**Unit-2**

**Q.1] What are objective of Software designing?**

* **1) Correctness :** Software design should be correct as per requirement.
* **2) Completeness :** The design should have all components like data structures, modules, and external interfaces, etc.
* **3) Efficiency :** Resources should be used efficiently by the program.
* **4) Flexibility :** Able to modify on changing needs.
* **5) Consistency :** There should not be any inconsistency in the design.
* **6) Maintainability :** The design should be so simple so that it can be easily maintainable by other designers.

**Q.2] What are quality attribute of Software designing?**

* **Performance:** It is measured by considering processing speed, response time, resource consumption, throughput and efficiency.
* **Supportability:** It combines the ability to extend the program, adaptability, serviceability. These three term defines the maintainability.
* **Usability:** It is accessed by considering the factors such as human factor, overall aesthetics, consistency and documentation.
* **Reliability:** It is evaluated by measuring parameters like frequency and security of failure, output result accuracy, the mean-time-to-failure(MTTF), recovery from failure and the program predictability.
* **Functionality:** It evaluates the feature set and capabilities of the program

**Q.3] Explain parameter of Software designing concept?**

* **1) Abstraction:** Is the process of hiding complex properties or characteristics from the software itself to keep things more simplistic.

-The developers will be able to hide the complicated and unnecessary details in the background while retaining core information in the foreground.

-Abstraction simply means to hide the details to reduce complexity and increases efficiency or quality.

* **2) Modularity**: Modularity simply means dividing the system or project into smaller parts to reduce the complexity of the system or project.

-Means subdividing a system into smaller parts and use them independently in different systems to perform different functions.

-Modularity make the system easy to manage.

* **3) Architecture-** Design a structure of something, Architecture simply means a technique to design a structure of something. Structure organization of program components (modules) and their interconnection Architecture Models are,

**(a) Structural Models--** An organized collection of program components

**(b) Framework Models--** Represents the design in more abstract way

**(c) Dynamic Models--** Represents the behavioural aspects indicating changes as a function of external events .

**(d) Process Models--** Focus on the design of the business or technical process.

* **4) Refinement- removes impurities:**  Refinement simply means to refine something to remove any impurities if present and increase the quality. Refinement is very necessary to find out any error if present and then to reduce it.
* **5)** **Pattern- a repeated form:** The pattern in the design process means the repetition of a solution to a common recurring problem within a certain context. Provides a description to enables a designer to determine the followings:

(a) whether the pattern is applicable to the current work

(b)Whether the pattern can be reused

(c) Whether the pattern can serve as a guide for developing a similar but functionally or structurally different pattern .

* **6) Information Hiding- hide the information**: Information hiding simply means to hide the information so that it cannot be accessed by an unwanted party.

-Hiding in such a way that the information gathered or contained in one module is hidden and can’t be accessed by any other modules.

* **7) Refactoring- reconstruct something:**Defined refactoring as “the process of changing a software system in a way that it won’t affect the behaviour of the design and improves the internal structure”.

-Refactoring simply means reconstructing something in such a way that it does not affect the behaviour of any other feature.

* **8) Information Hiding- hide the information**: Information hiding simply means to hide the information so that it cannot be accessed by an unwanted party.

- Hiding in such a way that the information gathered or contained in one module is hidden and can’t be accessed by any other modules.

|  |
| --- |
| design concept.png |

|  |
| --- |
| Design-Modeling-in-Software-Engineering.jpg |

**Q.4] Explain design modeling with diagram.**

* 1) Design modelling in software engineering represents the features of the software that helps engineer to develop it effectively,
* 2) Design modeling is useful in dealing with the architecture, the user interface, and the component level detail. It is mainly classified into four categories –
* **1) Data design: -** The data design element produced a model of data that represent a high level of abstraction. This model is then more refined into more implementation specific representation which is processed by the computer based system. The structure of data is the most important part of the software design.
* **2) Architectural design elements: a)** The architecture design elements provides us overall view of the system but the internal details of major components are ignored.

**b)** The architectural design element is generally represented as a set of interconnected subsystem that are derived from analysis packages in the requirement model.

**c)** Architectural design is the specification of the major components of a system, their responsibilities, properties, interfaces, and the relationships and interactions between them.

* **3. Interface design elements: a)**The interface design elements for software represents the information flow within it and out of the system.

**b)** They communicate between the components defined as part of architecture.

**c)** Interface design is the specification of the internal elements of all major system components, their properties, relationships, processing, and often their algorithms and the data structures.

* **4. Component level diagram elements: a)** The component level design for software is similar to the set of detailed specification of each room in a house. **b)** The component level design for the software completely describes the internal details of the each software component. **c)** The processing of data structure occurs in a component and an interface which allows all the component operations.
* **5. Deployment level design elements :** The deployment level design element shows the software functionality and subsystem that allocated in the physical computing environment which support the software.

**Q.5] What are designing principle of software.**

* The design process is a sequence of steps that enable the designer to describe all aspects of the software to be built. the design model that is created for software provides a variety of different views of the computer software.
* **1) The design process should not suffer from “tunnel vision.”** - A good designer should consider alternative approaches.
* **2) The design should be traceable to the analysis model.**- Because a single element of the design model often traces to multiple requirements, it is necessary to have a means for tracking how requirements have been satisﬁed by the design model
* **3) The design should not reinvent the wheel.** - Design time should be invested in representing truly new ideas and integrating those patterns that already exist.
* **4) The design should “minimize the intellectual distance” between the software and the problem as it exists in the real world**. -That is, the structure of the software design should (whenever possible) mimic the structure of the problem domain.
* **5)** **The design should exhibit uniformity and integration. -**Rules of style and format should be deﬁned for a design team before design work begins.
* **6) The design should be structured to accommodate change.-**The design concepts discussed in the next section enable a design to achieve this principle.
* **7) The design should be structured to degrade gently, even when aberrant data, events, or operating conditions are encountered.** -Well designed software should never “bomb.” It should be designed to accommodate unusual circumstances, and if it must terminate processing, do so in a graceful manner.
* **8) Design is not coding, coding is not design.** - The only design decisions made at the coding level address the small implementation details that enable the procedural design to be coded.

**Q.6] Explain taxonomy of software architecture with diagram.**

* **Taxonomy of Architectural styles & Patterns :**
* **1) Data centered architectures:** A data store (e.g., a ﬁle or database) resides at the center of this architecture and is accessed frequently by other components that update, add, delete, or otherwise modify data within the store. client software accesses the data independent of any changes to the data or the actions of other client software**.**
* **2) Data-ﬂow architectures : -**This architecture is applied when input data are to be transformed through a series of computational or manipulative components into output data.-  Each component, known as filter, transforms the data and sends this transformed data to other filters for further processing using the connector, known as pipe.

|  |
| --- |
| Data-flow-Architecture.jpg |

* **3) Call and return architecture**- A call and return architecture enables software designers to achieve a program structure, which can be easily modified. This style consists of the following two sub-styles.
* **Main program/subprogram architecture:** In this, function is decomposed into a control hierarchy where the main program invokes a number of program components, which in turn may invoke other components.
* **b) Remote procedure call architecture** : In this, components of the main or subprogram architecture are distributed over a network across multiple computers.
* **4) Object-oriented architectures:** The components of a system encapsulate data and the operations that must be applied to manipulate the data. Communication and coordination between components is accomplished via message passing. Object protect the system’s integrity.
* **5) Layered Architecture:** In layered architecture, several layers (components) are defined with each layer performing a well-defined set of operations.Each layer will do some operations that becomes closer to machine instruction set progressively. At the outer layer, components will receive the user interface operations and at the inner layers, components will perform the operating system interfacing(communication and coordination with OS).

**Q.7] What is UI design? Explain 3 golden rule.**

**UI design :** The design of interfaces between software components, the design of interfaces between the software and other nonhuman producers and consumers of information (i.e., other external entities), and the design of the interface between a human (i.e., the user) and the computer

**Three golden rule.: 1. Place the user in control:**  A number of design principles that allow the user to maintain control: 1) Define interaction modes in a way that does not force a user into unnecessary or undesired actions 2) Provide for ﬂexible interaction.

3) Allow user interaction to be interruptible and undo ( CTR+Z). Streamline interaction as skill levels advance and allow the interaction to be customized.

4) Hide technical internals from the casual user.

5) Design for direct interaction with objects that appear on the screen

**2.** **Reduce the User’s Memory Load:** The more a user has to remember, the more error-prone will be the interaction with the system. Principles that enable an interface to reduce the user’s memory load:

1) Reduce demand on short-term memory.

2) Establish meaningful defaults.

3) Deﬁne shortcuts that are intuitive.

4) The visual layout of the interface should be based on a real world metaphor.

5) Disclose information in a progressive fashion.

**3**. **Make the Interface Consistent:** A set of design principles that help make the interface consistent: 1)Allow the user to put the current task into a meaningful context.

2) Maintain consistency across a family of applications.

3) If past interactive models have created user expectations, do not make changes unless there is a compelling reason to do so.

**Q.8] Explain various UI user.**

* **1)Novices-** No syntactic knowledge of the system and little semantic knowledge of the application or computer usage in general.
* **2) Knowledgeable, intermittent users-** Reasonable semantic knowledge of the application but relatively low recall of syntactic information necessary to use the interface.
* **3) Knowledgeable, frequent users-** Good semantic and syntactic knowledge that often leads to the "power-user syndrome"; that is, individuals who look for shortcuts and abbreviated modes of interaction

**Q.9] What is UML?**

* The (UML) is a general-purpose visual modeling lan- guage that is used to specify, visualize, construct, and document the artifacts of a software system.
* It is used to understand, design, browse, configure, maintain, and control information about such systems.
* UML includes semantic concepts, notation, and guidelines.
* UML is not a programming language, it is rather a visual language.

**Q.10] What are Object oriented concept in UML? In brief.**

* **Class –**A class defines the blue print i.e. structure and functions of an object.
* **Objects** – Objects help us to decompose large systems and help us to modularize our system.
* **Inheritance** – Inheritance is a mechanism by which child classes inherit the properties of their parent classes.
* **Abstraction –**Mechanism by which implementation details are hidden from user.
* **Encapsulation –**Binding data together and protecting it from the outer world is referred to as encapsulation.
* **Polymorphism –**Mechanism by which functions or entities are able to exist in different forms.

**Q.11] What is Rumbaugh method? With class diagram.**

* The Rumbaugh methodology also known as OMT (Object Modeling Technique).

OMT is an approach used to develop manageable object-oriented systems and host object oriented programming. The purpose is to allow for class attributes, methods, inheritance, and association to be easily expressed. **OMT consists of four stages:** 1) Analysis, 2) Systems Design, 3) Object Design, 4) Implementation

1. **Additionally, OMT is always broken down into three separate parts. These parts are the: 1) An object model- object model & data dictionary**.

**–** Classes : a set of individual objects .

– Association lines: relationship among classes .

-Classes interconnected by association lines

**2) A dynamic model :**- state diagrams & event flow diagrams. OMT state transition diagram-network of states and events

**3) A functional model** - data flow & constraints. Shows flow of data between different processes in a business. Simple and intuitive method for describing business processes without focusing on the details of computer systems.

|  |
| --- |
| state transition.png |

**Q.12] Explain Booch Methodology with Diagrams of Booch method.**

* **Diagrams of Booch method:** • Class diagrams describe roles and responsibilities of objects

• Object diagrams describe the desired behaviour of the system in terms of scenarios

• State transition diagrams state of a class based on a stimulus

• Module diagrams to map out where each class & object should be declared

• Process diagrams to determine to which processor to allocate a process

• Interaction diagrams describes behaviour of the system in terms of scenarios

* **a) Macro Development Process**: • Steps for macro development process

Conceptualization: Analysis & Development of the model, Design or create the system architecture, Evolution or implementation, Maintenance.

* **b) Micro Development Process:** • Each macro process has its own micro development process Steps:- Identify classes & objects, Identify class & objects semantics, Identify class & object relationship, Identify class & objects interface and implementation

**Q.13] Describe Jacobson method ?**

* Also known as Object-Oriented Software Engineering (OOSE) or even Objectory, OOSE is a method used to plan, design, and implement object-oriented software. OOSE consist of five parts: a set of requirements, an analysis, a design, an implementation, and a testing model. the methodology or OOSE utilizes use cases in its design.

**Q.14] Explain with ATM diagram of Use case .**

* **The Jacobson methodology Use Cases:** -Use cases help us understand the how we want to design our system.

-Use cases are scenarios for helping us understanding the requirements of our system.

-We can see a use case as an interaction between a user and the system.

-A use case must contain these several elements: The how and when the use case begins and ends.

-Shows interactions between the various actors - when the interaction occurs and what material is exchanged

-The how and when of data storage and data usage Exceptions. The how the constraints of the problem domain are handled.

|  |
| --- |
| use case atm.jpg |

**Q.15] Describe Interaction diaram along with Log-In system.**

* Interaction diagrams is use to visualize the interactive behaviour of the system.
* Interaction Diagram are used to establish communication between objects
* This interactive behavior is represented in UML by two diagrams known as Sequence diagram and Collaboration diagram. The basic purpose of both the diagrams are similar
* Interaction diagrams mostly focus on message passing .
* Interaction diagrams capture the dynamic behavior of the system.

|  |
| --- |
| **E:\CSMSS\AI & DS\Software Engineering & Testing\Unit wise PPT\Unit 02\sequence diagram.png.crdownload** |

**Q.16] Explain transition (changing) states of a System with any one example**

* This is used to represent different transition (changing) states of a System.

-It is generally used to graphically represent all possible transition states a system can have and model such systems.

-The System consists of various states that are being represented using various symbols in the state transition diagram. You can see the symbols and their description given below :

**Initial State –** state dig. symbol.png

**Final State –** satate 2.png

**Simple State –** satate 3.png

**Composite State –** satate 4.png

|  |
| --- |
| UML-State-Diagram-15.png |