Name: Chetan Pardhi

Roll no.: B22 2nd Shift Vi Sem

Software Engineering Pract1

Aim

To study Unified Modeling Language (UML) perspectives and notations in StarUML

Theory

Define software engineering

Software engineering is a detailed study of engineering to the design, development and maintenance of software. Software engineering was introduced to address the issues of low-quality software projects. Problems arise when a software generally exceeds timelines, budgets, and reduced levels of quality. It ensures that the application is built consistently, correctly, on time and on budget and within requirements. The demand of software engineering also emerged to cater to the immense rate of change in user requirements and environment on which application is supposed to be working.

Need for Software engineering

The necessity of Software engineering appears because of a higher rate of progress in user requirements and the environment on which the program is working.

- Huge Programming: It is simpler to manufacture a wall than to a house or building, similarly, as the measure of programming becomes extensive engineering has to step to give it a scientific process.

- Adaptability: If the Software procedure were not based on scientific and engineering ideas, it would be simpler to re-create new Software than to scale an existing one.

- Cost: As the hardware industry has demonstrated its skills and huge manufacturing has let down the cost of computer and electronic hardware. But the cost of programming remains high if the proper process is not adapted.

- Dynamic Nature: The continually growing and adapting nature of programming hugely depends upon the environment in which the client works. If the quality of the Software is continually changing, new upgrades need to be done in the existing one.

- Quality Management: Better procedure of software development

  provides a better quality software product.

Define and Explain SDLC

The Software Development Life Cycle (SDLC) is a structured process that enables the production of high-quality, low-cost software, in the shortest possible production time. The goal of the SDLC is to produce superior software that meets and exceeds all customer expectations and demands. The SDLC defines and outlines a detailed plan with stages, or phases, that each encompass their own process and deliverables. Adherence to the SDLC enhances development speed and minimizes project risks and costs associated with alternative methods of production.

It is important because :

- It provides a standardized framework that defines activities and deliverables
- It aids in project planning, estimating, and scheduling
- It makes project tracking and control easier
- It increases visibility on all aspects of the life cycle to all stakeholders involved in the development process
- It increases the speed of development
- It improves client relations
- It decreases project risks
- It decreases project management expenses and the overall cost of production

UML and its need

In unified Modelling Language (UML) a use case diagram can summarize the details of the system's user. The uml diagram are really important since it really becomes very easy after designing the skeleton of the product. It can depicts the flaws in the project very easily. In UML, there are five diagrams in which we can design the nature of the project UML is a standardized modeling language that can be used across different programming languages and development processes, so the majority of software developers will understand it and be able to apply it to their work. This will help in greater collaboration and cooperation and also help in designing more creative and application oriented software for solving real world problems.

Define StarUML

StarUML is a sophisticated software modeler aimed to support agile and concise modeling.It is used for making class diagrams, state diagrams and all the other diagrams that can be used for designing efficient software. It is a universally accepted tool for designing software diagrams and models.

StarUMLTM 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. StarUMLTM excels in customizability to the user's environment and has a high extensibility in its functionality. Using StarUMLTM, one of the top leading software modeling tools, will guarantee to maximize the productivity and quality of your software projects.

Note on StarUML (intro) [overview, building blocks, modeling types]

Overview : StarUML is a sophisticated software modeler aimed to support *agile* and *concise* modeling. StarUML is an open source project to develop fast, flexible, extensible, featureful, and freely-available UML/MDA platform running on Win32platform. If you use StarUML(tm), you can easily and quickly design exact software models which is based on UML standard. It will guarantee to maximize the productivity and quality because of generating numerous results automatically from it.

Building Blocks : UML is composed of three main building blocks, i.e., things, relationships, and diagrams. Building blocks generate one complete UML model diagram by rotating around several different blocks. It plays an essential role in developing UML diagrams. The basic UML building blocks are enlisted below:

- Things : Anything that is a real world entity or object is termed as things. It can be further divided into structural, behavioral , grouping and annotational things.

- Relationships : It illustrates the meaningful connections between things. It shows the association between the entities and defines the functionality of an application. There are four types of relationships , namely dependency, association , generalization and realization.

- Diagrams : The diagrams are the graphical implementation of the models that incorporate symbols and text. Each symbol has a different meaning in the context of the UML diagram. There are thirteen different types of UML diagrams that are available in UML 2.0, such that each diagram has its own set of a symbol. And each diagram manifests a different dimension, perspective, and view of the system.

Model Types : **Model** or **software model** is a description of any aspect of a software system such as structure, behavior, requirement, and so on. A software model can be represented in textual, mathmatical or visual form. A **Model element** is a building block of a software model.

List of diagrams supported by StarUML

10 standard metamodel and diagrams:

- Class,
- Object,
- Use Case,
- Component,
- Deployment,
- Composite Structure,
- Sequence,
- Communication,
- Statechart,
- Activity,
- Timing,
- Interaction Overflow,
- Information Flow and
- Profile Diagram.

Define all diagrams with types(structural/behavioral) which are in list of StarUML (Question 9 is included)

The different diagrams in StarUml are :

Class Diagram (Structural Diagram)
Class diagram is a static diagram. It represents the static view of an application. Class diagram is not only used for visualizing, describing, and documenting

different aspects of a system but also for constructing executable code of the software application.
Class diagram shows a collection of classes, interfaces, associations, collaborations, and constraints. It is also known as a structural diagram.

Package diagram (Structural Diagram)
Package diagram, a kind of structural diagram, shows the arrangement and organization of model elements in middle to large scale project. Package diagram can show both structure and dependencies between sub-systems or modules, showing different views of a system, for example, as multi-layered (aka multi-tiered) application - multi-layered application model.

Object Diagram (Structural Diagram)
Object diagrams are derived from class diagrams so object diagrams are dependent upon class diagrams. Object diagrams represent an instance of a class diagram. The basic concepts are similar for class diagrams and object diagrams. Object diagrams also represent the static view of a system but this static view is a snapshot of the system at a particular moment.

Composite structure Diagram (Structural Diagram)
A composite structure diagram is a diagram that shows the internal structure of a classifier, including its interaction points to other parts of the system. It shows the configuration and relationship of parts, that together, perform the behavior of the containing classifier. Class elements have been described in great detail in the section on class diagrams. This section describes the way classes can be displayed as composite elements exposing interfaces and containing ports and parts.

Component Diagram (Structural Diagram)
Component diagrams are different in terms of nature and behavior. Component diagrams are used to model the physical aspects of a system. Now the question is, what are these physical aspects? Physical aspects are the elements such as executables, libraries, files, documents, etc. which reside in a node. Component diagrams are used to visualize the organization and relationships among components in a system. These diagrams are also used to make executable systems.

Deployment Diagram (Structural Diagram)
Deployment diagrams are used to visualize the topology of the physical components of a system, where the software components are deployed.

Deployment diagrams are used to describe the static deployment view of a system. Deployment diagrams consist of nodes and their relationships.

Use Case Diagram (Behavioral Diagram)
In the Unified Modeling Language (UML), a use case diagram can summarize the details of your system's users (also known as actors) and their interactions with the system. To build one, you'll use a set of specialized symbols and connectors.

Sequence Diagram (Behavioral Diagram)
UML Sequence Diagrams are interaction diagrams that detail how operations are carried out. They capture the interaction between objects in the context of a collaboration. Sequence Diagrams are time focus and they show the order of the interaction visually by using the vertical axis of the diagram to represent time what messages are sent and when.

Communication Diagram (Behavioral Diagram)
UML communication diagrams, like the sequence diagrams - a kind of interaction diagram, shows how objects interact. A communication diagram is an extension of object diagram that shows the objects along with the messages that travel from one to another. In addition to the associations among objects, communication diagram shows the messages the objects send each other.

Statechart Diagram (Behavioral Diagram)
The name of the diagram itself clarifies the purpose of the diagram and other details. It describes different states of a component in a system. The states are specific to a component/object of a system. A Statechart diagram describes a state machine. State machine can be defined as a machine which defines different states of an object and these states are controlled by external or internal events.

Activity Diagram (Behavioral Diagram)
Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system. The control flow is drawn from one operation to another. This flow can be sequential, branched, or concurrent. Activity diagrams deal with all type of flow control by using different elements such as fork, join, etc.

Profile Diagram (Structural Diagram)

Profile diagram, a kind of structural diagram in the Unified Modeling Language (UML), provides a generic extension mechanism for customizing UML models for particular domains and platforms. Extension mechanisms allow refining standard semantics in strictly additive manner, preventing them from contradicting standard semantics. Profiles are defined using stereotypes, tagged value definitions, and constraints which are applied to specific model elements, like Classes, Attributes, Operations, and Activities. A Profile is a collection of such extensions that collectively customize UML for a particular domain.

ER Diagram (Structural Diagram)
An entity relationship diagram (ERD) is a representation of data within a domain. It consists of entities as well as relationships between entities. An entity can be a tangible, physical object such as a school or student, or a concept such as a reply or a transaction. Entity can be identified by extracting objects that are relevant and meaningful to the problem domain and the system to develop. In entity relationship modeling, the term entity has synonyms "table", "database table", "entity-type". Yet, entity is the most commonly used term. Each entity brings along a set of columns, which are the properties of the entity the attributes belong to. For instance, entity Student has name, address and grade as columns (synonyms: attributes, properties, fields). Every entity must have at least one attribute that can be used to uniquely identify the entity, which is known as the entity's primary key(s).

Flowchart Diagram (Behavioral Diagram)
A flowchart is a diagram that depicts a process, system or computer algorithm. They are widely used in multiple fields to document, study, plan, improve and communicate often complex processes in clear, easy-to-understand diagrams. Flowcharts, sometimes spelled as flow charts, use rectangles, ovals, diamonds and potentially numerous other shapes to define the type of step, along with connecting arrows to define flow and sequence.

Dataflow Diagram (Behavioral Diagram)
A data-flow diagram is a way of representing a flow of data through a process or a system (usually an information system). The DFD also provides information about the outputs and inputs of each entity and the process itself. A data-flow diagram has no control flow — there are no decision rules and no loops. Specific operations based on the data can be represented by a flowchart.

Explain your case study with its title as per formats/explanation given in shared pdf

The idea of this case study is to design Agent Delivery System. There are many e-commerce online shopping portals running around, for instance amazon. One major task is to deliver an online books products to the customers as first as possible in a cost-effective (cheapest) manner. A delivery agent system, which would automatically receive a delivery request from an online portal and identify the couriers, whom the delivery job can be assigned.

Input:

- Shipping details (source and destination) locations
- Couriers' details in different localities.
- Service offering for each courier company.

Functions:

- Client can Track the delivery, aslo can all the delivery boy
- Client can order the product
- Client can check about the shipment and location of product
- Client can cancel the order
- Client can change the delivery date and time
- Reward and penalty calculation
- Archiving record of past services
- Client can provide feedback about the service
- User can Login and Logout from system

Output:

- Booking confirmation, if booking is successful.
- Reporting delivery status
- Cancellation of booking confirmation
- Update record on delivery rescheduling
- Update record and intimation on reward and penalty calculation
- Query generation on record of past service, given a courier agency.

Result

The need for Unified Modeling Language and StarUMLfor generating various UML diagrams has been studied.