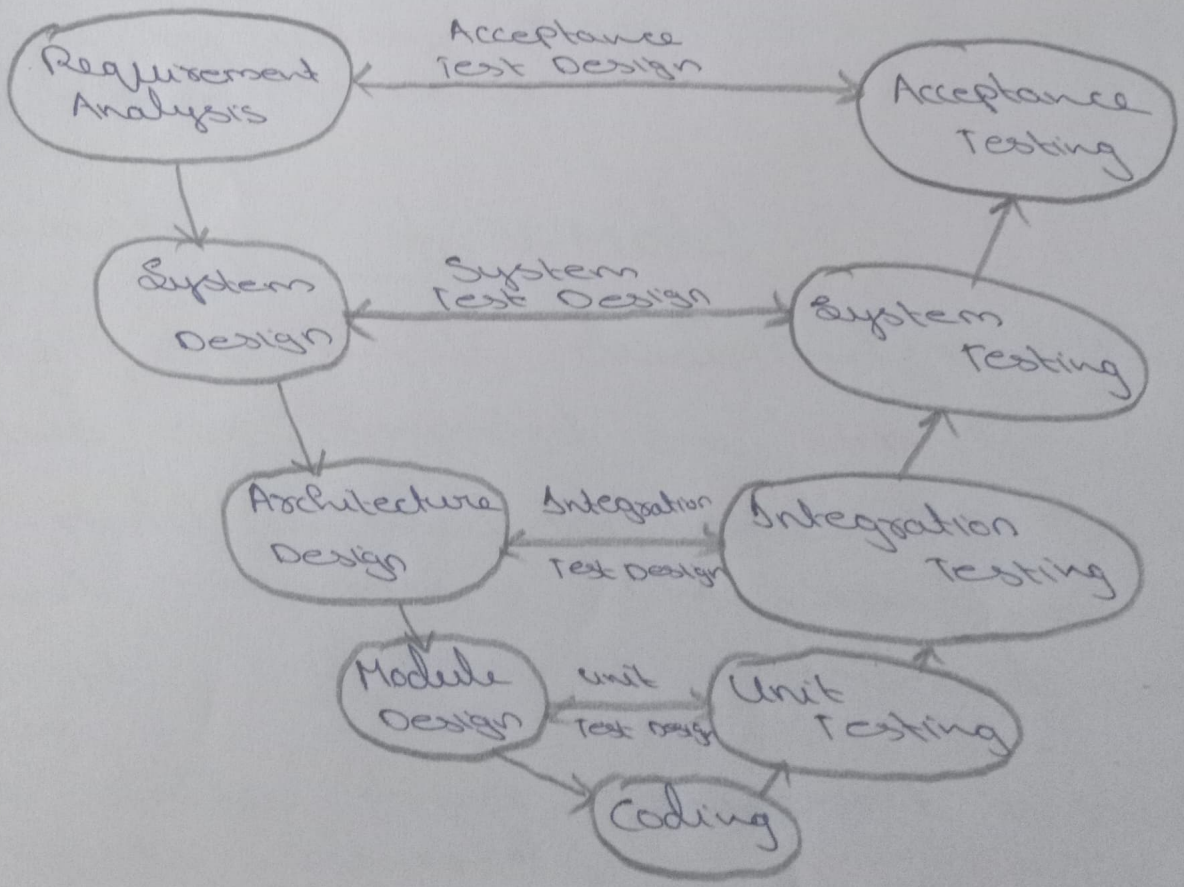
ADVANCED SOFTWARE ENGINEERING.

UMESH A. NAYAK
Roll no :- 58
Batch :- MCA-2021-23

Q1. What is V-model? Discuss about its advantages and disadvantages.

The V-model is a type of SDLC model where the process executes in a sequential manner in V-Shape. It is also known as Verification and Validation model. It is based on the association of a testing phase for each corresponding development stage. Development of each step directly associated with the testing phase. The next phase starts only after completion of the previous phase i.e. for each development activity, there is a testing activity corresponding to it.



Verification :- It involves static analysis technique (review) done without executing code. It is the process of evaluation of the product development phase to find whether specified requirements met.

Validation :- It involves dynamic analysis technique, testing done by executing code It is the process to evaluate the software after the completion of the development phase to determine whether software meets the customer expection and requirements.

Verification and validation phase are joined by coding phase in V-shape. Thus it is called V-Model.

Advantages of V-Model

- This is a highly disciplined model and phases are completed one at a time
- V-Model is used for small projects where project requirements are clear.
- Simple and easy to understand and use.
- It enables project management to track progress accurately.

Disadvantages

- High risk and uncertainty
- It is not good for complex and object oriented projects
- This model does not support iteration of phase.
- It does not easily handle concurrent events.

Q2 Discuss on Scenario based modelling with examples.

Requirements for a computer-based system can be seen in many different ways. The specific elements of the requirments model are dedicated to the analysis modeling method that is to be used.

- Scenario-based elements.

Using a scenario-based approach, system is described from user's point of view. For example, basic use cases and their corresponding use-case diagrams evolve into more elaborate template-based use cases. There are three levels of elaboration.

- Class-based elements:-

A collection of things that have similar attributes and common behaviors i.e; Objects are categorized into classes. For example, a UML case diagram can be used to depict a sensor class for the Safe Home security function.

- Flow-Oriented elements:-

As it flows through a computer-based system information is transformed. system accepts input, applies functions to transform it, and produces output in various forms.

Q3 Elaborate Webapp Design.

A web-application is an application program that is usually stored on a remote server and user can access it through the use of software known as web-browser.

A web application can be developed for several uses, which can be used by anyone like it can be

used as an individual or as a whole organization for several reasons.

In general, a web application can contain online shops, web mail's, calculators. There is also some kind of web application that usually requires a special kind of web browser to access them. We cannot access those kinds of web applications by using regular web-browsers. but can be accessed using a standard web browsers.

In general, web-application does not require downloading them because, as we already discussed, the web application is a computer program that usually resides on the remote server. Any user can access it by using one of the standard web browsers such as Google Chrome, Safari, Microsoft etc.

A web application are generally coded using the languages supported by almost every web browsers such as HTML, JavaScript because these are the languages that rely on the web browsers to render the program executable.

Q4 Explain the various UML Diagrams.

It is the general-purpose modeling language used to visualize the system. It is a graphical language that is standard to the software industry for specifying visualizing, constructing, and documenting the artifacts of the software systems, as well as for business modeling.

Benefits of UML

- Simplifies complex software design, can also implement OOPs like a concept that is widely used.
- It makes communication more clear and more real.

Types of UML

The UML diagrams are divided into two parts: structural UML diagrams and Behavioral UML diagrams which are listed below.

1. Structural UML diagrams
   - class diagram
   - Package diagram
   - Object diagram

- Component diagram
- Composite structure diagram
- Deployment diagram.

2. Behavioral UML diagrams
   - Activity diagram
   - Sequence diagram
   - Use case diagram
   - State diagram
   - Communication diagram
   - Interaction overview diagram
   - Timing diagram.

Structural Diagrams

Structural diagrams depict a static or structure of a system. It is widely used in the documentation of software architecture it embraces class diagrams, composite structure diagrams, component diagrams, deployment diagrams etc. It presents an outline for the system. It stresses the elements to be present that are to be modeled.

- Class Diagram:- They are one of the most widely used diagrams. It is the back bone of all the object-oriented software systems. It displays the system's class attributes and methods.

- Composite Structure Diagram:-

It show parts within the class. It displays the relationship between the parts and their configuration that as certain the behavior of the class

- Object Diagram

It describes the static structure of a system at a particular point in time. it can be used to test the accuracy of class diagrams. It represents distinct instances of classes and the relationship between them at a time.

- Component Diagram

It portrays the organization of the physical components within the system. It is used for modeling execution details. It determines whether the desired functional requirements have been considered by the planned development or not, as it depicts the structural relationships between the elements of a software system

- Deployment Diagram :-

It presents the system's software and it's hardware by telling what the existing physical components are and what

Software components are running on them.

- Package Diagram:-

It is used to illustrate how the packages are, and their elements are organized. It shows the dependencies between distinct packages. It is used for organizing the class and use case diagrams.

2. Behavioral Diagrams

Behavioral diagrams portray a dynamic view of a system or the behavior of a system, which describes the functioning of the system. It includes use case diagrams, state diagrams and activity diagrams.

- State machine Diagram

It is a behavioral diagram. It portrays the system's behavior utilizing finite state transitions. It is also known as the state charts diagram.

- Activity Diagram

It models the flow of control from one activity to the other. with the

help of an activity diagram, we can model sequential and concurrent activities

- Use case Diagrams

It represents the functionality of a system by utilizing actors and use cases It encapsulates the functional requirement of a system and its association with actors.

Q5. Discuss the different Software architecture Styles.

Software architecture is the base point of building Software. It shows the overall structure of the software, the collection of components in it, and how they interact with one another while hiding the implementation.

Different software architecture patterns

1. Layered Pattern
2. Client-server Pattern
3. Event-Driven Pattern
4. Microkernel Pattern
5. Micro services Pattern.

1. Layered Pattern:-

As the name suggests, components in this pattern are separated into layers of subtasks and they are arranged one above another.

Each layer has unique tasks to do and all the layers are independent of one another. Since each layer is independent, one can modify the code inside a layer without affecting others.

It is the most commonly used pattern for designing the majority software. This layer is also known as 'N-tier architecture'.

2. Client-Server Pattern:-

The client-server pattern has two major entities. They are a server and multiple clients.

Here the server has resources and a client requests the server for a particular resource. Then the server processes the request and responds back accordingly

3. Event - Driven Pattern:

Event - Driven Architecture is an agile approach in which services of the software are triggered by events. When a user takes action in the application built using the EDA approach a state change happens and a reaction is generated this called an event.

4. Microkernel Patterns

Micro kernel pattern has two major components. They are a core system and plug in modules.

- The core system handles the fundamental and minimal operations of the application.
- The plug-in modules handle the extended functionality and customized processing.

5. Micro Services Pattern:

The collection of small services that are combined to form the actual application is the concept of micro services pattern. Instead of building a bigger application small programs are built for every service of an application independently