|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | | **Hope Foundation’s**  **Finolex Academy of Management and Technology, Ratnagiri** | | | | | |
| **Department of Information Technology** | | | | | |
| Subject name: Software Design Lab | | | | | | Subject Code: ITL601 | |
| Class | | TEIT | | | Semester –VI (CBCGS) | Academic year: 2018-19 | |
| Name of Student | | **Kazi Jawwad A Rahim** | | | | **QUIZ Score : 6** | |
| Roll No | | 27 | | Assignment/Experiment No. | | 01 | |
| Title: **Introduction to UML** | | | | | | | |
|  | | | | | | | |
| **1. Lab objectives applicable: LOB4**. Understand the basics of UML with respect to use case, class, object and activity diagrams. | | | | | | | |
| **2. Lab outcomes applicable:**  **LO1**-Create designs of various scenarios using use case, class, object and activity diagrams. | | | | | | | |
| **3. Learning Objectives:**   1. To understand basics of UML. | | | | | | | |
| **4. Practical applications of the assignment/experiment:** Diagrams are used in order to give an insight for the development and modeling of any specific system. | | | | | | | |
| **5. Prerequisites**:   1. Not Required | | | | | | | |
| **6. Hardware Requirements**:  Not Required  **7. Software Requirements:**  Not required | | | | | | | |
|  | | | | | | | |
| **8. Quiz Questions (if any): (Online Exam will be taken separately batch-wise, attach the certificate/ Marks obtained)**   1. What is UML and why is it used? 2. Which are the various views of a software system? | | | | | | | |
|  | | | | | | | |
| **9. Experiment/Assignment Evaluation:** | | | | | | | |
| **Sr. No.** | **Parameters** | | | | | **Marks obtained** | **Out of** |
| **1** | Technical Understanding (Assessment may be done based on Q & A **or** any other relevant method.) Teacher should mention the other method used - | | | | |  | 6 |
| **2** | Neatness/presentation | | | | |  | 2 |
| **3** | Punctuality | | | | |  | 2 |
| **Date of performance (DOP)** | | |  | | **Total marks obtained** |  | **10** |

**Signature of the faculty**

**10. Theory:**

* **Introducing UML**

Object Oriented Analysis and Design (OOAD) is a technique for analyzing the requirements and creating the design for a software system. The analysis done using OOAD needs to be depicted in a graphical form. This method of displaying the information graphically is known as visual modeling.

The visual models of a software system can be created by using Unified Modeling Language (UML). UML provides a variety of diagrams to represent the structure and design of a software system. It helps project teams to communicate, explore potential designs, and validate the architectural design of the software.

* **Overview of UML**

UML is a standard language for creating blueprints that depict the structure and design of the software system. You can use UML for modeling systems that can range between enterprise information systems to distributed Web based applications.

UML is a notational language that enables the stakeholders to view the architecture of the system from various perspectives. There are several tools available, such as Visual Paradigm, StarUML, Rational Rose, ArgoUML,Visio, Poseidon, etc., which you can use to design software systems by using UML.

* **Evolution of UML**

The evolution of UML began with the need for a modeling language in which you can develop models of object-oriented software systems. During the mid1970s and late 1980s,object-oriented modeling languages were developed for the analysis and design of software. Of these modeling languages, the most prominently used languages were Booch's Booch’93, Jacobson's Object Oriented Software Engineering (OOSE), and Rumbaugh's Object Modeling Technique-2 (OMT). Each of these languages was complete in it and was recognized as having certain strengths and weaknesses.

Because several modeling languages were being used in the industry, there was a need for a standard language for modeling. In October 1994, GradyBooch, Ivor Jacobson, and Jim Rumbaugh of Rational Software Corporation began the unification of Booch’93, OMT, and OOSE. They developed a unified language and released version 0.9 and 0.91 of UML.

To standardize UML, Object Management Group (OMG), a non-profit organization for standardizing specifications, issued a Request for Proposal(RFP) in 1996. In response to the RFP issued by OMG, a UML consortium was established with the support of major software organizations such as HP,I-Logix, IBM, Microsoft, Oracle, Rational Software, and Unisys.

The UML consortium prepared and presented the revised version of UML 1.1 to OMG. Finally, OMG accepted UML as a standard modeling language and took over the responsibility for overseeing the maintenance of UML. Many versions of UML were released with more additions and modifications.

* **Viewing a Software System Architecture**

You can define the architecture of a software system as an arrangement of the static and dynamic constituents in a model. You can view the architecture of a software system from different perspectives. You need to do this because it helps the stakeholders of the system to make decisions about the project. The stakeholders may include analysts, architects, developers, testers, and end users.

The various views of a software system are:

**Use case view:** Indicates the functionalities that the system offers to each stakeholder.

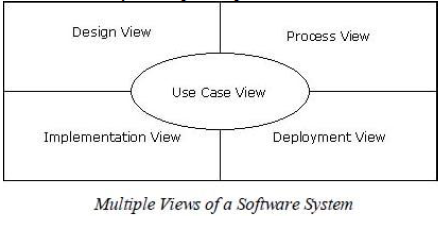
**Design view:** Focuses on the static and dynamic representation of the system. The stakeholders’ primary interest in viewing the system from the design perspective is to ensure that all the primary requirements of the system are implemented.

**Process view:** Represents the various processes executing in a system at a given instance of time. The primary focus of this view is to evaluate the execution of processes in terms of performance and scalability. The process view is again important for analysts, architects, and end users to view the information flow during process execution and how data is modified while processing.

**Implementation view:** Represents the physical system including the files and components required to assemble the system. This view focuses on various methods of assembling and configuring the components to release the required software system. This view is important for the client and project managers in terms of the plans for the release. It is important for developers and testers to know about the final components that need to be assembled for the system.

**Deployment view:** Represents the hardware components on which the software system will execute. This view is important to the client and developers because it enables them to identify the hardware and network configuration required to install the software system.

The following figure shows the multiple views that together represent the model of a software system designed using UML.



Each of the five views are interlinked. For example, the deployment view includes the classes, interfaces, and collaborations of the design view. In addition, the use case view is placed in the center because it is used to capture and represent the system requirements from the perspective of other views.

* **Types of UML Diagrams**

There are two broad categories of diagrams and then are again divided into sub-categories:

1. Structural Diagrams
2. Behavioral Diagrams

**Structural Diagrams:**

The *structural diagrams* represent the static aspect of the system. These static aspects represent those parts of a diagram which forms the main structure and therefore stable.

These static parts are represents by classes, interfaces, objects, components and nodes. The four structural diagrams are:

1. Class diagram
2. Object diagram
3. Component diagram
4. Deployment diagram

**Behavioral Diagrams:**

Any system can have two aspects, static and dynamic. So a model is considered as complete when both the aspects are covered fully.

Behavioral diagrams basically capture the dynamic aspect of a system. Dynamic aspect can be further described as the changing/moving parts of a system.

UML has the following five types of behavioral diagrams:

1. Use case diagram
2. Sequence diagram
3. Collaboration diagram
4. Statechart diagram
5. Activity diagram

**What is Star UML and rational rose?**

StarUML™ is a software modeling platform that supports UML (Unified Modeling Language). It is based on UML version 1.4 and provides eleven different types of diagram, and it accepts UML 2.0 notation It actively supports the MDA (Model Driven Architecture) approach by supporting the UML profile concept. StarUML™ excels in customizability to the user’s environment and has a high extensibility in its functionality. Using StarUML™, one of the top leading software modeling tools, will guarantee to maximize the productivity and quality of your software projects.

Rational Rose is an object-oriented Unified Modeling Language (UML) software design tool intended for visual modeling and component construction of enterprise-level software applications. In much the same way a theatrical director blocks out a play, a software designer uses Rational Rose to visually create (model) the framework for an application by blocking out classes with actors (stick figures), use case elements (ovals), objects (rectangles) and messages/relationships (arrows) in a sequence diagram using drag-and-drop symbols. Rational Rose documents the diagram as it is being constructed and then generates code in the designer's choice of C++, Visual Basic, Java, Oracle8, Corba or Data Definition Language.

**11. Learning Outcomes Achieved**

1. Students have got the overview of UML

**12. Conclusion:**

1. **Applications of the studied technique in industry**
   1. Every software industry uses UML modeling for designing software system and as a part of project documentation.
2. **Engineering Relevance** 
   1. Helpful in designing software system based on user requirements.
3. **Skills Developed**
   1. Familiarity with UML

**13. References**:

1. https://www.geeksforgeeks.org/unified-modeling-language-uml-introduction/
2. https://www.tutorialspoint.com/uml/
3. https://www.ibm.com/developerworks/rational/library/769.html