UML - An Overview

Before we begin to look at the theory of the UML, we are going to take a very brief run through some of the major concepts of the UML.

The first thing to notice about the UML is that there are a lot of different diagrams (models) to get used to. The reason for this is that it is possible to look at a system from many different viewpoints. A software development will have many stakeholders playing a part.

For Example:

- Analysts
- Designers
- Coders
- Testers
- QA
- The Customer
- Technical Authors

All of these people are interested in different aspects of the system, and each of them require a different level of detail. For example, a coder needs to understand the design of the system and be able to convert the design to a low level code. By contrast, a technical writer is interested in the behavior of the system as a whole, and needs to understand how the product functions. The UML attempts to provide a language so expressive that all stakeholders can benefit from at least one UML diagram.

Here's a quick look at each one of these 13 diagrams in as shown in the UML 2 Diagram Structure below:

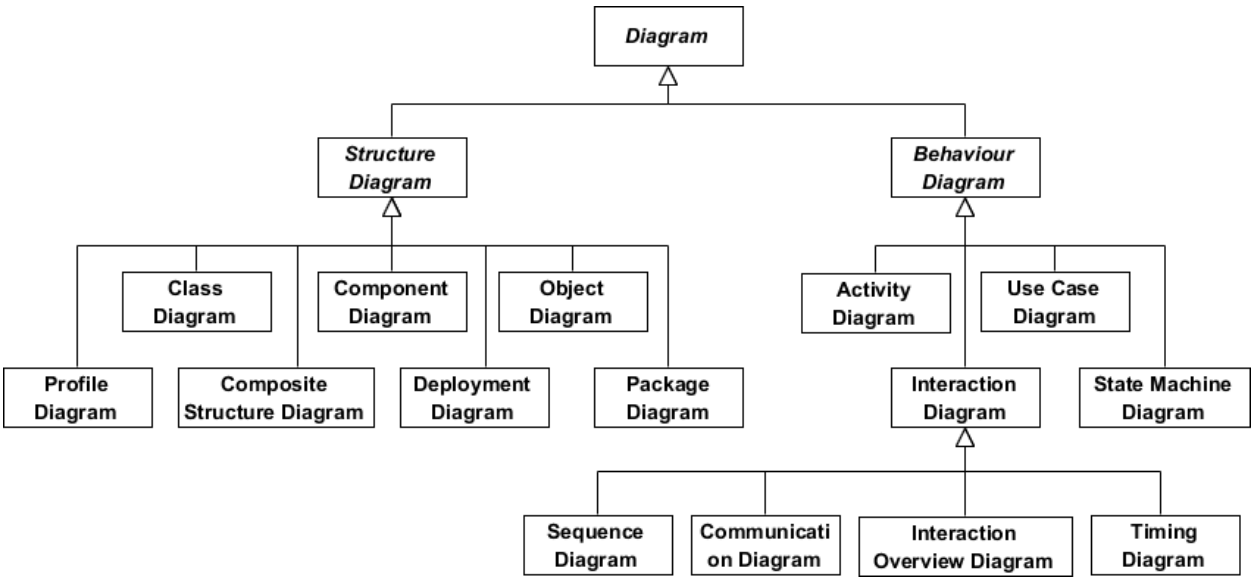
Structure diagrams show the static structure of the system and its parts on different abstraction and implementation levels and how they are related to each other. The elements in a structure diagram represent the meaningful concepts of a

system, and may include abstract, real world and implementation concepts, there are seven types of structure diagram as follows:

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

Behavior diagrams show the **dynamic behavior** of the objects in a system, which can be described as a series of changes to the system over **time**, there are seven types of behavior diagrams as follows:

- **Use Case Diagram**
- **Activity Diagram**
- **State Machine Diagram**
- **Sequence Diagram**
- **Communication Diagram**
- **Interaction Overview Diagram**
- **Timing Diagram**



What is a Class Diagram?

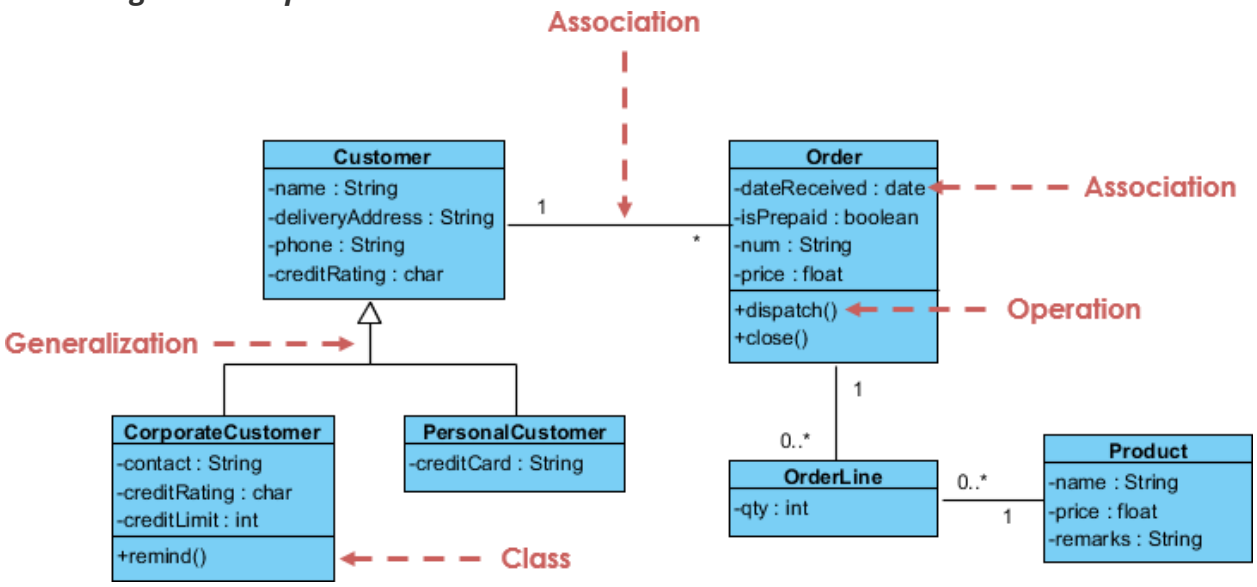
The class diagram is a central modeling technique that runs through nearly all object-oriented methods. This diagram describes the types of objects in the system and various kinds of static relationships which exist between them.

Relationships

There are three principal kinds of relationships which are important:

1. **Association** - represent relationships between instances of types (a person works for a company, a company has a number of offices.
2. **Inheritance** - the most obvious addition to ER diagrams for use in OO. It has an immediate correspondence to inheritance in OO design.
3. **Aggregation** - Aggregation, a form of object composition in object-oriented design.

Class Diagram Example

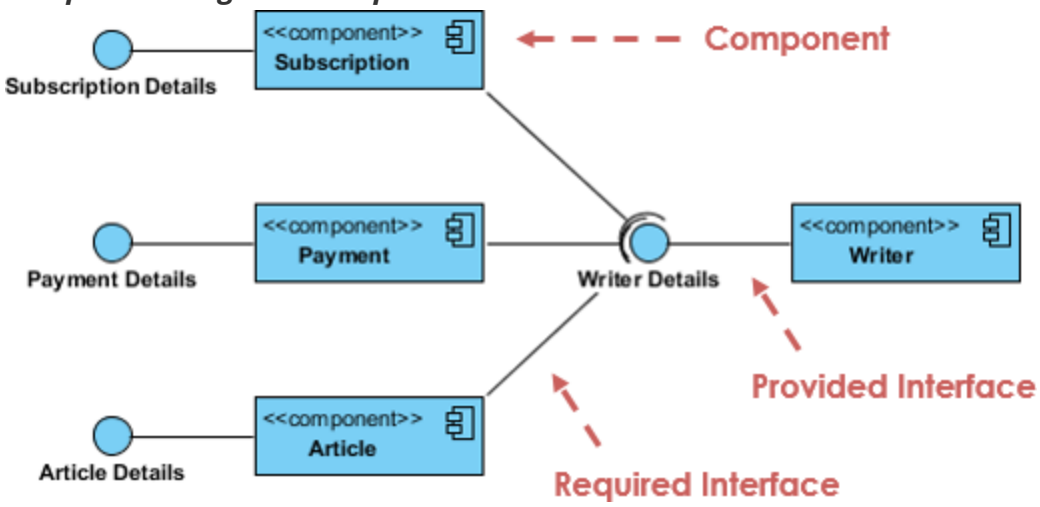


For more details about Class Diagram, please read the article **What is Class Diagram?**

What is Component Diagram?

In the Unified Modeling Language, a component diagram depicts how components are wired together to form larger components or software systems. It illustrates the architectures of the software components and the dependencies between them. Those software components including run-time components, executable components also the source code components.

Component Diagram Example

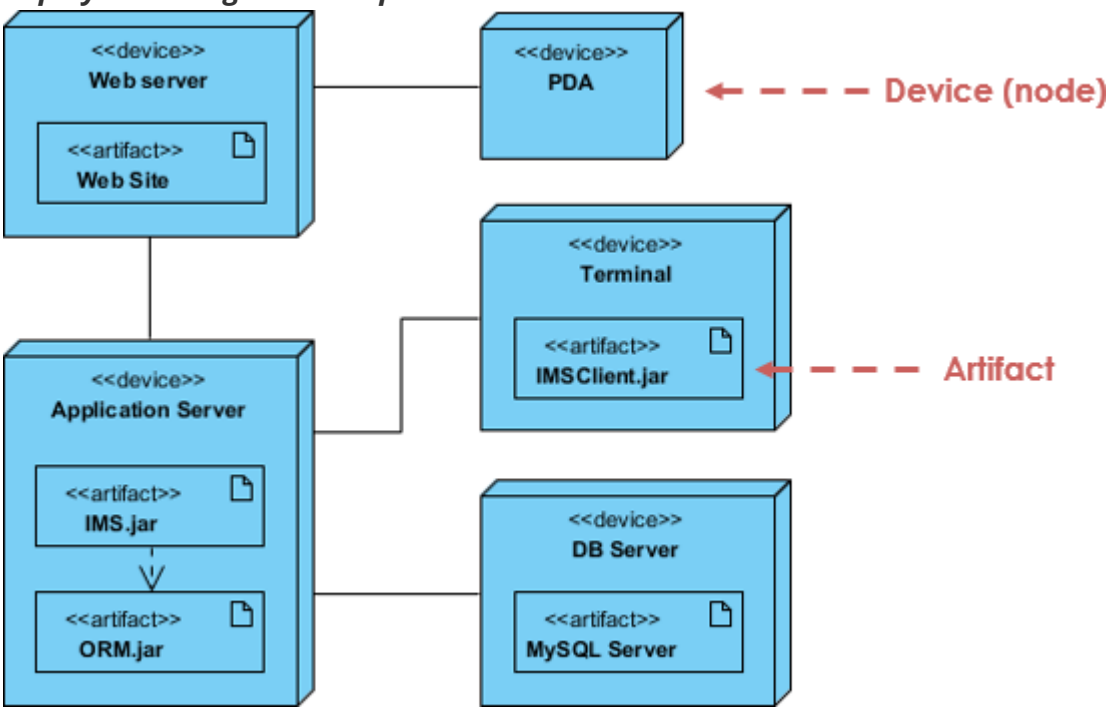


For more details about Component Diagram, please read the article **What is Component Diagram?**

What is a Deployment Diagram?

The Deployment Diagram helps to model the physical aspect of an Object-Oriented software system. It is a structure diagram which shows architecture of the system as deployment (distribution) of software artifacts to deployment targets. Artifacts represent concrete elements in the physical world that are the result of a development process. It models the run-time configuration in a static view and visualizes the distribution of artifacts in an application. In most cases, it involves modeling the hardware configurations together with the software components that lived on.

Deployment Diagram Example



For more details about Deployment Diagram, please read the article **What is Deployment Diagram?**

What is an Object Diagram?

An object diagram is a graph of instances, including objects and data values. A static object diagram is an instance of a class diagram; it shows a snapshot of the detailed state of a system at a point in time. The difference is that a class diagram represents an abstract model consisting of classes and their relationships. However, an object diagram represents an instance at a particular moment, which is concrete in nature. The use of object diagrams is fairly limited, namely to show examples of data structure.

Class Diagram vs Object Diagram - An Example

Some people may find it difficult to understand the difference between a UML Class Diagram and a UML Object Diagram as they both comprise of named "rectangle blocks", with attributes in them, and with linkages in between, which make the two UML diagrams look similar. Some people may even think they are the same because in the UML tool they use both the notations for Class Diagram and Object Diagram are put inside the same diagram editor - Class Diagram.

But in fact, Class Diagram and Object Diagram represent two different aspects of a code base. In this article, we will provide you with some ideas about these two UML diagrams, what they are, what are their differences and when to use each of them.

Relationship between Class Diagram and Object Diagram

You create "classes" when you are programming. For example, in an online banking system you may create classes like 'User', 'Account', 'Transaction', etc. In a classroom management system you may create classes like 'Teacher', 'Student', 'Assignment', etc. In each class, there are attributes and operations that represent the characteristic and behavior of the class. Class Diagram is a UML diagram where you can visualize those classes, along with their attributes, operations and the inter-relationship.

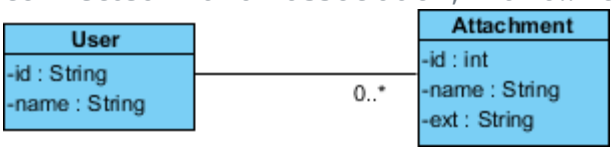
UML Object Diagram shows how object instances in your system are interacting with each other at a particular state. It also represents the data values of those objects at that state. In other words, a UML Object Diagram can be seen as a

representation of how classes (drawn in UML Class Diagram) are utilized at a particular state.

If you are not a fan of those definition stuff, take a look at the following UML diagram examples. I believe that you will understand their differences in seconds.

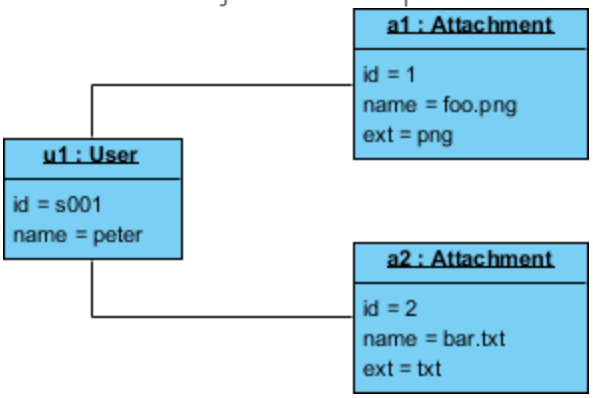
Class Diagram Example

The following Class Diagram example represents two classes - User and Attachment. A user can upload multiple attachment so the two classes are connected with an association, with 0..* as multiplicity on the Attachment side.



Object Diagram Example

The following Object Diagram example shows you how the object instances of User and Attachment class "look like" at the moment Peter (i.e. the user) is trying to upload two attachments. So there are two Instance Specification for the two attachment objects to be uploaded.

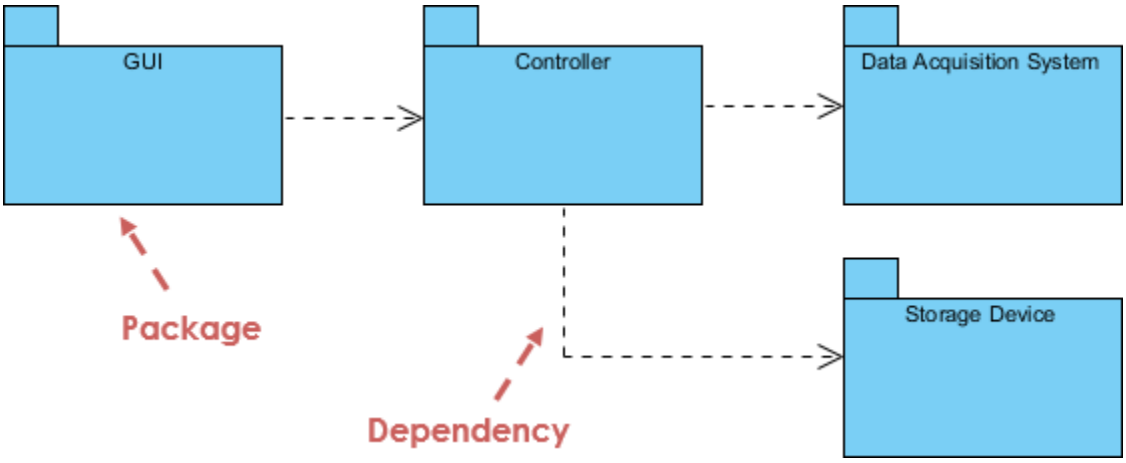


For more details about Object Diagram, please read the article **What is Object Diagram?**

What is a Package Diagram?

Package diagram is UML structure diagram which shows packages and dependencies between the packages. Model diagrams allow to show different views of a system, for example, as multi-layered (aka multi-tiered) application - multi-layered application model.

Package Diagram Example



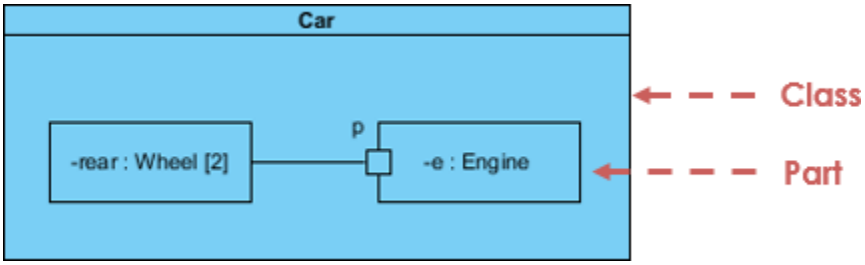
For more details about Package Diagram, please read the article **What is Package Diagram?**

What is a Composite Structure Diagram?

Composite Structure Diagram is one of the new artifacts added to UML 2.0. A composite structure diagram is similar to a class diagram and is a kind of component diagram mainly used in modeling a system at micro point-of-view, but it depicts individual parts instead of whole classes. It is a type of static structure diagram that shows the internal structure of a class and the collaborations that this structure makes possible.

This diagram can include internal parts, ports through which the parts interact with each other or through which instances of the class interact with the parts and with the outside world, and connectors between parts or ports. A composite structure is a set of interconnected elements that collaborate at runtime to achieve some purpose. Each element has some defined role in the collaboration.

Composite Structure Diagram Example

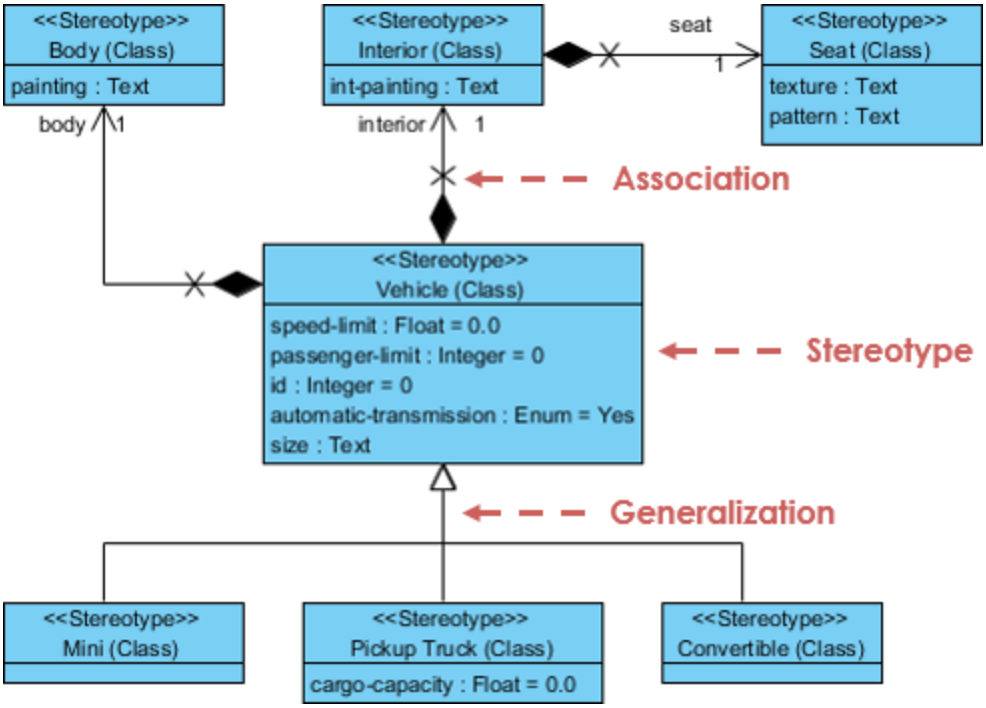


For more details about Composite Structure Diagram, please read the article **What is Composite Structure Diagram?**

What is a Profile Diagram?

A profile diagram enables you to create domain and platform specific stereotypes and define the relationships between them. You can create stereotypes by drawing stereotype shapes and relate them with composition or generalization through the resource-centric interface. You can also define and visualize tagged values of stereotypes.

Profile Diagram Example



For more details about Profile Diagram, please read the article **What is Profile Diagram in UML?**

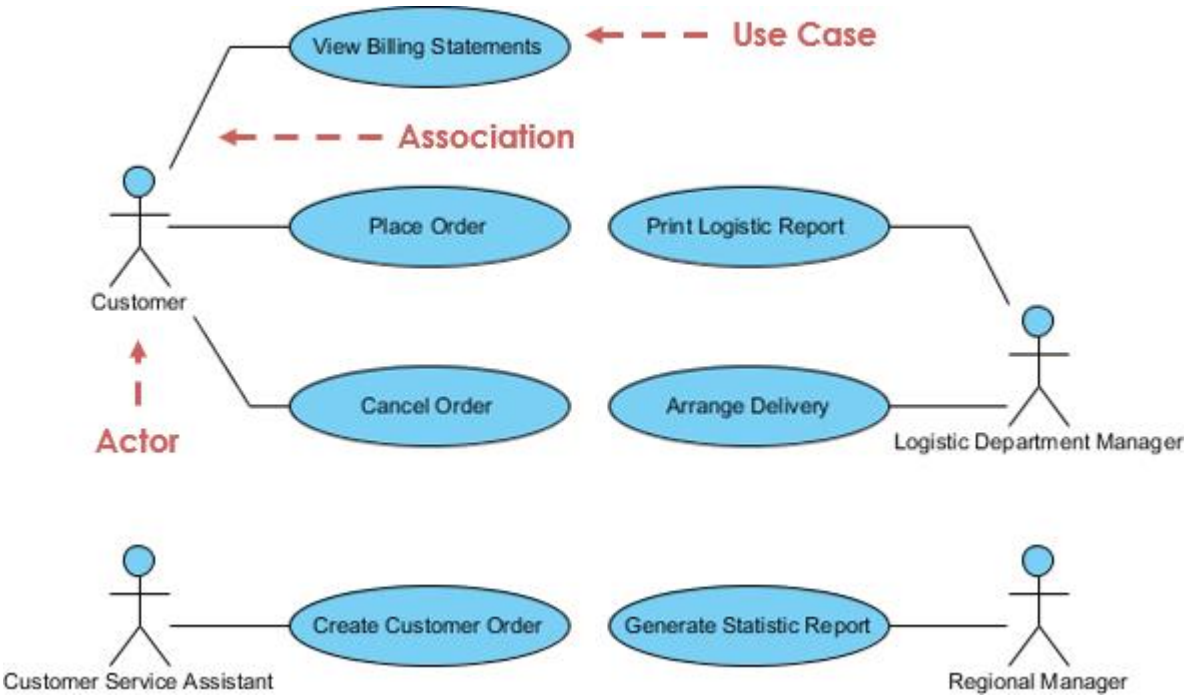
What is a Use Case Diagram?

A use-case model describes a system's functional requirements in terms of use cases. It is a model of the system's intended functionality (use cases) and its environment (actors). Use cases enable you to relate what you need from a system to how the system delivers on those needs.

Think of a use-case model as a menu, much like the menu you'd find in a restaurant. By looking at the menu, you know what's available to you, the individual dishes as well as their prices. You also know what kind of cuisine the restaurant serves: Italian, Mexican, Chinese, and so on. By looking at the menu, you get an overall impression of the dining experience that awaits you in that restaurant. The menu, in effect, "models" the restaurant's behavior.

Because it is a very powerful planning instrument, the use-case model is generally used in all phases of the development cycle by all team members.

Use Case Diagram Example

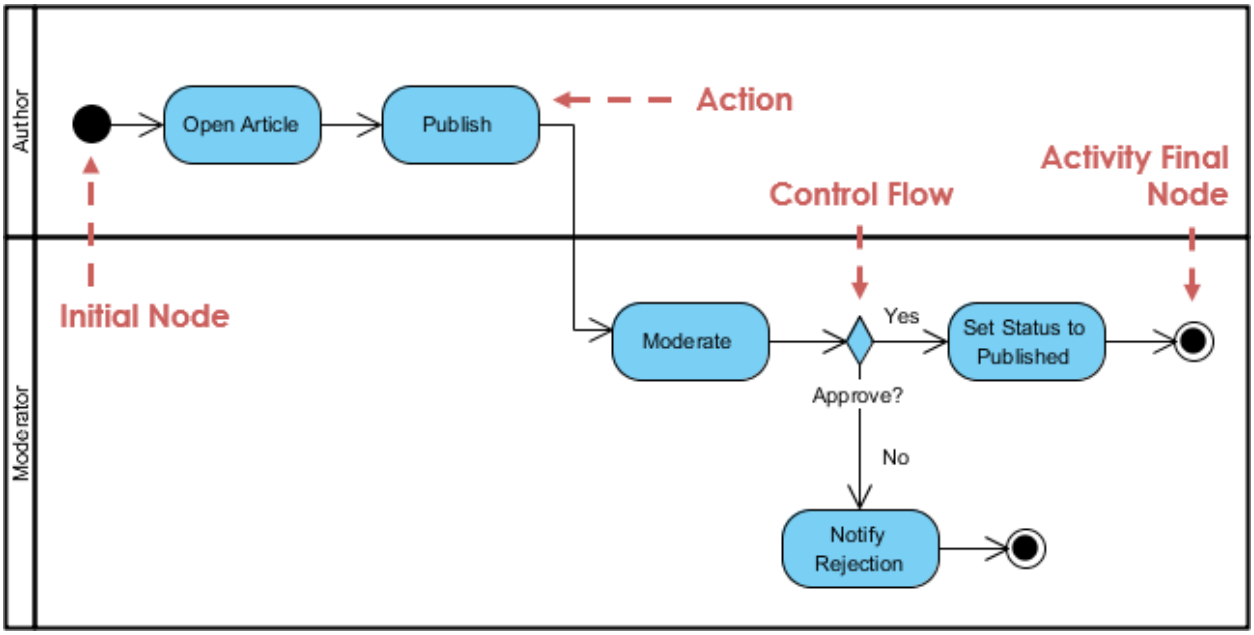


For more details about Use Case Diagram, please read the article **What is Use Case Diagram?**

What is an Activity Diagram?

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. It describes the flow of control of the target system, such as the exploring complex business rules and operations, describing the use case also the business process. In the Unified Modeling Language, activity diagrams are intended to model both computational and organizational processes (i.e. workflows).

Activity Diagram Example



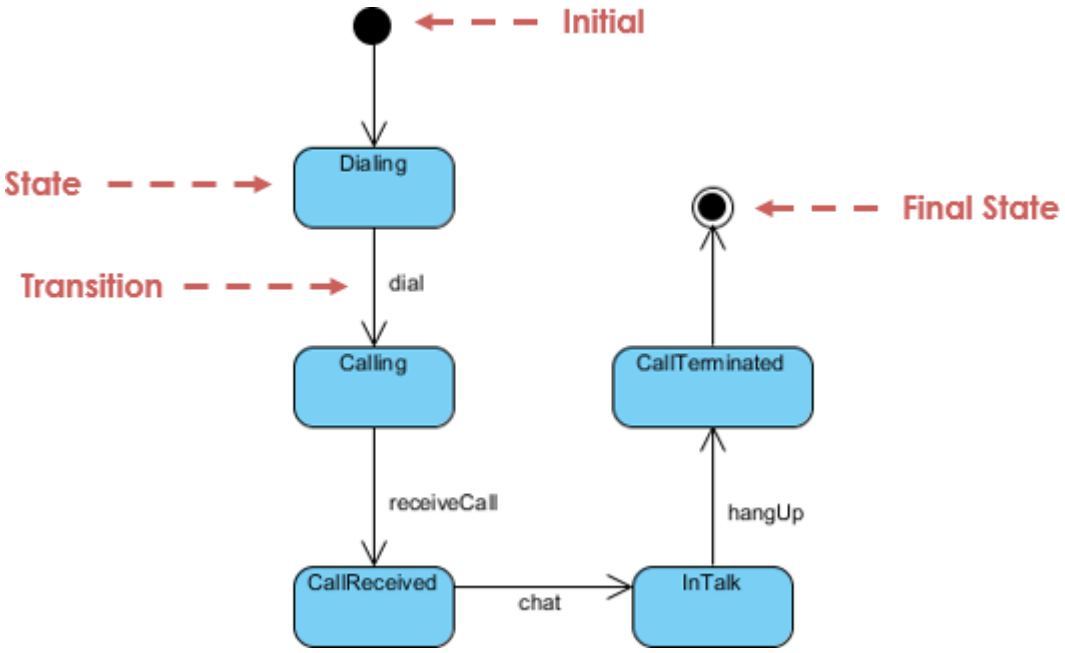
For more details about Activity Diagram, please read the article **What is Activity Diagram?**

What is a State Machine Diagram?

A state diagram is a type of diagram used in UML to describe the behavior of systems which is based on the concept of state diagrams by David Harel. State diagrams depict the permitted states and transitions as well as the events that

effect these transitions. It helps to visualize the entire lifecycle of objects and thus help to provide a better understanding of state-based systems.

State Machine Diagram Example

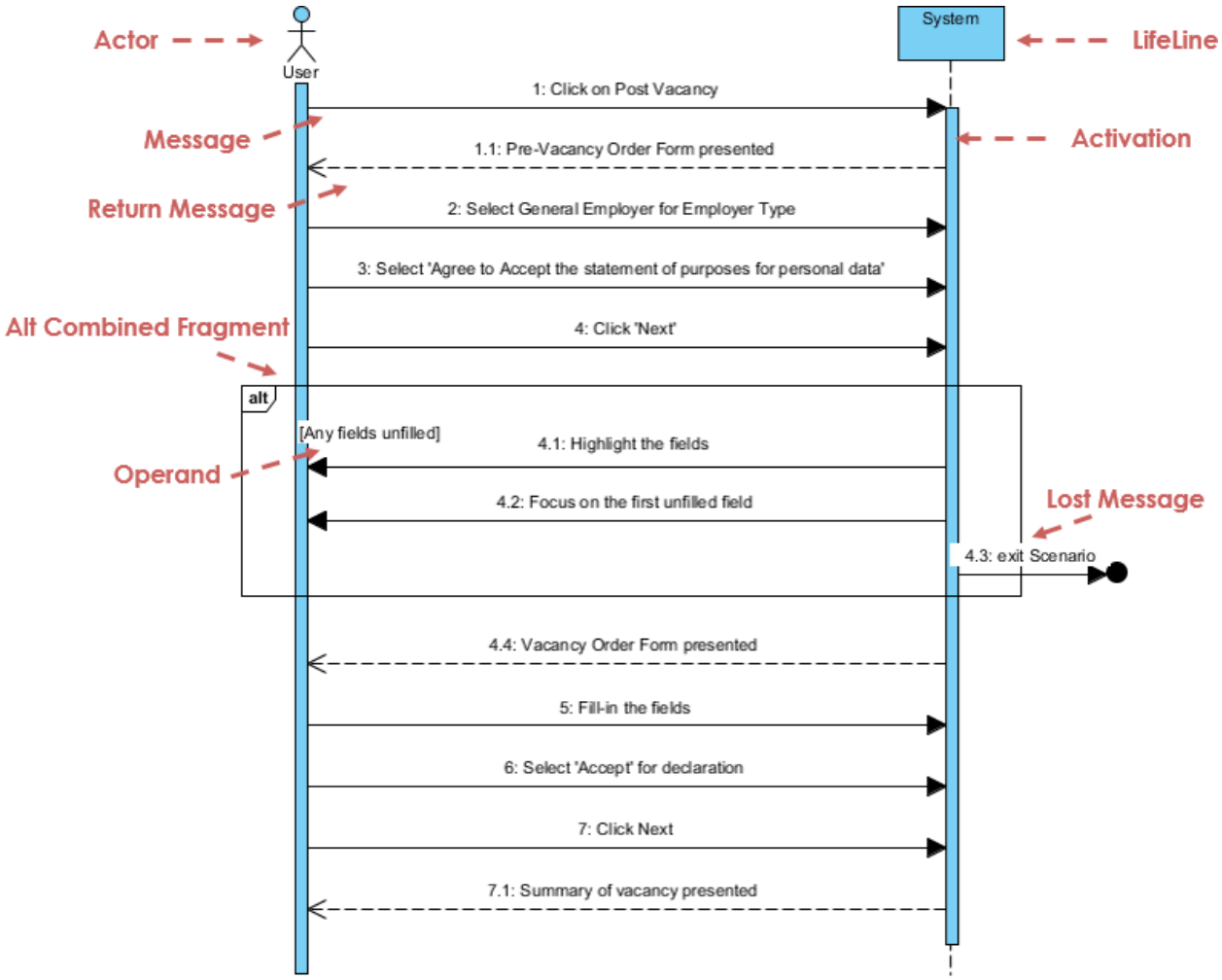


For more details about State Machine Diagram, please read the article **What is State Machine Diagram?**

What is a Sequence Diagram?

The Sequence Diagram models the collaboration of objects based on a time sequence. It shows how the objects interact with others in a particular scenario of a use case. With the advanced visual modeling capability, you can create complex sequence diagram in few clicks. Besides, some modeling tool such as Visual Paradigm can generate sequence diagram from the flow of events which you have defined in the use case description.

Sequence Diagram Example

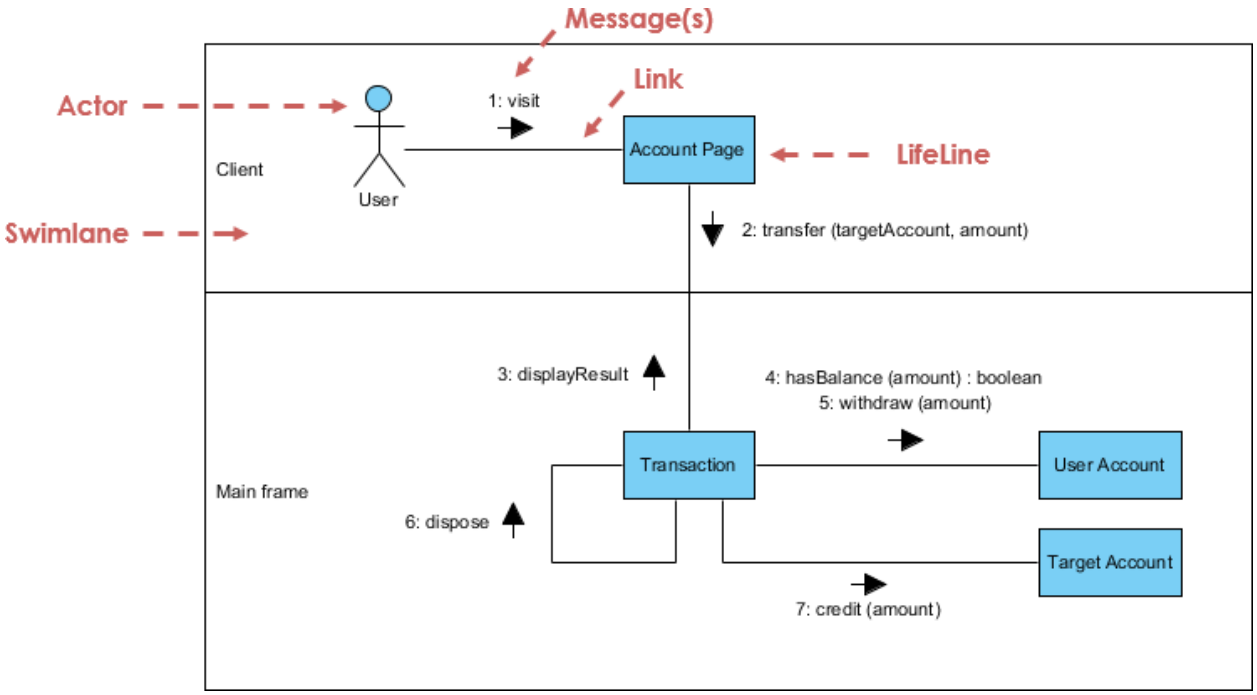


For more details about Sequence Diagram, please read the article **What is Sequence Diagram?**

What is a Communication Diagram?

Similar to Sequence Diagram, the Communication Diagram is also used to model the dynamic behavior of the use case. When compare to Sequence Diagram, the Communication Diagram is more focused on showing the collaboration of objects rather than the time sequence. They are actually semantically equivalent, so some of the modeling tool such as, Visual Paradigm allows you to generate it from one to the other.

Communication Diagram Example

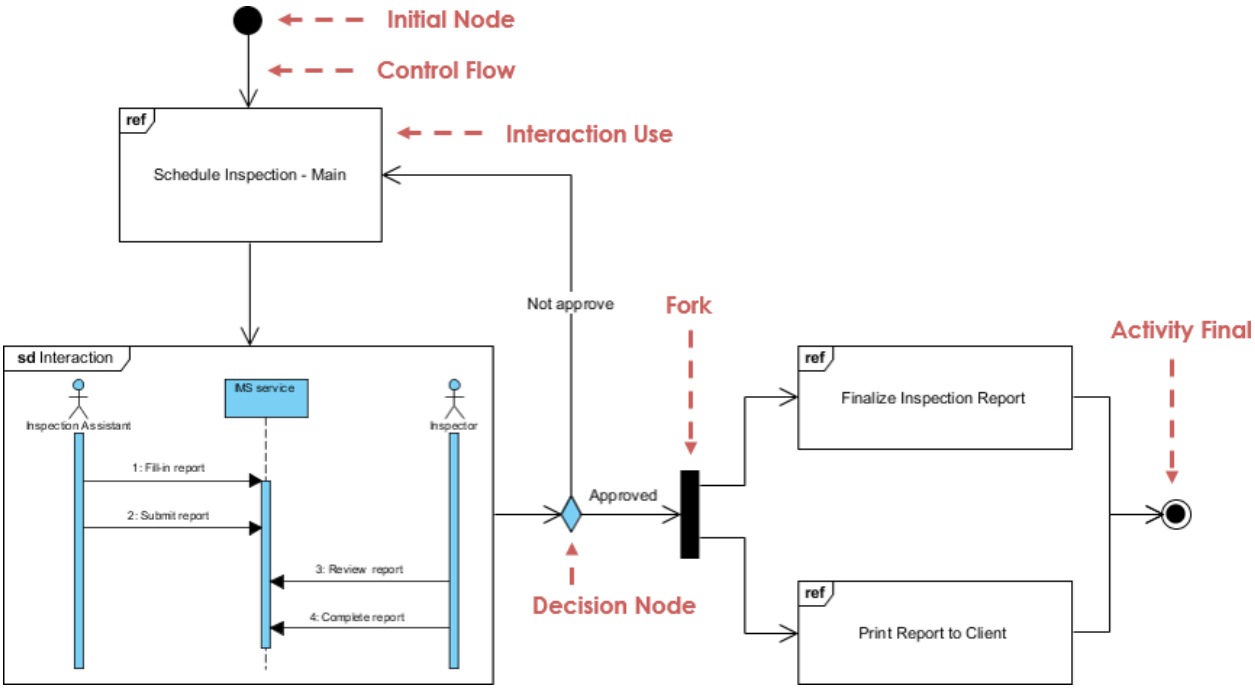


For more details about Communication Diagram, please read the article **What is Communication Diagram?**

What is Interaction Overview Diagram?

The Interaction Overview Diagram focuses on the overview of the flow of control of the interactions. It is a variant of the Activity Diagram where the nodes are the interactions or interaction occurrences. The Interaction Overview Diagram describes the interactions where messages and lifelines are hidden. You can link up the "real" diagrams and achieve high degree navigability between diagrams inside the Interaction Overview Diagram.

Interaction Overview Diagram Example



For more details about Interaction Overview Diagram, please read the article **What is Interaction Overview Diagram?**

What is Timing Diagram?

Timing Diagram shows the behavior of the object(s) in a given period of time. Timing diagram is a special form of a sequence diagram. The differences between timing diagram and sequence diagram are the axes are reversed so that the time are increase from left to right and the lifelines are shown in separate compartments arranged vertically.

Timing Diagram Example