UML (Use Cases)

- "Buy a Product" scenario on a web-based on-line store

The customer browses the catalog and adds desired items to the shopping basket. When the customer wishes to pay, the customer describes the shipping and credit card information and confirms the sale. The system checks the authorization on the credit card and confirms the sale both immediately and with a follow-up e-mail.

- What if credit card authorization fails
- There is no need to capture the shipping and credit card information for a regular customer
- All these scenarios are different yet similar.
- In all these three scenarios, the user has the **same goal**: to buy a product.
- This user goal is the key to use cases
- A **use case is a set of scenarios tied together** by a common user goal.

Use Cases

- A *use case* ...
  - Does not indicate how the specified behavior is implemented, only what the behavior is
  - Performs a service for some *user* of the system.
    - A user of the system is known as an *actor*.
- An actor is a role that a user plays with respect to the system
- An actor doesn't have to be human

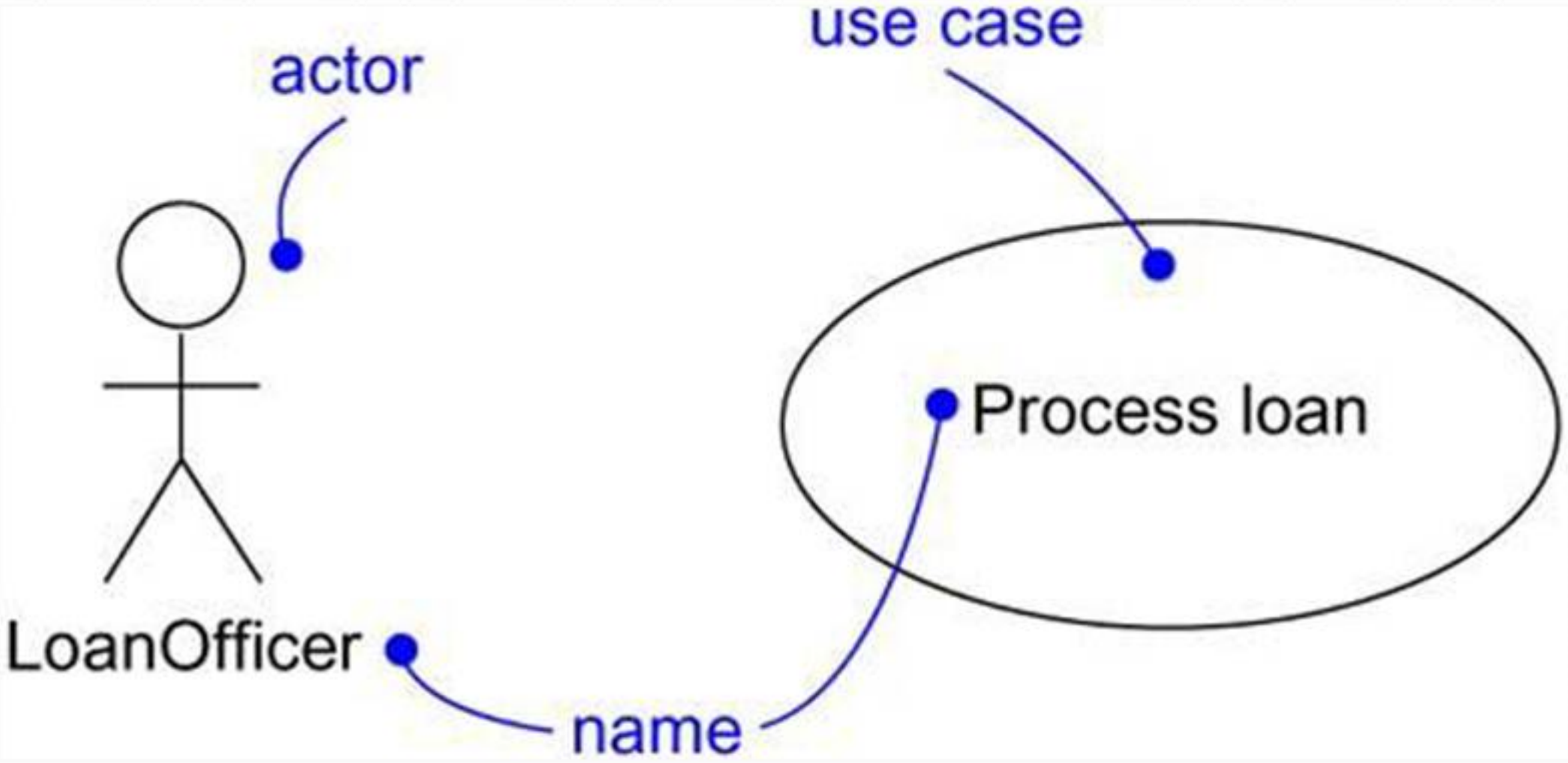
Use Cases

- A *use case* represents a functional requirement of the system
- A *requirement* ...
  - Is a design feature, property, or behavior of a system
  - States what needs to be done, but not how it is to be done
  - Is a contract between the customer and the developer
  - Can be expressed in various forms, including use cases

Actors

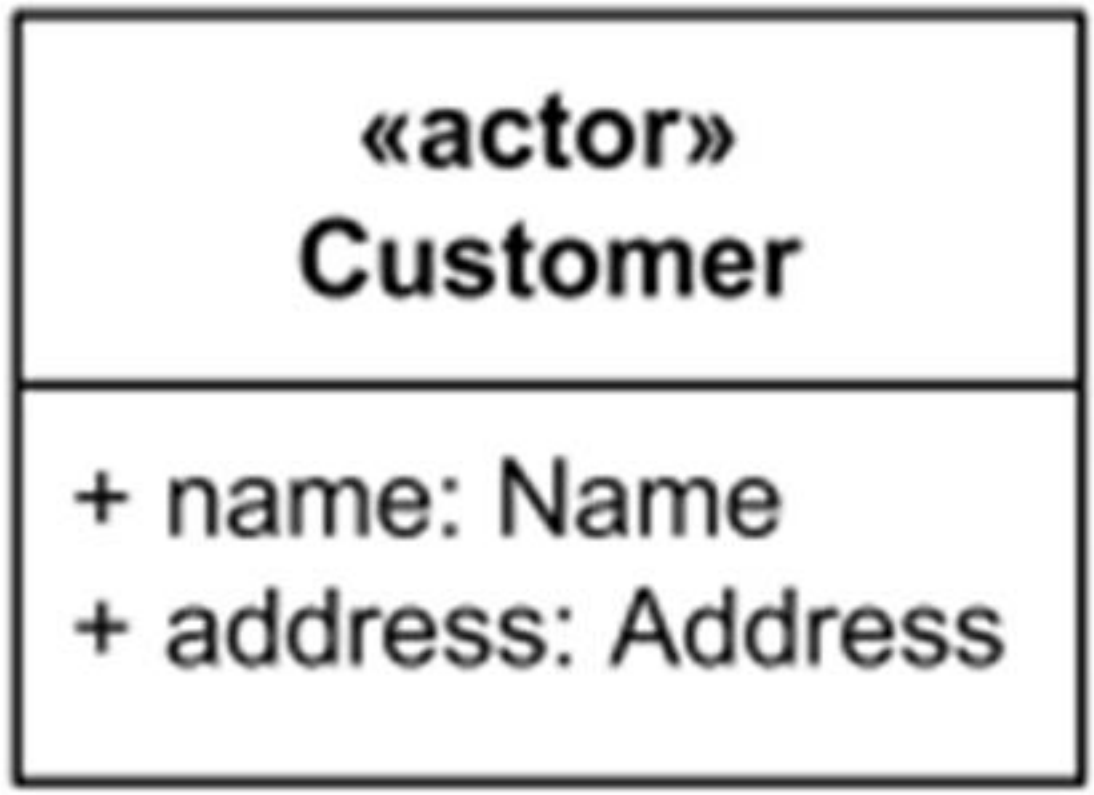
- An *actor* ...
  - is a role that the user plays with respect to the system
  - is associated with one or more use cases
  - is connected to use cases by associations
  - exists outside the system boundaries

Actors and Use Cases

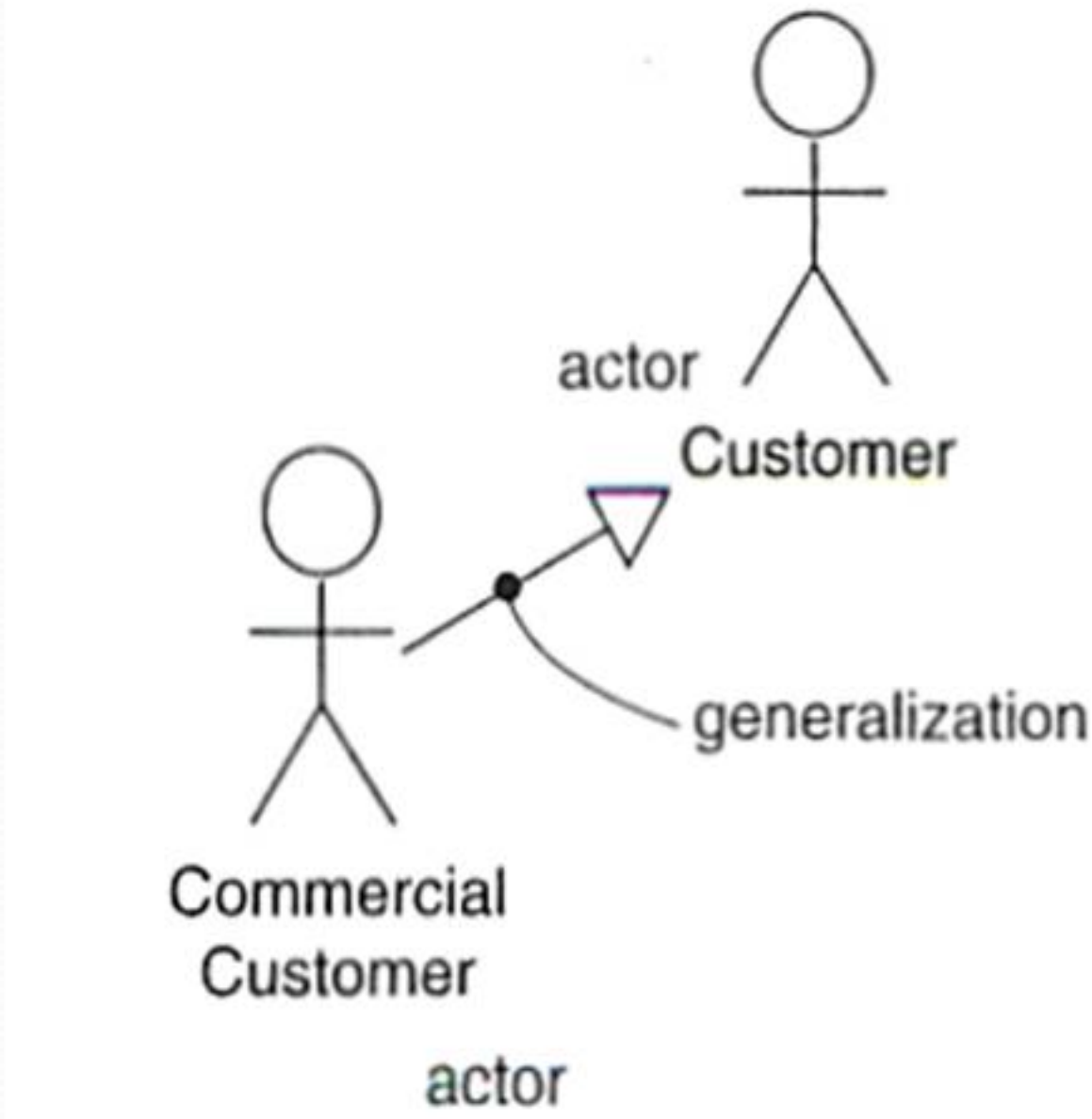


- Graphical representation of an actor (**stick man icon**) and a use case in UML
- An actor is labeled with its role name (should be noun)
- A Use Case is labeled with the name of its functionality (verb or a verb phrase) written inside the oval

- Actor names should follow guidelines for **classes**.
- The names of **abstract actors** should be shown in italics.
- An actor may also be shown as a **class** rectangle with the standard keyword **«actor»**



Actors and Use Cases



An actor may exist in a generalization relationships with other actors in the same way as classes

Use cases and Flow of Events

- A use case describes *what a system (or a subsystem, class, or interface) does but it does not* specify *how it does it*
- A use case does not describe the flow of events needed to carry out the use case.
- Flow of events can be described using informal text, pseudocode, or activity diagrams.
- Address *exception handling* (error conditions) when describing flow of events.

Use cases and Flow of Events

Buy a Product

Goal Level: Sea Level

Main Success Scenario:

1. Customer browses catalog and selects items to buy
2. Customer goes to check out
3. Customer fills in shipping information (address; next-day or 3-day delivery)
4. System presents full pricing information, including shipping
5. Customer fills in credit card information
6. System authorizes purchase
7. System confirms sale immediately
8. System sends confirming e-mail to customer

Extensions:

3a: Customer is regular customer

.1: System displays current shipping, pricing, and billing information

.2: Customer may accept or override these defaults, returns to MSS at step 6

6a: System fails to authorize credit purchase

.1: Customer may reenter credit card information or may cancel

Example Use Case text

Other common information to a use case.

- A **pre-condition** describes what the system should ensure is true before the system allows the use case to begin. This is useful for telling the programmers what conditions they don‘t have to check for in their code.
- A **guarantee** describes what the system will ensure at the end of the use case. Success guarantees hold after a successful scenario; minimal guarantees hold after any scenario.
- A **trigger** specifies the event that gets the use case started.

In addition each scenario includes the following:

- What actions the actors take
- What results or changes are caused by the system
- What feedback the actors receive
- Whether there are repeating activities, and what causes them to conclude
- A description of the logical flow of the scenario
- What causes the scenario to end
- **Each use case and each scenario has a name**

Use Case: Customer withdraws cash

Scenario: Successful cash withdrawal from checking

Preconditions: Customer is already logged in to system

Trigger: Customer requests "withdrawal"

Description: Customer chooses to withdraw cash from a checking account. There is sufficient cash in the account and in the ATM, and the network is up. The ATM asks the customer to indicate the amount of the withdrawal, and the customer asks for $300, a legal amount to withdraw at this time. The machine dispenses $300 and prints a receipt, and the customer takes the money and the receipt.

Post Conditions: Customer account is debited $300, and customer has $300 cash.

Generalization, Include and Extend

- A use case may have a relationship with other use cases
  - ***Generalization*** between use cases is used to extend the behavior of a parent use case.
  - An **<<include>>** relationship: Use case **explicitly incorporates** the behavior of another use case at a location specified in the base. (**delegation**)
  - An **extend** relationship: The base use case **implicitly incorporates** the behavior of another use case at a location specified indirectly by the extending use case. The behavior of base use case may be extended by the behavior of another use case under certain conditions

Generalization, Include and Extend

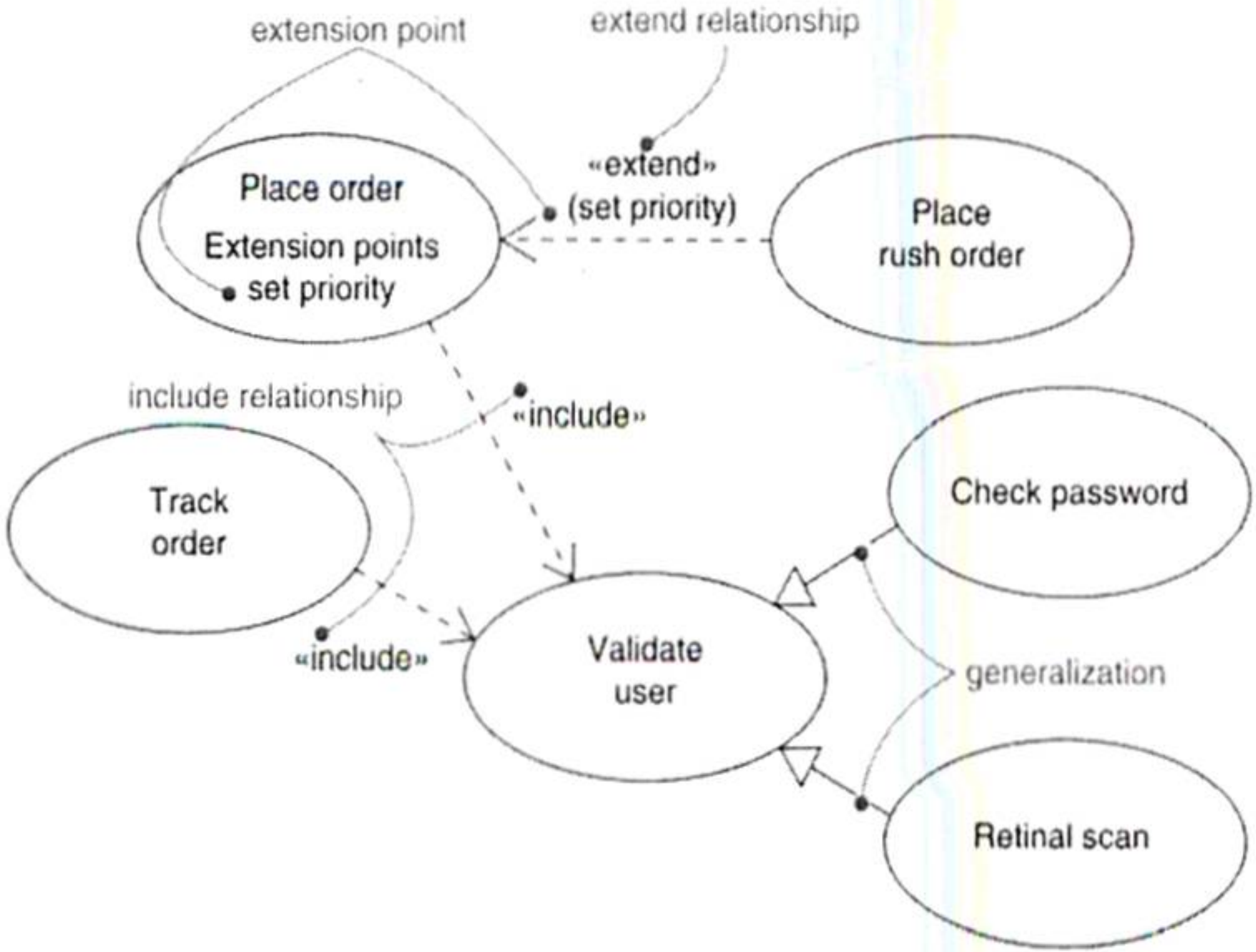


Table 5–1 Key Differences between *«include»* and *«extend»* Relationships[a]

	Included Use Case	Extending Use Case
Is this use case optional?	No	Yes
Is the base use case complete without this use case?	No	Yes
Is the execution of this use case conditional?	No	Yes
Does this use case change the behavior of the base use case?	No	Yes

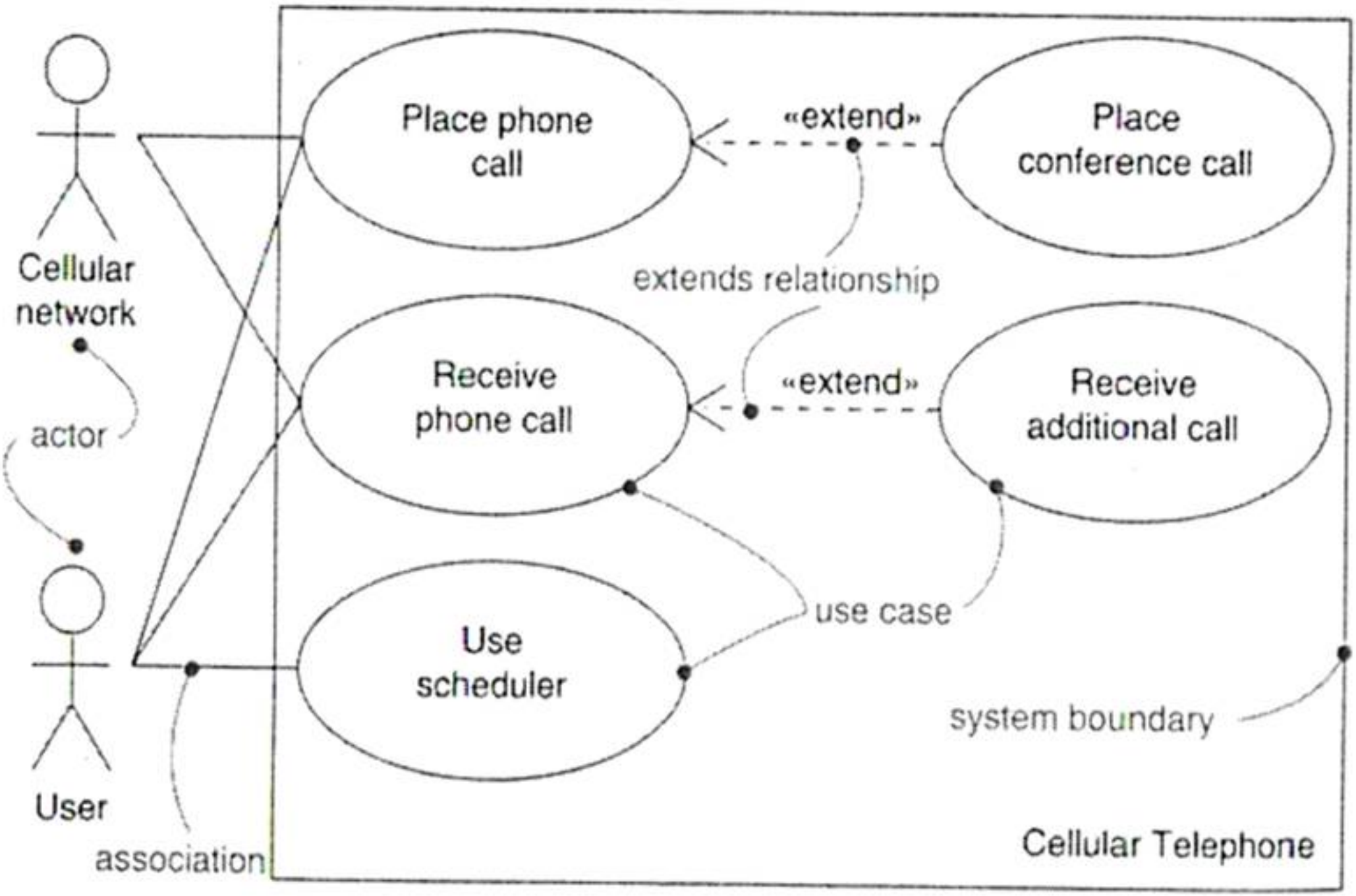
Use Cases

- Use Cases model the behavior of an element (system, subsystem, class)
- To model the behavior of an element
  - Identify the actors that interact with the element
  - Organize actors by identifying general and more specialized roles
  - For each actor consider the
    - primary ways in which that actor interacts with the element
    - exceptional ways in which each actor interacts with the element
  - Organize these behaviors as use cases, applying include and extend relationships to factor common behavior and distinguish exceptional behavior.

Use Case Diagrams

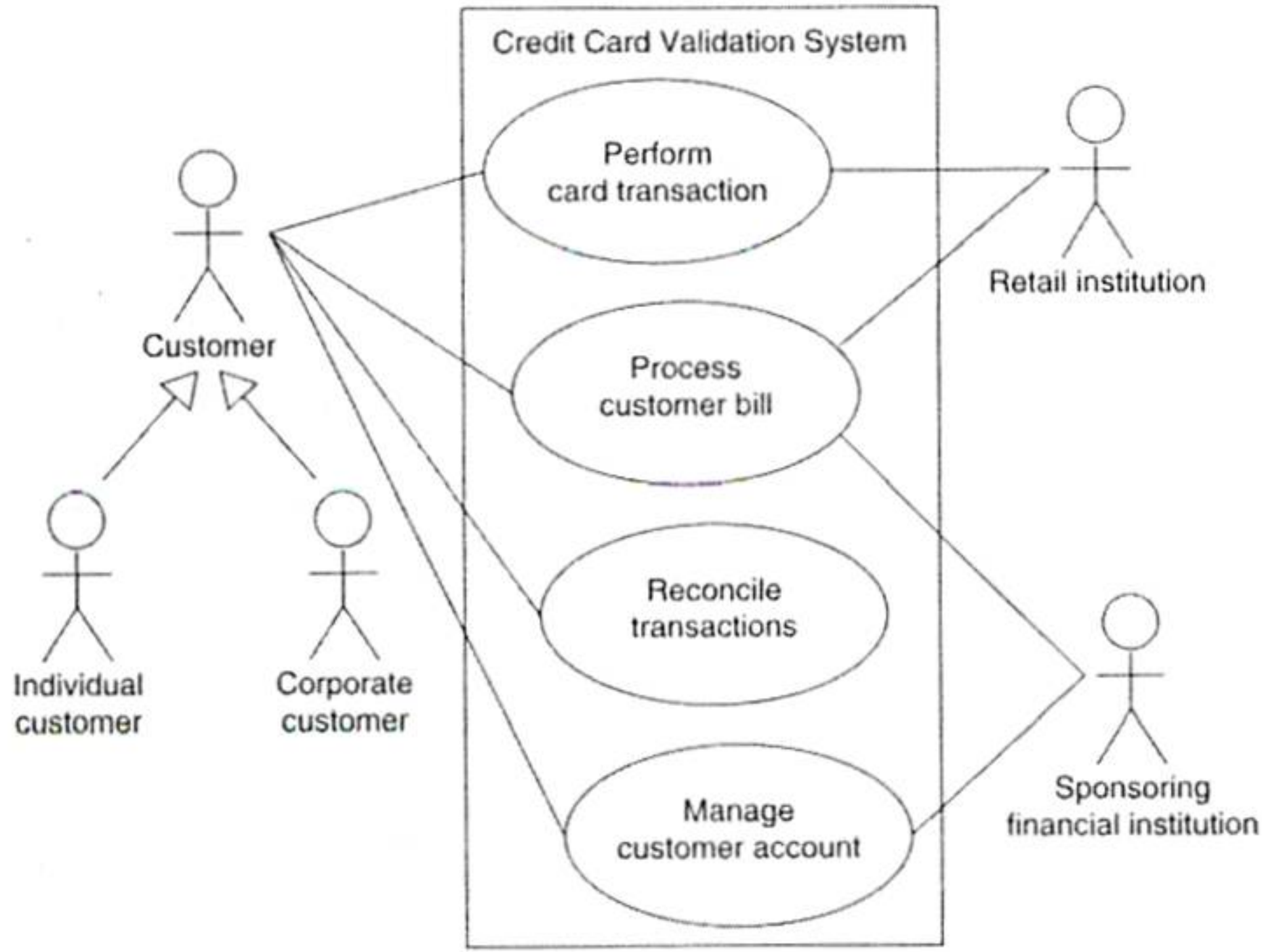
- A *use case diagram is a diagram that shows a set of use cases and actors and their* relationships
- Use case diagrams commonly contain
  - Use cases
  - Actors
  - Dependency, generalization, and association relationships

Use Case Diagram



A Use Case Diagram

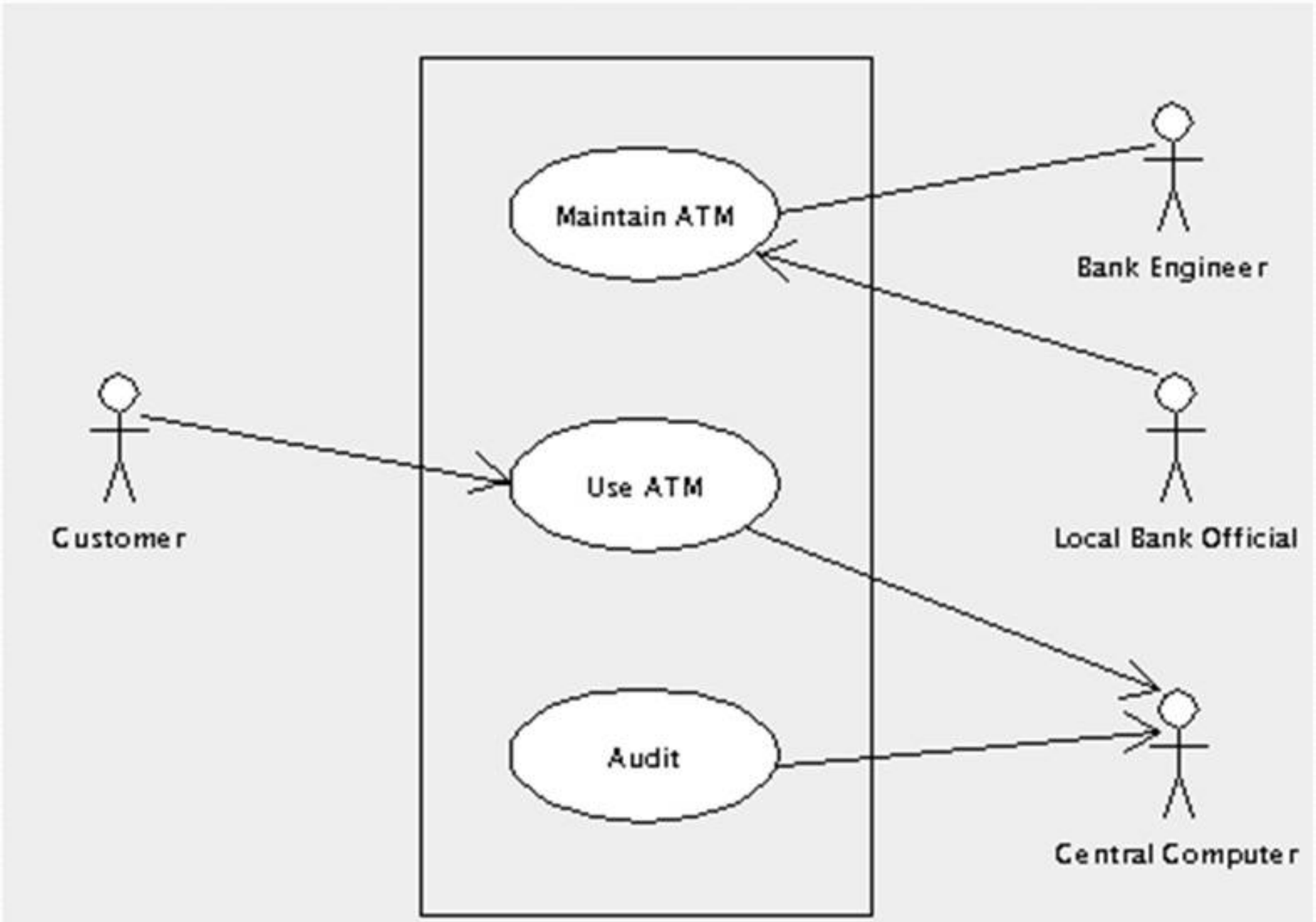
Use Case Diagrams



Modeling the Context of a System

Active and Passive Actors

- Active actors initiate interaction with the system. This can be shown by placing an arrow on the association from the actor pointing toward the use case.
- Interaction with passive actors is initiated by the system. This can be shown by placing an arrow on the association from the use case pointing toward the actor.


Maintain ATM
Use ATM
Audit
Customer
Bank Engineer
Local Bank Official
Central Computer

It can be useful to show the *multiplicity of associations between actors and use cases.*