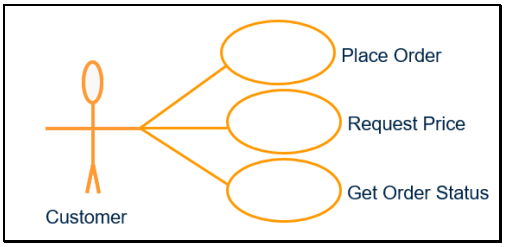
Use case

A formal way of representing how a business system interacts with its environment.

Use Case Diagrams

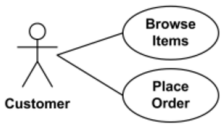
Use Case Diagrams model the functionality of system by using Actors and Use Cases:

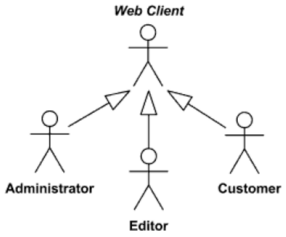


Notations used in the Use case diagram with examples.

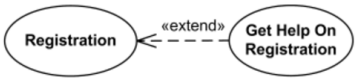
Actor	
 Student *Student* ***actor***.	Standard UML icon for actor is "stick man" icon with the name of the actor above or below the icon. Actor names should follow the capitalization and punctuation guidelines for **classes**. The names of abstract actors should be shown in italics. All actors must have names.

Use Case	
 User Registration *User Registration* ***Use Case***.	Every use case must have a name. Use case is shown as an ellipse containing the name of the use case.

Association	
 Browse Items Place Order Customer ***Actor*** *Customer associated with two* ***use cases***.	An **association** between an actor and a use case indicates that the actor and the use case **communicate** with each other. An actor could be associated to one or several use cases.

Generalization between actors	
 Web Client Administrator Editor Customer *Web Client actor is abstract superclass for Administrator, Editor and Customer.*	**Generalization** between actors is rendered as a solid directed line with a large arrowhead (the same as for generalization between classes).

Extend



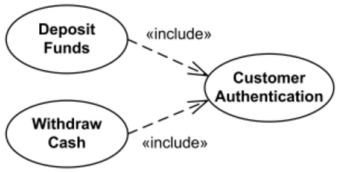
***Registration** use case is complete and meaningful on its own. It could be extended with optional **Get Help On Registration** use case.*

Extend is a **directed relationship** that specifies how and when the behavior defined in usually supplementary (optional) **extending use case** can be inserted into the behavior defined in the **extended use case**.

Extended use case is meaningful on its own, it is **independent** of the extending use case. **Extending** use case typically defines **optional** behavior that is not necessarily meaningful by itself.

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

Include



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

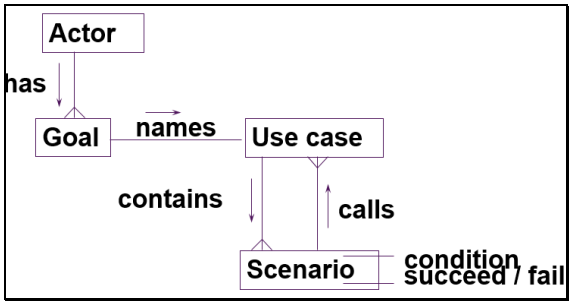
An **include** relationship is a **directed relationship** between two **use cases** when required, not optional behavior of the **included** use case is inserted into the behavior of the **including** (base) use case.

The **include** relationship is analogous to a subroutine call or macro and 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.

An **include** relationship between use cases is shown by a dashed arrow with an open arrowhead from the base use case to the included use case. The arrow is labeled with the keyword **«include»**.

Use cases: Actors and Goals



Other Use case aspects	Meaning
Alternate paths	**Result positive: An Alternate Flow is a step or a sequence of steps that achieves the use case's goal following different steps than described in the main success scenario. But the goal is achieved finally.**
Exception	**Result negative: An Exception is anything that leads to NOT achieving the use case's goal.**
Error	**Errors are when things unexpectedly go wrong. They can result from malformed data, bad programs or logic errors, or broken hardware**
Precondition	**Constraint on when a use case can start**
Post condition	**This should be true regardless of which alternative flows were executed**

Steps in Use Case Modeling

1. Identify the actors and their goals
2. List the use cases and sketch the system through usecase diagram.
3. Write failure conditions / alternatives as extensions
4. Follow the failure till it ends or rejoins
5. Note the data variations
6. Deferred variations note cases that must be handled eventually, by lower-level use cases
7. Useful for tracking requirements at high level.