It has a wide range of Applications e.g., distributed systems, analysis, system design, and deployment

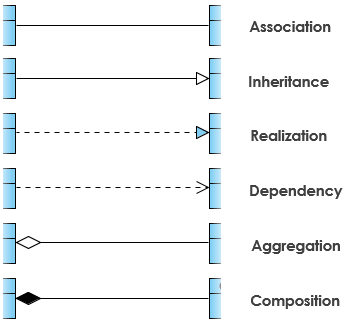
Structure diagrams and behavior diagrams

Structure Diagrams split into multiple diagrams:

* Composite Structure Diagram
* Deployment Diagram
* Package Diagram
* Profile Diagram
* Class Diagram
* Object Diagram
* Component Diagram

Behavioral Diagram splits into multiple Diagrams:

* activity Diagram
* Use case Diagram
* State machine Diagram
* Interaction Diagram
* Sequence Diagram
* Communication Diagram
* Timing Diagram

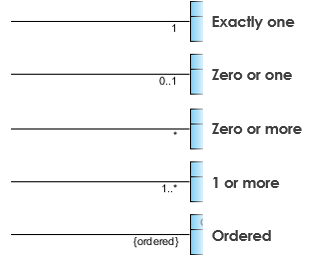


+ denotes public attributes of operations

- Denotes private attributes

# Denotes protected attributes or operations

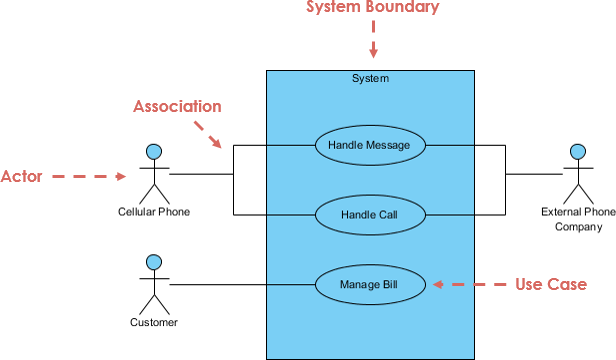
Class diagram is used in almost all OO, it describes the types of objects in the system and different types of static relationships that exist withing them. The three most important types of relationships in Class diagrams



Use Case Diagram

Use case diagrams are typically developed in the early stage of development and people often apply use case modeling for the following purposes:

* Specify the context of a system
* Capture the requirements of a system
* Validate a systems architecture
* Drive implementation and generate test cases
* Developed by analysts together with domain experts

A standard form of use case diagram is defined in the Unified Modeling Language as shown in the Use Case Diagram example below:  


Actor:

* Someone interacts with use case (system function)
* Named by noun
* Actor plays a role in the business
* Similar to the concept of user, but a user can play different roles
* For example:
* A professor can be instructor and also researcher
* Plays 2 roles with two systems
* Actor triggers use case
* Actor has a responsibility toward the system inputs and Actor has expectations from the system

Use Case

* System function (process – automated or manual)
* Named by verb + noun (or noun phrase)
* To do something
* Each actor must be linked to a use case, while some use cases may not be linked to actors

Communication link

* The participation of an actor in a use case is shown by connecting an actor to a use case by a solid link
* Actors may be connected to use cases by associations, indicating that the actor and the use case communicate with one another using messages

Boundary of System

* The system boundary is potentially the entire system as defined in the requirements document
* For large and complex systems, each module may be the system boundary
* For example, for and ERP system for an organization, each of the modules such as personnel, payroll, account, etc.
* Can form a system for use cases specific to each of these business functions
* The entire system can span all of these modules depicting the overall system boundary

Structuring Use case diagram with Relationships

Use cases share different kinds of relationships. Defining the relationship between two use cases is the decision of the software analysts of the use case diagram. A relationship between two use cases is basically modeling the dependency between the ttwo use cases. The reuse of an existing use case by using different types of relationships reduces the overall effort required in developing a system. Use case relationships are listed as the following:

* Use Case Relationship
* Visual representation

Extends:

* Indicates that an "Invalid Password" use case may include (subject to specified in the extension) the behavior specified by base use case "Login Account".
* Depict with a directed arrow having a dotted line. The tip of arrowhead points to the base use case and the child use case is connected at the base of the arrow.
* The stereotype "<<extends>>" identifies as an extend relationship



Include:

* When a use case is depicted as using the functionality of another use case, the relationship between the use cases is named as include or uses relationship.
* A use case includes the functionality described in another use case as a part of its business process flow.
* A uses relationship from base use case to child use case indicates that an instance of the base use case will include the behavior as specified in the child use case.
* An include relationship is depicted with a directed arrow having a dotted line. The tip of arrowhead points to the child use case and the parent use case connected at the base of the arrow.
* The stereotype "<<include>>" identifies the relationship as an include relationship.



Generalization:

* A generalization relationship is a parent-child relationship between use cases.
* The child use case is an enhancement of the parent use case.
* Generalization is shown as a directed arrow with a triangle arrowhead.
* The child use case is connected at the base of the arrow. The tip of the arrow is connected to the parent use case.

