14.2 UNIFIED MODELING LANGUAGE (UML)

Before we go into UA and OOSAD development, let us study the Unified Modeling Language (UML), which is extensively used in modeling the software system process development.

☞ In the OO software life cycle development, the following models are built to achieve the system objective

Model	Achievement
Use case model	Defines actors inside and outside of use case and their behaviour.
Business domain object model	Business domain modeled through objects.
Analysis object model	Presents information how the object model will be executed.
Design object model	Presents the detail design object model that will be created.
Implementation model	Converts design object model to implementation model, based on reusable component technology.
Test model	Stipulates test strategy, test plan, test specifications, test results and test recovery reports.

UML provides standard notations to model these different OO models, each having its own unique characteristics. UML is a language used to create an abstract system scenario, by visualising, specifying, constructing and documenting various parts and components of the system into a representative model enabling software system development.

UML uses 'object constraint language' (OCL), which uses simple logic for specifying system specifications. UML with OCL becomes powerful enough to map to OO languages, which developers use for coding the methods selected for object operations. The application of UML is one more step towards unification of best practices in the OO system development.

The primary goals in the design of UML (*The Unified Modeling Language, Notation Guide Version 1.1,* September 1997) are as follows:

☞
- Provide users with a ready-to-use, expressive and visual modeling language to develop models.
- Provide a language and notations to extend concepts to higher order representation.
- Independent of OO languages.

- Support higher-level development concepts like component technology, rapid application development, reusability, portability and inter-operability.

The inventors and promoters of UML are Grady Booch, Ivar Jacobson, and James Rumbaugh. They evolved UML notations and its semantics.

14.2.1 UML Diagrams

UML provides notations and diagrams to model the system in different views. The views and diagrams put together accurately represent the real-world scenario, as shown in Table 14.1.

Table 14.1 ***Views—UML Diagrams—Model***

View	*UML diagram*	*Model*
Domain/business	Class diagram	Static business model
Users	Use case diagrams	Static use case
Behaviour	Interactions diagrams ● Sequence diagram ● Collaboration diagram State chart diagram Activity diagram	Dynamic use case model similar to DFDs and work flow diagrams.
Implementation	Component diagram Deployment diagram	Deployment model

UML supports both static modeling and dynamic modeling, where static modeling shows the stand-alone static view, and dynamic modeling shows the changing behaviour view of the business system.

☞ Static models clearly bring out the structural aspect of the system at a point of time. The business model comprising various systems is modeled in the class diagram. It corresponds, to some extent, to the high-level modular structure of SSAD. Dynamic models, in contrast, model the behaviour of the system over time. They show the interaction of objects that achieve the goals of the system. DFD, work flow diagrams and system flow charts in SSAD are equivalent to interaction diagrams, state chart diagrams and activity diagrams.

The static model explains structure and relationship. For example, a delivery note object is made of more than one line item and belongs to more than one purchase order. The dynamic model explains the behavioural interaction of Worked Hours object with Salary object to compute salary. These two objects have an association through a relationship expressed in a method Compute Salary.

UML-based modeling of the system helps to represent the complexity of the system structure at all levels (data, application, system) and the behaviour of the system elements through interaction, based on rules, assumptions, constraints and conclusions.

☞ UML-based modeling offers the following benefits:

- Improves communication among project teams Reason: this leads to a uniform and common understanding of the system.
- Improves the developer's insight and visualization of the complex system.
- Developers learn faster to incorporate the system's intricacies correctly in the design.
- Prototype design is more appropriate, where the specific complexity of structure and behaviour is considered in each iteration. This improves the system in increments, and part by part.

In general terms, UML-based modeling benefits manager, designer and developer as it provides additional operational benefits as shown in Table 14.2.

Table 14.2

Additional Benefits of UML Modeling

Benefit	*Comment*
Clarity	Provides transparency in the system enabling detection of errors, mistakes and omissions in the life cycle development process.
Familiarity	Use case-based models bring familiarity to the system, as model represents, in parallel, the system flows (data, information, application).
Quicker Maintenance	Provides improved clarity and familiarity due to in-depth visibility in the system, which aids in the maintenance phase in early problem location, identification of solution and testing. This reduces maintenance cycle time.

14.2.2 UML Diagrams in UA

In this section, we study UML diagrams used in OOSAD. For OO modeling of the system, eight diagrams are proposed. Each of these plays a unique contributory role in improving the understanding of the system; they enhance clarity and visualization and aid in simplification of the system. The eight diagrams, with their modeling objective, are listed below.

Diagram	**Objective**
• Static class diagram • Use case diagram	Structure modeling
• Sequence diagram • Collaboration diagram • State chart diagram • Activity diagram	Behaviour modeling

● Component diagram ● Deployment diagram	Implementation and deployment modeling

We will now illustrate the diagrams in terms of the methods of drawing, the purpose of drawing and what their purpose. These models, shown in different diagrams, together contribute extensively to system development in UA.

Through structure modeling, we understand the system, and are in a better position to analyse it for software development. The static and stable view of the system helps to understand the partners and participants in the system. We are in a position to relate the system structure to organisation structure in terms of people and function.

Through behaviour modeling, we understand relationships, nature of interactions and responses between various elements of the system. The behaviour model throws light on functionality, features and outputs, within the framework of conditions and constraints, over a period of time.

Through implementation modeling, we learn how the system is moved from a development environment to a deployment environment.