

UML

Dr. Buddhadeb Pradhan

Overview

- UML is a standard language for specifying, visualizing, constructing, and documenting the artifacts of software systems.
- UML was created by the Object Management Group (OMG) and UML 1.0 specification draft was proposed to the OMG in January 1997.
- UML stands for **Unified Modeling Language**.
- UML is a pictorial language used to make software blueprints.
- UML can be described as a general purpose visual modeling language to visualize, specify, construct, and document software system.
- Although UML is also used to model non-software systems as well. For example, the process flow in a manufacturing unit, etc.
- UML is not a programming language but tools can be used to generate code in various languages using UML diagrams. UML has a direct relation with object oriented analysis and design. After some standardization, UML has become an OMG standard.

Goals of UML

- *A picture is worth a thousand words*, this idiom absolutely fits describing UML. Object-oriented concepts were introduced much earlier than UML. At that point of time, there were no standard methodologies to organize and consolidate the object-oriented development. It was then that UML came into picture.
- There are a number of goals for developing UML but the most important is to define some general purpose modeling language, which all modelers can use and it also needs to be made simple to understand and use.
- UML diagrams are not only made for developers but also for business users, common people, and anybody interested to understand the system. The system can be a software or non-software system. Thus it must be clear that UML is not a development method rather it accompanies with processes to make it a successful system.
- In conclusion, the goal of UML can be defined as a simple modeling mechanism to model all possible practical systems in today's complex environment.

A Conceptual Model of UML

To understand the conceptual model of UML, first we need to clarify what is a conceptual model? and why a conceptual model is required?

- A conceptual model can be defined as a model which is made of concepts and their relationships.
- A conceptual model is the first step before drawing a UML diagram. It helps to understand the entities in the real world and how they interact with each other.

As UML describes the real-time systems, it is very important to make a conceptual model and then proceed gradually. The conceptual model of UML can be mastered by learning the following three major elements –

- UML building blocks
- Rules to connect the building blocks
- Common mechanisms of UML

Object-Oriented Concepts

- ❖ UML can be described as the successor of object-oriented (OO) analysis and design.
- ❖ Objects are the real-world entities that exist around us and the basic concepts such as abstraction, encapsulation, inheritance, and polymorphism all can be represented using UML.
- ❖ UML is powerful enough to represent all the concepts that exist in object-oriented analysis and design. UML diagrams are representation of object-oriented concepts only. Thus, before learning UML, it becomes important to understand OO concept in detail.

Following are some fundamental concepts of the object-oriented world –

- ✓ **Objects** – Objects represent an entity and the basic building block.
- ✓ **Class** – Class is the blue print of an object.
- ✓ **Abstraction** – Abstraction represents the behavior of an real world entity.
- ✓ **Encapsulation** – Encapsulation is the mechanism of binding the data together and hiding them from the outside world.
- ✓ **Inheritance** – Inheritance is the mechanism of making new classes from existing ones.
- ✓ **Polymorphism** – It defines the mechanism to exists in different forms.

OO Analysis and Design

- OO can be defined as an investigation and to be more specific, it is the investigation of objects. Design means collaboration of identified objects.
- The purpose of OO analysis and design can be described as –
 - ❖ Identifying the objects of a system.
 - ❖ Identifying their relationships.
 - ❖ Making a design, which can be converted to executables using OO languages.

There are three basic steps where the OO concepts are applied and implemented. The steps can be defined as

OO Analysis → OO Design → OO implementation using OO languages
The above three points can be described in detail as –

- During OO analysis, the most important purpose is to identify objects and describe them in a proper way.
- The second phase is OO design. During this phase, emphasis is placed on the requirements and their fulfilment. In this stage, the objects are collaborated according to their intended association. After the association is complete, the design is also complete.
- The third phase is OO implementation. In this phase, the design is implemented using OO languages such as Java, C++, etc.

Role of UML in OO Design

- UML is a modeling language used to model software and non-software systems. Although UML is used for non-software systems, the emphasis is on modeling OO software applications. Most of the UML diagrams discussed so far are used to model different aspects such as static, dynamic, etc. Now whatever be the aspect, the artifacts are nothing but objects.
- If we look into class diagram, object diagram, collaboration diagram, interaction diagrams all would basically be designed based on the objects.
- Hence, the relation between OO design and UML is very important to understand. The OO design is transformed into UML diagrams according to the requirement. Before understanding the UML in detail, the OO concept should be learned properly. Once the OO analysis and design is done, the next step is very easy. The input from OO analysis and design is the input to UML diagrams.

UML - Building Blocks

As UML describes the real-time systems, it is very important to make a conceptual model and then proceed gradually. The conceptual model of UML can be mastered by learning the following three major elements –

- ✓ UML building blocks
- ✓ Rules to connect the building blocks
- ✓ Common mechanisms of UML

The building blocks of UML can be defined as –

- Things
- Relationships
- Diagrams

Things

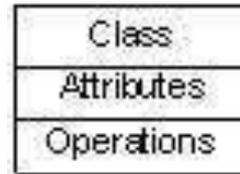
Things are the most important building blocks of UML. Things can be –

- Structural
- Behavioral
- Grouping
- Annotational

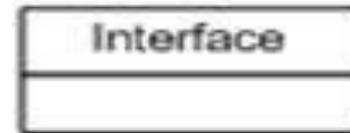
Structural Things

Structural things define the static part of the model. They represent the physical and conceptual elements. Following are the brief descriptions of the structural things.

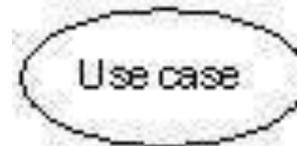
Class – Class represents a set of objects having similar responsibilities.



Interface – Interface defines a set of operations, which specify the responsibility of a class.

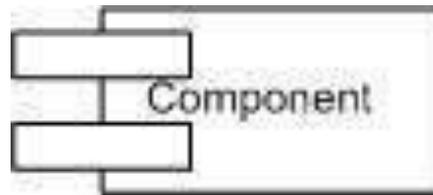


Use case – Use case represents a set of actions performed by a system for a specific goal.

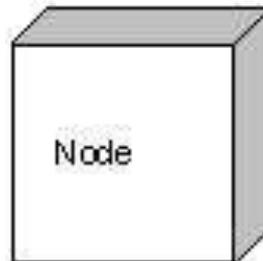


Structural Things

Component – Component describes the physical part of a system.



Node – A node can be defined as a physical element that exists at run time.

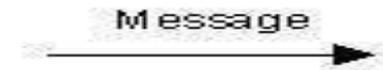


Behavioral Things

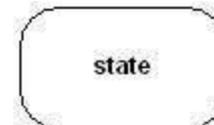
A behavioral thing consists of the dynamic parts of UML models.

Following are the behavioral things.

Interaction – Interaction is defined as a behavior that consists of a group of messages exchanged among elements to accomplish a specific task.



State machine – State machine is useful when the state of an object in its life cycle is important. It defines the sequence of states an object goes through in response to events. Events are external factors responsible for state change



Grouping Things

Grouping things can be defined as a mechanism to group elements of a UML model together. There is only one grouping thing available –

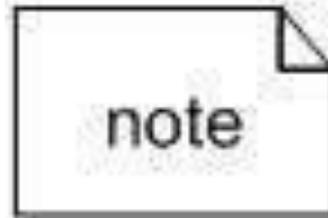
- **Package** – Package is the only one grouping thing available for gathering structural and behavioral things.



Annotational Things

- **Annotational things** can be defined as a mechanism to capture remarks, descriptions, and comments of UML model elements.

Note - It is the only one Annotational thing available. A note is used to render comments, constraints, etc. of an UML element.



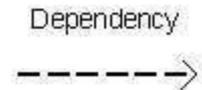
Relationship

- **Relationship** is another most important building block of UML. It shows how the elements are associated with each other and this association describes the functionality of an application.

There are four kinds of relationships available.

Dependency

- Dependency is a relationship between two things in which change in one element also affects the other.



Association

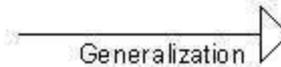
Association is basically a set of links that connects the elements of a UML model. It also describes how many objects are taking part in that relationship.



Relationship

Generalization

- Generalization can be defined as a relationship which connects a specialized element with a generalized element. It basically describes the inheritance relationship in the world of objects.



Realization

- Realization can be defined as a relationship in which two elements are connected. One element describes some responsibility, which is not implemented and the other one implements them. This relationship exists in case of interfaces.

