**Descriptive Questions on UML**

1. **What is pattern? how do patterns help the software?**

**Answer:**

"A pattern is the abstraction from a concrete form which keeps recurring in specific non-arbitrary contexts."-Riehle and Zullighoven(1996)

A pattern has

* A context
* Forces
* A software configuration

Patterns have been applied widely in software development

* + Organization patterns describe structures, roles and interactions in the software development organization itself

1. **What is the difference between synchronous and asynchronous?**

Answer:

A *synchronous message* or *procedural call*, shown with a full arrowhead, causes the invoking operation to suspend execution until the focus of control has been returned to it.

An *asynchronous message*, drawn with an open arrowhead, does not cause the invoking operation to halt execution while it awaits a return .

1. **Define event, state and transition?**

Answer:

**Event:** An occurrence that is of significance to the information system.

**State:** ‘A state is a condition during the life of an object or an interaction during which it satisfies some condition, performs some action or waits for some event.... .Conceptually, an object remains in a state for an interval of time’ (OMG, 2001)

**Transition:** the movement from one state or activity to another, triggered by an event. A transition may start and end at the same state.

1. **Difference between Cohesion and Coupling.**

Answer:

**Cohesion:** a measure of the degree to which an element of a model contributes to a single purpose.

**Coupling:** relates to the degree of interconnectedness between design components and is reflected by the number of links and the degree of interaction an object has with other objects.

1. **Explain briefly about MVC?**

Answer:

*Model—*provides the central functionality of the application and is aware of each of its dependent view and controller components.

*View—*corresponds to a particular style and format of presentation of information to the user. The view retrieves data from the model and updates its presentations when data has been changed in one of the other views. The view creates its associated controller.

*Controller*—accepts user input in the form of events that trigger the execution of operations within the model. These may cause changes to the information and in turn trigger updates in all the views ensuring that they are all up to date.

1. **Define Integrity constraint, Normalization**

Answer:

**Integrity constraint:** Systems analysis will have identified a series of Integrity constraints that have to be enforced to ensure that the application holds data that is mutually consistent and manipulates it correctly. These Integrity constraints come in various forms:

* Referential Integrity
* Dependency Constraints
* Domain Integrity

**Normalization:** a technique that groups attributes based upon functional dependencies according to several rules to produce normalized data structures that are largely redundancy free.

Normalization may be useful in OO approaches

* + when using a relational database management
  + as a guide to decomposing a large, complex (and probably not very cohesive) objects

Objects need not be normalised but it is important to remove redundancy

1. **What are the advantage and disadvantage of singleton pattern?**

Answer:

Advantage:

* It provides controlled access to the sole object instance as the Singleton class encapsulates the instance.
* The namespace is not unnecessarily extended with global variables.
* The Singleton class may be subclassed.
* A variation of this pattern can be used to create a specified number of instances if required.

Disadvantage:

* Using the pattern introduces some additional message passing.
* The pattern limits the flexibility of the application.
* The singleton pattern is quite well known and developers are tempted to use it in circumstances that are inappropriate. Patterns must be used with care.

1. **List the name of the fact finding techniques**

Answer:

* Background Reading
* Interviewing
* Observation
* Document Sampling

Questionnaires

1. **What is multiplicity? Write down advantages of components.**

Answer:

**Multiplicity:** Multiplicity (the range of allowed cardinalities) is the term used to describe constraints on the number of participating objects. Multiplicity reflects enterprise rules, which are constraints on the way that business activities can take place.

Advantages of components:

* Saves time – if we are building everything from scratch, probably we wasting time and efforts.
* Saves money – Developing from pre-existing elements certainly saves production cost.
* Increases confidence – Components are already tested, evaluated and tried and so we have little concerns on them.

1. **Quality criteria for good design?**

**Answer:**

* Cohesion is a measure of the degree to which an element contributes to a single purpose
* The concepts of coupling and cohesion are not mutually exclusive but actually support each other
* Coad and Yourdon (1991) suggested several ways in which coupling and cohesion can be applied within an object-oriented approach

**11. Advantages & disadvantages of traditional waterfall life cycle.**

Answer : Advantages of TLC

* Tasks in phases may be assigned to specialized teams
* Project progress evaluated at the end of each phase
* Manage projects with high levels of risks

Disadvantages of TLC

* Real projects rarely follow such a simple sequential life cycle
* Iterations are almost inevitable
* Time elapses between system engineering and the final installation
* The design is unresponsive to business changes during the project

**12.Phases of waterfall life cycle.**

Answer: The steps are:

1. System Engineering
2. Requirement analysis
3. Design
4. 4. Construction
5. Testing
6. Installation
7. Maintenance

**13.What do you mean by Prototyping? What are the steps tp prepare prototype?**

Answer : A prototype is a system or partially complete system that is built quickly to explore some aspects of the system requirements. It is not intended as the final system.

The main stages required to prepare a prototype are as follows:

1. Perform an initial analysis
2. Define prototype objectives
3. Specify prototype
4. Construct prototype
5. Evaluate prototype and recommend changes.

**14. What do you mean by Incremental Development?**

Answer: Involves some initial analysis to scope the problem and deliver the major requirements. The requirements are then reviewed and those that deliver most benefit to the client become the focus of the first increment of development and delivery. The installation of the first increment provides valuable feedback to the development team and informs the development of the second increment and so on.

**15. What is the difference between model and diagram?**

Answer:Amodel provides a complete view of a system at a particular stage and from a particular perspective.

A model may consist of a single diagram, but most consist of many related diagrams and supporting data and documentation.

A diagram illustrates some aspect of a system.

**16. What is the purpose of Activity Diagram?**

Answer: Purpose

* to model a task (for example in business modelling)
* to describe a function of a system represented by a use case
* to describe the logic of an operation
* to model the activities that make up the life cycle in the Unified Process

**17. What do you mean by Guard Condition?**

Answer: A Boolean expression associated with a transition that is evaluated at the time the event fires. The transition only takes place if the condition is true. A guard condition is a function that may involve parameters of the triggering event and also attributes and links of the object that owns the statechart.

**18. What is use case? What is the purpose of use case?**

Answer: Use Cases are the descriptions of the functionality of the system from the users’ perspective. Use case diagrams are used to show the functionality that the system will provide and to show which users will communicate with the system in some way to use that functionality.

Purpose

* document the functionality of the system from the users’ perspective
* document the scope of the system
* document the interaction between the users and the system using supporting use case descriptions (behaviour specifications)

**19. What is Stereotypes? Describe include & exclude.**

Answer: **Stereotypes**: A stereotype is a specialized UML modeling element that is constrained to behave in a particular way. For example- <<include>>, <<boundary>> are stereotypes used in UML.

**Include relationship:** A relationship between use cases where one use case includes modeling the implementation of a system.

**Extend relationship:** A relationship between use cases where one use case extends of adds new actions to another. Written as a stereotype: <<extend>>.

**20. Define boundary class, entity class and control class?**

Answer: **Boundry class:** A stereotyped class that provides an interface to users or other systems.

**Entity class:**  A setereotyped class that represents objects in the business domain modle.

**Control class:** A stereotyped class that controls the interaction between boundary classes and entity classes.

**21. How does a collaboration diagram differ from class diagram?**

Answer:

A collaboration diagram only shows those collaborate to provide the functionality of a particular use case.

A class diagram shows all the classes in a package.

A collaboration diagram only shows links those are required.

A class diagram shows all association between classes in a package.

**22. Distinguish between composition from aggregation.**

Answer:

An aggregation is a whole-part relationship between two or more object where a one represents the whole and the others represent the parts of the whole.

Composition is a “strong form of aggregation”. Here a lifetime dependency exists between objects – if whole is destroyed the parts are also destroyed. Moreover, each part may belong to only one whole at a time.

For example, consider the relation between a class and students. A student may attend many different classes and it is aggregation.

Again consider the relationship between a car and wheels. If the car is destroyed the wheels are also destroyed. It is composition.

Aggregation:

Class

Student

0..\*

1..\*

Composition:

Meal

Ingredient

1..\*

1

**23. Difference between sequence diagram and collaboration diagram**

Answer:

* Sequence diagrams have a time dimension (normally vertically down the page) while collaboration diagrams do not.
* Collaborations show the links between objects, which are not shown on sequence diagrams.

**24. What is an object lifeline and focus of control?**

Answer:

An object lifeline represents the existence of an object during an interaction in a sequence diagram.



* Focus of control indicates times during an activation when processing is taking place within that object
* Parts of an activation that are not within the focus of control represent periods when, for example, an operation is waiting for a return from another object
* Focus of control may be shown by shading those parts of the activation rectangle that correspond to active processing by an operation

Object A

Object B

message ()

**25. What is Layering and partitioning?**

Ans:

Two general approaches to the division of a software system into sub-systems

Layering, so called because the different sub-systems usually represent different levels of abstraction

Partitioning, this usually means that each sub-system focuses on a different aspect of the functionality of the system as a whole.

Both approaches are often used together on one system.

**26.Difference between patterns and framework:**

**Ans:**

Frameworks are partially completed software systems that may be targeted at a specified type of application.

However patterns

-Are more abstract and general than frameworks.

-Cannot be directly implemented in a particular software environment.

-are more primitive than frameworks.

**27. What is software architecture?**

**Ans:**

**A software architecture is a description of the sub-systems and components of a software system and the relationships between them. Subsystem and components are typically specified in different views to show the relevant functional and non-functional properties of a software system. The software architecture of a system is an artifact. It is the result of the software design activity.**

**28. Difference between algorithmic and non-algorithmic technique to operation specification?**

**Ans:**

**Algorithmic specification techniques describe the sequence of internal logical steps that an operation is to follow. Non-algorithmic techniques describe the result for a given set of inputs. Non-algorithmic methods of operation specification emphasize encapsulation.**

**29. Define Blackbox testing and whitebox testing?**

**Ans:**

Black Box Testing: It is also known as Skin Box testing or Closed Box Testing. This is used to find External functionalities of software.

White Box Testing: It is also known as Clear Box Testing or Glass Box or Open Box. This is used to find internal functionalities of software like conditional loops; statement coverage. It is mainly done by the Developers.

(OR)

Black Box Testing is a software testing method in which the internal structure/ design/ implementation of the item being tested is NOT known to the tester .

White Box Testing is a software testing method in which the internal structure/ design/ implementation of the item being tested is known to the tester.

**30. Define various levels of testing such as**

* **unit testing**
* **integration testing**
* **sub-system testing**
* **system testing**
* **acceptance testing**

**Answer:**

* **unit testing**

A unit is the smallest piece of software that can be tested. This usually means the software

can be compiled, linked or loaded into memory. In an object-oriented programming language, this typically refers to simple classes and interfaces.

* **integration testing**

The beginning of integration testing assumes a reliable and complete set of unit tests. At

this stage we begin to combine the different tested units or components to form a working

subsystem. Despite the fact that a unit has been through a successful unit test, it might still

behave unpredictably when interacting with other components in the system.

* **system testing**

System test planning phase is very dependent on the high-level design specification in the

development process. As a result any errors made in translating the requirements specification and

the design specification would be very drastic as it would propagate downwards to the lower levels

of test and development.

* **acceptance testing**

Acceptance testing is concerned with showing that the end product does not meet the user

requirement i.e. that the user is not satisfied with the system. Since acceptance testing is based

solely on user requirements specs, it is usually the first to receive full planning.

* **sub-system testing**

Use cases that share the same persistent data should be tested together. This is one form of sub-system testing in which the sub-systems are build around different business functions that make use of the same stored data.

**For MCQ Chapter 6to11,13to15,17to18**