Overview: Use Case

To model a system the most important aspect is to capture the dynamic behaviour. To clarify a bit in details, *dynamic behaviour* means the behaviour of the system when it is running /operating.

So only static behaviour is not sufficient to model a system rather dynamic behaviour is more important than static behaviour. In UML there are five diagrams available to model dynamic nature and use case diagram is one of them. Now as we have to discuss that the use case diagram is dynamic in nature there should be some internal or external factors for making the interaction.

These internal and external agents are known as actors. So use case diagrams are consists of actors, use cases and their relationships. The diagram is used to model the system/subsystem of an application. A single use case diagram captures a particular functionality of a system.

So to model the entire system numbers of use case diagrams are used.

Purpose:

The purpose of use case diagram is to capture the dynamic aspect of a system. But this definition is too generic to describe the purpose.

Because other four diagrams (activity, sequence, collaboration and Statechart) are also having the same purpose. So we will look into some specific purpose which will distinguish it from other four diagrams.

Use case diagrams are used to gather the requirements of a system including internal and external influences. These requirements are mostly design requirements. So when a system is analyzed to gather its functionalities use cases are prepared and actors are identified.

Now when the initial task is complete use case diagrams are modelled to present the outside view.

So in brief, the purposes of use case diagrams can be as follows:

- Used to gather requirements of a system.
- Used to get an outside view of a system.
- Identify external and internal factors influencing the system.
- Show the interacting among the requirements are actors.

How to draw Use Case Diagram?

Use case diagrams are considered for high level requirement analysis of a system. So when the requirements of a system are analyzed the functionalities are captured in use cases.

So we can say that uses cases are nothing but the system functionalities written in an organized manner. Now the second things which are relevant to the use cases are the actors. Actors can be defined as something that interacts with the system.

The actors can be human user, some internal applications or may be some external applications. So in a brief when we are planning to draw an use case diagram we should have the following items identified.

- Functionalities to be represented as an use case
- Actors
- Relationships among the use cases and actors.

Use case diagrams are drawn to capture the functional requirements of a system. So after identifying the above items we have to follow the following guidelines to draw an efficient use case diagram.

- The name of a use case is very important. So the name should be chosen in such a way so that it can identify the functionalities performed.
- Give a suitable name for actors.
- Show relationships and dependencies clearly in the diagram.
- Do not try to include all types of relationships. Because the main purpose of the diagram is to identify requirements.
- Use note whenever required to clarify some important points.

The following is a sample use case diagram representing the order management system. So if we look into the diagram then we will find three use cases (Order, SpecialOrder and NormalOrder) and one actor which is customer.

The *SpecialOrder* and *NormalOrder* use cases are extended from *Order* use case. So they have extends relationship. Another important point is to identify the system boundary which is shown in the picture. The actor *Customer* lies outside the system as it is an external user of the system.

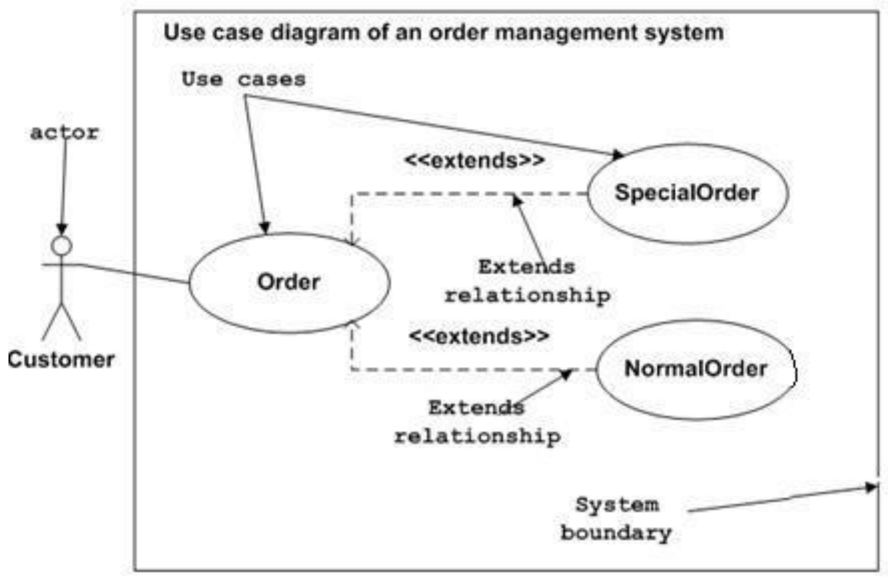


Figure: Sample Use Case diagram

Where to Use Case Diagrams?

As we have already discussed there are five diagrams in UML to model dynamic view of a system. Now each and every model has some specific purpose to use. Actually these specific purposes are different angles of a running system.

So to understand the dynamics of a system we need to use different types of diagrams. Use case diagram is one of them and its specific purpose is to gather system requirements and actors.

Use case diagrams specify the events of a system and their flows. But use case diagram never describes how they are implemented. Use case diagram can be imagined as a black box where only the input, output and the function of the black box is known.

These diagrams are used at a very high level of design. Then this high level design is refined again and again to get a complete and practical picture of the system. A well structured use case also describes the pre-condition, post condition, exceptions. And these extra elements are used to make test cases when performing the testing.

Although the use cases are not a good candidate for forward and reverse engineering but still they are used in a slight different way to make forward and reverse engineering. And the same is true for reverse engineering. Still use case diagram is used differently to make it a candidate for reverse engineering.

In forward engineering use case diagrams are used to make test cases and in reverse engineering use cases are used to prepare the requirement details from the existing application.

So the following are the places where use case diagrams are used:

- Requirement analysis and high level design.
- Model the context of a system.
- Reverse engineering.
- Forward engineering.

Overview: class diagram

The class diagram is a static diagram. It represents the static view of an application. Class diagram is not only used for visualizing, describing and documenting different aspects of a system but also for constructing executable code of the software application.

The class diagram describes the attributes and operations of a class and also the constraints imposed on the system. The class diagrams are widely used in the modelling of object oriented systems because they are the only UML diagrams which can be mapped directly with object oriented languages.

The class diagram shows a collection of classes, interfaces, associations, collaborations and constraints. It is also known as a *structural diagram*.

Purpose:

The purpose of the class diagram is to model the static view of an application. The class diagrams are the only diagrams which can be directly mapped with object oriented languages and thus widely used at the time of construction.

The UML diagrams like activity diagram, sequence diagram can only give the sequence flow of the application but class diagram is a bit different. So it is the most popular UML diagram in the coder community.

So the purpose of the class diagram can be summarized as:

- Analysis and design of the static view of an application.
- Describe responsibilities of a system.
- Base for component and deployment diagrams.
- Forward and reverse engineering.

How to draw Class Diagram?

Class diagrams are the most popular UML diagrams used for construction of software applications. So it is very important to learn the drawing procedure of class diagram.

Class diagrams have lot of properties to consider while drawing but here the diagram will be considered from a top level view.

Class diagram is basically a graphical representation of the static view of the system and represents different aspects of the application. So a collection of class diagrams represent the whole system.

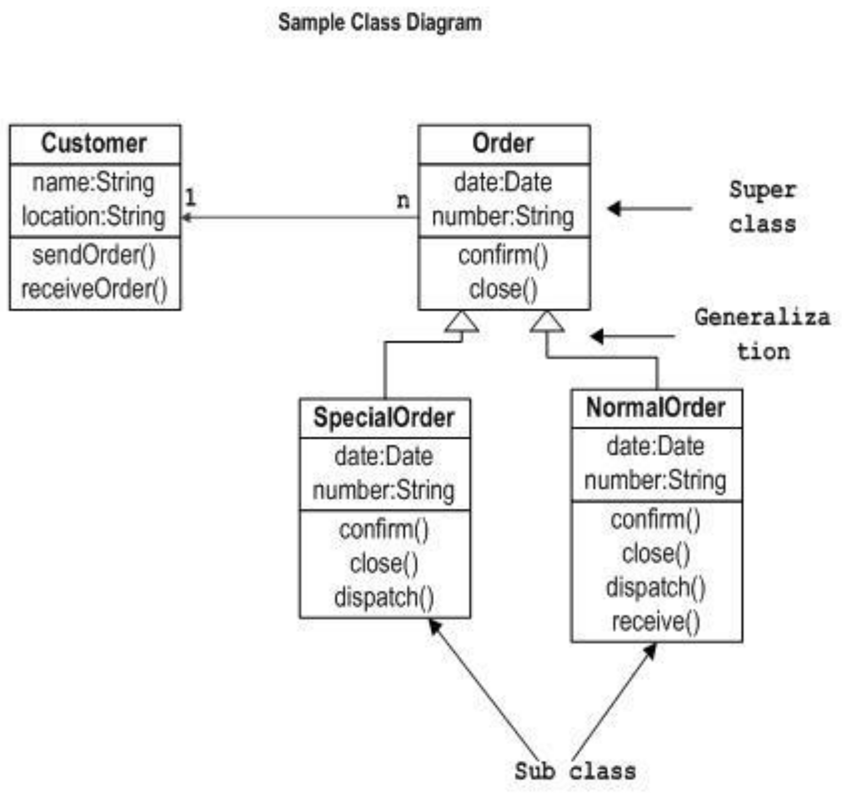
The following points should be remembered while drawing a class diagram:

- The name of the class diagram should be meaningful to describe the aspect of the system.
- Each element and their relationships should be identified in advance.
- Responsibility (attributes and methods) of each class should be clearly identified.
- For each class minimum number of properties should be specified. Because unnecessary properties will make the diagram complicated.
- Use notes whenever required to describe some aspect of the diagram. Because at the end of the drawing it should be understandable to the developer/coder.
- Finally, before making the final version, the diagram should be drawn on plain paper and rework as many times as possible to make it correct.

Now the following diagram is an example of an *Order System* of an application. So it describes a particular aspect of the entire application.

- First of all *Order* and *Customer* are identified as the two elements of the system and they have a *one to many* relationship because a customer can have multiple orders.
- We would keep Order class is an abstract class and it has two concrete classes (inheritance relationship) SpecialOrder and NormalOrder.
- The two inherited classes have all the properties as the *Order* class. In addition they have additional functions like *dispatch()* and *receive ()*.

So the following class diagram has been drawn considering all the points mentioned above:



Where to use Class Diagrams?

Class diagram is a static diagram and it is used to model static view of a system. The static view describes the vocabulary of the system.

Class diagram is also considered as the foundation for component and deployment diagrams. Class diagrams are not only used to visualize the static view of the system but they are also used to construct the executable code for forward and reverse engineering of any system.

Generally UML diagrams are not directly mapped with any object oriented programming languages but the class diagram is an exception.

Class diagram clearly shows the mapping with object oriented languages like Java, C++ etc. So from practical experience class diagram is generally used for construction purpose.

So in a brief, class diagrams are used for:

- Describing the static view of the system.
- Showing the collaboration among the elements of the static view.
- Describing the functionalities performed by the system.
- Construction of software applications using object oriented languages.