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

(A constituent college of Somaiya Vidyavihar University)

**Batch: A1**

**Roll No.: 16010120015, 16010120013, 16010120020**

**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. https://www.redhat.com/architect/dynamic-UML-diagramming

**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.

## Lucidchart

### Introduction:

Lucidchart is a web-based diagramming application that allows users to visually collaborate on drawing, revising and sharing charts and diagrams, and improve processes, systems, and organizational structures. It is produced by Lucid Software Inc., based in Utah, United States and co-founded by Ben Dilts and Karl Sun Lucidchart is used by companies such as Google, GE, NBC Universal, and Amazon.

### Features:

Lucidchart is entirely browser-based, running on browsers that support HTML5. This means it does not require plugins or updates of a third-party software like Adobe Flash. The platform supports real-time collaboration, allowing all users to work simultaneously on projects and see each user's additions reflected in real time. All data is encrypted and stored in secure data centers.

Its features include:

#### 1. Easily style shapes

When creating a diagram, chances are you've got specific fonts and colors in mind that you want used throughout. You could go back and update these preferences for every single shape—but that would a) decrease productivity and b) be cumbersome

Instead, before you drag any shapes onto the canvas, choose the font, font size, font color, and line style from the Properties bar at the top of the editor. After doing so, the styling you've chosen is automatically applied to every shape you drag and drop onto the canvas.

Lucidchart makes mass formatting changes easy, too. Click Select > All Shapes, and then style your entire diagram the way you want.

#### 2. Create clean diagrams

You may notice that as you drag shapes around, sometimes the connecting lines have a mind of their own. There's an easy hack for getting lines to make the most efficient (rather than the most creative) connections between shapes: smart lines.

To create a smart line, rather than dragging and dropping a line directly to the edge of the next shape, pull the line just past the edge until the shape gets the blue "halo" effect. Release your mouse, and you've got yourself a smart line that will make smart connections.

### 3. Work faster with keyboard shortcuts

Make life easier with our nifty keyboard shortcuts. Lucidchart has quite a few, which can be accessed in the editor by clicking F1 or selecting Help > Hotkey Reference. Here are two of our favorites:

- To easily zoom in and out of your diagram, hold down the space bar and scroll up and down with your mouse—this is especially helpful when you're dealing with large diagrams.
- To make more precise adjustments to your shapes, select the desired shape, hold down the Shift key, and use the arrow key for subtle movements that get your shape just the way you want it.

### 4. Include links and layers for easy-to-read diagrams

There's always additional detail you could add to a diagram. But what's the point of creating a visual if you stuff it full of text? Provide all the necessary info while still keeping your diagram clean and easy to read using shape actions.

Select a shape, click the lightning icon in the Properties bar, and select an action for the shape. You can toggle, show, or hide layers, link to another page inside the document, link to an external page, or link to an email. For example, if you choose "External Link," you can link the shape to a Google Doc containing additional context needed for that particular step.

### 5. Collaborate without leaving Lucidchart

Once you've got a diagram built, you'll want to gather feedback from others. Avoid giant email threads that lose or construe information by having these conversations without ever leaving Lucidchart.

With our commenting feature, you can leave comments on specific shapes or the document as a whole. To make a shape-specific comment, select a shape, choose "Comment" from the right-hand Dock, and select the "+" symbol. You can even @mention in your comment to alert certain people that feedback or review is needed. To find the location of a comment, click the pin icon in the top left corner. You can also use our chat feature to hold conversations inside your document.

### 6. Add necessary context

When you have thoughts or feedback that needs to stay permanently attached to a particular shape, use the notes function in Lucidchart. Right-click a shape, choose "Add Note," and include whatever information is needed. A grey icon indicates that a shape has a note.

Additional features include:

- A drag-and-drop interface
- Real-time co-authoring, shape-specific comments, and collaborative cursors
- Data linking
- Auto-visualization to generate org charts and ERDs
- SQL import and export capabilities

Lucidchart also supports importing files from draw.io, Gliffy, OmniGraffle, and Microsoft Visio. The platform is integrated with Google Workspace and Drive, Microsoft Teams and other Office products, Atlassian's Jira and Confluence, Salesforce, GitHub, Slack, and others.

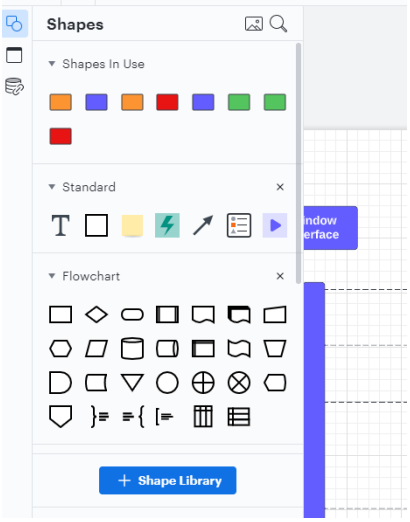
**Advantages :**

- Easy to use
- Clarify Complexity
- Great Versatility
- A large library of shapes & icons
- Practical Templates to Give the Chart Flavours

**Disadvantages :**

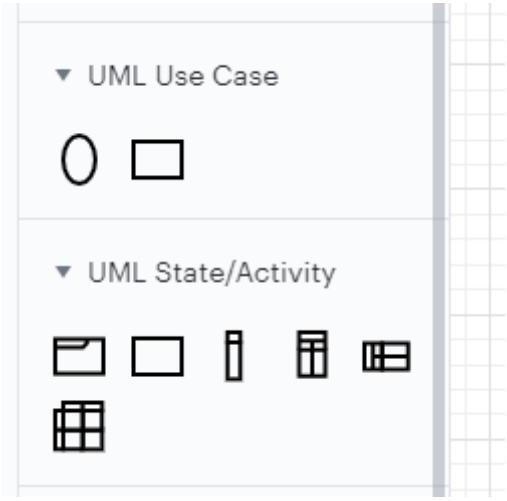
- No option to customize the color and size defaults
- Difficulty to find a certain diagram
- Connectors are difficult to navigate
- No option for recreating a new chart on your personalized chart
- Pricing

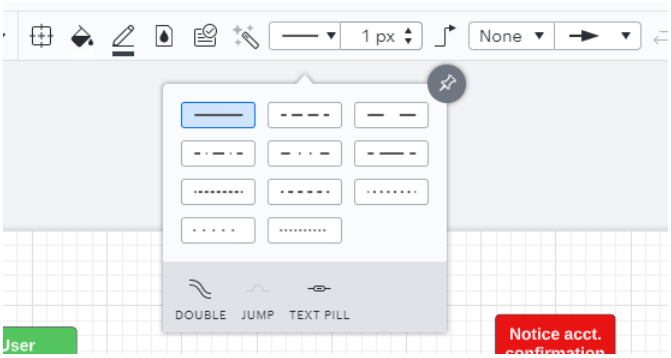
**Screenshots:**

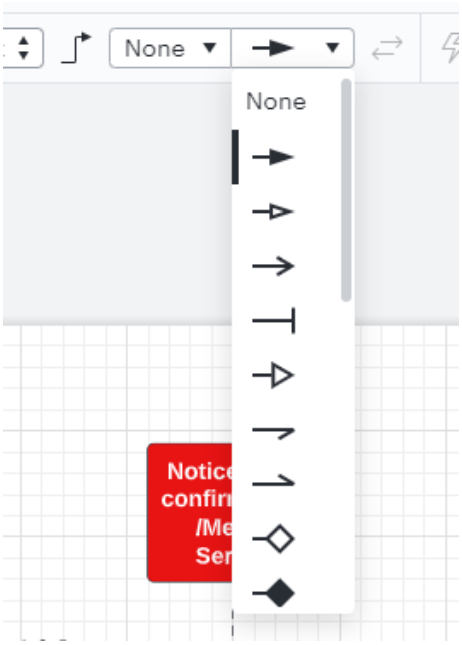


# K. J. Somaiya College of Engineering, Mumbai-77

(A constituent college of Somaiya Vidyavihar University)



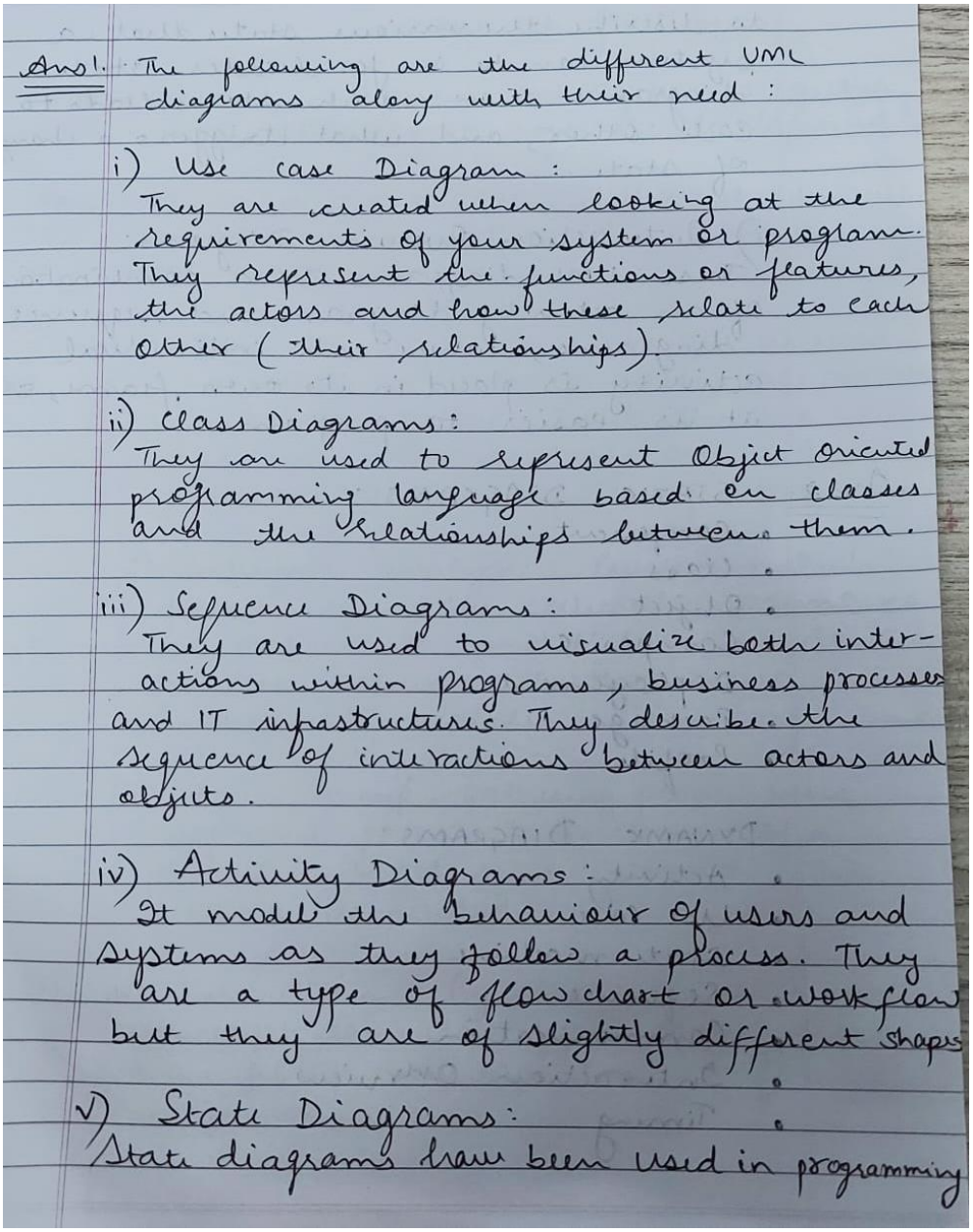
# K. J. Somaiya College of Engineering, Mumbai-77

(A constituent college of Somaiya Vidyavihar University)

**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?

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



Ans 1. The following are the different UML diagrams along with their need :

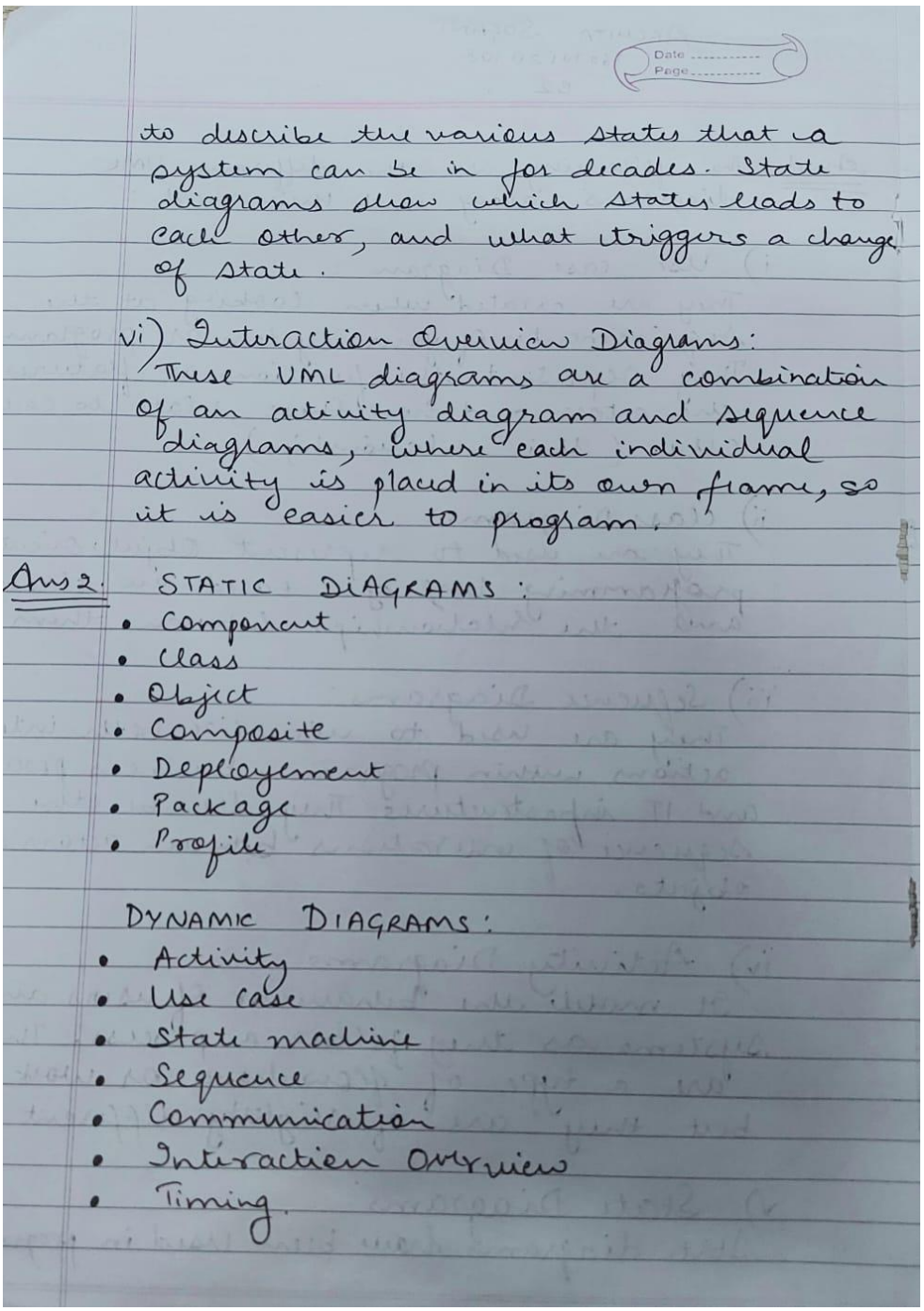
i) Use case Diagram :
They are created when looking at the requirements of your system or program. They represent the functions or features, the actors and how these relate to each other (their relationships).

ii) Class Diagrams :
They are used to represent Object oriented programming language based on classes and the relationships between them.

iii) Sequence Diagrams :
They are used to visualize both interactions within programs, business processes and IT infrastructure. They describe the sequence of interactions between actors and objects.

iv) Activity Diagrams :
It model the behaviour of users and systems as they follow a process. They are a type of flow chart or workflow but they are of slightly different shapes

v) State Diagrams :
State diagrams have been used in programming



to discribe the various states that a system can be in for decades. State diagrams show which states leads to each other, and what triggers a change of state.

vi) Interaction Overview Diagrams:
These UML diagrams are a combination of an activity diagram and sequence diagrams, where each individual activity is placed in its own frame, so it is easier to program.

Ans 2. STATIC DIAGRAMS:
- Component
- Class
- Object
- Composite
- Deployement
- Package
- Profile

DYNAMIC DIAGRAMS:
- Activity
- Use case
- State machine
- Sequence
- Communication
- Interaction Overview
- Timing.