UML Use Case

Use cases allow to capture requirements of systems under design or consideration, describe functionality provided by those systems, and determine the requirements the systems pose on their environment.

A **use case** is a kind of behaviored classifier that specifies a [complete] unit of [useful] functionality performed by [one or more] subjects to which the use case applies in collaboration with one or more actors, and which [for complete use cases] yields an observable result that is of some value to those actors [or other stakeholders] of each subject.

Naming Use Cases

UML Specification provides no guidelines on use case names. The only requirement is that each use case must have a name. We should just follow use case definition to give some name to that unit of functionality performed by a system which provides some observable and useful result to an actor. Examples of use case names:

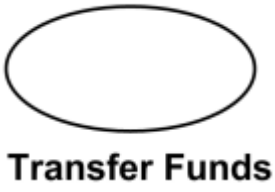
- Make Purchase
- Place Order
- Update Subscription
- Transfer Funds
- Hire Employee
- Manage Account

Notation

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

User Registration ***use case***

Name of the use case could also be placed below the ellipse.



Transfer Funds ***use case***

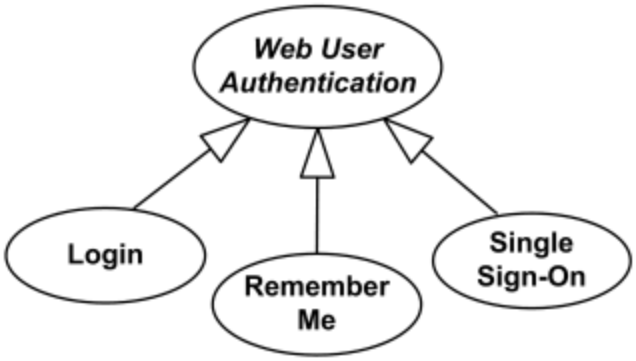
Relationships Between Use Cases

Use cases could be organized using following relationships:

- **generalization**
- **association**
- **extend**
- **include**

Generalization Between Use Cases

Generalization between use cases is similar to generalization between classes – 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.



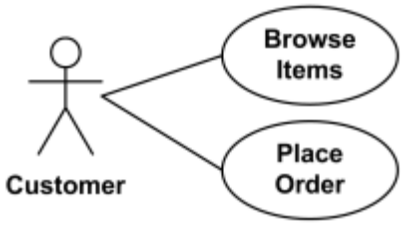
*Web User Authentication use case is **abstract use case***

specialized by Login, Remember Me and Single Sign-On use cases

UML Association Between Actor and Use Case

Each **use case** represents a unit of useful functionality that **subjects** provide to **actors**. An **association** between an actor and a use case indicates that the actor and the use case somehow interact or communicate with each other.

Only **binary associations** are allowed between actors and use cases.
An actor could be associated to one or several use cases.



Customer actor is associated with two use cases -Browse Items and Place Order.

A use case may have one or several associated actors.

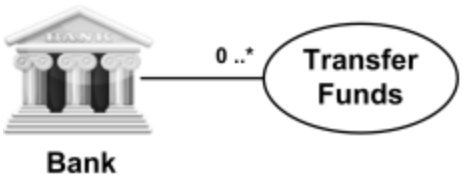
Manage Account use case is associated with Customer and Bank actors.

Multiplicity of Association Ends

UML allows the use of **multiplicity** at one or both ends of an association between the actor and the use case.

Multiplicity of a Use Case

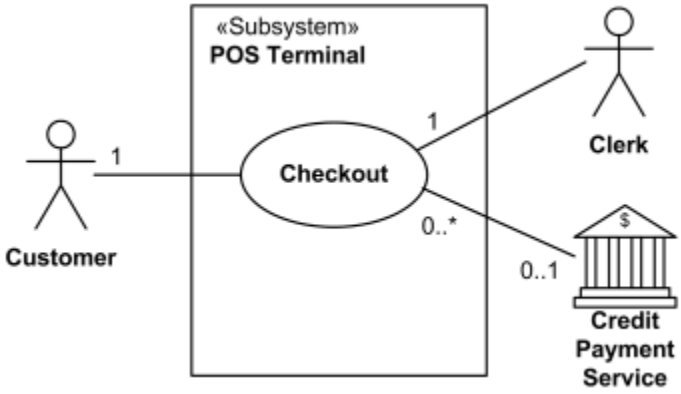
When an actor has an association to a **use case** with a **multiplicity** that is greater than one at the use case end, it means that a given actor can be involved in **multiple use cases** of that type.



Bank actor is involved in multiple Transfer Funds use cases.

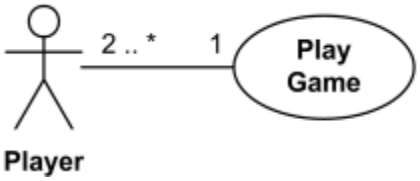
Multiplicity of an Actor

Required actor may be explicitly denoted using multiplicity 1 or greater. UML 2.5 also allows actor to be optional. Multiplicity **0..1** of actor means that the actor may or may not participate in any of their associated use cases.



Checkout use case requires Customer actor, hence the 1 multiplicity of Customer. The use case may not need Credit Payment Service (for example, if payment is in cash), thus the 0..1 multiplicity.

When a use case has an association to an **actor** with a **multiplicity** that is greater than one at the actor end, it means that more than one actor instance is involved in the use case.



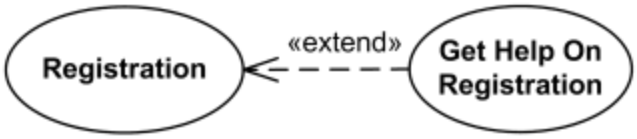
Two or more Player actors are involved in the Play Game use case.

Each Player participates in one Play Game.

UML Use Case Extend

Extending use case typically defines **optional behavior** that is not necessarily meaningful by itself. The extend relationship is **owned** by the extending use case. The same extending use case can extend more than one use case, and extending use case may itself be extended.
The extension takes place at one or more **extension points** defined in the **extended use case**.

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 complete and meaningful on its own.*

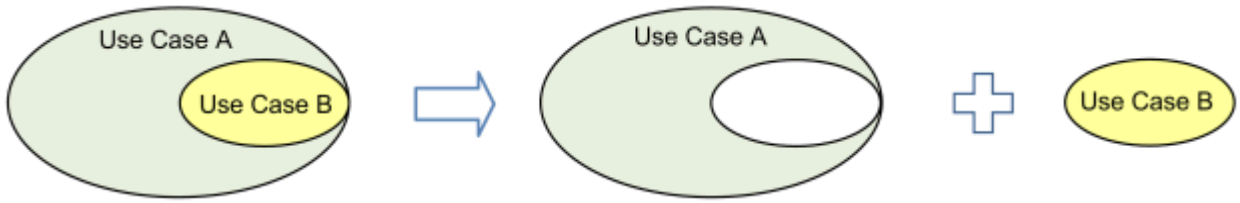
It could be extended with optional ***Get Help On Registration*** *use case.*

UML Use Case Include

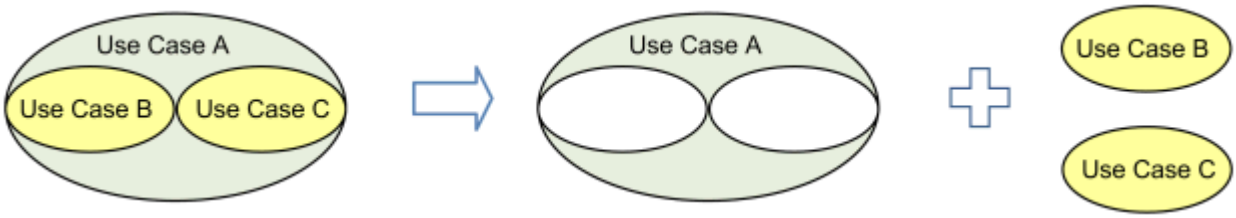
Use case include is a **directed relationship** between two **use cases** which is used to show that behavior of the **included** use case (the addition) is inserted into the **behavior** of the **including** (the base) use case.
The **include** relationship could be used:

- to simplify large use case by splitting it into several use cases,
- to extract **common parts** of the behaviors of two or more use cases.

A large use case could have some behaviors which might be detached into distinct smaller use cases to be included back into the base use case using the UML **include** relationship. The purpose of this action is modularization of behaviors, making them more manageable.

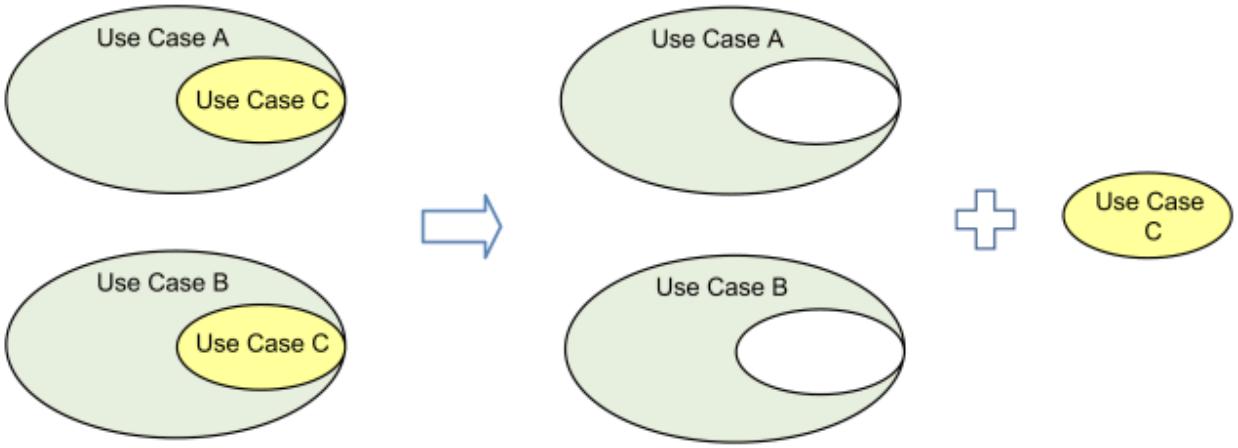


Use case B is extracted from larger use case A into a separate use case.



Use cases B and C are extracted from larger use case A into separate use cases.

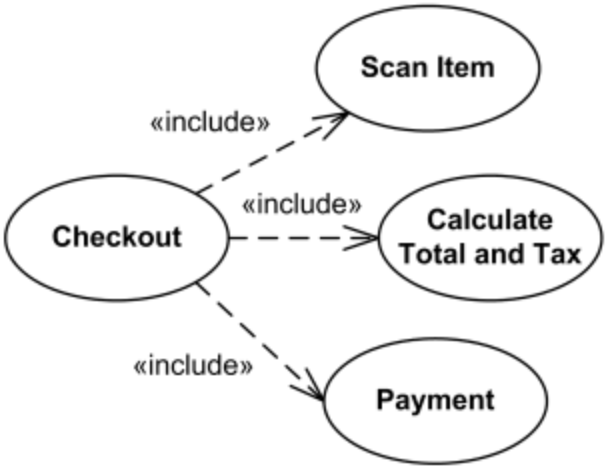
When two or more use cases have some **common behavior**, this common part could be extracted into a separate use case to be included back by the use cases with the UML **include** relationship.



Use case C is extracted from use cases A and B to be reused by both use cases using UML include relationship.

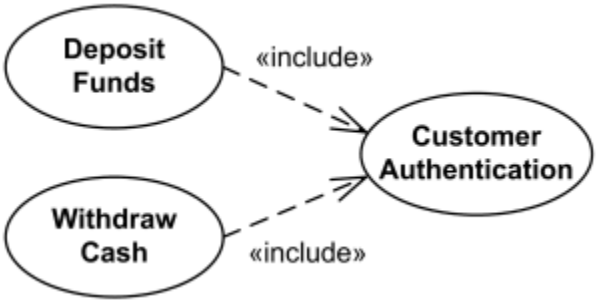
Including use case depends on the addition of the included use case, which is required and not optional. It means that including use case is **not complete** by itself, and so it would make sense to refer to the including use cases as **abstract use cases**.

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».



Checkout use case includes several use cases - Scan Item, Calculate Total and Tax, and Payment

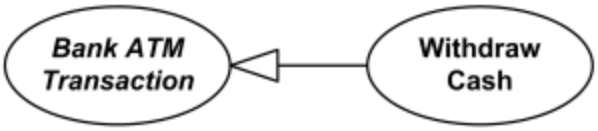
Large and complex Checkout use case has several use cases extracted, each smaller use case describing some logical unit of behavior. Note, that including Checkout use case becomes incomplete by itself and requires included use cases to be complete.



***Deposit Funds** and **Withdraw Cash** use cases include **Customer Authentication** use case.*

Use Case Relationships Compared

Generalization



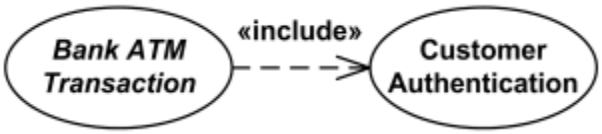
Base use case could be **abstract use case** (incomplete) or concrete (complete).
Specialized use case is required, not optional, if base use case is abstract.
No explicit location to use specialization.
No explicit condition to use specialization.

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.