K. J. Somaiya College of Engineering, Mumbai-77

(A constituent college of Somaiya Vidyavihar University)

Batch: D-2 **Roll No.:** 16010122151

Experiment / assignment / tutorial No. 3

TITLE: Study of Umbrello, a Unified Modelling Language tool

AIM: To understand importance of a common language used by various developers and practitioners for planning and coding.

Expected Course outcome of Experiment:

Learn Umbrello, the tool, used to create diagrams of software and other systems in the industry-standard UML format.

Books/ Journals/ Websites referred:

1.
2.
3.

Pre Lab/ Prior Concepts:

The Unified Modelling Language (UML) is a diagramming language or notation to specify, visualize and document models of Object Oriented software systems. UML is not a development method, that means it does not tell you what to do first and what to do next or how to design your system, but it helps you to visualize your design and communicate with others. UML is controlled by the Object Management Group (OMG) and is the industry standard for graphically describing software.

UML is designed for Object Oriented software design and has limited use for other programming paradigms.

UML is composed of many model elements that represent the different parts of a software system. The UML elements are used to create diagrams, which represent a certain part, or a point of view of the system.

The following types of diagrams are supported by Umbrello UML Modeller:

- Use Case Diagrams show actors (people or other users of the system), use cases (the scenarios when they use the system), and their relationships
- Class Diagrams show classes and the relationships between them
- Sequence Diagrams show objects and a sequence of method calls they make to other objects.

- Collaboration Diagrams show objects and their relationship, putting emphasis on the objects that participate in the message exchange
- State Diagrams show states, state changes and events in an object or a part of the system
- Activity Diagrams show activities and the changes from one activity to another with the events occurring in some part of the system
- Component Diagrams show the high level programming components (such as KParts or Java Beans).
- Deployment Diagrams show the instances of the components and their relationships.
- Entity Relationship Diagrams show data and the relationships and constraints between the data.

The above types of diagrams are drawn by different levels of designers/ coders to understand the product to be developed. They model the system and be used by developers, tester, project managers for various purposes.

The Umbrello software can be downloaded from https://umbrello.kde.org/

Installation steps :https://umbrello.kde.org/installation.php

Post Laboratory Activity:

1. Install Umbrello on the desktop
2. Study various options and menus, and get acquainted with the IDE.

Post Lab Descriptive Questions answers must be handwritten and to be submitted BEFORE the next tern.

1. Why do we need to draw different types of diagram during software development process?

   Different types of diagrams are needed in software development because each type serves a unique purpose and provides specific insights.

2. List various types of diagrams used in static and dynamic model of a software system.

   Static Model Diagrams
   Class Diagram: Represents classes, attributes, methods, and relationships.

Object Diagram: Shows instances of classes and their relationships at a specific point in time.
Component Diagram: Depicts the organization and dependencies among software components.
Deployment Diagram: Illustrates the physical deployment of artifacts on nodes.
Package Diagram: Organizes classes and components into packages to show their relationships.
ER Diagram (Entity-Relationship Diagram): Represents data entities and their relationships in a database.
Dynamic Model Diagrams
Use Case Diagram: Illustrates interactions between actors and the system, capturing functional requirements.
Sequence Diagram: Shows how objects interact in a particular scenario over time.
Collaboration Diagram: Similar to sequence diagrams but focuses on the relationships and interactions among objects.
State Diagram: Represents the states of an object and transitions between those states based on events.
Activity Diagram: Describes workflows and processes, showing the flow of control and data.
Timing Diagram: Focuses on the timing of messages and events in interactions between objects.