Biyani's Think Tank

Concept based notes

Object Oriented Software Engginering

MCA

Ms Kritika Saxena
Deptt. of IT
Biyani Girls College, Jaipur

Published by :
Think Tanks
Biyani Group of Colleges

Sector-3, Vidhyadhar Nagar,
Jaipur-302 023 (Rajasthan)
Ph : 0141-2338371, 2338591-95 • Fax : 0141-2338007
E-mail : acad@biyanicolleges.org
Website :www.gurukpo.com; www.biyanicolleges.org

Edition : 2012

Leaser Type Setted by :
Biyani College Printing Department

Preface

I am glad to present this book, especially designed to serve the needs of the students. The book has been written keeping in mind the general weakness in understanding the fundamental concepts of the topics. The book is self-explanatory and adopts the “Teach Yourself” style. It is based on question-answer pattern. The language of book is quite easy and understandable based on scientific approach.

Any further improvement in the contents of the book by making corrections, omission and inclusion is keen to be achieved based on suggestions from the readers for which the author shall be obliged.

I acknowledge special thanks to Mr. Rajeev Biyani, *Chairman* & Dr. Sanjay Biyani, *Director* (*Acad.*) Biyani Group of Colleges, who are the backbones and main concept provider and also have been constant source of motivation throughout this endeavour. They played an active role in coordinating the various stages of this endeavour and spearheaded the publishing work.

I look forward to receiving valuable suggestions from professors of various educational institutions, other faculty members and students for improvement of the quality of the book. The reader may feel free to send in their comments and suggestions to the under mentioned address.

Author

Chapter -1
Introduction to object oriented software engineering

Q-1 Define the term software.

Ans: In general, software can be defined as a collection of computer programs; which in turn is a collection of commands. It is a list of instruction designed to perform certain processing on the input & to produce certain results.

It is written to handle an input-process-output system to achieve predetermined goals.

Q-2 What is object oriented software engineering.

Ans: OOSE is an object modeling language and methodology. OOSE is called its appropriate design patterns are recognized they may also be received. OO design establishes a design model to define the OO architecture thus improving development speed and end product quality.

In OO approach, software is built unit objects that encapsulate (hide) the function and data. OOSE is self contained objects that can be replaced, modified and received.

In the OO system everything is an object and each object is responsible for itself.

Q-3 What is Unified Modeling Language?

Ans: UML is used as a universal approach to all problems of s/w design. To solve this problem, a unified approach evolved from the combined efforts of Booth, Rumbaugh and Jacobson.

UML is organized into two major design activities system design and object design.

Conceptual Model of UML

UML requires three major elements to build its conceptual model-

(i) UML is building blocks
(ii) Rules that dictate how those building blocks be put together.
(iii) Some common mechanisms that apply throughout UML

Q-4 Define basic structure and modeling classes.

Ans.: Some of major consideration in basic structural modeling includes

(i) Class
(ii) Relationships
(iii) Common mechanism
(iv) Diagrams
(v) Class diagram

Q-5 What are relationships? Define their Types?

Ans: Relationship provides a path way for communication b/w objects. There are mainly five

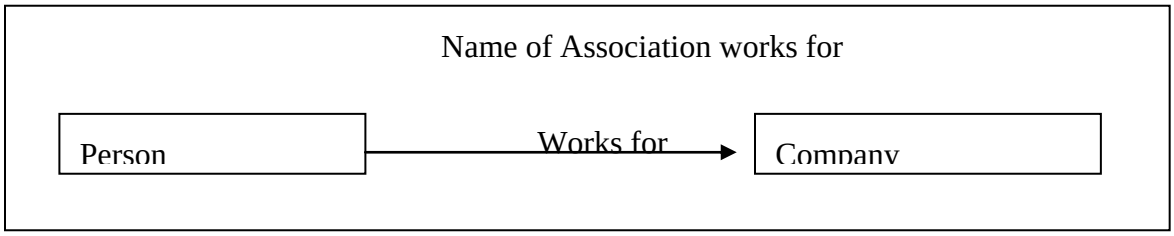
Types of relationships in UML.

(i) Association
(ii) Aggregation/Composition
(iii) Dependency
(iv) Generalization/specialization
(v) Realization

1- **Association:-** Association is a bidirectional structural relationship that describes connection among the classes. If the association is between two classes then it is called binary Association. It is represented by solid line.

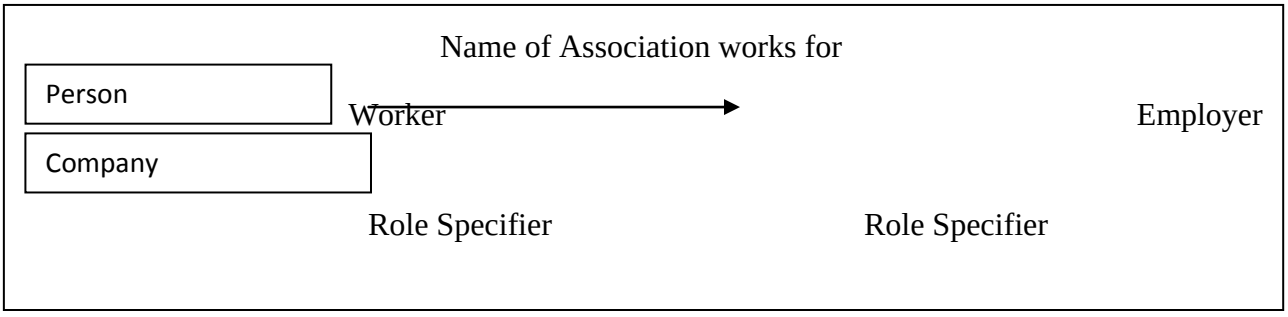
Key Points about association are as follows:

I - The nature of relationship between two classes is described by the name of association.



A Simple association

II- A class participates in an association with another class and has some specific roles. One class may have different roles in different association.

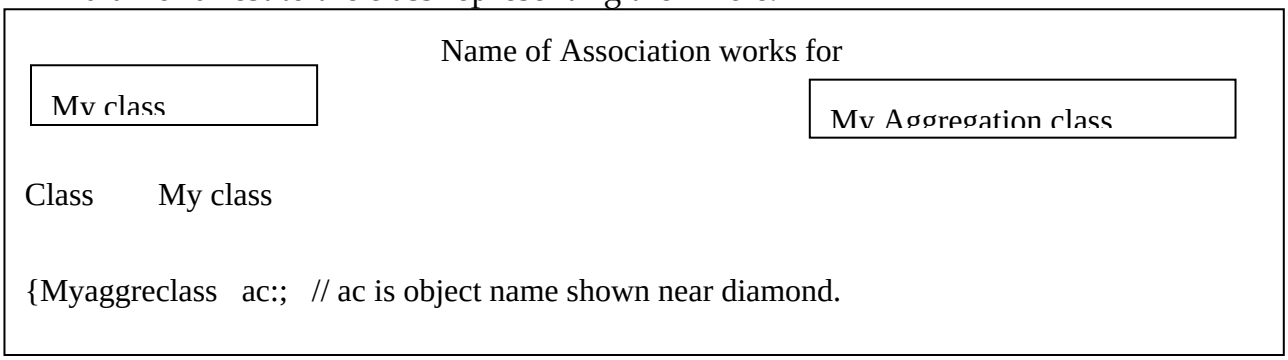


A Simple Association Showing Specific Roles

2- Aggregation/ Composition:- Aggregation is used to connect two classes but in a stronger way. It is in fact type of association where one of the class participating in a relations.

Key Points about Aggregation are as follows:

I- An aggregation is shown as a line connecting the related class with a write diamond rest to the class representing the whole.

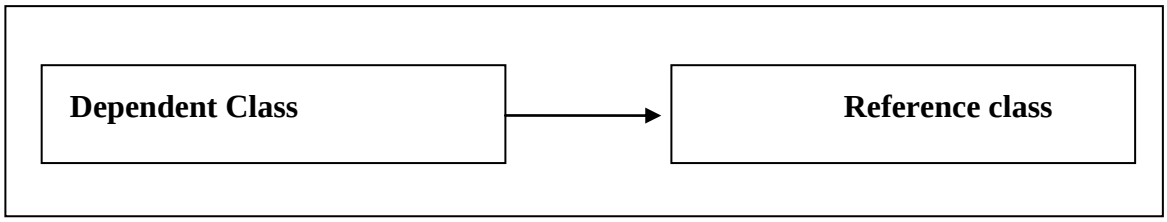


3. Dependency: - A dependency states that change in specification of one class may affect another class.

Key points about depending as follows:-

II- They are represented by dashed directed lines.
III- A depending may have a name too, although it is used very often.

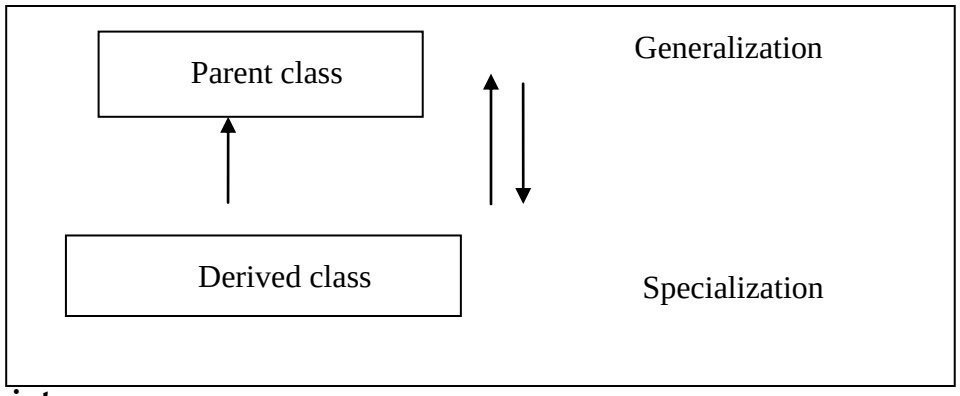
Showing Dependency



4. Generalization/Specification: - It comes from the property of inheritance of OOPS. The term generalization is for the inheritance in the bottom to up direction that is, form parent class to the derived class while another relation is specialization.

Which is in direction from the parent class to the derived class?

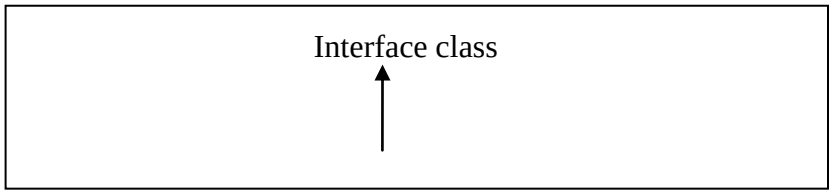
Fig.:-



Key points are:-

1- It is represented by a solid line with a large arrow head pointing towards the parent class
2- There may be single, multi level and inheritance.

5. Realization: - Realization is derived from an interface instead of a base class. The derived class implements the operations declared by the interface and thus the relation b/w an interface and the class that implements the interface called realization.



Q-6 Explain common mechanism in brief?

Ans: UML has four common mechanisms that we can apply throughout the language

(i) Specification (ii) Common Divisions

(iii) Adornments (iv) Extensibility mechanisms.

(i) Specifications:- The UML is more than a graphical language. Rather, behind every part of its graphical notation there is a specification that provides a textual statement of the syntax and semantics of the building block.

(ii) Adornments:- Notes are the most important kind of adornment that stand alone. A note is a graphical symbol for rendering constraints attached to an element or a collection of elements.

(iii) Common Division:- In modeling Object Oriented systems, the orld often gets divided in several ways. First, there is the division of class and object.

A class is an abstraction, an object is one concrete manifestation of that abstraction.

Graphically, the UML distinguish an object by using the same symbol as its class and then simply underlying the objects name

(iv) Extensibility Mechanisms: - UML is opened ended language, making it possible for You to extend the language in controlled ways :

There are three:-

(1) **Stereotype:-** Stereotype is used when you are modeling a network, you might want to have symbols for routers and hubs; then we use stereotype nodes to make these things appear as primitive building blocks.

(2) **Tagged value:-** A tagged value extends the properties of UML building

blocks ,
Allowing user to create new information in that element's specification.

(3) **Constraint**: - This allows user to add new rules or modify existing ones.

Q-7 Define class diagram, their attributes and operations?

Ans: Class Diagram: - Class diagram show a set of classes, interfaces, Collaborations & Their relationships. Class diagram are specifically used to model the static design view of system.

We all know that class is known as collection of objects with common structure, common behavior & common relationships. A class diagram shows the existence of classes and their relationships in the logical view of a system

Class Name
Attributes
Operations

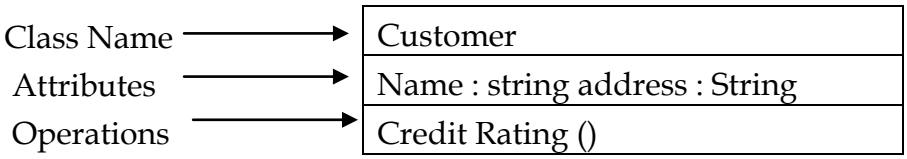
Format of Object oriented Classes.

- Visibility of Attributes and operations:- Visibility rules defines the concept of encapsulation. There are 4 visibility levels defined in UML
- (I) Public (II) Private (III) Protected (iv) Package

Key-points:-

- The class name should be at the top, it is a compulsory field and rests are optional field.
- The attributes are mentioned next. The public attributes start with + sign. Protected with # sign and the private attribute with –sign.
- The operations are also represented in a similar ways like the sign #, + & - are used for protected, public, private.
- Lastly, there is an optional field for writing comments.

Example:-



Q- 8 Explain common Modeling Techniques?

Ans:- Common Modeling techniques are defined by three types:-

(i) Common Modeling Techniques for Class
(ii) Common Modeling Techniques for Relationships
(iii) Common Modeling Techniques for Common Mechanism

- **Common Modeling Techniques for Classes :-**
  (i) Modeling the vocabulary of a system
  (ii) Modeling the Distribution of responsibilities in a systems
  (iii) Modeling Non-S/W Things
  (iv) Modeling primitive types

- **Common modeling Techniques for Relationships**
  (i) Modeling Single Dependencies
  (ii) Modeling Single Inheritance
  (iii) Modeling Structural Relationship

- **Common Modeling Techniques for Common Mechanism :-**
  (i) Modeling Comments
  (ii) Modeling New Building Blocks
  (iii) Modeling New Properties
  (iv) Modeling new Semantics

Q.9 What are UML Diagram Types?

Ans. There are several types of UML diagrams:

Use-case Diagram

The use case diagram Shows actors, use-cases, and the relationships

between them.

Class Diagram
The Class Diagram Shows relationships between classes and pertinent information about classes themselves.

Object Diagram
The Object diagram Shows a configuration of objects in time.

Interaction Diagrams
It Show an interaction between a group of collaborating objects.
Two types: Collaboration diagram and sequence diagram

State Diagram
It Describes behavior of instances of a class in terms of states and transitions.

Activity Diagram
It shows actions and decision points, but with the ability to accommodate concurrency.

Deployment Diagram
It Shows configuration of hardware and software in a distributed system.

Component Diagram
This depicts how components are wired together to form larger components and or software systems.

Q.10. What is the Component Diagram?

Ans. Effective use of UML

1) Keep in mind the purpose of a particular diagram and use appropriate detail for that purpose.
2) Beware of arguments about notional choice, such as aggregation versus composition, that are ultimately inconsequential.
3) Don't fill design documents with unnecessary diagrams.
4) Don't get carried away with drawing tools.

Q.10. What is the Role of UML in OO design?

Ans. UML is a modeling language used to non software systems and model software. Although UML is used for non software systems the Supports on modeling object oriented software applications. If we look into class diagram, object diagram, collaboration diagram, interaction diagrams all would basically be designed based on the objects.

So the relation between OO design and UML is very important to understand for us. The OO design is transformed into UML diagrams according to the requirement. Once the OO analysis and design is done the next step is very easy. The input from the OO analysis and design is the input to the UML diagrams.

Q.11 What is the purpose of OO analysis and design ?

Ans. The purpose of OO analysis and design can described as:

- Identifying the objects of a system.
- Identify their relationships.
- Make a design which can be converted to executables using OO languages.

Case Study

A sample case study following UML 2.0

Case Study – Hospital Management System

Problem Statement:

A system to manage the activities in a hospital:

- Patients request for appointment for any doctor. The details of the existing patients are retrieved by the system. New patients update their details in the system before they request for appointment with the help of assistant.
- The assistant confirms the appointment based on the availability of free slots for the respective doctors and the patient is informed.
- Assistant may cancel the appointment at any time.

Now in this case study, we have exploded only one use case which has the "include" and "extend" conditions. The corresponding sequence diagram and design level class diagrams for this use case diagram is also shown.

Chapter-2
Advanced Structured Modeling

Q-1 Define advanced structured modeling and what is advanced class and relationships?

Ans: Advanced structural Model: - A view of a system that emphasizes the structure of the advanced objects includes their classifiers, advanced relationships attributes and operations.

- Advanced Class: - Class is the most important building blocks of any OO systems. Class is one kind of even more general building block in UML classifier. Classes have a name attribute, operations, and responsibilities. Classifier is a mechanism that describes structural and behavioral feature. Classifier in classes, interfaces, data types, signals, components, nodes, use cases and subsystems.

- **Advantages of Class :-**

(i) Multiplicity (ii) Visibility
(iii) Template class (iv) Abstract, root, leaf, multiple inheritances.
(v) Scope

1.Visibility: - Means when we specify the attributes and operations of a class, we also have the ability to determine the visibility of each feature with the UML.

- Visibility is applied to both attributes and operations in a class.
- Level of Visibility :-

(i) Private Scope:- Within a class
(ii) Package Scope:- Within the some Package
(iii) Public Scope:- Publicly

(iv) Protected Scope:- Within an Inheritance tree.

2. Scope:- In scope another detail that can be specified for a classfier's attributes and operations is its owner scope.

Kinds of Scope:-

(i) Instance: - Each instances of the classifier holds its own value of the feature.

(ii) Classifier: - There is just one value of the feature for all instances of the classifier.

Advanced Relationships:-

When things are modeled that form that vocabulary of system one must also model how those things stand in relation with each other. Dependencies, generalizations and associations are the most important relational building blocks of UML. One can also model multiple inheritance, navigation, composition, refinement and other characteristics. Using a fourth kind of relationship realization one can model the connection between an interface and a class or component or b/w a use case and collaboration.

- **Links:** - It is a physical or conceptual connection between object instances. Once a link is established between two objects, each object may request the services of the other object. In Object Oriented modeling, all links are considered as bidirectional.

- **Role Name :-** A role name is a name that uniquely identifies one end of an association.

- **Ordering:** - In one to many association, the objects on many side are treated as a set and normally they are not in any ordered. If required, objects can be explicitly ordered.

- **Visibility:-** Visibility refers to who can access the role name.

- **Composition:-** Composition is a type of aggregation. Aggregation is merely can conceptual, meaning is does not affect any of the elements.

- **Realization:-** Realization is a relationship between classifiers that allows one classifier to specify a contract that and classifier promises to carry out.

Q-2 Define Interfaces, Types & Roles?

Ans:- Interface:- Interface is a collection of operations that are used to specify of a class or a component.

Interface is a named collection of operations used to specify a service of a class or a component. Interfaces do not include any attributes.

(i) **Names:-** Interfaces must be identified by a unique name. The name given may be either a simple name or path name (prefixed by the name of package).There are class that have nothing but pure virtual functions.

(ii) **Operations:-** An interface is a named collection of operations used to specify a service of a class of a component. These operations have visibility properties, concurrency properties, stereo types, tagged values and constrains.

(iii) **Relationships:**- Interfaces participates primarily in realization relationships. They could however, participate in general association and dependency relationships.

You can show that an element realizes an interface in two ways, which are as –

(i) You can use the simple form in which the interface and its realization relationship are rendered as lollipop sticking –off to one side of a class or component.

(ii) You can use the expanded form in which you rendered an interface as a stereotyped class which allows you to visualize its operations & other properties, than draw a realization from the classifier or

component to the interface.

- **Roles:** - Role, names a behavior of an entity, participating in a particular context.
  A role is the face that an abstraction presents to the world.

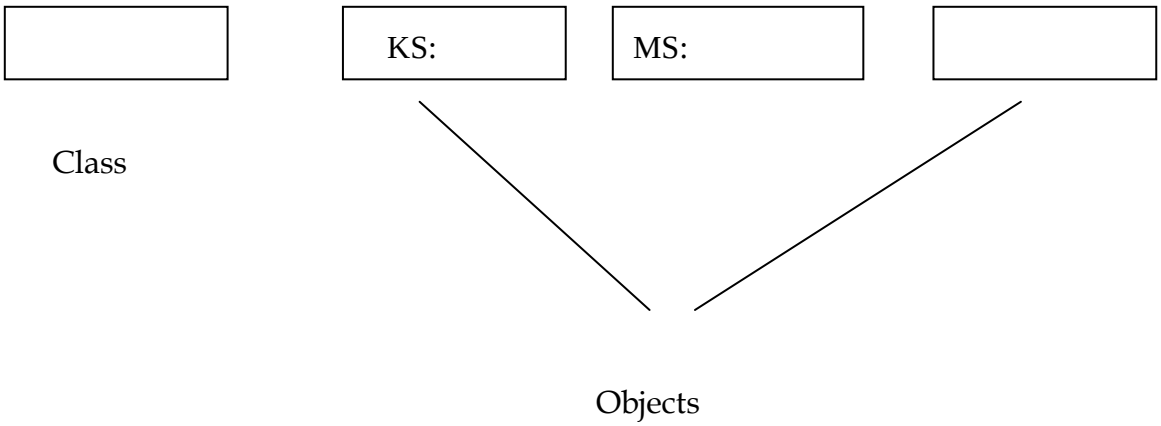
- **Type**: - A type is a stereotype of a class used to specify a domain of object, together with the operation (but not the methods) applicable to the objects.

Q-3 Define instances and object diagram?

Ans: An object is an instance or occurrence of a class. Each class describes a possibly set of individual objects. Each object is said to be an instance of its class. An object has its own value, for each attribute but shares the attributes names & operations with other instances of the class.

Object diagrams: - An object diagram shows individual objects and their relationship, object diagrams are help for documenting test cases.

An object diagram in the unified modeling language is a diagram that shows a complete or partial view of the structure of a modeled system at a specific time.
An object diagram focuses uses on some particular set of object instances and attributes, the link between the instances.



Q-4 What is Basic idea of behavioral modeling?

Ans:- Behavioral models describe the internal dynamic aspects of an information system that supports the business processes in an organization. During the analysis phase, behavioral models describe what the internal logic of the processes is without specifying how the processes are to be implemented.

Creating behavioral models is an iterative process that not only iterates over the individual behavioral models (eg.Interactions sequence & communication) diagrams & behavioral state machines but also over the functional and structural models. As the behavioral models are created, it is not unusual to make changes to the functional and structural models.

Interaction: - In every interesting system, objects don't just sit idle; they interact with one another by passing massages. An interaction is a behavior that comprises a set of massages exchanged among a set of objects within a context to accomplish a purpose. A message is a specification of communication between objects that conveys information, with the expectation that activity will ensure.

In the UML, we model the dynamic aspect of a system of using interactions.

One uses, interactions to model the flow of control within an operation, a class, a component, a use case, or the system as a whole. There can be one reason about these flows in two ways.

1- One can focus on how messages are dispatched across time.
2- One can focus on the structural relationships among the objects in an interaction then consider how messages are passed within the context of that structure.

Q.5 How to Construct Object-Behavior Model ?

Ans. We can construct a Object Behavior model using some steps-

- Evaluate all use-cases to understand the sequence of interaction within the system
- Identify events that drive the interaction sequence and how events relate to specific objects
- Create an event-trace for each use-case
- Build a state transition diagram for the system

Case study

You are in a coffee shop across the street from office having lunch. A customer walks up to the counter. You observe the following:

Customer: Hi Dimple, I'd like a burger to go. Everything but onions.

Dimple(waitress): Anything else?

Customer: Yes, a small order of fries and cold coffee.

Dimple: That'll be 82.35 Rs only.

She collects the cash and places the order through an electronic cash register that automatically displays the order on a TV screen in the back room where

orders are prepared. When the order is ready, Dimple puts it in a bag and hands it to the customer.

Assignment

a.) Explain the pattern of this system in action. Specifically discuss the following:

i.) The organization system's characteristics.

ii.) The subsystem, information flow, and interfaces.

iii.) The types of interdependence in the organization structure and the nature of feedback.

iv.) Input/output and environment

v.) Formal and Informal information system

b.) If you were to improve the performance of the system , what would you do? How? Explain.

----------x----------

Chapter-3

Object Oriented Concepts and Project Management

Q-1 What object-oriented concepts and need of Object Oriented model?

Ans: **Object :-** Object is an identifiable entity which has own characteristics and behavior Object Oriented Programming is a programming language model , organized around "object" rather than "actions" & data rather than logic.

Object oriented approach is a new way of thinking about the process of decomposing problem developing programming solutions. It views a program as a collection of objects. Each object is responsible for specific task.

Need of Object Oriented Model: - When program becomes larger, a single list of instructions becomes unwieldy. So, the program is divided into functions and procedures and each functions or procedure has a clearly defined purpose and a clearly defined interface to other function or procedure in the program.

1- In OOP, the fundamental construct is an object, which combines both structural (data) and behavioral (functions) aspects in a single entity.
2- Data and procedures or functions are loosely coupled in a procedural paradigm where as in an OO paradigm, data functions are tightly coupled to constitute objects.

Q-2 What are elements of OO Model?

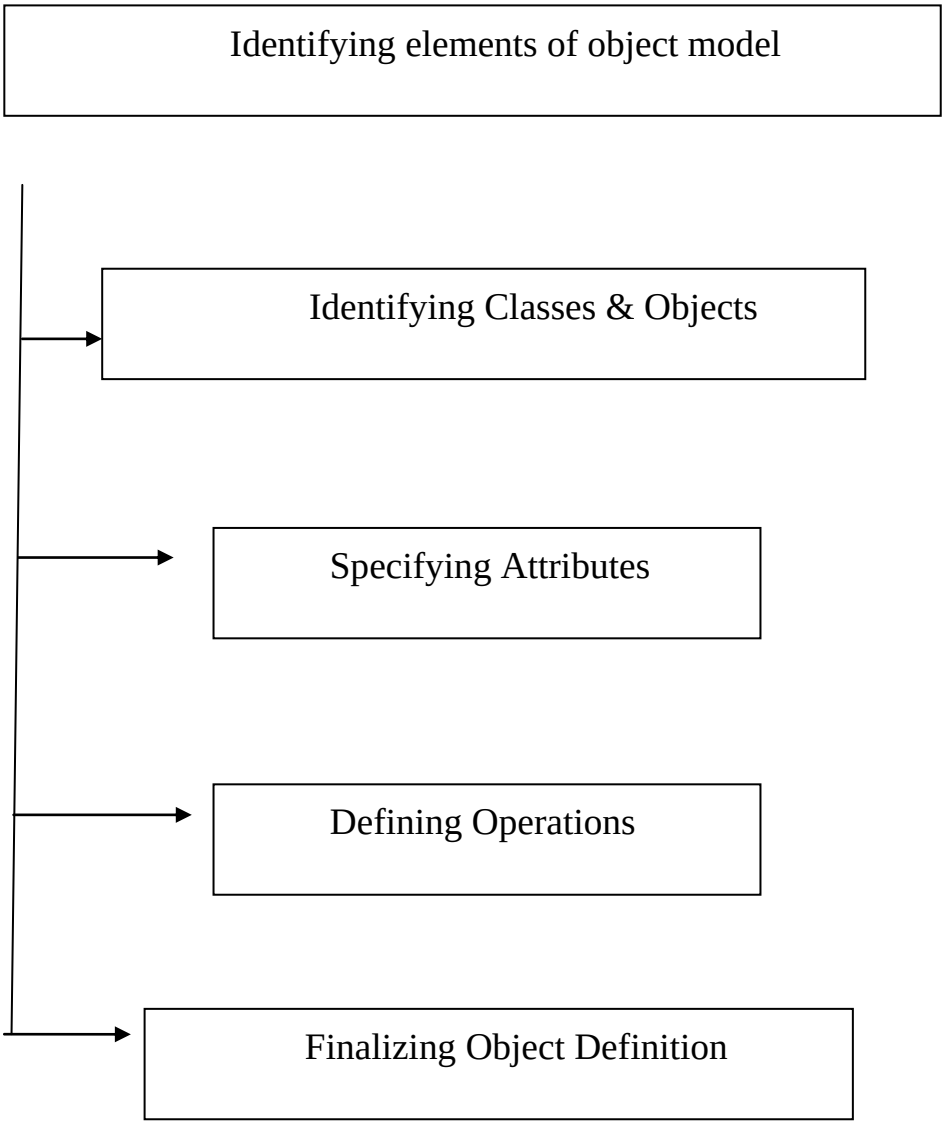
Ans: (i) Object (ii) Class (iii) Instance

Q-3 What are object oriented principles?

Ans: (i) Inheritance (ii) Data abstraction (iii) Encapsulation
(iv) Polymorphisms (v) Message Passing

Q-4 Define identifying elements of object model?

Ans : The main elements of an object model are the class and object , attributes & messages.



(Identifying elements of object model)

1) **Identifying Classes and objects:** - Objects are determined by underlining each noun or noun clause & entering it in a simple table. Synonym should be noted.

Objects can be:-

(1) External Entities
(2) Things
(3) Occurrences or events
(4) Roles
(5) Organization units
(6) Places
(7) Structures.

2- **Specifying Attributes:** - An attribute is defined as data that has a value for all objects in a class that is an attribute describe an object more closely within the chosen context. Thus, attributes are used as representation of states. Which must readable in the model.

To find the attribute, the following pieces of advice can be used:-

1- Investigate how the object can be described generally & specifically in relation to the context for the desired system.
2- Attributes should always have a meaningful value classes have as a rule more than one attribute.

3- Defining Operations:- Operations define the behavior of an object and change the objects attributes in same way.

The operations defined by a class specify how the data may be manipulated. Generally, a complete set of primitive operations maximizes reusability.

Analysis models class operations as the recipients of objects messages (on class, objects, sequence, & collaboration diagrams) state event acceptors and as actions and activities. Details design often adds operations that are used

only internally.

4- Finalizing the object definition:- Additional operations may be determined by considering the "life history" of an object and the messages that are passed among objects defined for the system.
The generic life history of an object can be defined by recognizing that the object must be created, modified, manipulated or read in other ways, and possibly deleted. For the system object, this can be expanded to reflect non activities that occur during its life.

Q-5 Define object oriented project metrics and Estimation?

Ans: Conventional software project estimation techniques require estimates of line-of-code (LOC) or function point (fp) as the primary driver for estimations. Because an overriding goal for Object Oriented projects should be revise, estimates make little sense. Fp estimates can be used effectively because the information domain counts that required are readily obtainable from the problem statement.

Project Metrics:-

Lorenz & kidd suggested the following set of project Metrics:-

1- Number of Scenario scripts: - A scenario script is a detailed sequence of steps that describe the interaction b/w the user and the application.

2- Number of key classes:- Key classes are the highly independent components that are defined early in OOA.

3- Number of support classes:- Support classes are required to implement the system but are not immediately related to the problem domain.

4- Average Number of Support classes per key class: - In general, key class are known early in the project. Support is defined throughout.

5- Number of Subsystems: - A sub systems is an aggregation of class that support of function that is visible to the end user of a system.

Q-6 Explain technical metric for object oriented system?

Ans: OO metrics have been introduced to help a s/w engineers quantitative analysis to access the quality of the design before a system is built.

OO technical metric can provide this guidance. In this, the collect the necessary data and computer metric. Once computed, Metrics are analyzed based on pre-established guidelines and historical data.

- **Goals of Object Oriented Metrics :**

(i) To better understand quality
(ii) To access process effectiveness
(iii) To improve quality of the work performed at the project level.

- **Metrics for Object Oriented Design Model :-** In a detailed treatment of s/w metrics for OOs systems, there is nine distinct and measure characteristics of an OO design.

1. Size: Size is defined in terms of four views
   (i) Population (ii) Volume (iii) length (iv) Functionality.

2. Complexity:- How class interrelated to one another.
3. Coupling:- Physical Connections b/w design elements.
4. Sufficiency: - How well design components reflect all properties of the problem domain.
5. Completeness:- Coverage of all parts of problem domain.
6. Cohesion: - An OO component should be designed in a manner.
7. Primitiveness: - Degree to which a attributes & operations are atomic.
8. Similarity:- Degree to which two or more classes are alike.

Q-7 Define class oriented metrics?

Ans: The metric suite for object oriented design is intended to be a comprehensive approach to evaluating the class in a system.

The CK metrics Suite :-

(i) **Metric-:-Weighted methods per class**: - The weighted methods per class metric are based on the intuition that the number of methods per class is a significant indication of the complexity of the s/w.

(ii) **Matric-2 :- Depth of Inheritance tree**: - The depth of inheritance tree metric is just the maximum length from any node to the root of the inheritance tree for that class.

(iii) **Metric-3:- Number of Children:-** The number of children metric is the number of immediate sub class sub ordinate to a class.

(iv) **Metric -4 :- Coupling between Object (CBO) classes:-** We can define coupling as the use of methods or attributes in another class. The coupling b/w object class metric will be the count of the number of other class to which it is coupled.

(v) **Metric -5:- Response for a class**: - The response let of a class, is the set of methods that can potentially be executed in response to a merge received by an object of that class.

Q-8 What is metrics for object oriented project?

Ans: A key issue that faces a project manager during panning is an estimate of the implemented size of the s/w size is directly proportional to effort & durations.

There are:-

(i) Number of Scenario scripts
(ii) Number of Key class
(iii) Number of support class
(iv) Average number of support class per key class
(v) Number of sub systems

Q.9 Define Measures and Metrixs?

Ans. **Measure** - provides a quantitative indication of the size of some product or process attribute

Measurement - is the act of obtaining a measure

Metric - is a quantitative measure of the degree to which a system, component, or process possesses a given attribute.

Q.10 Explain Project Metrics?

Ans A software team can use software project metrics to adapt project workflow and technical activities.

2.Project metrics are used to avoid development schedule delays, to mitigate potential risks, and to assess product quality on an on-going basis.

3.Every project should measure its inputs , outputs , and results .

Q.11 Define Size-Oriented and Function-Oriented Metrices?

Ans. **1.Size-Oriented Metrics**

• Derived by normalizing any direct measure associated with the product or project by LOC.

• Size-oriented metrics are widely used but their validity and applicability is a matter of some debate.

2.Function-Oriented Metrics

• Function points are computed from direct measures of the information domain of a business software application and assessment of its complexity.

• Once computed function points are used like LOC to normalize measures for software productivity, quality, and other attributes.

• The relationship of LOC and function points depends on the language used to implement the software.

Q.12. What are the software engineering tools in oose?

Ans. There are some tools-

1.Correctness-Correctness is the ability of components to perform their tasks exactly, as designed by the requirements and specifications.

2.Reusability -Reusability allows software components to be used as building blocks for future software developments. Taking components

created by others rather than creating new ones from scratch.

3.Extensibility -Extensibility permits new functionalities to be added easily with little modi_cation to existing software systems. With this property, software systems can be extended easily to meet new requirements.

4.Compatibility- Compatibility is the ease with which software components may be combined and assembled to build useful programs. The Ei_el approach to building software can rely on the mere notion of assembling software components.

5. **Robustness-** Robustness is the ability of software components to function even in abnormal conditions.

6.Testability Testability is the ease of preparing validation suites for software components.

7. **Effciency** Effciency is the good use of resources (e.g., processor or memory), to make good trade-offs among various strategies.

Q.13. How we Establish a Software Metrics Program?

Ans.
1. Identify business goal
2. Identify what you want to know
3. Identify subgoals
4. Identify subgoal entities and attributes
5. Formalize measurement goals
6. Identify quantifiable questions and indicators related to subgoals
7. Identify data elements needed to be collected to construct the indicators
8. Define measures to be used and create operational definitions for them
9. Identify actions needed to implement the measures
10. Prepare a plan to implement the measures

Case study

Allied concrete, Inc., has had to renovate its approach to maintain a computer system and converting application. Recently management has established a course-plotting committee to supervise and endorse all applications before they are run on the mainframe. The committee consists of one member from each of the following areas: accounting, sales, production, and information system. The committee is chaired by the vice president in charge of production. The primary charge is to review each user request and approve or disapprove it based on feasibility and priority. If a request is approved, the user department is billed for its includes computer time, analyst and programmer time, and supplies. All department heads have agreed to the new policy.

In formalizing the committee's authority and responsibilities, serious questions were raised by several user departments about whether the committee has the authority to turn down a project even if it is project out of their budget, there is no reason for it to be rejected.

Assignment:

a.) Should all user projects that are operationally and technically feasible be developed as long as the user is paying the price? If so, what should be the role of the steering committee?

b.) What do you think of the makeup of the steering committee? What role should the analyst, programmer, or data base specialist play in a steering committee? Elaborate.

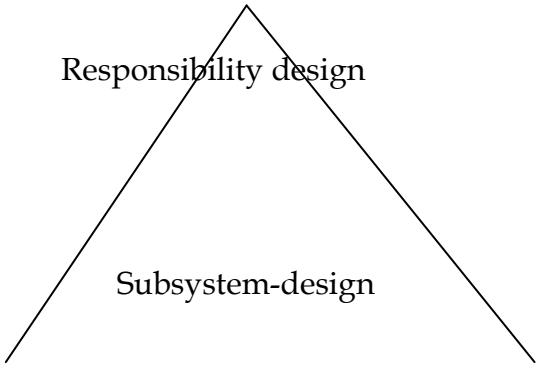
Chapter- 4

Object :- Oriented design and testing

Q-1 Define layers for object oriented system?

Ans: For Object Oriented systems we can also define a design pyramid, but the layers are a bit different.

The four layers of the Object Oriented design pyramid are:-



1. **The subsystem layer** contains a representation of each of the

Subsystem that enable the s/w to achieve its customer define requirements and to implement the technical infrastructure that supports customer requirements .

Message design

Class & object design

2. **The class and object layer**: - Contains the class hierarchies that enable the system to be created using generalizations and increasingly more targeted specializations. This design also contains representation of each object.

3. **The Message layer: -** Contains the design details that enable each object to communicate with its collaborators. This layer establishes the external and internal interfaces for the system.

4. **The Responsibilities layer: -** Contains the data structure and algorithmic design for all attributes and operations for each object.

Q-2 Explain the steps of system design process?

Ans: There are 7 Steps for system design process are:-

(i) **Partitioning the Analysis Model**: - Use partition the analysis model to define collection of classes, relationships, and behavior. In general, all the elements of a subsystem share some property in common.

When a system is partitioned into subsystems, another design activity, called layering.

There are four layer architecture:-

(i) A presentation layer
(ii) An application layer
(iii) A data formatting layer
(iv) A database layer.

(ii) **Concurrency & Subsystem Allocation**:- The dynamic aspect of the object behavior model provides an indication of concurrency among classes. If classes are not active at the same time, there is no need for concurrent processing.

This means that the classes can be implemented at the same processor

hardware.

When subsystems are concurrent, two allocating options are there-

(i) Allocate each subsystem to an independent processor.

(ii) Allocate the subsystems to the same processor and provide concurrency through operating system features.

(iii) The task management component:-

(i) The characteristics of the task are determined.

(ii) A coordinator task and associated objects are defined.

(iii) The coordinator and other tasks are integrated.

(iv) The User Interface Component: - Although the user interface component is implemented within context of the problem domain.

The OO analysis model contain usage scenarios (called use-cases) & a description of the roles that use play(called actors) as they interact with the system.

(v) The Data Management Component:-

(i) Management of data that is critical to the application itself.

(ii) Creation of an infrastructure for storage and retrieval of objects

(vi) The Resource Management Component: - A variety of d/f resources are available to an OO system or product, and in many instances, subsystems compete for there resources at the same time.

(vii) Inter subsystem Communication: - Once each substances has been specified, it increasing to define the colorations that exit b/w the subsystems.

Q-3 What is the object oriented texting?

Ans:- Testing is the process of examining something with the intention of finding errors. Testing was reveal symptom of error, but it may not uncover the precise cause of that errors. Once source code has been generated, s/w must be tented to uncover as many errors a possible before delivery to informer.

Features of S/W testing:-

i) To perform effective testing, a s/w team should conduct effective format technical reviews

ii) Testing begins at the component level and works "outward" toward the integration of the entire computer based system.

iii) Different testing techniques are appropriate at different points in time.

iv) Testing is conducted by the developer of the s/w and (for large projects) an independent text groups.

Factors effecting OO texting

(i) Information hiding

(ii) Encapsulation

(iii) Inheritance.

Q-4 What are two factors for OOA and OOD models testing?

Ans: There are 2 Factors:-

(i) Accuracy of the OOA and OOD models

(ii) Consistency of the OOA and OOD models.

(i) **Accuracy of the OOA & OOD Models**:- There are many methodology for creating the analysis and design models. The symbols, language, and notation of these models will be linked to their aspects. Now it is important to ensure that the correct symbols and notations are used and that correct modeling techniques and conventions are followed. The class definitions and inter-relationships should be examined to iron out any problems.

(ii) **Consistency of OOA & OOD models**:- It is important to verify the consistency of these models. That is, the same class and its relationships with other classes should be represented exactly in the same manner everywhere in the models. Models such as the class responsibility-collaboration should be used for this purpose.

Q-5 Define the strategies of object-oriented testing?

Ans: Strategies –

(i) **Unit testing in OO context**:- When object oriented software is considered, concept of the unit testing changes. Encapsulation drives the definition of classes and objects. This means that each class and each instance of a class packages attributes (data) & the operations that manipulate these data. Rather than testing on individual module, the smallest testable unit is the encapsulated class or object.

(ii) **Integration Testing in OO Context**:- There are two strategies for integration testing .

(i) Thread based testing, integrates the set of class required to respond to one input or event for the system.

(ii) Use –based testing: - Begins the construction of the system by testing the classes (called independent class) that use very few of server class.

(iii) Validation testing in an OO context:- At the validation or system level, the details of class connections disappear. The validation of OO s/w focuses on user –visible actions and inner –reorganization output from the system.

To assist in the derivation of validation test, the tester should draw upon the use-cases that are part of analysis model.

Q-6 What is inter class testing? Define types?

Ans: Inter-class testing means how class work each other to perform a given set of functions, users can use such as random and partitioning testing.

Types of Inter-Class Testing:-

(1) **Random Testing** :-

(i) Use list of operations to generate random test sequences.

(ii) For each message that is generated – determine collaborator classes and corresponding operations server class.

(2) **Partition Testing**:-

(i) State based partitioning

(ii) Attribute based partitioning

(iii) Category –based partitioning

(3) **Scenario Based Testing**:- Scenario Based testing concentrates an what the user does.

(4) **Behavioral Testing**:- State diagrams help in deriving sequence of tests that exercise dynamic behavior of a class and classes.

(5) **Multiple Class Testing**: -]

(i) For each client class, use the list of class operations to generate a services of random test sequences.

(ii) For each message that is generated, determine the collaborator class and the corresponding operations in the server object.

(iii) For each operation in the server object, determine the messages that it transmits.

(iv) For each of the messages, determine the next level of operations that are invoked and incorporate there into the test sequences.

Q.7 What are Object-oriented Design Issues?

Ans.

- Decomposability - facility of design method that allows the designer or user to decompose the problem into solved subproblems
- Composability - degree to which design method ensures that modules constructed for one project can be reused in another
- Understandablity - ease with which a component can be understood without examining other components
- Continuity - ability to isolate changes made in one module and restrict the propagation of changes to other modules
- Protection - architectural characteristic that reduces the propagation of side effects when errors occur

Q.8 How we Design Test Case for OO Software?

Ans.

- Each test case should be uniquely identified and be explicitly associated with a class to be tested
- State the purpose of each test
- List the testing steps for each test including:
  1. list of states to test for each object involved in the test

2. list of messages and operations to exercised as a consequence of the test
3. list of exceptions that may occur as the object is tested
4. list of external conditions needed to be changed for the test
5. supplementary information required to understand or implement the test

Q.9 What are the Goals for Using Object-Oriented Metrics

Ans.
- To better understand product quality
- To assess process effectiveness
- To improve quality of the work performed at the project level

Q.10 How many types are Object Oriented Design?

Ans. There are two types-

Architectural Design

Architectural design divide the system into different sub systems known as packages. Then the dependency, relationship and communication between the packages are also identified.Package diagram is use to represent architectural design using UML.

Detailed Design

It describes the detailed description of the classes, that is all the attributes (Variables and functions). The detailed class diagram represents the detailed design using UML.

Case study

The vice president of a large retail store wants to modify is order entry system. He states the problem as follows: "I need a report that gives me information about the previous or old records."

Assignment:

a.) Outline the procedure which you follow. [hint: follow the steps of System Design Process]

b.) What questions would you ask?

Chapter-5

Advanced Topic in S/W engineering

Q-1 Define component based s/w engineering?

Ans Define component based s/w engineering begins with the identification of the system requirements and the architecture that define the need for components. Every system needs in interface to provide a service, and an interface to access this service.

Components that have been identified for reuse must next be examined for implementation in the system under development. Very often, there is a mismatch between component architecture and system architecture and , in such cases, changes have to be made in the component before use.

Essential of Component based s/w Engineering-

CBSE is the process of defining, implementing, and integrating or composing oosely coupled independent components into systems. It has become as an important s/w development approach because s/w systems are becoming larger and more complex and customers are demanding more dependable s/w that is developed more quickly.

(i) **Independent components that are completely Specified by their interfaces: -** There should be a clear separation between the component interface and its implementation so that one implementation of a component can be replaced by another without changing the system.

(ii) **Component standards that facilitate the integration of components**: - There

standards are embodied in a component model and define, at the very minimum, how component interfaces should be specified and how components communicate. Components written in d/f language can be integrated into the same systems.

(iii) **Middleware** :- It provides s/w support for component integration to make independent, distributed components work together , you need middleware support that handles component communications.

(iv) **Development process** :- It is geared to component based s/w engineering. If you try to add a component based approach to a development process that is geared to original s/w production, you will find that the assumptions inherent in the process limit the potential of CBSE.

Q-2 What is goals of CBSE?

Ans: (i) CBSE uses s/w engineering principals to apply the same idea as OOP's to the whole process of designing and constructing s/w systems.

(ii) It focus on reusing and adopting existing components, as opposed to just coding in a particular style.

(iii) The software systems built using CBSE are not only simpler to bold and cheaper but usually turn out to be more robust, adaptable.

Q-3 Define methodologies of component based s/w development?

Ans: The seven methodologies that have been studied for extracting process patterns are –UML Components, select perspective

(i) **UML Components**:- UML components is UML based methodology aiming to help developer use technologies such as COM+ and JAVA beans for defining and specifying components.

(ii) **Selective Perspective** :- Select perspective was the result of combining the object modeling language introduce in with objects we case driven process(later integrated in to RUP)
It has three types of services, which are as follows:-

(i) user Services (ii) Business Services

(iii) Data Services.

(iii) **FORM: -** Feature –oriented domain analysis (FODA) was introduced in 1990, and presented the Idea of using features in requirements engineering.

(iv) **Kobra: -** It is based an a number of advance s/w engineering technologies , including product-line engineering, frameworks, architecture –centric development , quality modeling and process modeling.

(v) **Catalysis: -** This methodology is a component based approach on object oriented analysis and design. Catalysis provides a framework for constructing component based s/w.

(vi) **Adaptive s/w development**:- This was introduced in 1997, and is the only agile method that specifically targets component bared development.

(vii) **Rational Unified Process: -** The Rational unified process (RUP) is the most well known of the selected methodologies. RUP is use-case-driven and architecture centric and in corporate specific guidelines for component based development.

Q-4 How a component can be classified and retrieving?

Ans (i) Describing Reusable components

(ii) Classification for s/w components.

(i) **Describing reusable components**: - The basic ideal description has 3 model.

a. Concept b. Content c. context

The content of a component describes how the concept is realized.

The context places a reusable s/w component within its domain of applicability.

(2) Classification for s/w component:-

(i) **Encapsulation classification:** - Components are described by a hierarchical structure in which classes and very fine levels of subclasses of s/w components are defined.

(ii) **Faceted classification:-**A domain area is analyzed and a set of basic

descriptive features are identified. A facet can describe the function that the component performs, the data that are manipulated, or any other feature.

(iii) **Attribute value classification:** - A set of attributes is defined for all components in a domain area, values are then arranged to these attribute in much the same way as faceted classification.

Classification

(i) No limit is placed on the number of attributes that can be used.
(ii) Attributes are not arranged priorities.
(iii) A thesaurus function is not used.

Q.5 What are the objective of CBSE?

Ans. There are some objective-

1.The main objective is developing standardized components and composing these into applications
2. To describe components and component models
3. To show the principal activities in the CBSE process
4. To discuss approaches to component composition and problems that may arise

Q.6 What is the analysis of Component Requirement?

Ans Component requirement analysis is the process of discovering, understanding, documenting, validating and managing the requirements for a component. The main objectives of component requirement analysis are to produce complete, consistent and relevant requirements that a component should realize, as well as the programming language, the platform and the interfaces related to the component.

Q.7 What is Formal Method?

Ans. Formal Method is a Techniques and tools based on mathematics and formal logic.This Can assume various forms and levels of rigor.
There are some Formal Specification Methods like as-
1.Formal Specification
2.Formal proof

3.Model Checking
4.Abstraction

Q.8 What are the benefits of Formal Specifications?

Ans. 1 Defects are uncovered that would likely go unnoticed with traditional specification methods
2.Identify defects earlier in life cycle
3.Formal specification language semantics allow checks for self-consistency of a problem specification
4.Formal specifications enable formal proofs which can establish fundamental system properties and invariants
5. Abstract formal view helps separate specification from design

Q.9. What are component Composition and also define types ?

Ans. The process of assembling components to create a system. Composition involves integrating components with each other and with the component infrastructure.

Types of Component Composition-

1.Sequential composition where the composed components are executed in sequence. This involves composing the provides interfaces of each component.

2.Hierarchical composition where one component calls on the services of another. The provides interface of one component is composed with the requires interface of another.

3.Additive composition where the interfaces of two components are put together to create a new component.

Case Study

A large wholesale liquor distributer has been having difficulty keeping inventory up to date because incoming shipments are not processed quickly enough. The sales force can never be certain which brands and quantities are available for sale. The vice president of sales asks an outside analyst to investigate the problem.

The analyst arrived at the computer center Monday at 8:00 A.M. He asked to see the manager. The receptionist told him that Mr. Sibley came around 9.00 A.M. Not wanting to waste an hour, the analyst decided to interview the programmers to learn about inventory control.

Mr. Sibley arrived at 8.45. He was furious to find that the analyst has taken the liberty to his staff without his consent. He promptly told the analyst that Monday is a bad day. The programmers could no be interrupted before 3.30 P.M on Tuesday. The analyst agreed to come then. Meanwhile, he went to stockroom to observe stock keeping activities.

The next day, the analyst interviewed more employees in the stockroom and the clerical staff of the warehouse. It was 4.30 P.M then he remember his appointment with manager of the computer center. He abruptly terminated his work

Multiple Choice Questions

1. **Which of the following should be represented on an information flow diagram?**

a) Database
b) Source
c) Process
d) Entity

2. A web blueprint depicts:

a) the layout of an index page
b) the layout of a sitemap
c) the layout of an individual web page
d) the layout of a website

3. **UML depicts information systems as a collection of:**

a) data
b) entities
c) objects
d) processes

4. **Use-case analysis focuses upon:**

a) objects
b) information
c) entities
d) actors

5. **Which of the following is part of a static view of information?**

   a) Data flow model
   b) Logical data model
   c) Information process model
   d) Meta data

6. **Divide a complex system into small, self-contained pieces that can be managed independently. How is it called?**

   a) Abstraction
   b) Modularity
   c) Encapsulation
   d) Hierarchy

7. In order to model the relationship "a course is composed of 5 to 20 students and one or more instructors", you could use:

a) **Aggregation**
b)Association
c) Composition
d) Realization

8 . Which of the following statements are true?

a . All operations defined in a sub-class are inherited by the super-class
b. Generalization allows abstracting common features and defining them in a super-class
c. A super-class is a class that must not have associations
d . Association is a "part-of" relationship

9 . What is the relationship between these two use cases?

a .Generalization

b .Extend
c .Include
d. Association

10. Which of these diagrams shows interactions between objects?
a .Activity diagram
b .Class diagram
c. Sequence diagram
d .Component diagram

11. A statechart diagram describes:
a .Attributes of objects
b .Nodes of the system
c .Operations executed on a thread
d .Events triggered by an object

12. An interface is:
a. A set of objects used to provide a specific behaviour
b .A set of classes used on a collaboration
c .A set of attributes used on an operation
d .A set of operations used to specify a service of a class or component

13. The sequence diagram models:
a .The order in which the class diagram is constructed
b. The way in which objects communicate
c .The relationship between states
d .The components of the system

14. The activity diagram:
a. Focuses on flows driven by internal processing
b. Models the external events stimulating one object
c .Focuses on the transitions between states of a particular object
d .Models the interaction between objects

15. The deployment diagram shows:

a .Objects of a system

b. Distribution of components on the nodes in a system

c .Functions of a system

d. Distribution of nodes

16. Unified Process is a software development methodology which is:

a .Use-case driven

b .Component-driven

c .Related to Extreme Programming

d. None in only one iteration

17. Ordering abstractions into a tree-like structure. How is it called?

a .Abstraction

b. Modularity

c. Encapsulation

d .Hierarchy

18. In order to model the relationship "a hotel has rooms", between Hotel and Room,you could use:

a .Aggregation

b .Association

c .Composition

d .Realization

19. Structured programming codes includes ?

a. sequencing

b. alteration

c. iteration

d. only A, B and C

20. An important aspect of coding is ?
- **a. Readability**
- b. Productivity
- c. To use as small memory space as possible
- d. brevity

21. Data structure suitable for the application is discussed in ?
- **a. data design**
- b. architectural design
- c. procedural design
- d. interface design

22. In object oriented design of software , objects have ?
- a. attributes and names only
- b. operations and names only
- **c. attributes, name and operations**
- d. None of above

23. Function oriented metrics were first proposed by ?
- a. John
- b. Gaffney
- **c. Albrecht**
- d. Basili

24. A good specification should be ?
- a. Unambiguous
- b. Distinctly Specific
- c. Functional
- **d. All of Above**

25. Which of the following is a tool in design phase ?
- a. Abstraction
- b. Refinement
- c. Information Hiding
- **d. All of Above**

26. Information hiding is to hide from user, details ?

a. that are relevant to him
b. that are not relevant to him
c. that may be maliciously handled by him
d. that are confidential

27. Which of the following comments about object oriented design of software, is not true ?

a. Objects inherit the properties of class
b. Classes are defined based on the attributes of objects
c. an object can belong to two classes
d. classes are always different

28. Design phase includes?

a. data, architectural and procedural design only
b. architectural, procedural and interface design only
c. data, architectural and interface design only
d. data, architectural, interface and procedural design

29. What is true about UML stereotypes?

a.A stereotype is used for extending the UML language.
b.A stereotyped class must be abstract.
c. The stereotype {frozen} indicates that the UML element cannot be changed.
d. UML Profiles can be stereotyped for backward compatibility.

30. If you need to show the physical relationship between software components and the hardware in the delivered system, which diagram can you use?

a. component diagram
b. deployment diagram
c. class diagram
d. network diagram

31. Which diagram is NOT commonly used for illustrating use cases?

a.system sequence diagram
b. activity diagram
c.use case diagram

d.collaboration diagram

32. Process indicators enable a software project manager to
a.assess the status of an on-going project
b.track potential risks
c.adjust work flow or tasks
d.all of the above

33. Which of the following items are not measured by software project metrics?
a.Inputs
b.Markets
c.Outputs
d.results

34. Software quality and functionality must be measured indirectly.
a.TRUE
b.FALSE

35. Which of following are advantages of using LOC (lines of code) as a size-oriented metric?
a.LOC is easily computed.
b.LOC is a language dependent measure.
c.LOC is a language independent measure.
d.LOC can be computed before a design is completed.

36. There is no need to reconcile LOC and FP measures since each is meaningful in its own right as a project measure.
a.TRUE
b.FALSE

37. Which of the following provide useful measures of software quality?
a.correctness, business relevance, integrity, usability
b.reliability, maintainability, integrity, sales
c.correctness, maintainability, size, satisfaction
d.**correctness, maintainability, integrity, usability**

38. The software metrics chosen by an organization are driven by the business or technical goals an organization wishes to accomplish.
a.TRUE
b.FALSE

39. Which of the following is not one of the CBSE activities that take place for requirements that can be addressed with commercial off-the-shelf (COTS) components?
a.component adaptation
b.component composition
c.component design
d.component qualification

40. What are the two parallel engineering activities found the CBSE process model?
a. component-based development and library development
b. domain engineering and component-based development
c. domain engineering and process development
d. none of the above

41. Which of the following is not one of the major activities of domain engineering?
a.Analysis
b.Construction
c.dissemination
d.validation

42. Domain analysis is only applicable to CBSE or object-oriented software engineering.
a.True
b.False

43. Which of the following is an example of a structure point for some software domain?
a. bounds setting mechanism
b. control mechanism
c. user interface
d. all of the above

44. Which of the following should be part of an infrastructure for effective component integration?
a.automation
b.data exchange model
c.underlying object model
d.all of the above

45. Which of the following is <u>not</u> one of the classification schemes used for software components?
a.attribute-value classification
b.domain classification
c.enumerated classification
d.faceted classification

46. In a reuse environment, library queries are often characterized using the _____ element of the 3C Model.
a.concept
b.content
c.context
d.all of the above

47. Which of the following is not improved by the effective use of CBSE?
a.cost
b. performance
c. productivity
d.quality

48. The effort required to qualify, adapt, and integrate structure points into new systems must be based on historical data collected for qualification,

adaptation, and integration of these reusable components in other applications.
a.True
b.False

49. It is impossible to develop effective metrics for software reuse.
a.True
b.False

50. Effective software project management focuses on four P's which are
a. people, performance, payoff, product
b. people, product, performance, process
c. people, product, process, project
d. people, process, payoff, product

51. The first step in project planning is to
a.determine the budget.
b.select a team organizational model.
c.determine the project constraints.
d.establish the objectives and scope.

52. Process framework activities are populated with
a.Milestones
b.work products
c.QA points
d.All of the above

53. The best project team organizational model to use when tackling extremely complex problems is the
a.closed paradigm
b.open paradigm
c.random paradigm
d.synchronous paradigm

54. One of the best ways to avoid frustration during the software development process is to
a. give team members more control over process and technical decisions.

b. give team members less control over process and technical decisions.
c. hide bad news from the project team members until things improve.
d. reward programmers based on their productivity.

55. Which of these software characteristics is <u>not</u> a factor contributing to project coordination difficulties?
a.i nteroperability
b. performance
c. scale
d. uncertainty

56. Which of these software characteristics are used to determine the scope of a software project?
a. context, lines of code, function
b. context, function, communication requirements
c. information objectives, function, performance
d. communications requirements, performance, information objectives

57. Product and process decomposition often occurs simultaneously as the project plan evolves.
a.True
b.False

58. When can selected common process framework activities be omitted during process decomposition?
a. when the project is extremely small in size
b. any time the software is mission critical
c. rapid prototyping does not require their use
d. never—the activities should always occur

59. How does a software project manager need to act to minimize the risk of software failure?
a. double the project team size
b. start on the right foot
c. track progress
d.both b and c

60. Which of these are critical practices for performance-based project management?

a. defect tracking against quality targets
b. empirical cost estimation
c. formal risk management
d.all of the above

KeyTerms

Abstraction

Abstraction in programming is the process of identifying common patterns that have systematic variations, an abstraction represents the common pattern and provides a means for specifying which variation to use.

Asset

A collection of artefacts.

Aspect migration

The process of *migrating* a software system that is written in a non aspect-oriented way into an aspect-oriented equivalent of that system.

Audit trails

An audit trail establishes additional information about when, why, and by whom changes are made.

Benchmark

(1) A standard against which measurements or comparisons can be made. (2) A recovery file.

Business model

A model of real-world objects and their interactions -or rather, some users' understanding of them

Business rule

A step or set of steps in a process or procedure or guide (algorithmic or heuristic) used by a customer for doing its business, work, or function, and often embodied in whole or in part in the software of a system .

Capability Maturity Model (CMM)

Defined by the Software Engineering Institute (SEI) at Carnegie Mellon University. Describes the level of capability and maturity a software team could aim for and could be assessed against.

Case study

A case study is a research technique where you identify key factors that may affect the outcome of an activity and then document the activity: its inputs, constraints, resources, and outputs.

Cliché

(French) A pattern describing salient features of a concept that supports recognition of that concept in some specified context by application of some specified comparison algorithm.

Clone

Clones are segments of code that are similar according to some definition of similarity.

Commonalities

The set of feature or properties of a component (or system) that are the same, or common, between systems

Decay

Decay is the antithesis of evolution. While the evolution process involves progressive changes, the changes are degenerative in the case of decay.

Domain

A problem area. Typically, many application programs exist to solve the problems in a single domain.

Evolvability

The capability of software products to be evolved to continue to serve its customer in a cost effective way.

Fragile base class problem

Refers to the problem that occurs when independently developed subclasses are broken when their base class evolves.

Framework

A framework is a reusable design of all or part of a software system described by a set of abstract classes and the way instances of those classes collaborate.

Heuristic

Involving or serving as an aid to learning, discovery or problem solving by experimental and especially trial-and-error methods.

Hypothesis

A tentative explanation that accounts for a set of facts and can be tested by further investigation; a theory.

Inconsistency

A state in which two or more overlapping elements of different software models make assertions about aspects of the system they describe which are not jointly satisfiable.

Integrity

The ability of software systems to protect their various components (programs, data) against unauthorized access and modification.

Intercession

Intercession is the ability of a program to modify its own execution state or to alter its own interpretation or meaning.

Introspection

Introspection is the ability of a program to observe and therefore reason about its own state.

Metric

A quantitative measure of the degree to which a system, component or process possesses a given attribute.

Mixin

A mixin is a subclass definition that may be applied to different superclasses to create a related family of modified classes.

Paradigm

A point of view in which some principles, approaches, concepts, and even theories, have been stated uniformly.

Pattern

A standard (object-oriented) design for addressing frequently occuring problems, described in a standard way.

Piecemeal growth

The process of design and implementation in which software is embellished, modified, reduced, enlarged, and improved through a process of repair rather than replacement.

Product line

A collection of existing and potential products that address a coherent business area and share a set of similar characteristics.

Refinement

A refinement is a detailed description that conforms to another (its *abstraction*). Everything said about the abstraction holds, perhaps in a somewhat different form, in the refinement.

Reflection

Reflection is the ability of a program to manipulate as data, something representing the state of the program during its own execution.

Reliability

Software reliability is the probability of a failure-free operation of a computer program in a specified environment for a specified time.

Repairability

Repairability is the ability to facilitate the repair of defects.

Replication

The collection of two or more observations under a set of identical experimental conditions.

Research hypothesis

A tentative theory or supposition provisionally adopted to account for certain facts and to guide in the investigation of others.

Restructuring

A restructuring transformation is often one of appearance, such as altering code

to improve its structure in the traditional sense of structured design.

Ripple effect

The phenomenon where a change in one piece of a software system affects at least one other area of the same software system (either directly or indirectly)

Robustness

The ability of software systems to react appropriately to abnormal condition.

Separation of concerns (SOC)

Separation of concerns is closely related to the well-known Roman principle of "divide and conquer".

Software entropy

The amount of disorder in a software system

Software quality assurance

A planned and systematic pattern of all actions necessary to provide adequate confidence that the item or product conforms to established technical requirements.

Traceability

The degree to which a relationship can be established between two or more products of the development process, especially products having a predecessor-successor or master-subordinate relationship to one another.

Treshold

A fixed value (typically an upper bound or lower bound) that distinguishes normal values from abnormal metric values.

Software variability

Refers to the ability of a software sysem or artefact to be efficiently extended, changed, customised or configured for use in a particular context.

Viewpoint

A viewpoint is a specification of the conventions for constructing and using a view.

Wrapper

A software component that encapsulates a system component (a procedure, a program, a file, an API) in order to transform its interface with its environment.

Bibliography

Bibliography

1. David E. Monarchi and Gretchen I. Puhr. A research typology for object-oriented analysis and design.

2. Object-Oriented Software Engineering Using UML, Patterns, and Java (3rd Edition)
Bernd Bruegge (Author), Allen H. Dutoit (Author)

3. Title -Advances in object-oriented software engineering
Prentice Hall object-oriented series
Author -Dino Mandrioli
Editors -Dino Mandrioli, Bertrand Meyer

4.Title- Fundamentals of software engineering
Authors -Carlo Ghezzi, Mehdi Jazayeri, Dino Mandrioli
Edition- illustrated