UNIT -2

SOFTWARE MODELING

Modeling is widely used in science and engineering to provide abstractions of a system at some level of precision and detail. The model is then analyzed in order to obtain a better understanding of the system being developed. According to the Object Modeling Group (OMG), "modeling is the designing of software applications before coding."

In model-based software design and development, software modeling is used as an essential part of the software development process. Models are built and analyzed prior to the implementation of the system, and are used to direct the subsequent implementation.

A better understanding of a system can be obtained by considering it from different perspectives (also referred to as multiple views) (Gomaa 2006; Gomaa and Shin 2004), such as requirements models, static models, and dynamic models of the software system. A graphical modeling language such as UML helps in developing, understanding, and communicating the different views.

Importance of Modeling

To know the importance of modeling let us assume that you are going to build a dog house, a house for your family and a high rise office for a client. In the case of a dog house you need minimal resources and the satisfaction of the dog is not that important.

In the case of building a house for your family, you need to satisfy the requirements of your family members and the amount resources are non-trivial. In the case of building a high rise office, the amount of risk is very high.

Curiously, a lot of software development organizations start out wanting to build high rises but approach the problem as if they are knocking at a dog house. Sometimes you get lucky, If you have the right people at the right moment and if all the planets align properly, then you might, get your team to create a software product that satisfies its users. This happens very rarely.

Unsuccessful software projects fail in their own unique ways, but all successful software projects are alike in many ways. There are many elements that contribute to a successful software organization; one common element is the use of modeling.

Modeling is a proven and well-accepted engineering technique. We build architectural models of houses and high rises to help their users visualize the final product.

Modeling is not only limited to the construction industry. Modeling is applied in the fields of aeronautics, automobile, picture, sociology, economics, software development and many more. We build models so that we can validate our theories or try out new ones with minimal risk and cost.

A model is a simplification of reality.

A good model includes those elements that have broad effect and omits those minor elements that are not relevant to the given level of abstraction. A model may be structural, emphasizing the organization of the system, or it may be behavioral, emphasizing the dynamics of the system.

Why do we model? There is one fundamental reason:

We build models so that we can better understand the system we are developing.

Through modeling, we achieve four aims:

- Models help us to visualize a system as it is or as we want it to be.
- Models permit us to specify the structure or behavior of a system.
- Models gives us a template that guides us in constructing a system.
- Models document the decisions we have made.

The larger and more complex the system becomes, the more important modeling becomes, for one very simple reason:

We build models of complex systems because we cannot comprehend such a system in its entirety. Every project can benefit from modeling. Modeling can help the development team better visualize the plan of their system and allow them to develop more rapidly by helping them build the right thing. The more complex your project, the more likely it is that you will fail or that you will build the wrong thing if you do on modeling at all.

model architecture defines on a logical level technological and domain software layers, where each layer consists of defined elements and their connectors. The layers themselves are not interconnected, only their elements, so we cannot specify rules relating to the layers. However, we can limit the relationships between elements (i.e., class libraries, components, black-box and white-box frameworks) of the layers by defining the following dependence rules:

- Elements of one layer may use only elements of the same or the next lower protocol-based layer.
- Elements of one layer may use only elements of the same or an arbitrary lower object-oriented layer.
- Elements of a product domain or use context should not use elements of another product domain or use context.

Principles of Modeling

First principle of modeling:

The choice of what models to create has a profound influence on how a problem is attacked and how a solution is shaped.

Choose your models well. The right models will highlight the most nasty development problems. Wrong models will mislead you, causing you to focus on irrelevant issues.

Second principle of modeling:

Every model may be expressed at different levels of precision.

Sometimes, a quick and simple executable model of the user interface is exactly what you need. At other times, you have to get down to complex details such as cross-system interfaces or networking issues etc.

In any case, the best kinds of models are those that let you choose your degree of detail, depending on who is viewing it. An analyst or an end user will want to focus on issues of what and a developer will want to focus on issues of how.

Third principle of modeling:

The best models are connected to reality.

In software, the gap between the analysis model and the system's design model must be less. Failing to bridge this gap causes the system to diverge over time. In object-oriented systems, it is possible to connect all the nearly independent views of a system into one whole.

Fourth principle of modeling:

No single model is sufficient. Every non-trivial system is best approached through a small set of nearly independent models.

In the case of a building, you can study electrical plans in isolation, but you can also see their mapping to the floor plan and perhaps even their interaction with the routing of pipes in the plumbing plan.

Object-oriented modeling is the process of preparing and designing what the model's code will actually look like. During the construction or programming phase, the modeling techniques are implemented by using a language that supports the object-oriented programming model.

OOM consists of progressively developing object representation through three phases: analysis, design, and implementation. During the initial stages of development, the model developed is abstract because the external details of the system are the central focus. The model becomes more and more detailed as it evolves, while the central focus shifts toward understanding how the system will be constructed and how it should function.

What is Object Oriented Methodology?

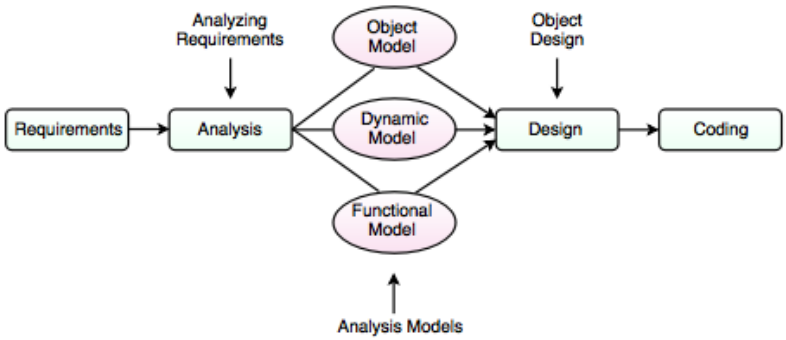
- It is a new system development approach, encouraging and facilitating re-use of software components.
- It employs international standard Unified Modeling Language (UML) from the Object Management Group (OMG).
- Using this methodology, a system can be developed on a component basis, which enables the effective re-use of existing components, it facilitates the sharing of its other system components.
- Object Oriented Methodology asks the analyst to determine what the objects of the system are?, What responsibilities and relationships an object has to do with the other objects? and How they behave over time?

There are three types of Object Oriented Methodologies

1. Object Modeling Techniques (OMT)
2. Object Process Methodology (OPM)
3. Rational Unified Process (RUP)

1. Object Modeling Techniques (OMT)

- It was one of the first object oriented methodologies and was introduced by Rumbaugh in 1991.
- OMT uses three different models that are combined in a way that is analogous to the older structured methodologies.



a. Analysis

- The main goal of the analysis is to build models of the world.
- The requirements of the users, developers and managers provide the information needed to develop the initial problem statement.

b. OMT Models

I. Object Model

- It depicts the object classes and their relationships as a class diagram, which represents the static structure of the system.
- It observes all the objects as static and does not pay any attention to their dynamic nature.

II. Dynamic Model

- It captures the behavior of the system over time and the flow control and events in the Event-Trace Diagrams and State Transition Diagrams.
- It portrays the changes occurring in the states of various objects with the events that might occur in the system.

III. Functional Model

- It describes the data transformations of the system.
- It describes the flow of data and the changes that occur to the data throughout the system.

c. Design

- It specifies all of the details needed to describe how the system will be implemented.
- In this phase, the details of the system analysis and system design are implemented.
- The objects identified in the system design phase are designed.

2. Object Process Methodology (OPM)

- It is also called as second generation methodology.
- It was first introduced in 1995.
- It has only one diagram that is the Object Process Diagram (OPD) which is used for modeling the structure, function and behavior of the system.
- It has a strong emphasis on modeling but has a weaker emphasis on process.
- It consists of three main processes:

I. Initiating: It determines high level requirements, the scope of the system and the resources that will be required.

II. Developing: It involves the detailed analysis, design and implementation of the system.

III. Deploying: It introduces the system to the user and subsequent maintenance of the system.

3. Rational Unified Process (RUP)

- It was developed in Rational Corporation in 1998.
- It consists of four phases which can be broken down into iterations.
  I. Inception
  II. Elaboration
  III. Construction
  IV. Transition
- Each iteration consists of nine work areas called disciplines.
- A discipline depends on the phase in which the iteration is taking place.
- For each discipline, RUP defines a set of artefacts (work products), activities (work undertaken on the artefacts) and roles (the responsibilities of the members of the development team).

Objectives of Object Oriented Methodologies

- To encourage greater re-use.
- To produce a more detailed specification of system constraints.
- To have fewer problems with validation (Are we building the right product?).

Benefits of Object Oriented Methodologies

1. It represents the problem domain, because it is easier to produce and understand designs.

2. It allows changes more easily.

3. It provides nice structures for thinking, abstracting and leads to modular design.

4. Simplicity:

- The software object's model complexity is reduced and the program structure is very clear.

5. Reusability:

- It is a desired goal of all development process.
- It contains both data and functions which act on data.
- It makes easy to reuse the code in a new system.
- Messages provide a predefined interface to an object's data and functionality.

6. Increased Quality:

- This feature increases in quality is largely a by-product of this program reuse.

7. Maintainable:

- The OOP method makes code more maintainable.
- The objects can be maintained separately, making locating and fixing problems easier.

8. Scalable:

- The object oriented applications are more scalable than structured approach.
- It makes easy to replace the old and aging code with faster algorithms and newer technology.

9. Modularity:

- The OOD systems are easier to modify.
- It can be altered in fundamental ways without ever breaking up since changes are neatly encapsulated.

10. Modifiability:

- It is easy to make minor changes in the data representation or the procedures in an object oriented program.

11. Client/Server Architecture:

- It involves the transmission of messages back and forth over a network.

UML (Unified Modeling Language) is a general-purpose, graphical modeling language in the field of Software Engineering. UML is used to specify, visualize, construct, and document the artifacts (major elements) of the software system. It was initially developed by Grady Booch, Ivar Jacobson, and James Rumbaugh in 1994-95 at Rational software, and its further development was carried out through 1996. In 1997, it got adopted as a standard by the Object Management Group.

The UML stands for Unified modeling language, is a standardized general-purpose visual modeling language in the field of Software Engineering. It is used for specifying, visualizing, constructing, and documenting the primary artifacts of the software system. It helps in designing and characterizing, especially those software systems that incorporate the concept of Object orientation. It describes the working of both the software and hardware systems.

Goals of UML

- Since it is a general-purpose modeling language, it can be utilized by all the modelers.
- UML came into existence after the introduction of object-oriented concepts to systemize and consolidate the object-oriented development, due to the absence of standard methods at that time.
- The UML diagrams are made for business users, developers, ordinary people, or anyone who is looking forward to understand the system, such that the system can be software or non-software.
- Thus it can be concluded that the UML is a simple modeling approach that is used to model all the practical systems.

Characteristics of UML

The UML has the following features:

- It is a generalized modeling language.
- It is distinct from other programming languages like C++, Python, etc.
- It is interrelated to object-oriented analysis and design.
- It is used to visualize the workflow of the system.
- It is a pictorial language, used to generate powerful modeling artifacts.

Conceptual Modeling

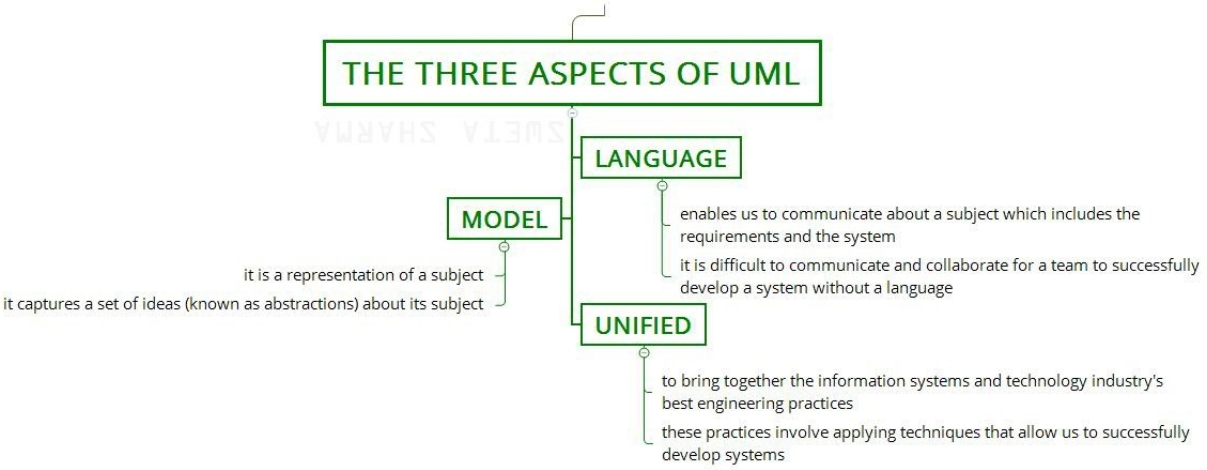
Before moving ahead with the concept of UML, we should first understand the basics of the conceptual model.

Role of UML in OO design

As the UML is a modeling language used to model software as well as non-software systems, but here it focuses on modeling OO software applications. It is essential to understand the relation between the OO design and UML. The OO design can be converted into the UML as and when required. The OO languages influence the programming world as they model real world objects.

The UML itself is an amalgamation of object-oriented notations like Object-Oriented Design (OOD), Object Modeling Technique (OMT), and Object-Oriented Software Engineering (OOSE). The strength of these three approaches is utilized by the UML to represent more consistency.

Conceptual Model of the Unified Modeling Language (UML)



1. Language:

- It enables us to communicate about a subject which includes the requirements and the system.
- It is difficult to communicate and collaborate for a team to successfully develop a system without a language.

2. Model:

- It is a representation of a subject.
- It captures a set of ideas *(known as abstractions)* about its subject.

3. Unified:

- It is to bring together the information systems and technology industry's best engineering practices.
- These practices involve applying techniques that allow us to successfully develop systems.

A Conceptual Model:

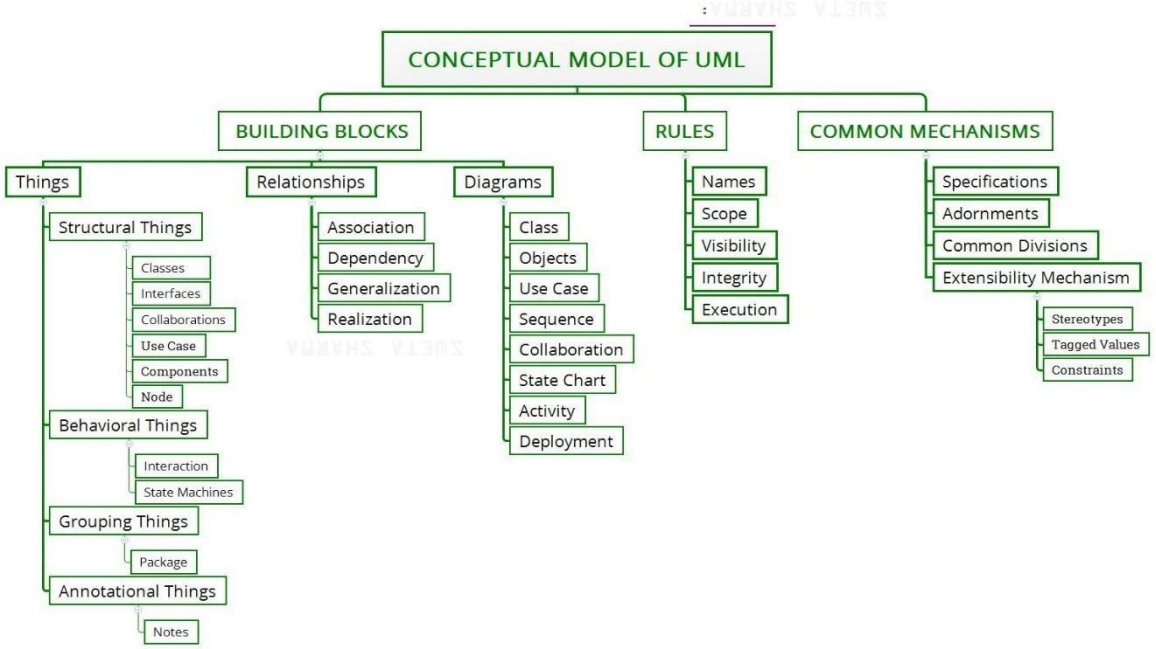
A conceptual model of the language underlines the three major elements:

• The Building Blocks

• The Rules

• Some Common Mechanisms

Once you understand these elements, you will be able to read and recognize the models as well as create some of them.



Building Blocks:
The vocabulary of the UML encompasses three kinds of building blocks:
Things:
Things are the abstractions that are first-class citizens in a model; relationships tie these things together; diagrams group interesting collections of things.
There are 4 kinds of things in the UML:

1. Structural things
2. Behavioral things
3. Grouping things
4. Annotational things

These things are the basic object-oriented building blocks of the UML. You use them to write well-formed models.

Relationships:
There are 4 kinds of relationships in the UML:
1. Dependency
2. Association
3. Generalization
4. Realization

These relationships are the basic relational building blocks of the UML.

1. **Diagrams:**
   It is the graphical presentation of a set of elements. It is rendered as a connected graph of vertices (things) and arcs (relationships).
2. **1.** Class diagram
3. **2.** Object diagram
4. **3.** Use case diagram
5. **4.** Sequence diagram
6. **5.** Collaboration diagram
7. **6.** Statechart diagram
8. **7.** Activity diagram
9. **8.** Component diagram
   9. Deployment diagram

Rules:
The UML has a number of rules that specify what a well-formed model should look like. A well-formed model is one that is semantically self-consistent and in harmony with all its related models.
The UML has semantic rules for:

1. **Names –** What you can call things, relationships, and diagrams.
2. **Scope –** The context that gives specific meaning to a name.
3. **Visibility –** How those names can be seen and used by others.
4. **Integrity –** How things properly and consistently relate to one another.
5. **Execution –** What it means to run or simulate a dynamic model.

Common Mechanisms:
The UML is made simpler by the four common mechanisms. They are as follows:
1. Specifications
2. Adornments
3. Common divisions
4. Extensibility mechanisms
Explain the Architecture of UML with Diagram

The unified Modelling language is not a programming language as such, but it is a visual language.

UML diagrams are used to represent different possible structures of the system. Architecture is a

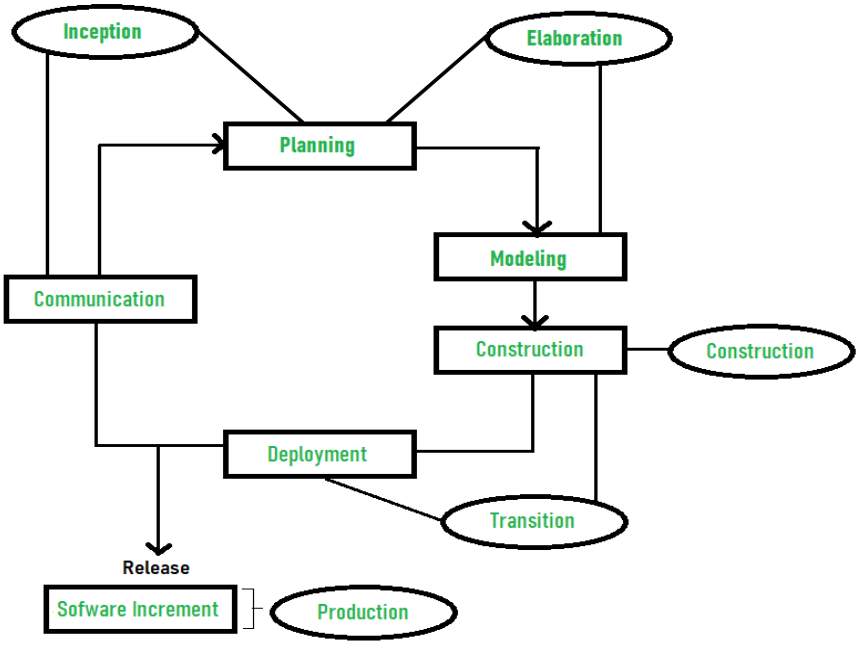
combination of several views of different users such as developer, analyst, tester, project manager,

technical writer, and end-user. For best practice, we consider 5 views: use case, design, development,

process, and deployment view. None the less use case remains the center of the other views.

Example- Class diagram , use case diagram etc

Rational Unified Process (RUP) is a software development process for object-oriented models. It is also known as the Unified Process Model. It is created by Rational corporation and is designed and documented using UML (Unified Modeling Language). This process is included in IBM Rational Method Composer (RMC) product. IBM (International Business Machine Corporation) allows us to customize, design, and personalize the unified process. RUP is proposed by Ivar Jacobson, Grady Bootch, and James Rambaugh. Some characteristics of RUP include use-case driven, Iterative (repetition of the process), and Incremental (increase in value) by nature, delivered online using web technology, can be customized or tailored in modular and electronic form, etc. RUP reduces unexpected development costs and prevents wastage of resources.
Phases of RUP: There is total of five phases of the life cycle of RUP:



1. **Inception –**
    - Communication and planning are the main ones.
    - Identifies the scope of the project using a use-case model allowing managers to estimate costs and time required.
    - Customers' requirements are identified and then it becomes easy to make a plan for the project.
    - The project plan, Project goal, risks, use-case model, and Project description, are made.
    - The project is checked against the milestone criteria and if it couldn't pass these criteria then the project can be either canceled or redesigned.
2. **Elaboration –**
    - Planning and modeling are the main ones.
    - A detailed evaluation and development plan is carried out and diminishes the risks.
    - Revise or redefine the use-case model (approx. 80%), business case, and risks.
    - Again, checked against milestone criteria and if it couldn't pass these criteria then again project can be canceled or redesigned.
    - Executable architecture baseline.
3. **Construction –**
    - The project is developed and completed.
    - System or source code is created and then testing is done.
    - Coding takes place.
4. **Transition –**
    - The final project is released to the public.
    - Transit the project from development into production.
    - Update project documentation.
    - Beta testing is conducted.
    - Defects are removed from the project based on feedback from the public.
5. **Production –**
    - The final phase of the model.
    - The project is maintained and updated accordingly.

Advantages:

1. It provides good documentation, it completes the process in itself.
2. It provides risk-management support.
3. It reuses the components, and hence total time duration is less.
4. Good online support is available in the form of tutorials and training.

Disadvantages:

1. Team of expert professional is required, as the process is complex.
2. Complex and not properly organized process.
3. More dependency on risk management.
4. Hard to integrate again and again.