**Descriptive Questions on UML**

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

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?/**

**Event:** An occurrence that 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.

**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.?/**

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

**Coupling:** relates to the degree of interconnection 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? /**

*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.

*Controller*—accepts user input in the form of events that trigger the execution of operations within the model.

1. **Define Integrity constraint, Normalization/**

**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.

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

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 subclasses.
* 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.

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

* 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.

1. **What do you mean by Guard Condition?/**

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.

1. **What is Stereotypes? Describe include & exclude./**

**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>>.

1. **Difference between sequence diagram and collaboration diagram/**

* 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.

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

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 ()

1. **What is Layering and partitioning?/**

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.

1. **Difference between patterns and framework:/**

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.

1. **What is software architecture?**

A software architecture is a description of the sub-systems and components of a software system and the relationships between them. The software architecture of a system is an artifact. It is the result of the software design activity.

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

-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.

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

-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.

1. **Define various levels of testing such as/**

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

* **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.

* **integration testing**

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

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. 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.