dependency stereotypes is the direction of the dependency arrow. The <<include>> dependency arrow points from the main Use Case (the one currently executing) to the one that it needs help from. The <<extend>> dependency arrow points from the extension Use Case (the one providing the extra help) to the main Use Case that it is helping (see Figure 3-9).

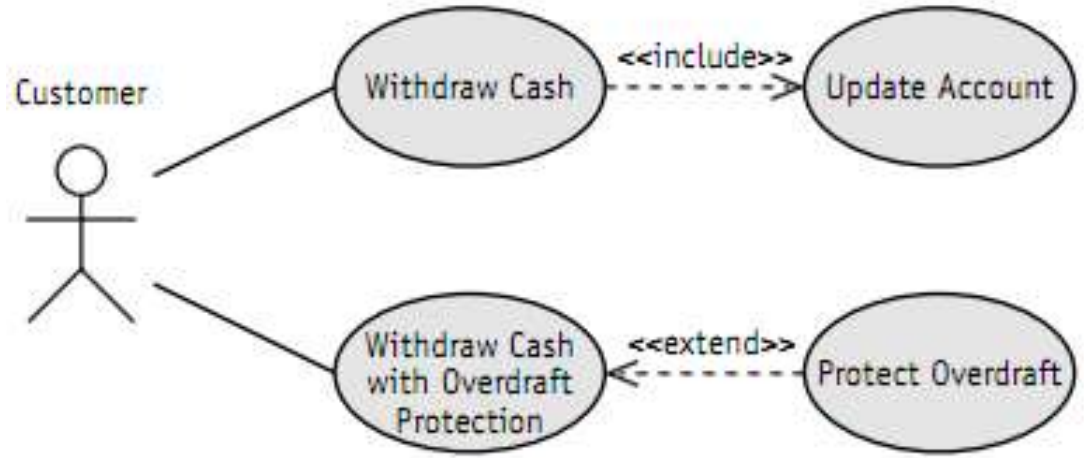


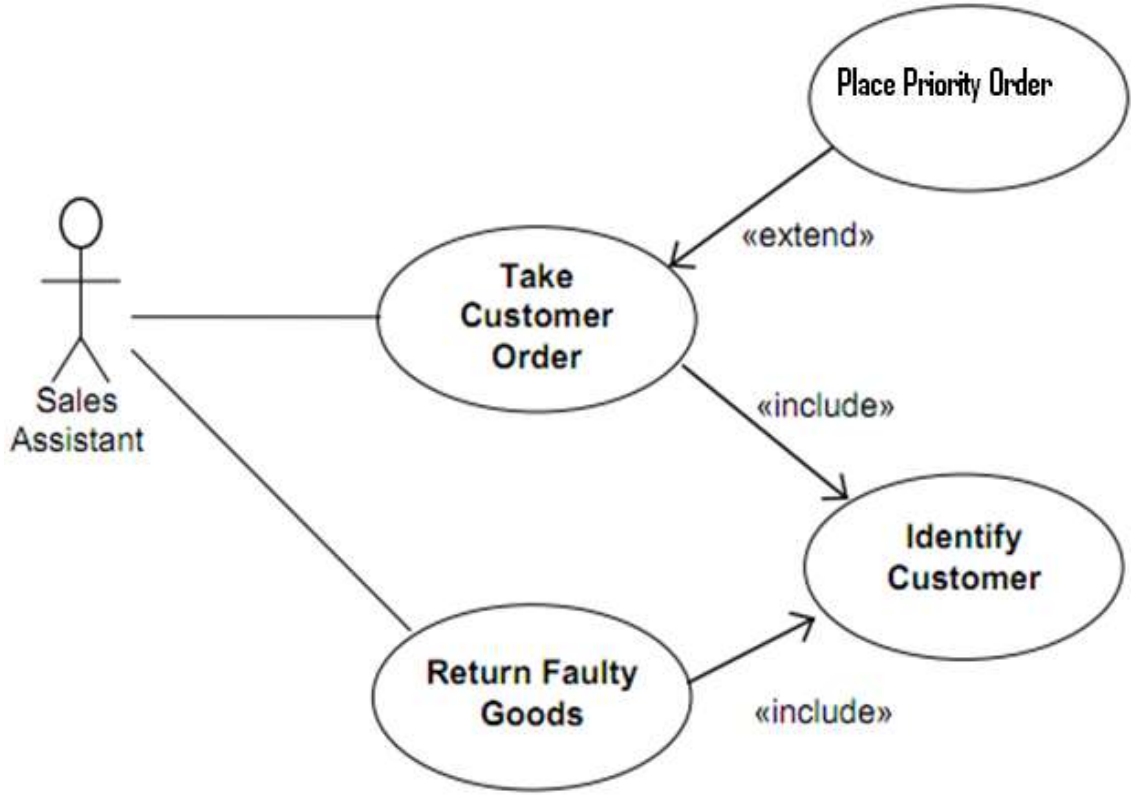
Figure 3-9 <<extend>> dependency notation for the Use Case diagram

Dependency: **<< extend >>**

>> **<<extend>>** relationship - specify **additional** or **alternative** steps that should be taken under certain condition.

>> The extension use case is not meaningful on its own.

>> Another example **<< extend >>** based on scenario of product order:



Generalization

>> Inheritance is a key concept in object-oriented programming, and OOAD.

>> The same idea, applied to actors and to Use Cases, is called generalization, and often goes by the nickname, an "is a" relationship.

>> A generalization relationship between Use Cases or Actors indicates that the child Use Cases or Actors inherit the properties of the parent Use Cases or Actors .

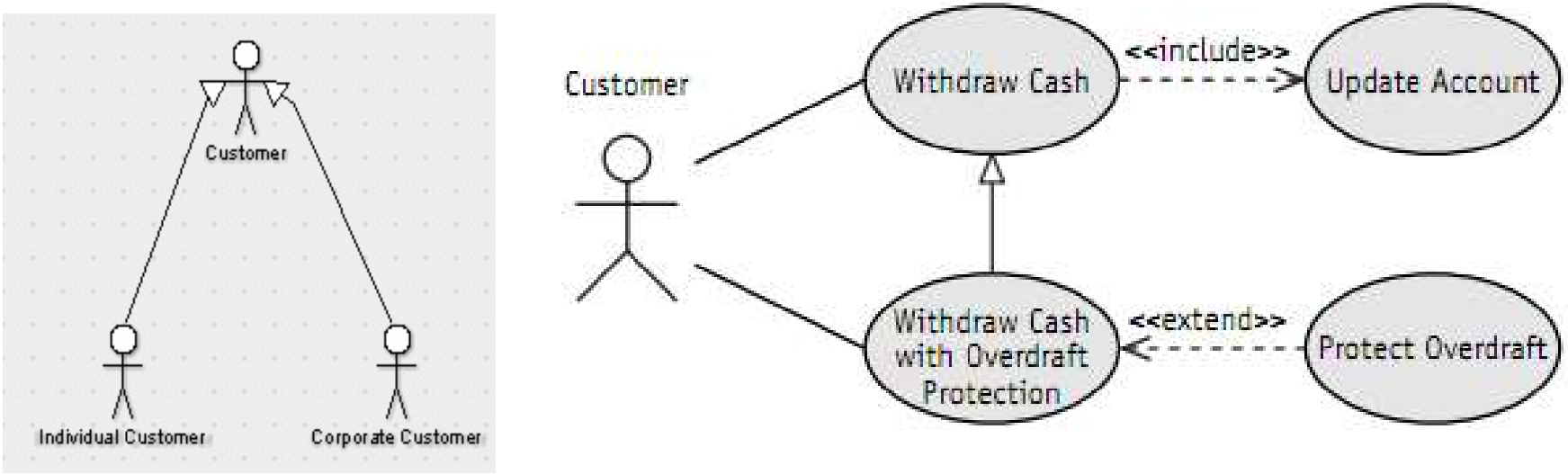
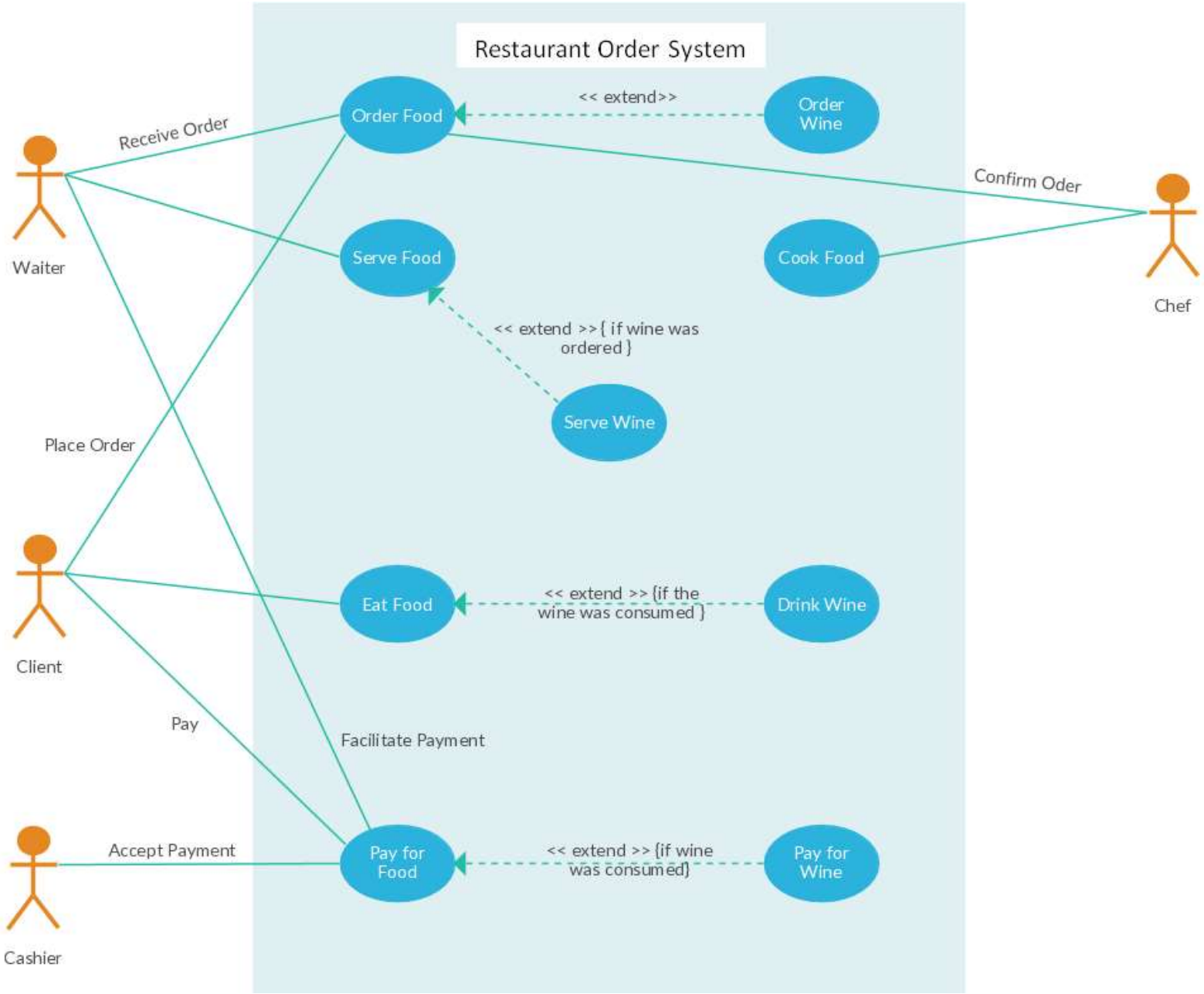
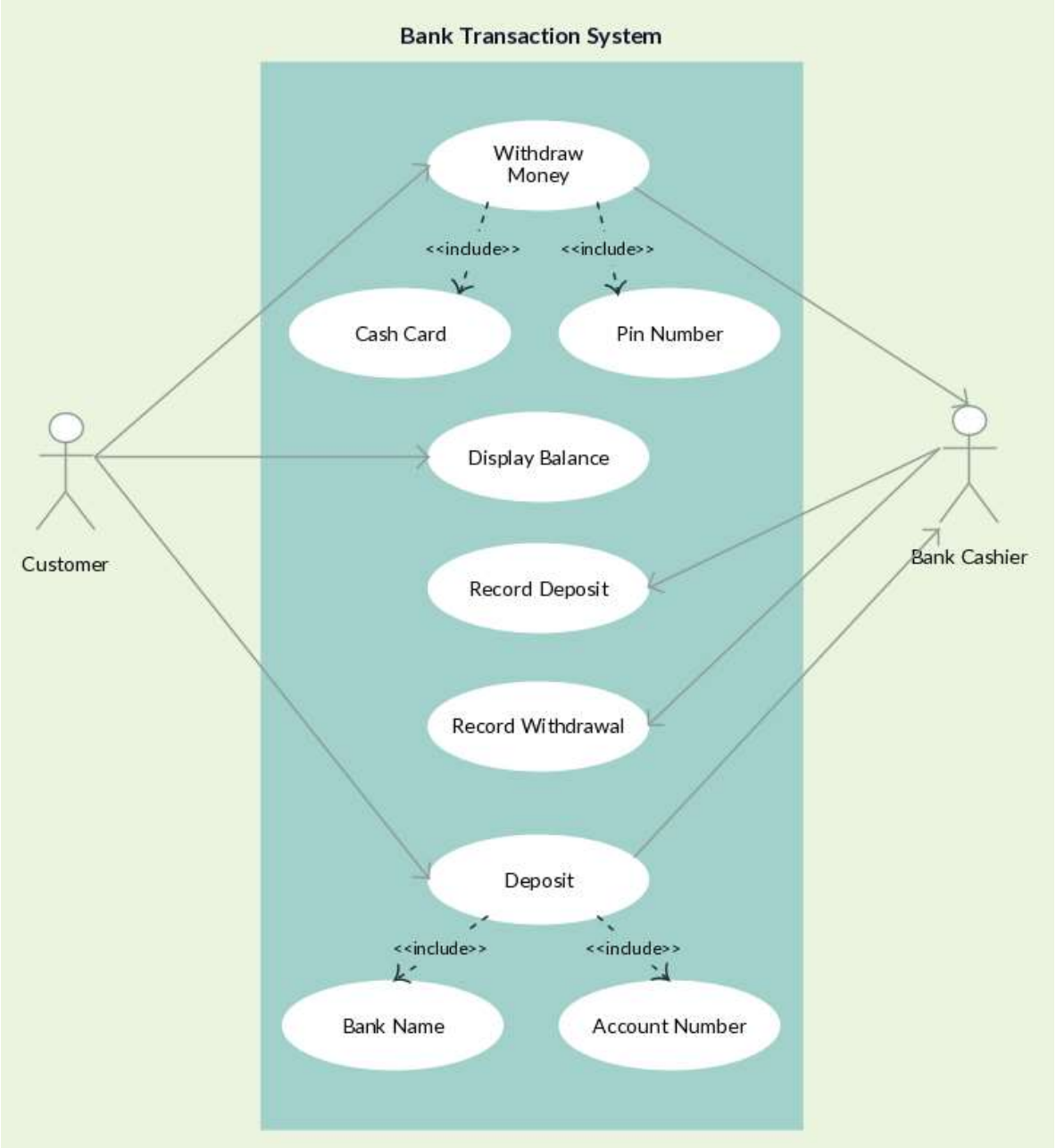


Figure 3-10 Generalization notation for the Use Case diagram

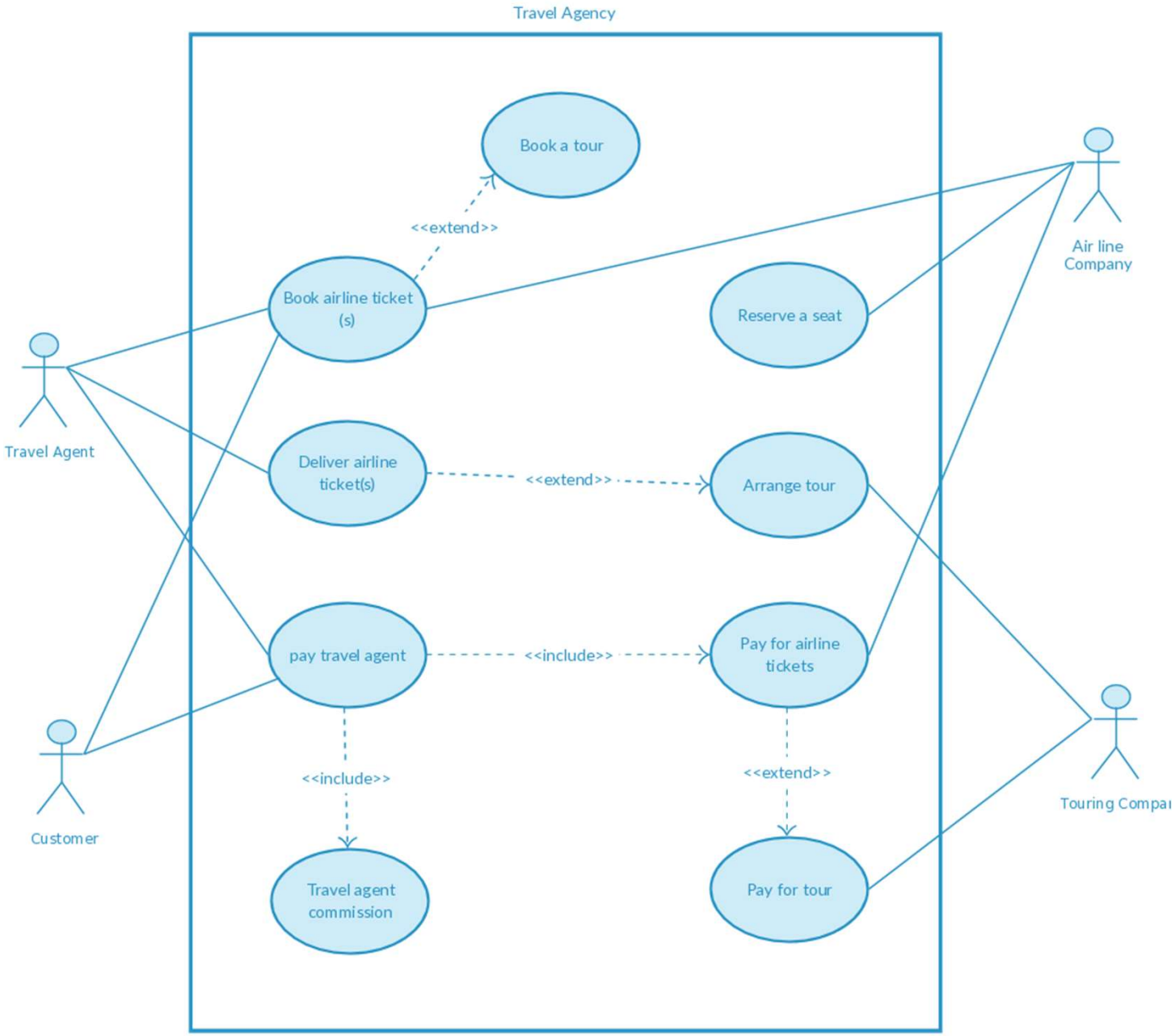
Use case Diagram Examples



Use case Diagram Examples

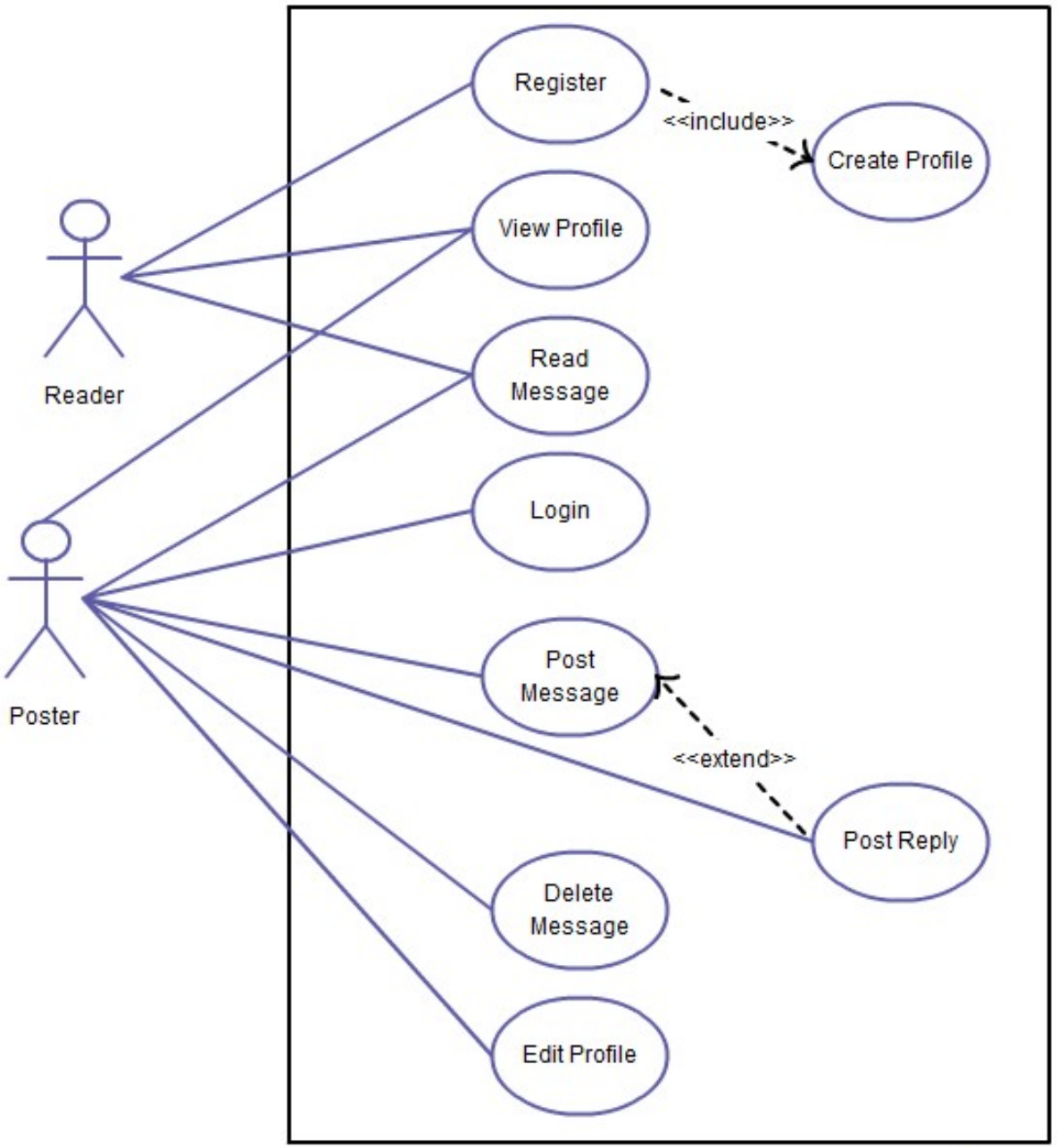


Use case Diagram Examples



Use case Diagram Examples

Blog site Usage System



Use case Diagram Exercises

Case 1:

A student can login to his/her profile using his/her student ID and Password. S/he could enter to the system or fail to enter because of wrong password. Student ID and password is verified by a server. If the student is new then s/he can register. After login they can check their profile to see their class routine or change their password.

Use case Diagram Exercises

Case 2:

In a **hotel management system** a **guest** can rent rooms. Hotel **receptionist** uses the system to assist in the renting. A guest can also book a room for future renting with the help of the receptionist, but he has to check if the room has prior booking or not. A Guest pays for the rooms at the time they check out. He can pay by cash, check or credit card. Receptionists has to logon to the system before they can use it, but to logon he has go through username/password verification.

Use case Diagram Exercises

Case 3:

In a hospital management system a **patient's medical history** is created by a **doctor**. A patient may be **referred** by a doctor to be **admitted** in the hospital. A patient can **rent** hospital facilities. An **administrative officer** deals with the renting. The **system** automatically checks whether the patient is referred by a doctor before he can rent any hospital facility. The types of facilities are rooms, beds or ICUs. Admitted patients are regularly visited by doctors and a **nurse** updates patient medical history after each visit. A doctor writes the **discharge** note of the patient when he leaves which is also included in the patient's medical history. An **account clerk** prepares the **bill**. The bill is calculated from the elements written in the medical history, i.e. **number of doctor's visit**, **prescribed medication**, **tests**. When the nurse **updates** the medical history she writes either the date-time of doctor's visit or prescribed medication or test. The patient may **pay** by **cash** or **card** when the bill is prepared.

Use case Diagram Exercises

Case 4:

Music Today is a audio recording company. There are several recordists in the company. Clients record their music with the help of the recordists. The recording is done in a session. A session is usually created by the office clerk. A session needs to be booked by the client beforehand and before booking the time needs to be verified by the booking clerk for availability. The company maintains a list of preferred clients who book sessions like regular clients but also can create sessions according to their needs. Clerks have to log in to use the system. Clients can pay in cash or card. The accountant deals with the payment. The payment is only cleared when the accountant receives clearance from the booking clerk.

References

→ **Chapter 17 & 18**
The Unified Modeling Language User Guide
SECOND EDITION
By Grady Booch, James Rumbaugh, Ivar Jacobson

→ **Session 5 & 6**
UML Weekend Crash Course
Thomas A. Pender