**LAB : 1**

**OBJECTIVE :**

(a) Introduction to Unified Modelling Language (UML) and its type.

(b) Draw Use Case diagram for ATM application

**Requrements :**

(a) Windows PC (Windows 7/8/10) / Mac

(b) Star UML Tool

**Procedure :**

**(1.a) Introduction to Unified Modelling Language (UML) and its type.**

**Unified Modeling Language (UML) :**

UML, as the name implies, is a modeling language. It may be used to visualize, specify, construct, and document the artifacts of a software system. It provides a set of notations (e.g. rectangles, lines, ellipses, etc.) to create a visual model of the system. Like any other language, UML has its own syntax (symbols and sentence formation rules) and semantics (meanings of symbols and sentences). Also, we should clearly understand that UML is not a system design or development methodology, but can be used to document object-oriented and analysis results obtained using some methodology.

UML was developed to standardize the large number of object-oriented modeling notations that existed and were used extensively in the early 1990s. The principles ones in use were:

• Object Management Technology [Rumbaugh 1991]

• Booch’s methodology [Booch 1991]

• Object-Oriented Software Engineering [Jacobson 1992]

• Odell’s methodology [Odell 1992]

• Shaler and Mellor methodology [Shaler 1992]

It is needless to say that UML has borrowed many concepts from these modeling techniques. Especially, concepts from the first three methodologies have been heavily drawn upon. UML was adopted by Object Management Group (OMG) as a de facto standard in 1997.

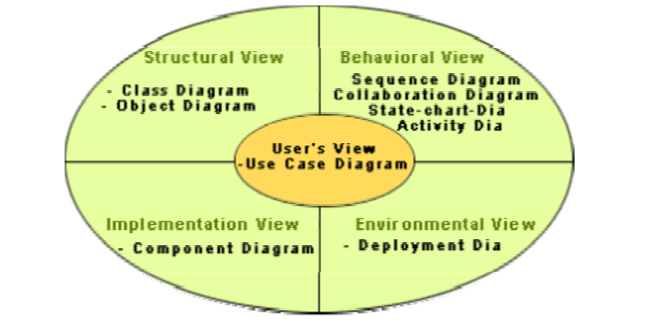
**UML diagrams**

UML can be used to construct nine different types of diagrams to capture five different views of a system. Just as a building can be modeled from several views (or perspectives) such as ventilation perspective, electrical perspective, lighting perspective, heating perspective, etc.; the different UML diagrams provide different perspectives of the software system to be developed and facilitate a comprehensive understanding of the system. Such models can be refined to get the actual implementation of the system.

The UML diagrams can capture the following five views of a system:

1. **User’s view** : This view defines the functionalities (facilities) made available by the system to its users. The users’ view captures the external users’ view of the system in terms of the functionalities offered by the system. The users’ view is a black-box view of the system where the internal structure, the dynamic behavior of different system components, the implementation etc. are not visible.
2. **Structural view** : The structural view defines the kinds of objects (classes) important to the understanding of the working of a system and to its implementation. It also captures the relationships among the classes (objects). The structural model is also called the static model, since the structure of a system does not change with time.
3. **Behavioral view** : The behavioral view captures how objects interact with each other to realize the system behavior. The system behavior captures the time-dependent (dynamic) behavior of the system.
4. **Implementation view** : This view captures the important components of the system and their dependencies.
5. **Environmental view** : Environmental view: This view models how the different components are implemented on different pieces of hardware.

Fig. 1.1 shows the UML diagrams responsible for providing the different views



**Fig. 1.1** Different types of diagrams and views supported in UML

Different UML Diagrams are as following :

**Use Case Diagram :** As the most known diagram type of the behavioral UML types, Use case diagrams give a graphic overview of the actors involved in a system, different functions needed by those actors and how these different functions interact.It’s a great starting point for any project discussion because you can easily identify the main actors involved and the main processes of the system.

**Class Diagram :** Class diagrams are the main building block of any object-oriented solution. It shows the classes in a system, attributes, and operations of each class and the relationship between each class.

**Object Diagram :** Object Diagrams, sometimes referred to as Instance diagrams are very similar to class diagrams. Like class diagrams, they also show the relationship between objects but they use real-world examples.

**Sequence Diagram :** Sequence diagrams in UML show how objects interact with each other and the order those interactions occur. It’s important to note that they show the interactions for a particular scenario. The processes are represented vertically and interactions are shown as arrows.

**Collaboration Diagram :**The collaboration diagram is used to show the relationship between the objects in a system. Both the sequence and the collaboration diagrams represent the same information but differently. Instead of showing the flow of messages, it depicts the architecture of the object residing in the system as it is based on object-oriented programming. An object consists of several features. Multiple objects present in the system are connected to each other. The collaboration diagram, which is also known as a communication diagram, is used to portray the object's architecture in the system.

**State Machine Diagram :** The state machine diagram is also called the Statechart or State Transition diagram, which shows the order of states underwent by an object within the system. It captures the software system's behavior. It models the behavior of a class, a subsystem, a package, and a complete system.

**Activity Diagram :** In UML, the activity diagram is used to demonstrate the flow of control within the system rather than the implementation. It models the concurrent and sequential activities.

The activity diagram helps in envisioning the workflow from one activity to another. It put emphasis on the condition of flow and the order in which it occurs. The flow can be sequential, branched, or concurrent, and to deal with such kinds of flows, the activity diagram has come up with a fork, join, etc.

**Component Diagram :** A component diagram is used to break down a large object-oriented system into the smaller components, so as to make them more manageable. It models the physical view of a system such as executables, files, libraries, etc. that resides within the node.

It visualizes the relationships as well as the organization between the components present in the system. It helps in forming an executable system. A component is a single unit of the system, which is replaceable and executable. The implementation details of a component are hidden, and it necessitates an interface to execute a function. It is like a black box whose behavior is explained by the provided and required interfaces.

**Deployment Diagram :** The deployment diagram visualizes the physical hardware on which the software will be deployed. It portrays the static deployment view of a system. It involves the nodes and their relationships.

It ascertains how software is deployed on the hardware. It maps the software architecture created in design to the physical system architecture, where the software will be executed as a node. Since it involves many nodes, the relationship is shown by utilizing communication paths.

Some other UML Diagrams are :

**Interaction Diagram :** As the name suggests, the interaction diagram portrays the interactions between distinct entities present in the model. It amalgamates both the activity and sequence diagrams. The communication is nothing but units of the behavior of a classifier that provides context for interactions.

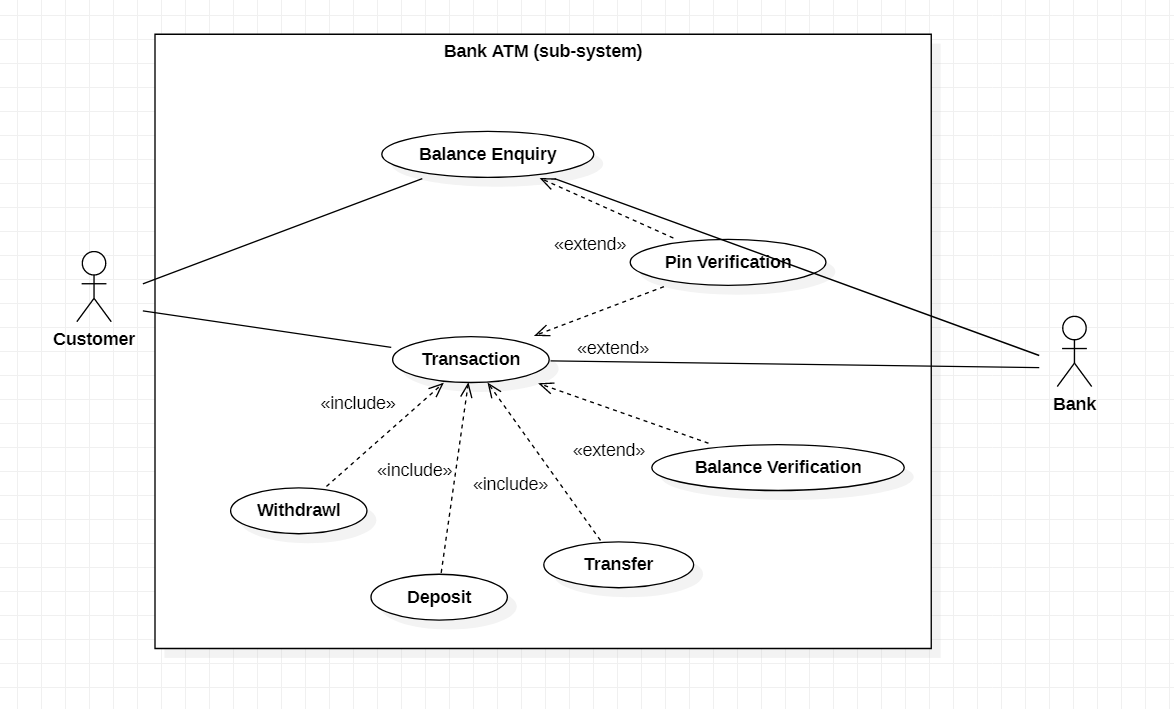
A set of messages that are interchanged between the entities to achieve certain specified tasks in the system is termed as interaction. It may incorporate any feature of the classifier of which it has access. In the interaction diagram, the critical component is the messages and the lifeline.

**Timing Diagram :** In UML, the timing diagrams are a part of Interaction diagrams that do not incorporate similar notations as that of sequence and collaboration diagram. It consists of a graph or waveform that depicts the state of a lifeline at a specific point of time. It illustrates how conditions are altered both inside and between lifelines alongside linear time axis.

The timing diagram describes how an object underwent a change from one form to another. A waveform portrays the flow among the software programs at several instances of time.

**(1.b) Draw Use Case diagram for ATM application**

- On Star UML Tool



**Fig. 1.2** UML Diagram for a Bank Atm (sub-system)