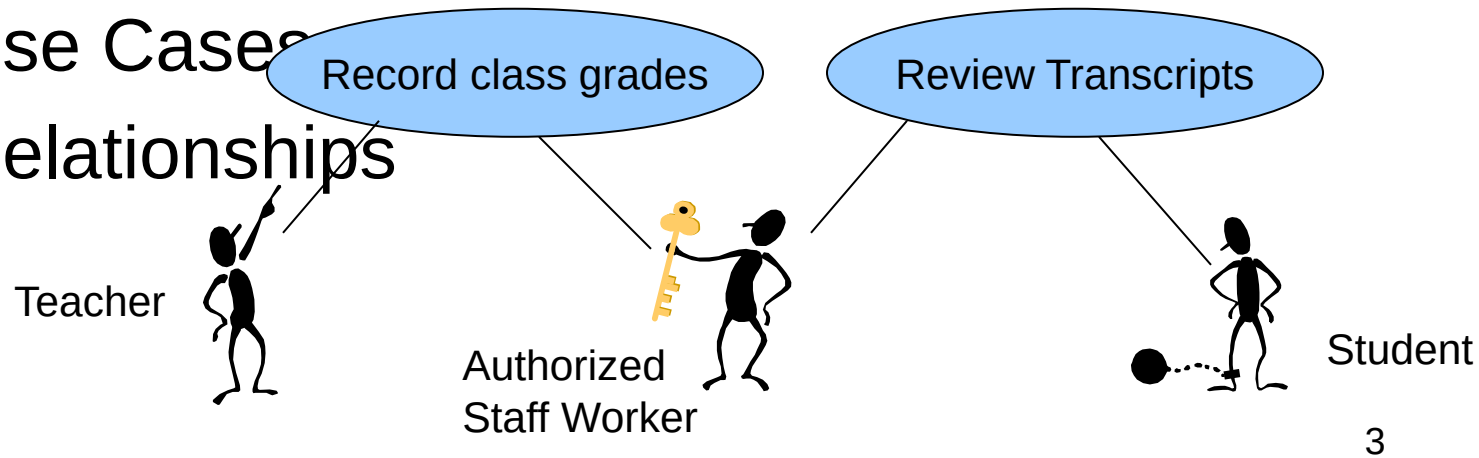
Use case diagrams

- behavior diagrams used to describe a set of actions (use cases)
- that some system or systems (subject) should or can perform in collaboration with one or more **external users** of the system (actors).
- Each use case should provide some observable and valuable result to the actors or other stakeholders of the system.

Use Case Diagrams

- Use Case Diagrams provide a visual way to document user goals and explore possible functionality
- Three primary modeling components:
  - Actors
  - Use Cases
  - Relationships



Actors

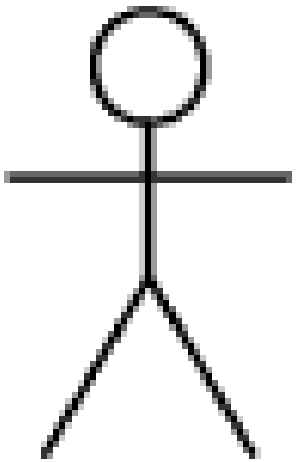
- An actor models an external entity which communicates with the system:
  - User
  - External system
  - Physical environment
- An actor has a unique name and an optional description.
- Examples:
  - Passenger: A person in the train
  - GPS satellite: Provides the system with GPS coordinates

Actors

All actors must have names according to the assumed role.

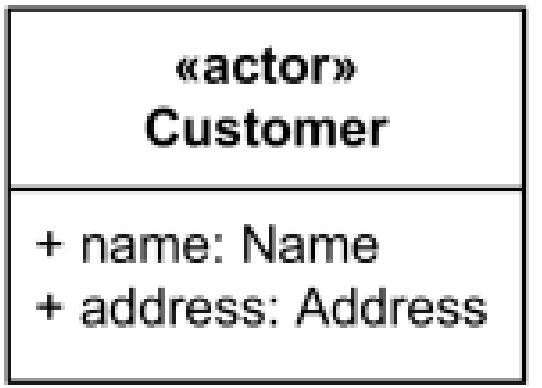
- Customer
- Web Client
- Student
- Passenger
- Payment System

Standard UML notation for actor is "**stick man**" icon with the name of the actor above or below of the icon.



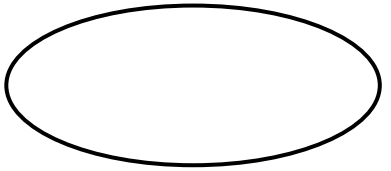
Custom icons that convey the kind of actor may also be used to denote an actor, such as using a separate icon(s) for non-human actors

- An actor may also be shown as a class rectangle with the standard keyword **«actor»**, having usual notation for class compartments, if needed.



Use Case

A use case represents a functionality provided by the system as an event flow.

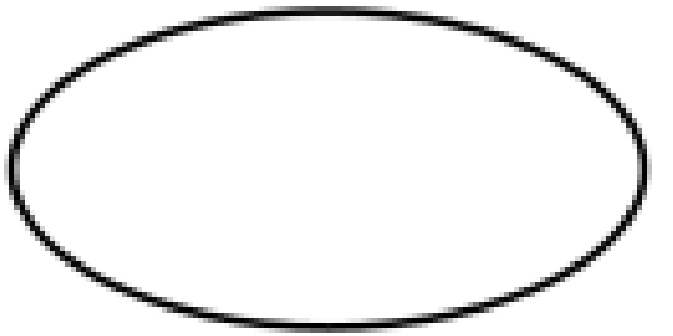


PurchaseTicket

A use case consists of:

- Unique name
- Participating actors
- Entry conditions
- Flow of events
- Exit conditions
- Special requirements

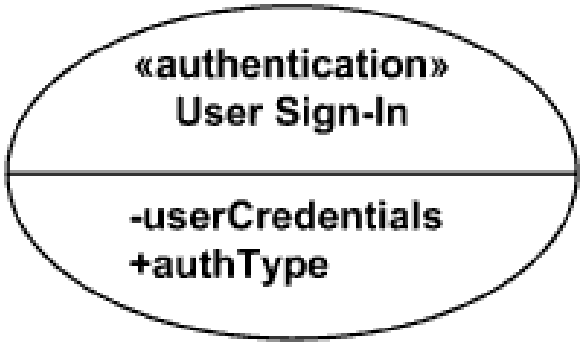
Use case is usually shown as an ellipse containing the name of the use case.



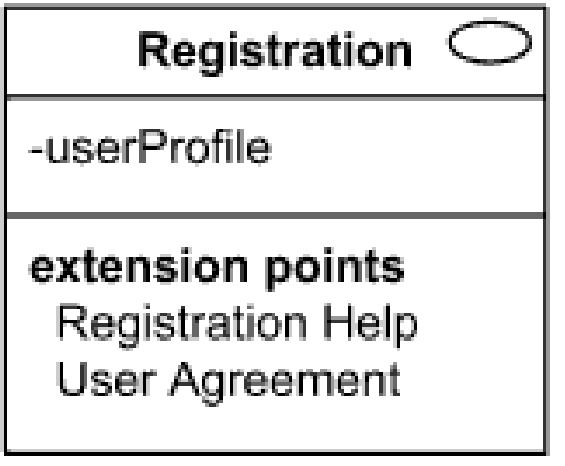
Transfer Funds

Use case could be shown with a **custom stereotype** above the name

Use case with extension points may be listed in a compartment of the use case with the heading **extension points**.



A use case can also be shown using the standard rectangle notation for classifiers with an ellipse icon in the upper right-hand corner of the rectangle and with optional separate list compartments for its features.

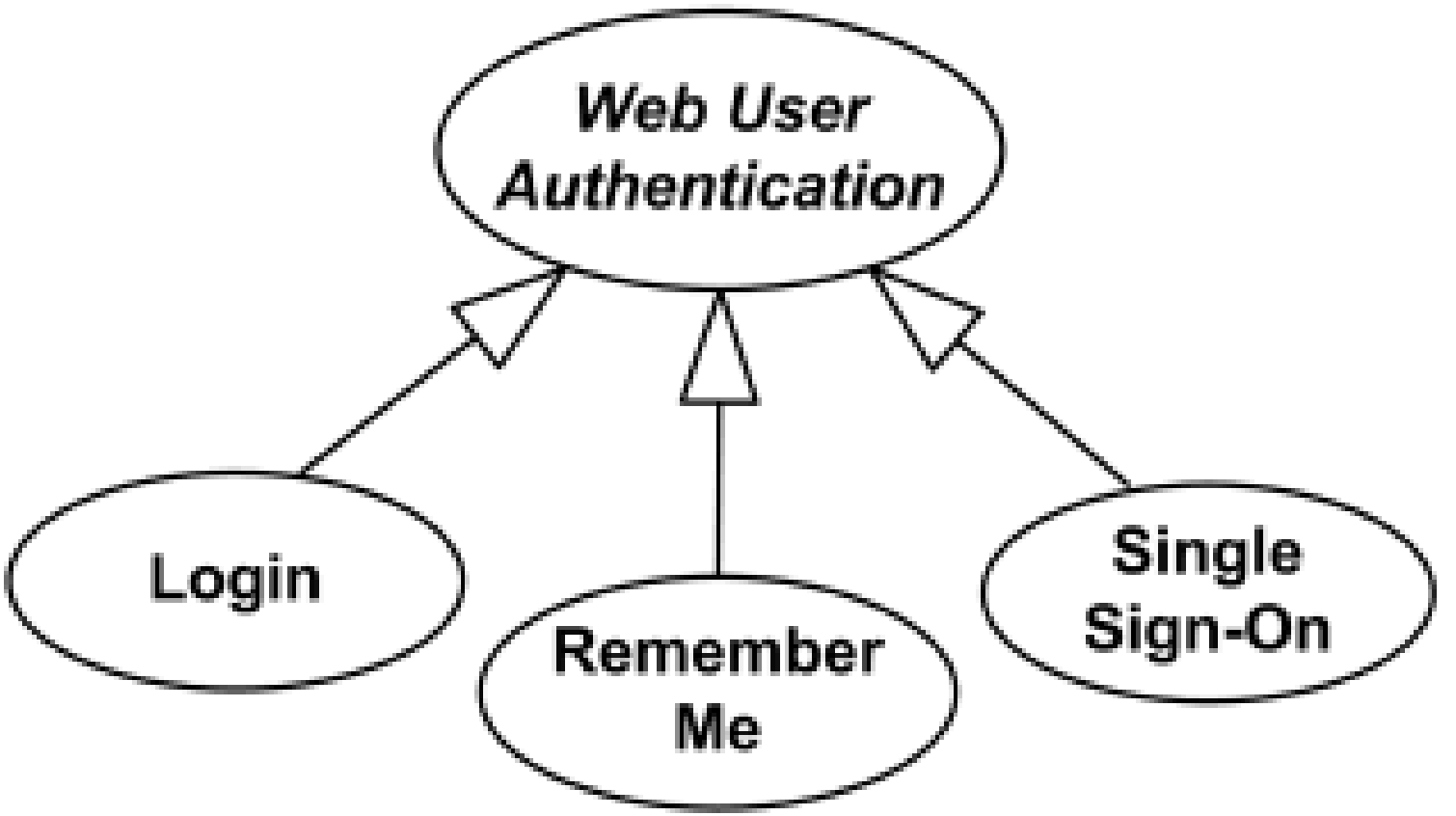


Relationships Between Use Cases

- Use cases could be organized using following relationships:
- generalization
- association
- extend
- include

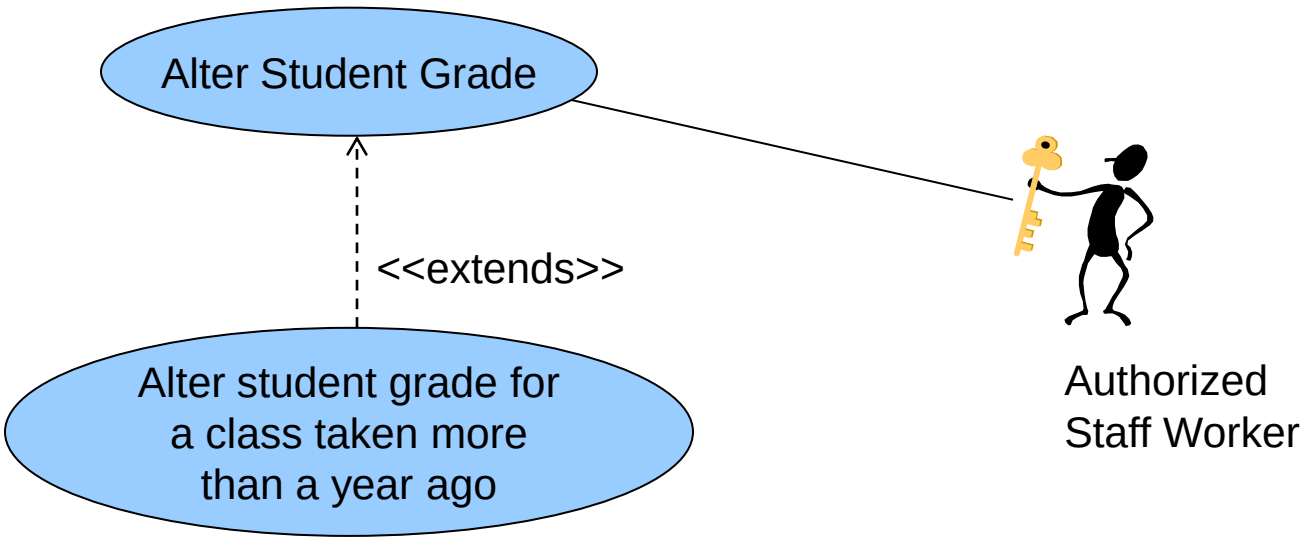
Generalization Between Use Cases

- child use case inherits properties and behavior of the parent use case and may override the behavior of the parent.
- Generalization is shown as a solid directed line with a large hollow triangle arrowhead, the same as for generalization between classifiers, directed from the more specific use case to the general use case.

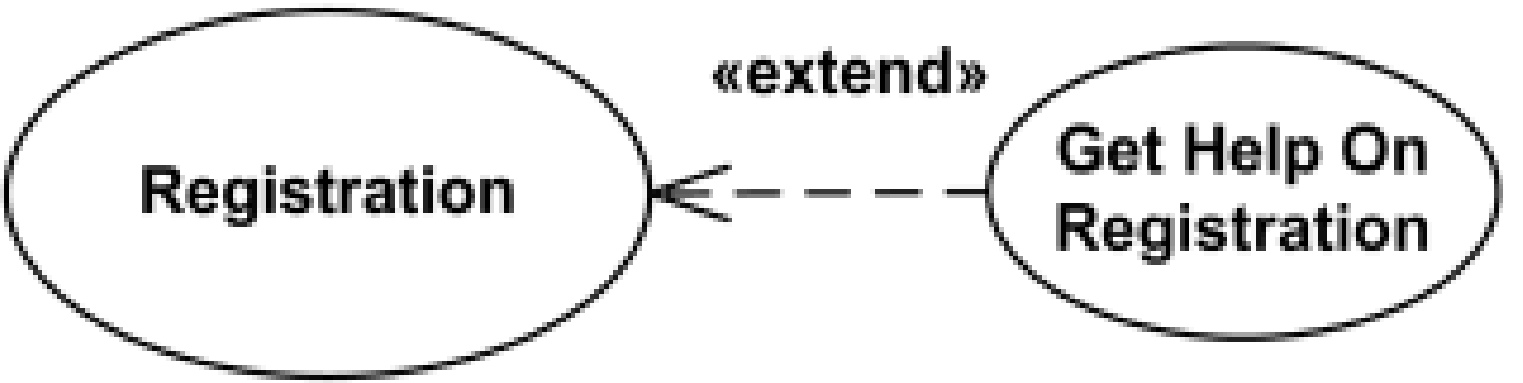

Web User Authentication
Login
Remember Me
Single Sign-On

Extend Relationship

- *Extends dependency*: defines a use-case that is a variation of another, usually for handling an abnormal situation



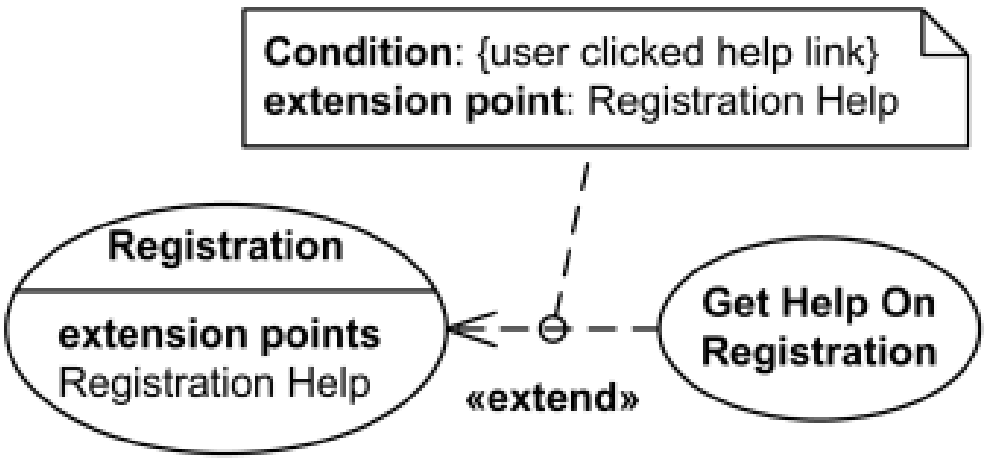
Extend relationship is shown as a dashed line with an open arrowhead directed from the **extending use case** to the **extended (base) use case**. The arrow is labeled with the keyword **«extend»**.



Registration use case is meaningful on its own.

It could be extended with optional **Get Help On Registration** use case

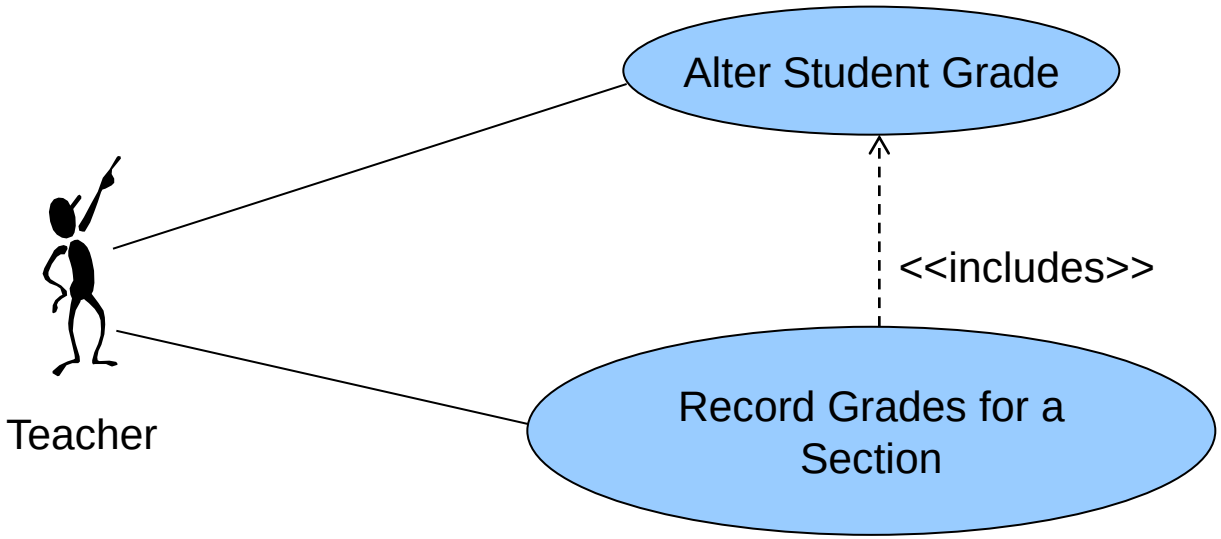
- The **condition** of the extend relationship as well as the references to the extension points are optionally shown in a comment note attached to the corresponding extend relationship.



- Registration use case is conditionally extended by Get Help On Registration
  use case in extension point Registration Help

Includes

- *Includes Dependency*: Defines how one use case can invoke behavior defined by another use case

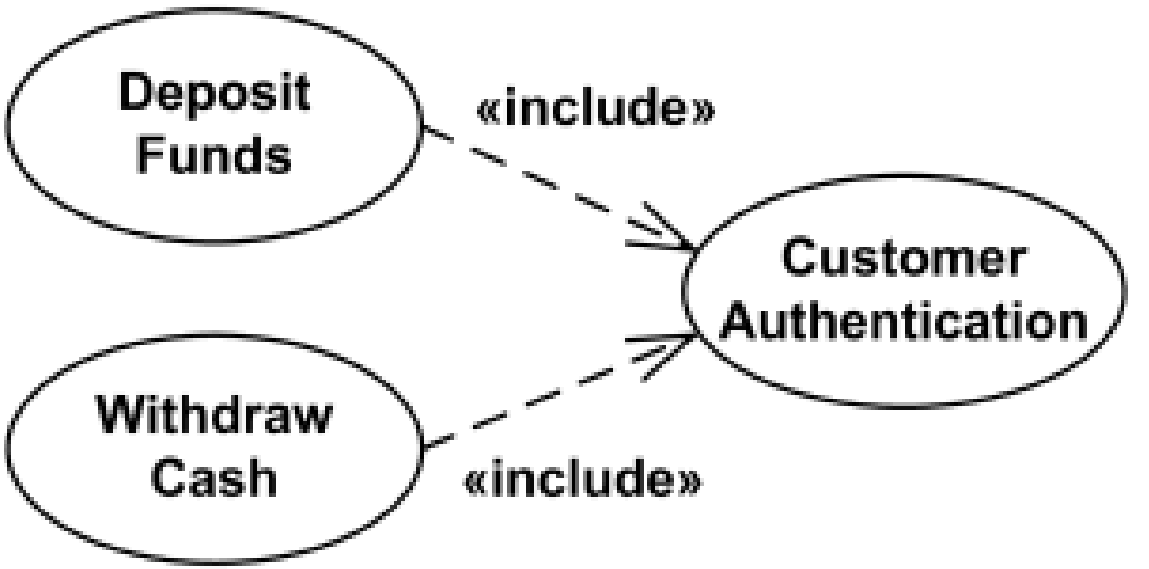


- **Include** relationship between use cases is shown by a dashed arrow with an open arrowhead from the including (base) use case to the included (common part) use case. The arrow is labeled with the keyword **«include»**.

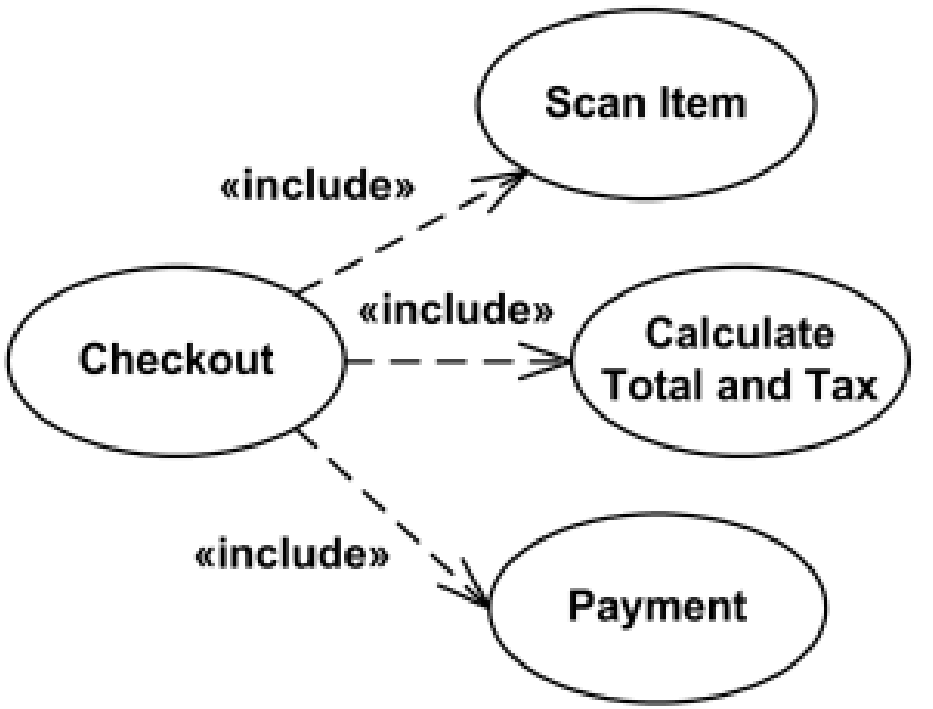
The **include** relationship could be used:

- when there are **common parts** of the behavior of two or more use cases,
- to simplify large use case by splitting it into several use cases.

Both **Deposit Funds** and **Withdraw Cash** use cases require (include) **Customer Authentication** use case.



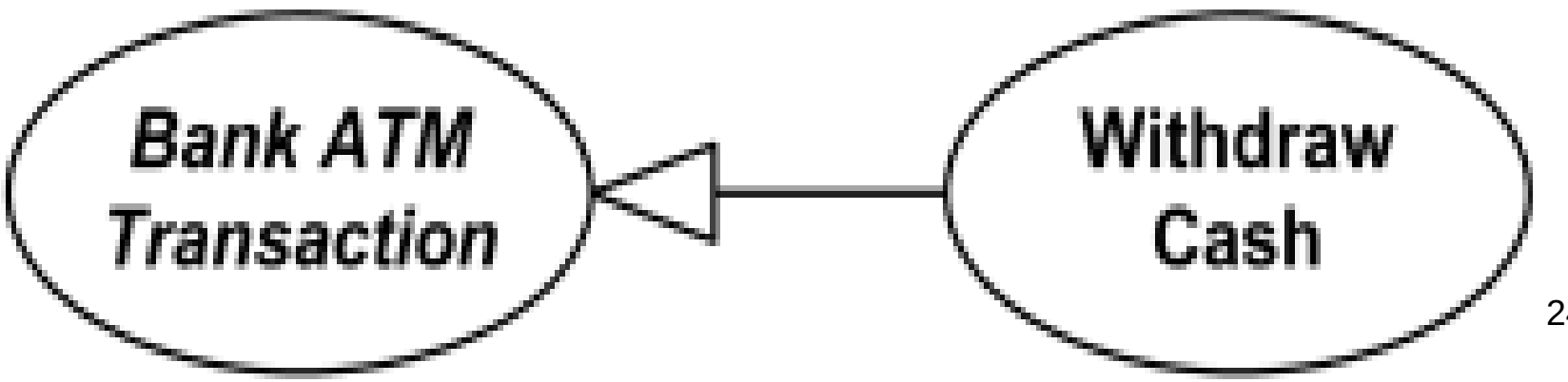
Large and complex use case could be simplified by splitting it into several use cases
including use case becomes incomplete by itself and requires included use cases to be complete.



Differences among generalization, include & extend

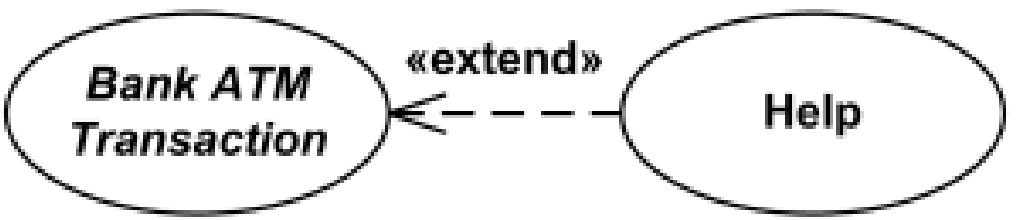
Generalization

- Base use case could be abstract use case (incomplete) or concrete (complete).
- Specialized use case is required, not optional, if base use case is abstract.
- No explicit location to use specialization.
- No explicit condition to use specialization.



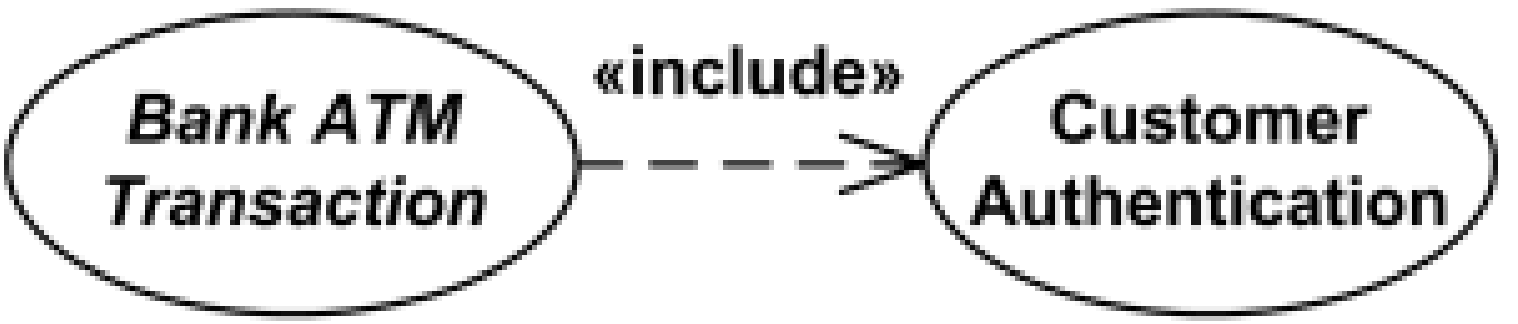
Extend

- Base use case is complete (concrete) by itself, defined independently.
- Extending use case is optional, supplementary.
- Has at least one explicit extension location.
- Could have optional extension condition.



Include

- Base use case is incomplete (abstract use case).
- Included use case required, not optional.
- No explicit inclusion location but is included at some location.
- No explicit inclusion condition.



Use Case Diagram Example