



- **Systems Analysis and Design**

- **Course Code: CSE 305**

**Use Case Diagram**



**Instructor:**  
*Dr. Shah Murtaza Rashid Al Masud*  
*Assoc. Prof. CSE Dept., UAP*



# The Unified Modelling Language (UML)

## UML properties/things:

---

- Objects.
- Relationships
- Diagrams, categorized as either structural/theoretical/conceptual/physical or behavioural.

**Structure diagrams show the things in the modeled system.**

**Behavioral diagrams show what should happen in a system.**

# Two General Groupings of Things

---

There are two general groupings of things in UML:

- Structural things that define the conceptual (theoretical) and physical structures of an O-O system and are described by **nouns**.
- Behavioral things, the verbs of a UML model that represent the behavior of the system and the states of the system before, during, and after the behaviors occur.

# Structural Relationships

---

Structural relationships are:

- Dependencies.
- Aggregations.
- Associations.
- Generalizations.

# Behavioral Relationships

---

- Behavioral relationships are:
  - Communicates.
  - Includes.
  - Extends.
  - Generalizes.

# Structural Diagrams

---

Structural things are the most common and include:

- Class (and object) diagrams.
- Component diagrams.
- Deployment diagrams.

# Behavioral Things/Diagrams

---

Behavioral things include:

- Use case diagrams.
- Sequence diagrams.
- Collaboration diagrams.
- Activity diagrams.

# Commonly Used UML Diagrams

---

The most commonly used UML diagrams are:

- Use case diagram, describing what the system does.
  - The starting point for UML modelling.
- Activity diagram.
  - Each use case may create one activity diagram.



# Commonly Used UML Diagrams

---

The most commonly used UML diagrams  
(continued):

- Sequence diagram, showing the sequence of activities and class relationships.
  - Each use case may create one or more sequence diagrams.

# Commonly Used UML Diagrams

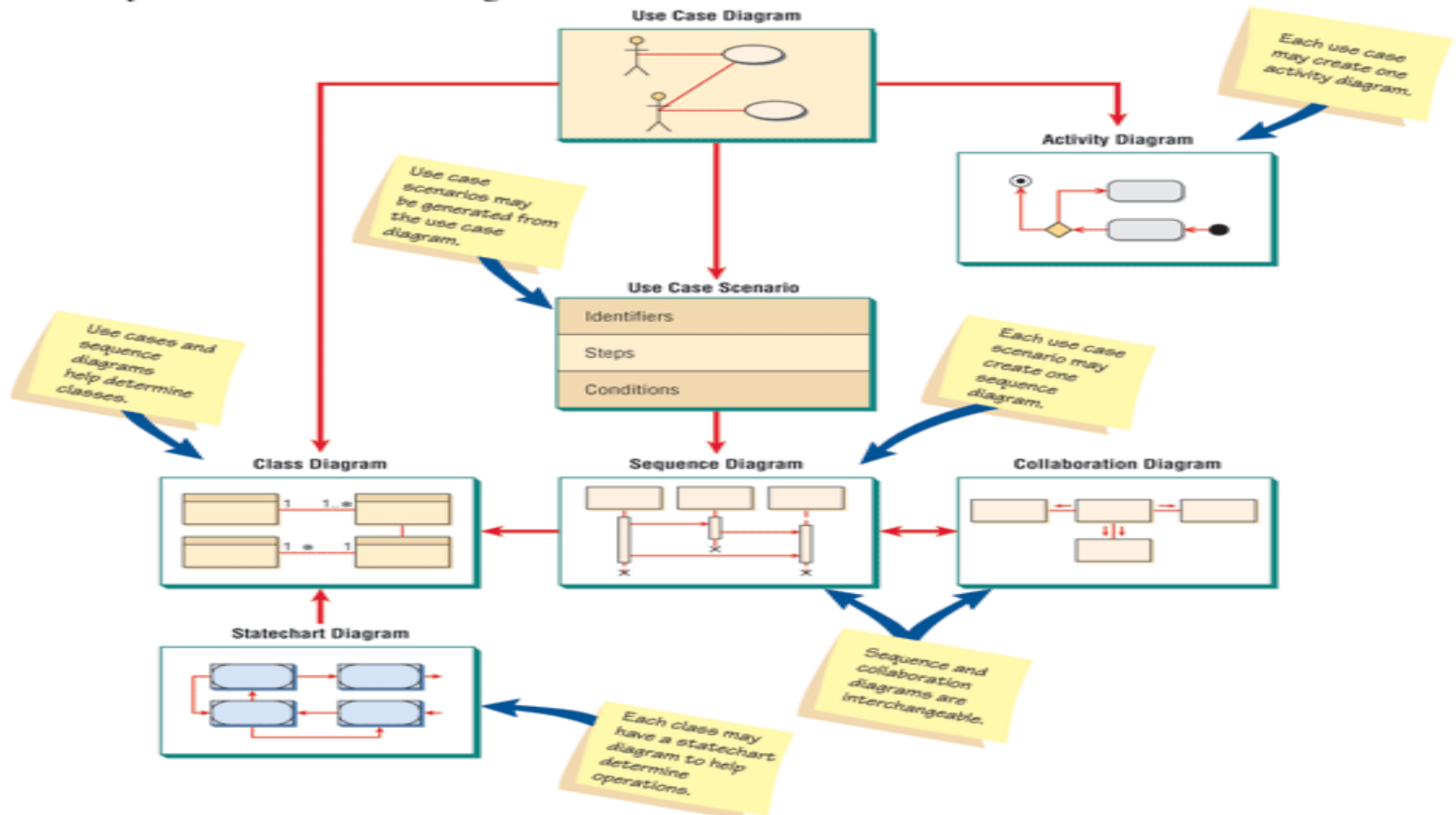
---

The most commonly used UML diagrams (continued):

- Class diagram, showing classes and relationships.
  - Sequence diagrams are used to determine classes.

# Overview of UML Diagrams

**Figure 18.5** An overall view of UML diagrams showing how each diagram leads to the development of other UML diagrams.



# Use Case Diagram

Use-case diagrams describe the high-level functions and scope of a system.

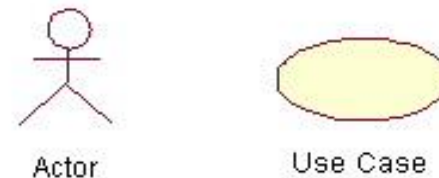
---

These diagrams also identify the interactions between the system and its actors.

The use cases and actors in use-case diagrams describe what the system does and how the actors use it, but not how the system operates internally.

# Use Case Diagram

- A use (yoos) case describes **what the system does**, **not how it does the work**. It is a logical model of the system. **Primarily used to visualize** the Use cases, corresponding Actors, and their interactions
- The use case model reflects the view of the system of the user out
- Symbols are:
  - **Actor**, a stick figure.
  - **Use case**, an oval.
  - **Connecting** lines (relations).



# Use Case Diagram

Use-case diagrams illustrate and define the context and requirements of either an entire system or the important parts of the system.

---

You can model a complex system with a single use-case diagram, or create many use-case diagrams to model the components of the system.

You would typically develop use-case diagrams in the early phases of a project and refer to them throughout the

# Use Case Diagram

- Before starting a project, you can create use-case diagrams to model a business so that all participants in the project share an understanding of the workers, customers, and activities of the business.
- While gathering requirements, you can create use-case diagrams to capture the system requirements and to present to others what the system should do.
- During the analysis and design phases, you can use the use cases and actors from your use-case diagrams to identify the classes that the system requires.
- During the testing phase, you can use use-case diagrams to identify tests for the system.

# Actors

---

- Actor is an external entity that interacts with the system.
- Most actors represent user roles, but actors can also be external systems.
- Exist outside of the system
- May interact with one or more use cases and a use case may involve one or more actors



# Use Case (Continued)

---

- Better to create fewer use cases
- 20 use cases for a large system
- 50 use cases would be the maximum for a large system
- Can nest use cases, if needed

# Boundary

---

- A boundary is the dividing line between the system and its environment.
- Use cases are within the boundary.
- Actors are outside of the boundary.

# What is a Connection?

---

- A connection is an association between an actor and a use case.

# Use Case Relationships

---

- Communicates
  - Connect an actor to a use case using a line with no arrowhead
- Includes
  - Use case contains a behavior that is common to more than one use case.
  - The common use case is included in other use cases.
  - Dotted arrow points toward common use case.

# Use Case Relationships (Continued)

---


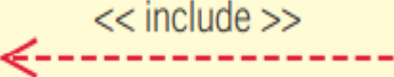
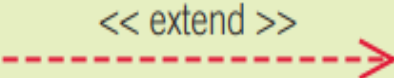

- Extends/Extend

- A different use case handles variations or exceptions from the basic/base use case.
- Dotted arrow goes from extended to basic use case.

- Generalizes

- One thing is more general than another thing.
- Arrow points to the general thing.

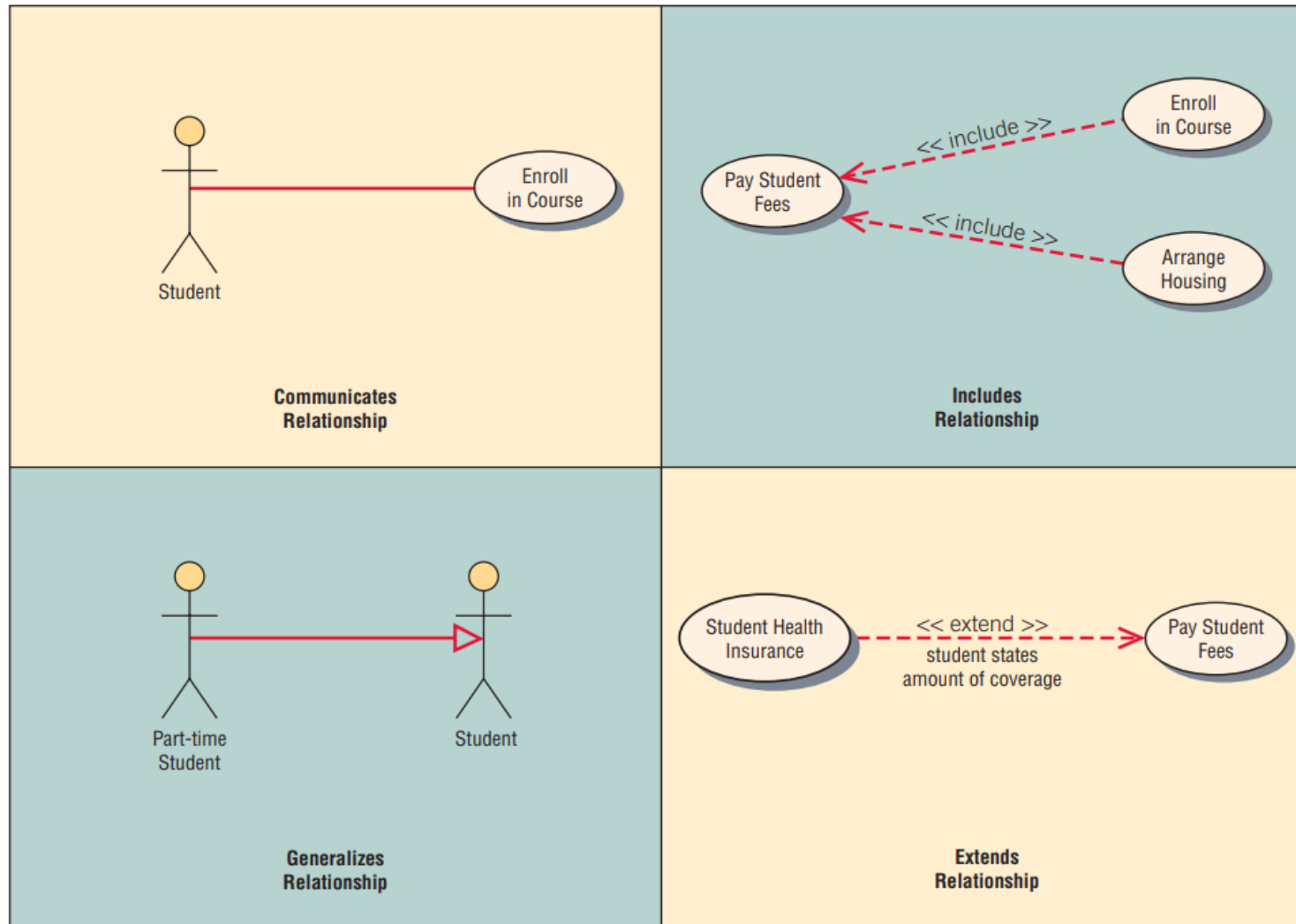
# Use Case Relationships

Relationship	Symbol	Meaning
<b>Communicates</b>		An actor is connected to a use case using a line with no arrowheads.
<b>Includes</b>		A use case contains a behavior that is common to more than one other use case. The arrow points to the common use case.
<b>Extends</b>		A different use case handles exceptions from the basic use case. The arrow points from the extended to the basic use case.
<b>Generalizes</b>		One UML “thing” is more general than another “thing.” The arrow points to the general “thing.”

*Include implies Mandatory*

*Extend implies Optional*

# Use Case Relationships



**FIGURE 2.14**

Examples of use cases and behavioral relationships for student enrollment.

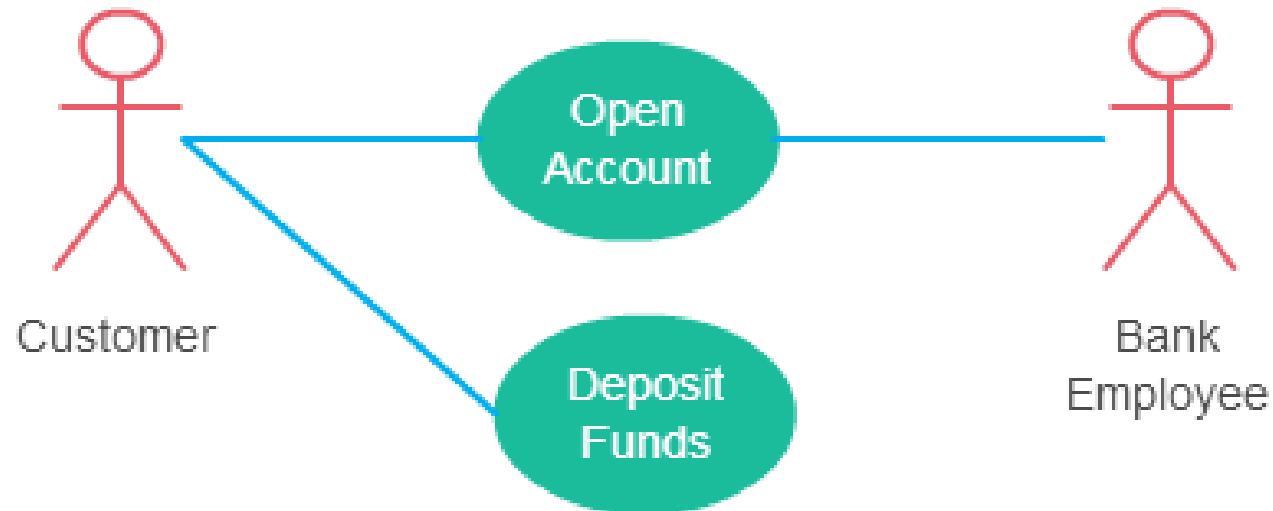
# Association Between Actor and Use Case

## Few things to note.

An actor must be associated with at least one use case.

An actor can be associated with multiple use cases.

Multiple actors can be associated with a single use case.





## Extend Relationship Between Two Use Cases

Here are a few things to consider when using the <<extend>> relationship.

---

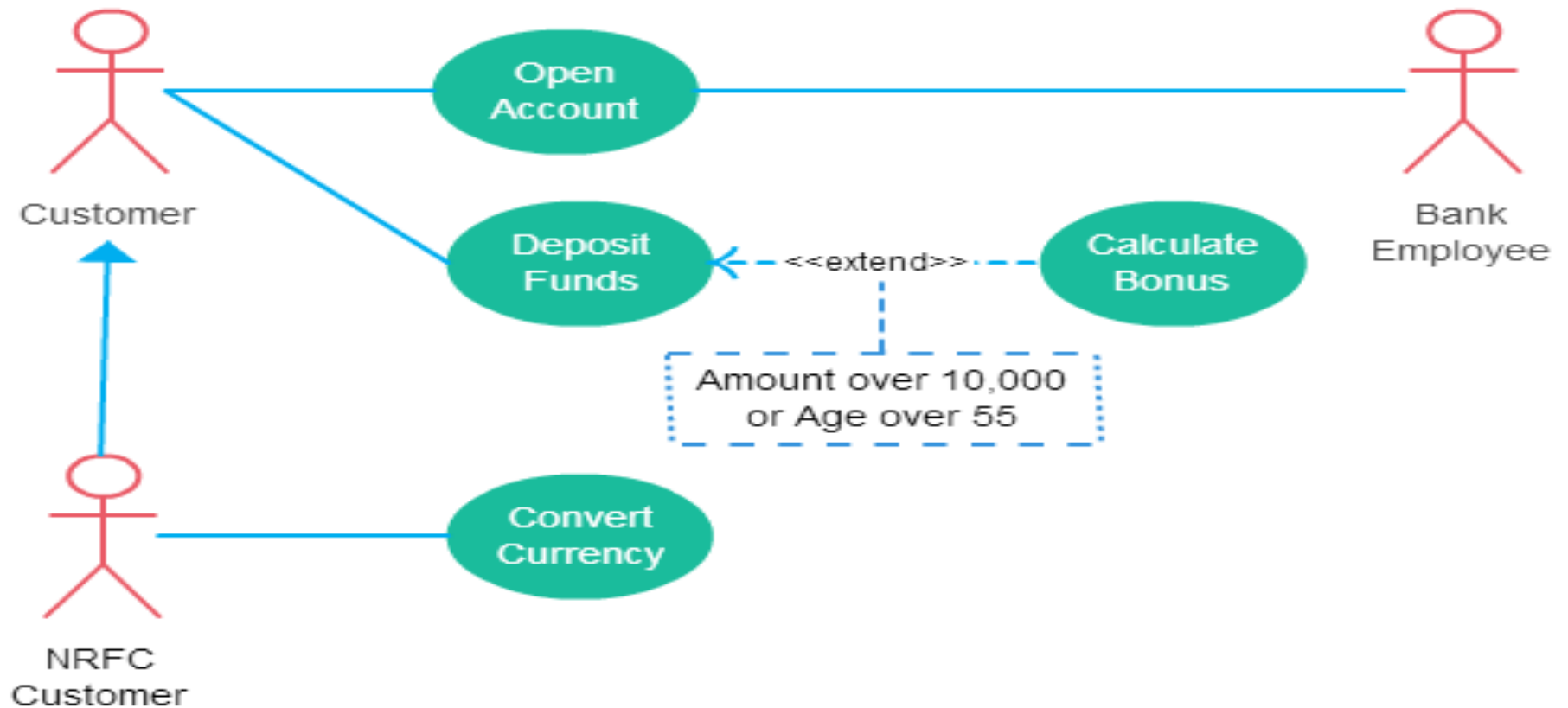
**The extending use case is dependent on the extended (base) use case.** In the below diagram the “Calculate Bonus” use case doesn’t make much sense without the “Deposit Funds” use case.

**The extending use case is usually optional** and can be triggered conditionally. In the diagram, you can see that the extending use case is triggered only for deposits over 10,000 or when the age is over 55.

**The extended (base) use case must be meaningful on its own.** This means it should be independent and must not rely on the behavior of the extending use case.

Lets expand our current example to show the <<extend>> relationship.

# Extend Relationship Between Two Use Cases



Extend relationship in use case diagrams

## **Include Relationship Between Two Use Cases**

Include relationship show that the behavior of the included use case is part of the including (base) use case.

---

The main reason for this is to reuse common actions across multiple use cases.

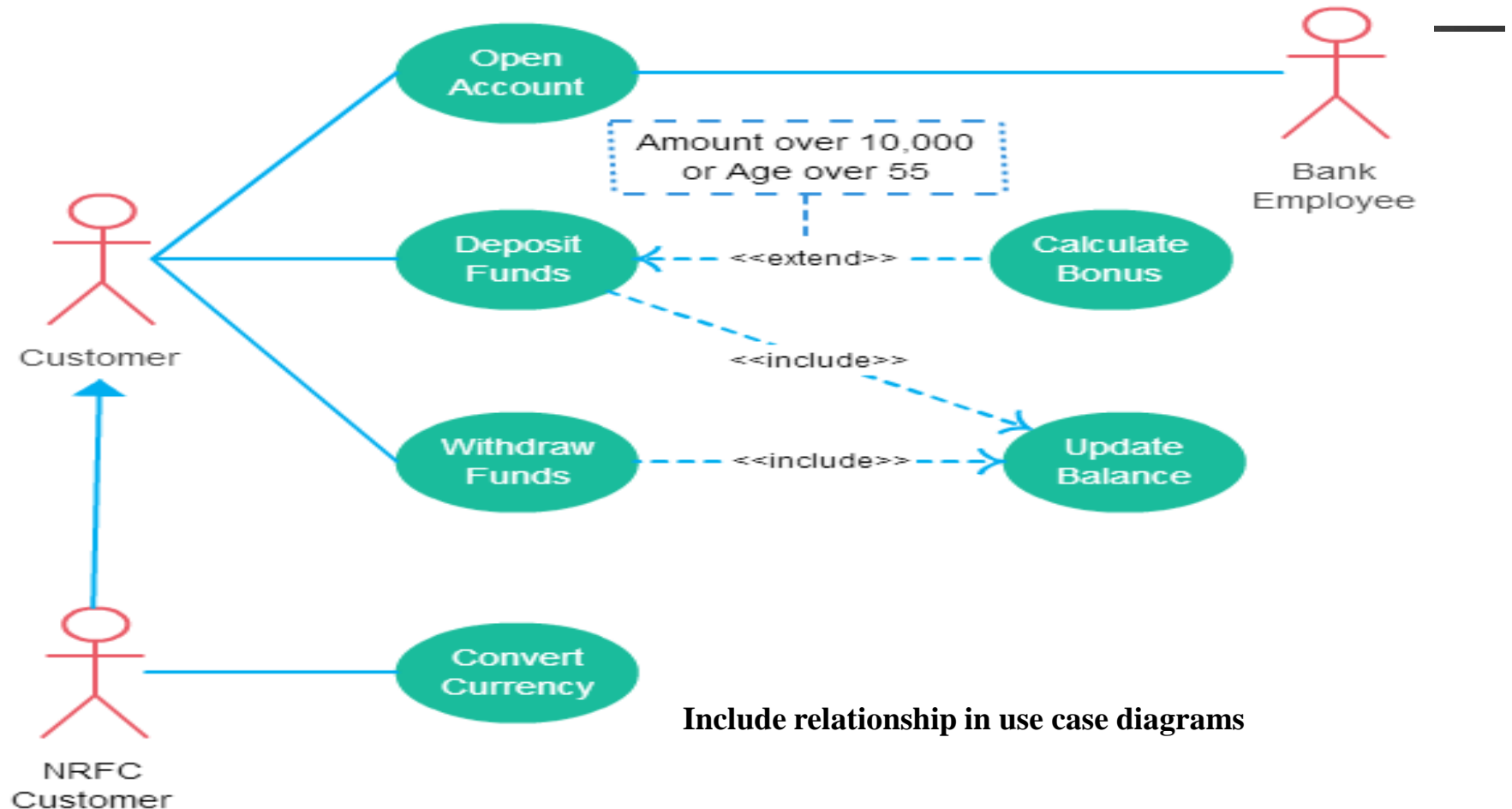
In some situations, this is done to simplify complex behaviors.

**Few things to consider when using the <<include>> relationship.**

- ☐ The base use case is incomplete without the included use case.
- ☐ The included use case is mandatory and not optional.

# Include Relationship Between Two Use Cases

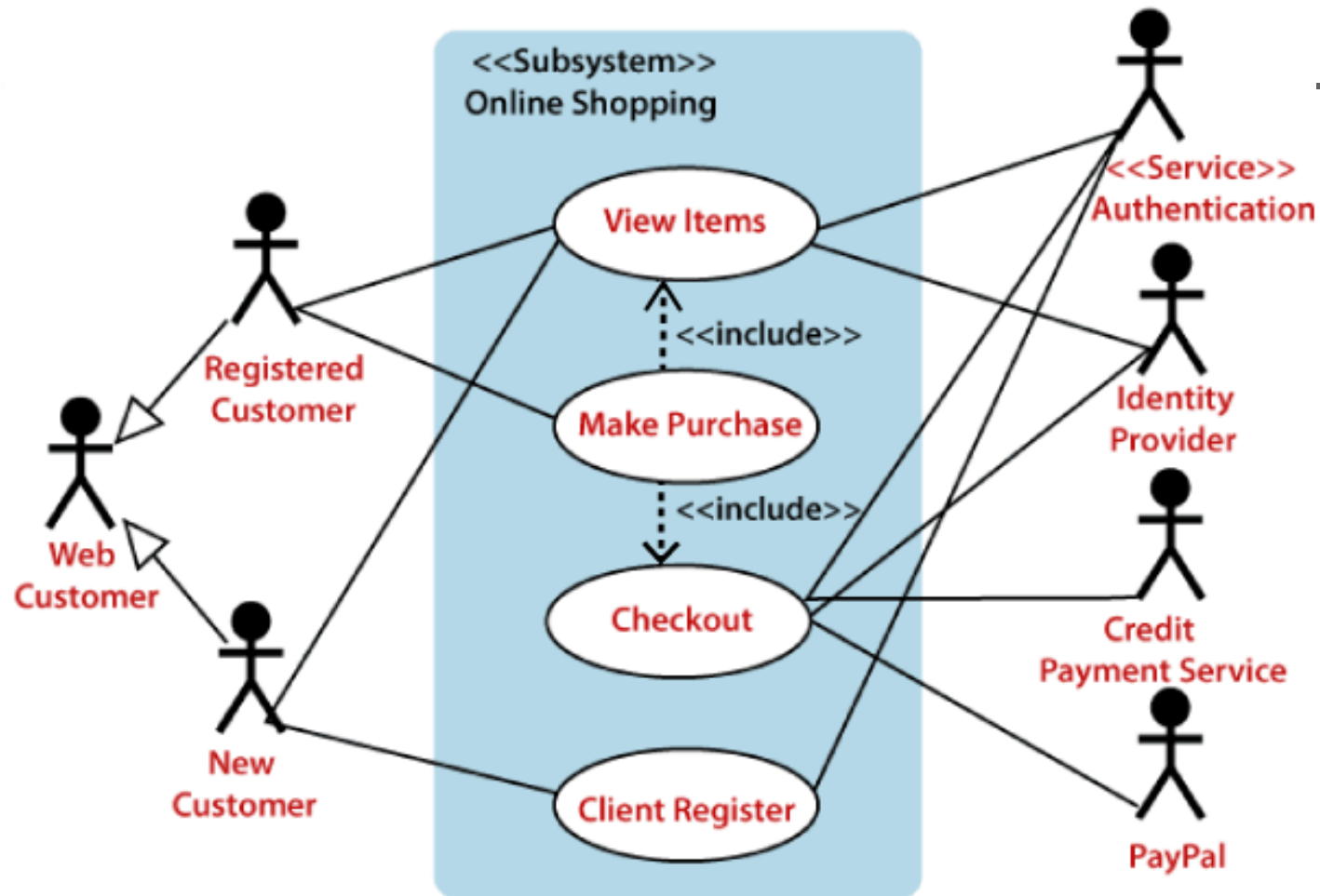
Let's expand our banking system use case diagram to show include relationships as well.



## A use case diagram depicting the Online Shopping website

Here the Web Customer actor makes use of any online shopping website to purchase online. The top-level uses are as follows; View Items, Make Purchase, Checkout, Client Register. The **View Items** use case is utilized by the customer who searches and view products. The **Client Register** use case allows the customer to register itself with the website for availing gift vouchers, coupons, or getting a private sale invitation. It is to be noted that the **Checkout** is an included use case, which is part of **Making Purchase**, and it is not available by itself.

## A use case diagram depicting the Online Shopping website



## A use case diagram depicting the Online Shopping website

The **View Items** is further extended by several use cases such as;

---

Search Items, Browse Items, View Recommended Items, Add to Shopping Cart, Add to Wish list. All of these extended use cases provide some functions to customers, which allows them to search for an item. The View Items is further extended by several use cases such as; Search Items, Browse Items, View Recommended Items, Add to Shopping Cart, Add to Wish list. All of these extended use cases provide some functions to customers, which allows them to search for an item.

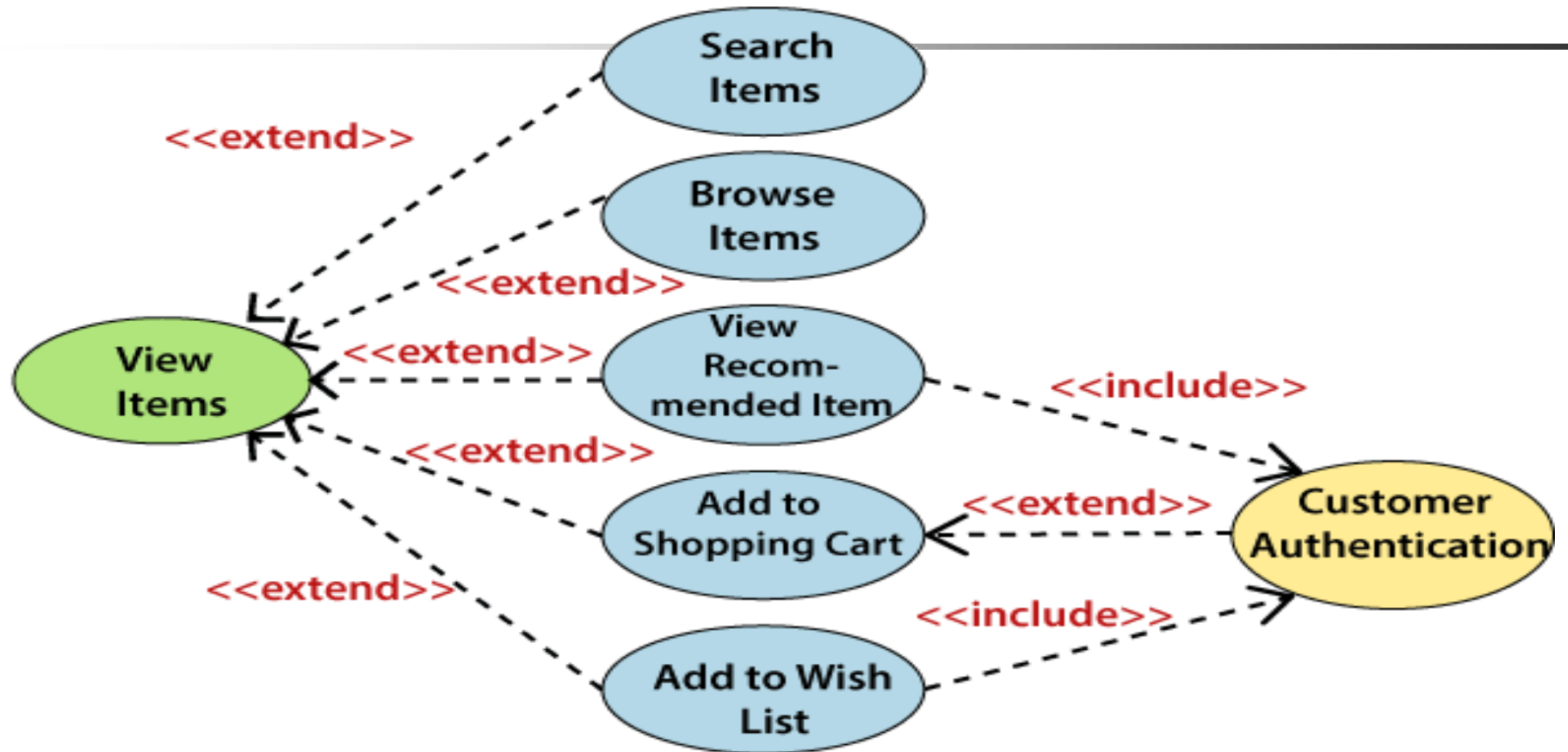


A use case diagram depicting the Online Shopping website

Both **View Recommended Item** and **Add to Wish List** include the Customer Authentication use case, as they necessitate authenticated customers, and simultaneously item can be added to the shopping cart without any user authentication.



## A use case diagram depicting the Online Shopping website

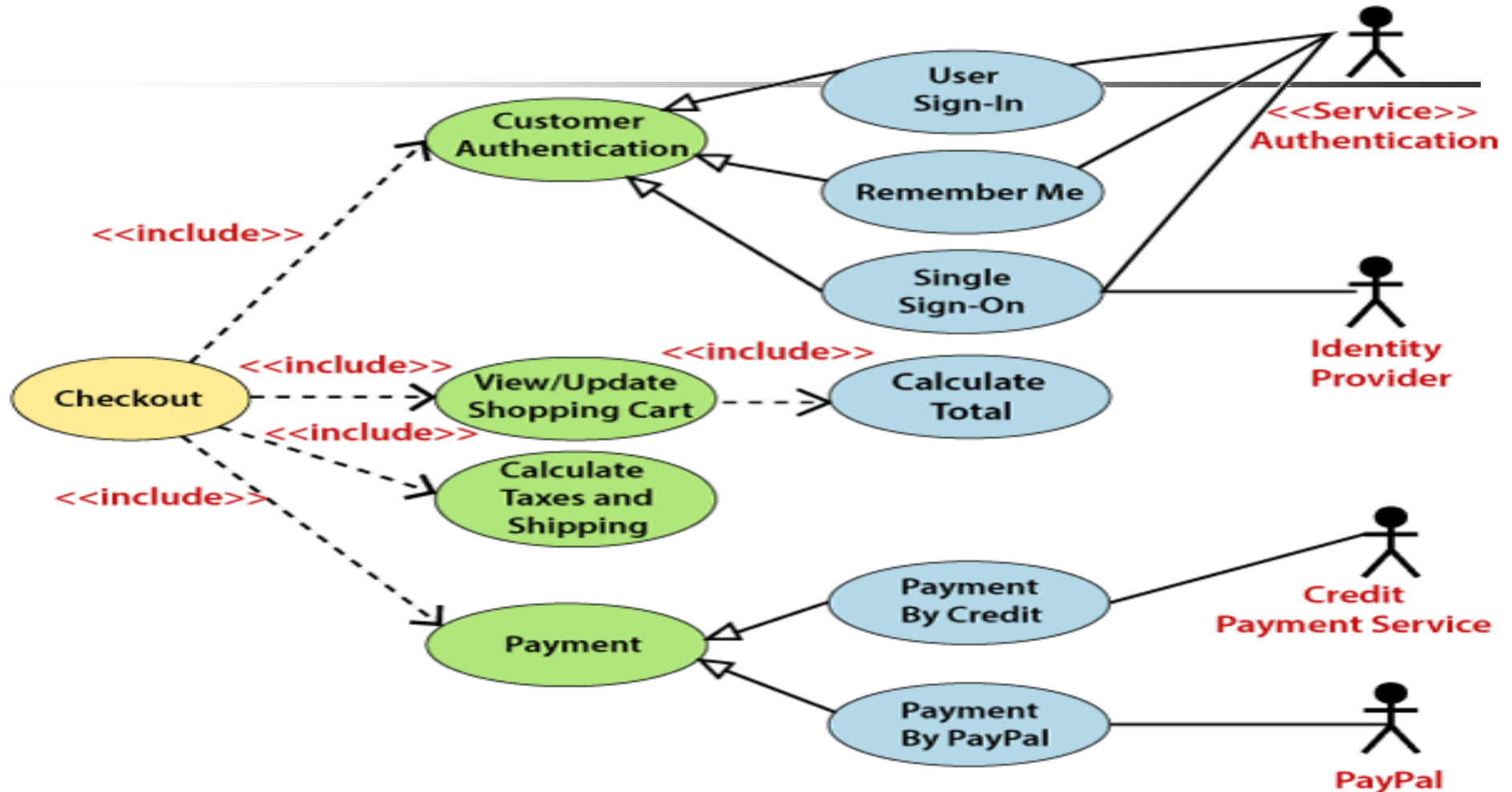


## A use case diagram depicting the Online Shopping website

Similarly, the **Checkout** use case also includes the following use cases, as shown below. It requires an authenticated Web Customer, which can be done by login page, user authentication cookie ("Remember me"), or Single Sign-On (SSO). SSO needs an external identity provider's participation, while Web site authentication service is utilized in all these use cases.

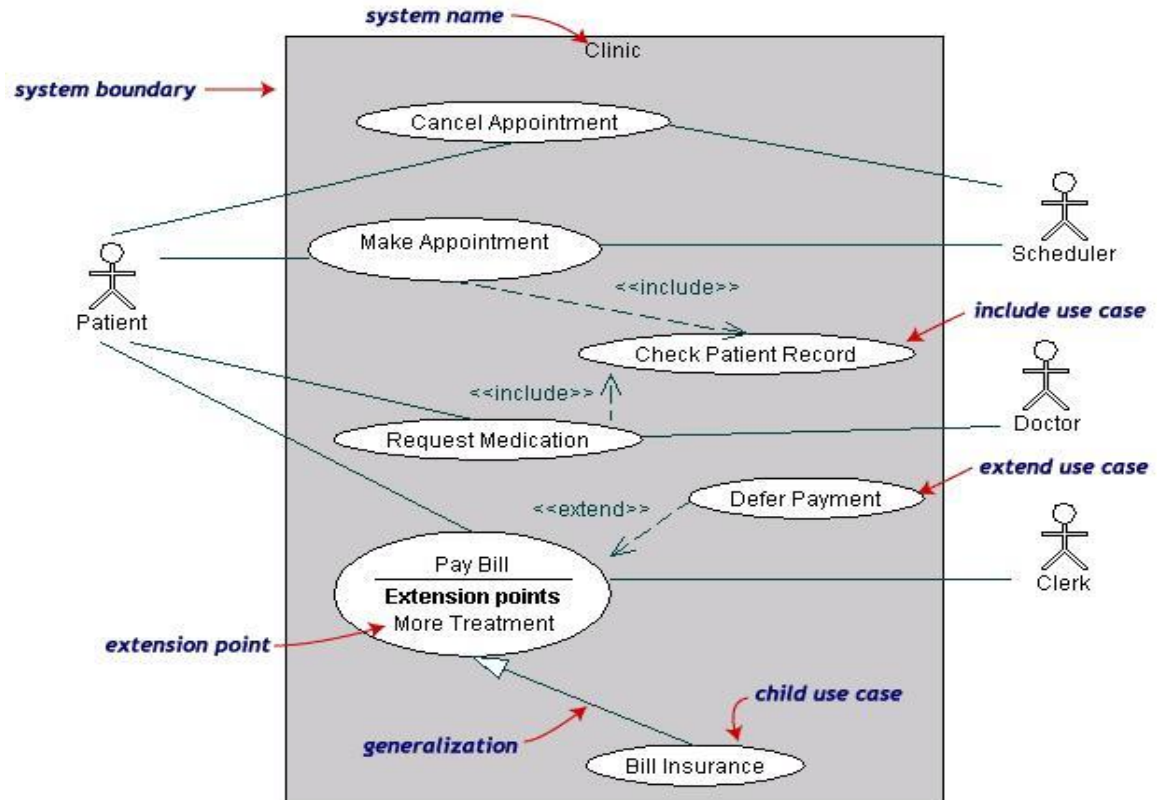
The Checkout use case involves Payment use case that can be done either by the credit card and external credit payment services or with PayPal.

## A use case diagram depicting the Online Shopping website



# Use-Case Diagrams

- 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

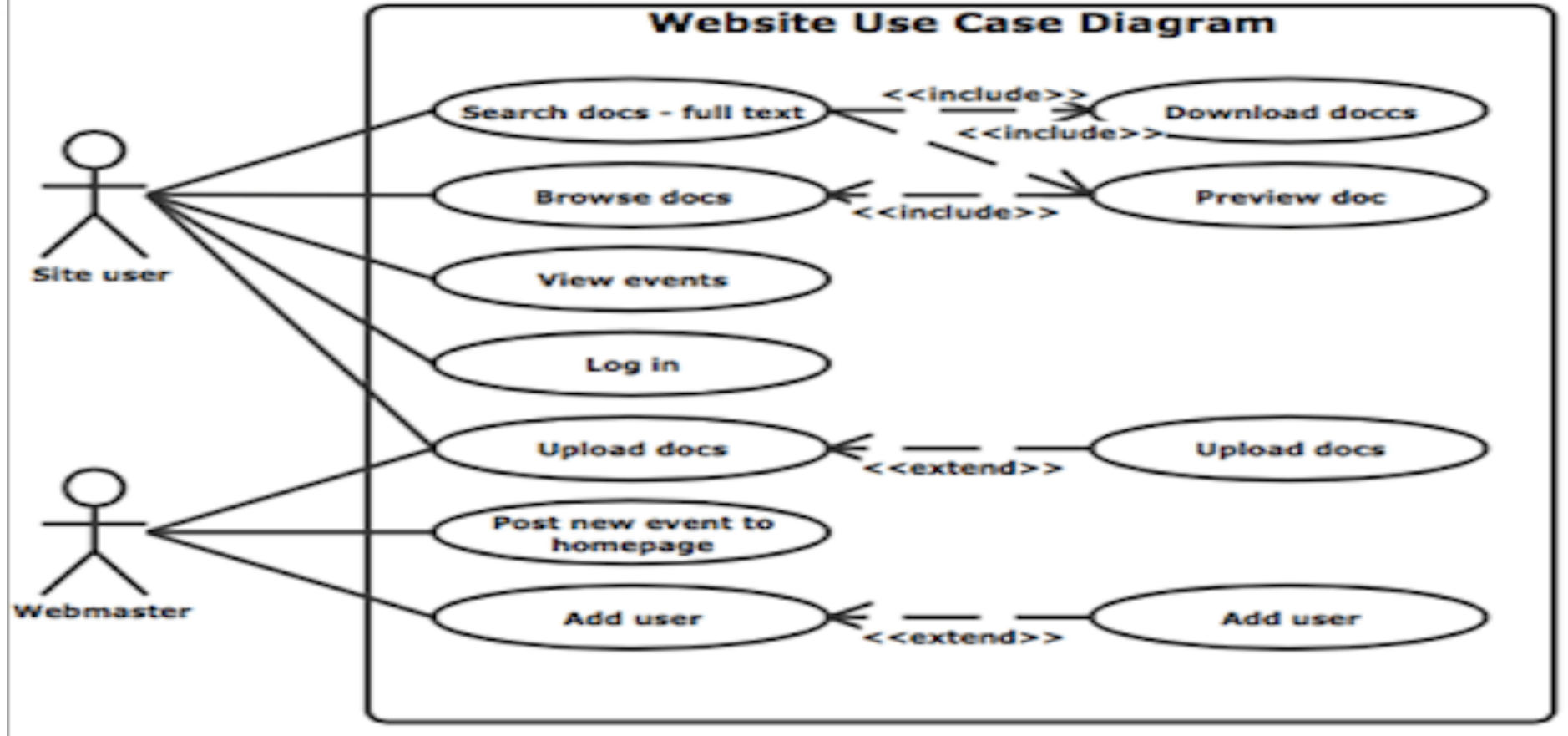
## Use Case - Example



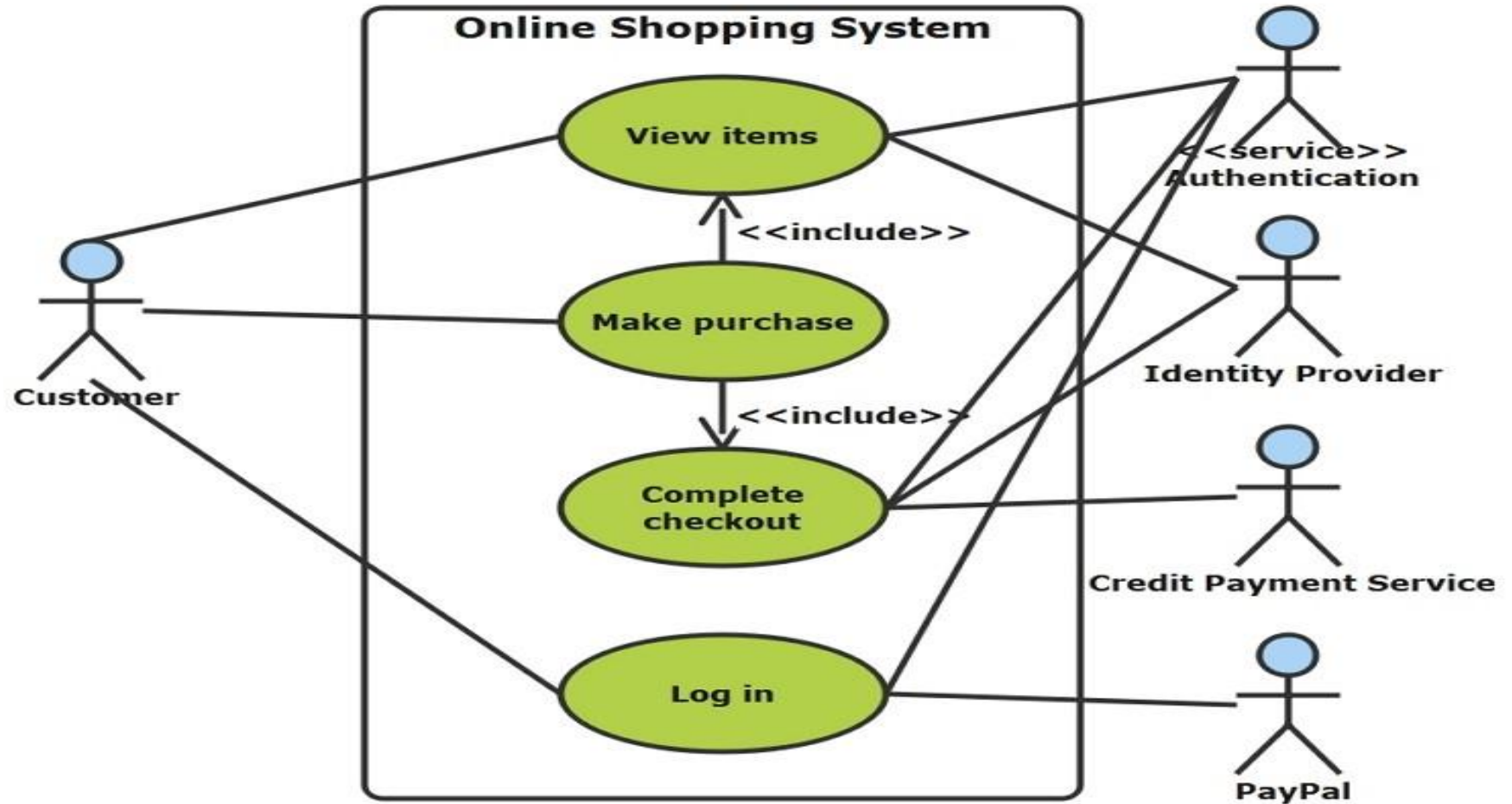


# Use-Case Diagrams

## Use Case - Example



# Use-Case Diagrams



# Use Case Relationships

## recap

**COMMUNICATES.** The behavioral relationship communicates is used to connect an actor to a use case. Remember that the task of the use case is to give some sort of result that is beneficial to the actor in the system. Therefore, it is important to document these relationships between actors and use cases. In our first example, a **Student** communicates with **Enroll in Course**. Examples of some components of a student enrollment example are shown in the use case diagrams in Figure 2.14.

**INCLUDES.** The includes relationship (also called uses relationship) describes the situation in which a use case contains behavior that is common to more than one use case. In other words, the common use case is included in the other use cases. A dotted arrow that points to the common use case indicates the includes relationship. An example would be a use case **Pay Student Fees** that is included in **Enroll in Course** and **Arrange Housing**, because in both cases students must pay their fees. This may be used by several use cases. The arrow points toward the common use case.



# Use Case Relationships

**EXTENDS.** The extends relationship describes the situation in which one use case possesses the behavior that allows the new use case to handle a variation or exception from the basic use case.

For example, the extended use case **Student Health Insurance** extends the basic use case **Pay Student Fees**. The arrow goes from the extended to the basic use case.

**GENERALIZES.** The generalizes relationship implies that one thing is more typical than the other thing. This relationship may exist between two actors or two use cases. For example, a **Part-Time Student** generalizes a **Student**. Similarly, some of the university employees are professors. The arrow points to the general thing.

# Steps for Creating a Use Case Model

---

The steps required to create a use case model are:

- Review the business specifications and identify the actors within the problem domain.
- Identify the high-level events and develop the primary use cases that describe the events and how actors initiate them.

# Steps for Creating a Use Case Model

---

- The steps required to create a use case model are (continued):
  - Review each primary use case to determine possible variations of flow through the use case.
  - Develop the use case documents for all primary use cases and all important use case scenarios.

# Use Case Scenario

## Developing Use Case Scenarios

Each use case has a description. We will refer to the description as a use case scenario. As mentioned, the primary use case represents the standard flow of events in the system, and alternative paths describe variations to the behavior. Use case scenarios may describe what happens if an item purchased is out of stock, or if a credit card company rejects a customer's requested purchase.

There is no standardized use case scenario format, so each organization is faced with specifying what standards should be included. Often the use cases are documented using a use case document template predetermined by the organization, which makes the use cases easier to read and provides standardized information for each use case in the model.

# Use Case Scenario

---

- A use case scenario may be created for the standard flow through the use case.
- Other scenarios may be created for variations on the main flow.
- A use case includes:
  - Use case identifiers and initiators.
  - Steps performed.
  - Conditions, assumptions, and questions.



## Why Use Case Diagrams Are Helpful

No matter what method you use to develop your system (traditional SDLC methods, agile methods, or object-oriented methods), you will find that use cases are very valuable. The use case diagrams identify all the actors in the problem domain, and a systems analyst can concentrate on what humans want and need to use the system, extend their capabilities, and enjoy their interaction with technology.

The actions that need to be completed are also clearly shown on the use case diagram. This not only makes it easy for the analyst to identify processes, but it also aids in communication with other analysts on the team and business executives.

The use case scenario is also worthwhile. Since a lot of the information the users impart to the analyst already takes the form of stories, it is easy to capture the stories on a use case scenario form. The use case scenario always documents the triggering event so that an analyst can always trace the steps that led to other use cases. Since the steps performed are noted, it is possible to employ use case scenarios to write logical processes.

Use case diagrams are becoming popular because of their simplicity and lack of technical detail. They are used to show the scope of a system, along with the major features of the system and the actors who work with those major features. The users see the system and they can react to it and provide feedback. They may also help to determine whether to build or buy the software.