TABLE OF CONTENTS

Unit III

Unit IV

Unit V

Unit VI

Unit III

3 Design Engineering

Part I : Design Engineering

3.1 : Design Process and Quality

Q.1 What is software design ?

Ans. : Definition : Software design is model of software which translates the requirements into finished software product in which the details about the software data structures, architecture, interfaces, and components that are necessary to implement the system are given.

Software design is an **iterative process** using which the user requirements can be translated into the software product.

Q.2 Explain the quality attributes, considered in software design.

☞ **[SPPU : Dec.-12, Marks 8, Dec.-16, Marks 5, Dec.-22, Marks 9]**

Ans. : The design quality attributes popularly known as **FURPS** (**F**unctionality, **U**sability, **R**eliability, **P**erformance and **S**upportability) is a set of criteria developed by Hewlett and Packard.

Following table represents meaning of each quality attribute

Quality Attribute	Meaning
Functionality	Functionality can be checked by assessing the set of **features and capabilities** of the functions. The functions should be **general** and should not work only for particular set of inputs. Similarly the **security** aspect should be considered while designing the function.

Usability	The usability can be assessed by knowing the **usefulness** of the system.
Reliability	Reliability is a measure of frequency and severity of **failure**. Repeatability refers to the consistency and repeatability of the measures. The **Mean Time to Failure (MTTF)** is a metric that is widely used to measure the product's performance and reliability.
Performance	It is a measure that represents the response of the system. Measuring the performance means measuring the processing speed, memory usage, response time and efficiency.
Supportability	It is also called maintainability. It is the ability to adopt the enhancement or changes made in the software. It also means the ability to withstand in a given environment.

Q.3 What are the design quality guidelines ?

☞ **[SPPU : May-14, Marks 8; Dec.-22, Marks 9]**

Ans. : The **quality guidelines** are as follows -

1. The design architecture should be created using following issues -
   - The design should be created using **architectural styles and patterns.**
   - Each component of design should posses **good design characteristics.**
   - The implementation of design should be **evolutionary,** so that testing can be performed at each phase of implementation.
2. In the design the data, architecture, interfaces and components should be **clearly represented.**

3. The design should be **modular.** That means the subsystems in the design should be logically partitioned.
4. The **data structure** should be appropriately chosen for the design of specific problem.
5. The **components** should be used in the design so that functional independency can be achieved in the design.
6. Using the information obtained in software **requirement analysis** the design should be created.
7. The **interfaces** in the design should be such that the complexity between the connected components of the system gets reduced. Similarly interface of the system with external interface should be simplified one.
8. Every design of the software system should **convey its meaning** appropriately and effectively.

Q.4 What are the software design quality attributes and quality guidelines ? ☞ **[SPPU : Dec.-19, Marks 7]**

Ans. : Quality Attributes : Refer Q.2.

Quality Guidelines : Refer Q.3.

3.2 : Design Concepts

Q.5 Explain about various design concepts considered during design. ☞ **[SPPU : April-15, Marks 6]**

Ans. : The software design concept provides a framework for implementing the right software.

Following are certain issues that are considered while designing the software -

- Abstraction
- Modularity
- Architecture
- Refinement
- Pattern
- Information hiding
- Functional independence
- Refactoring
- Design classes

Refer Q.6, Q.7, Q.8.

Q.6 Explain the following design concepts.
i) Refinement ii) Modularity iii) Architecture
☞ [SPPU : May-11, Marks 6, May-13, Marks 8; Dec.-22, Marks 9]

Ans. : i) Refinement :

- Refinement is actually a process of elaboration.
- Stepwise refinement is a top-down design strategy proposed by Niklaus WIRTH.
- The architecture of a program is developed by successively refining levels of procedural detail.
- The process of program refinement is analogous to the process of refinement and partitioning that is used during requirements analysis.
- Abstraction and refinement are complementary concepts. The major difference is that - In the abstraction low-level details are suppressed. Refinement helps the designer to elaborate low-level details.

ii) Modularity :

- The software is divided into separately named and addressable components that called as **modules.**
- Monolithic software is hard to grasp for the software engineer, hence it has now become a trend to divide the software into number of products. But there is a co-relation between the number of modules and overall cost of the software product.
- The overall complexity of two problems when they are combined is greater than the sum of complexity of the problems when considered individually. This leads to **divide and conquer strategy** (according to **divide and conquer strategy** the problem is divided into smaller subproblems and then the solution to these subproblems is obtained).

- Thus dividing the software problem into manageable number of pieces leads to the concept of modularity.
- It is possible to conclude that if we subdivide the software indefinitely then effort required to develop each software component will become very small. But this conclusion is invalid because the total number of modules get increased the efforts required for developing each module also gets increased.
- That means the cost associated with each effort gets increased.
- The effort (cost) required for integration these modules will also get increased.
- The total cost required to develop such software product is shown by Fig. Q.6.1.
- The Fig. Q.6.1 provides useful guideline for the modularity and that is - **overmodularity or the undermodularity while developing the software product must be avoided.** We should modularize the software but the modularity must remain near the region denoted by **M.**

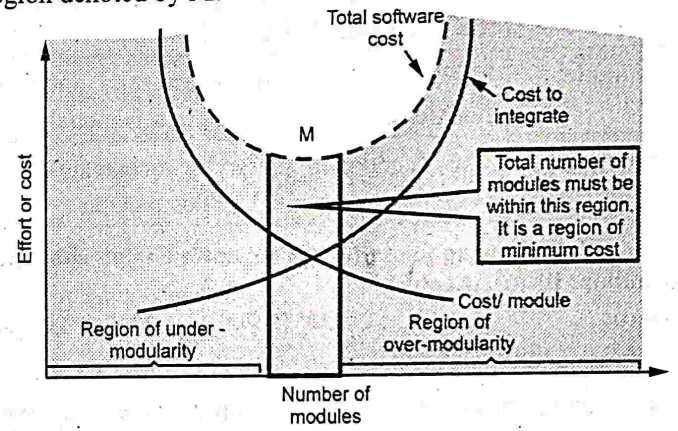


Fig. Q.6.1 Modularity and software cost

- Modularization should be such that the development can be planned easily, software increments can be defined and delivered,

changes can be more easily accommodated and long term maintenance can be carried out effectively.

iii) Architecture :

Architecture means representation of **overall structure** of an integrated system. In architecture various components interact and the data of the structure is used by various components. These components are called system elements. Architecture provides the basic framework for the software system so that important framework activities can be conducted in systematic manner.

- In architectural design **various system models** can be used and these are

Model	Functioning
Structural model	Overall architecture of the system can be represented using this model.
Framework model	This model shows the architectural framework and corresponding applicability.
Dynamic model	This model shows the reflection of changes on the system due to external events.
Process model	The sequence of processes and their functioning is represented in this model.
Functional model	The functional hierarchy occurring in the system is represented by this model.

Q.7 Explain the following fundamental software design concepts :
i) Abstraction ii) Architecture
iii) Patterns ☞ **[SPPU : June-22, Dec.-22, Marks 9]**

Ans. : i) Abstraction :

- The abstraction means an ability to cope up with the complexity.
- Software design occurs at different levels of abstraction. At each stage of software design process levels of abstractions should be applied to refine the software solution.

- At the **higher level** of abstraction, the solution should be stated in **broad terms** and in the **lower level** more **detailed description** of the solution is given.
- While moving through different levels of abstraction the **procedural abstraction** and **data abstraction** are created.
- The **procedural abstraction** gives the named sequence of instructions in the specific function. That means the functionality of procedure is hidden. For example : **Search the record** is a procedural abstraction in which implementation details are hidden (i.e. Enter the name, compare each name of the record against the entered one, if a match is found then declare success !! Otherwise declare 'name not found')
- In **data abstraction** the collection of data objects is represented. For example for the procedure **search** the data abstraction will be **record**. The record consists of various attributes such as record ID, name, address and designation.

ii) Architecture : Refer Q.6 (iii)

iii) Pattern :

- According to **Brad Appleton** the design pattern can be defined as - It is a named nugget (something valuable) of insight which conveys the essence of proven solution to a recurring problem within a certain context.
- In other words, design pattern acts as a design solution for a particular problem occurring in specific domain. Using design pattern designer can determine whether-
- Pattern can be **reusable**.
- Pattern can be used for **current work**.
- Pattern can be used to **solve similar kind of problem** with different functionality.

Q.8 What do you understand by refactoring ? Give the importance of refactoring in improving quality of software.
☞ **[SPPU : May-15, Marks 6]**

Ans. : Refactoring is necessary for simplifying the design without changing the function or behaviour. **Fowler** has defined refactoring as "The process of changing a software system in such a way that the external behavior of the design do not get changed, however the internal structure gets improved".

Importance of refactoring :

- The **redundancy** can be achieved.
- **Inefficient algorithms** can be eliminated or can be replaced by efficient one.
- Poorly constructed or **inaccurate data structures** can be removed or replaced.
- Other **design failures** can be rectified.

The decision of refactoring particular component is taken by the designer of the software system.

Q.9 Differentiate between abstraction and refinement.

☞ **[SPPU : Dec.-12, Marks 4]**

Ans. : Refer Q.6 (i) and Q.7 (i)

Q.10 Discuss the statement, "Abstraction and refinement are complementary concepts.

Ans. : Abstraction and refinement are complementary concepts. The major difference is that - in the abstraction low-level details are suppressed.

Refinement helps the designer to elaborate low-level details.

Q.11 What do you mean by the term cohesion and coupling in context of software design ? How are these concepts useful in arriving at a good design of a system ? ☞ **[SPPU : Dec.-16, Marks 5]**

Ans. : Cohesion

- With the help of cohesion the information hiding can be done.
- A cohesive module performs only "one task" in software procedure with little interaction with other modules. In other words cohesive module performs only one thing.

- Different types of cohesion are :
  1. **Coincidentally cohesive** - The modules in which the set of tasks are related with each other loosely then such modules are called coincidentally cohesive.
  2. **Logically cohesive** - A module that performs the tasks that are logically related with each other is called logically cohesive.
  3. **Temporal cohesion** - The module in which the tasks need to be executed in some specific time span is called temporal cohesive.
  4. **Procedural cohesion** - When processing elements of a module are related with one another and must be executed in some specific order then such module is called procedural cohesive.
  5. **Communicational cohesion** - When the processing elements of a module share the data then such module is communicational cohesive.
- The goal is to achieve high cohesion for modules in the system.

Coupling

- Coupling effectively represents how the modules can be "connected" with other module or with the outside world.
- Coupling is a measure of interconnection among modules in a program structure.
- Coupling depends on the interface complexity between modules.
- The goal is to strive for lowest possible coupling among modules in software design.
- The property of good coupling is that it should reduce or avoid change impact and ripple effects. It should also reduce the cost in program changes, testing and maintenance.
- Various types of coupling are :
  i) **Data coupling** - The data coupling is possible by parameter passing or data interaction.

ii) **Control coupling** - The modules share related control data in control coupling.

iii) **Common coupling** - In common coupling common data or a global data is shared among the modules.

iv) **Content coupling** - Content coupling occurs when one module makes use of data or control information maintained in another module.

Q.12 Give the difference between Cohesion and Coupling.

☞ [SPPU : Dec.-22, Marks 9]

Ans. :

Sr. No.	Coupling	Cohesion
1.	Coupling represents how the **modules are connected** with other modules or with the outside world.	In cohesion, the cohesive module performs only **one thing.**
2.	With coupling **interface complexity** is decided.	With cohesion, **data hiding** can be done.
3.	The goal of coupling is to achieve lowest **coupling.**	The goal of cohesion is to achieve **high cohesion.**
4.	Various types of couplings are -Data coupling, Control coupling, Common coupling and Content coupling.	Various types of cohesion are - Coincidental cohesion, Logical cohesion, Temporal cohesion, Procedural cohesion and Communicational cohesion.

Q.13 Justify " A cohesive design should have high cohesiveness and low coupling".

☞ [SPPU : April-15, Marks 4]

Ans. :

- Low coupling is an approach to interconnecting the components in a system or network so that those components, are less dependant upon each other.
- High cohesiveness indicate that module perform only one task.
- Ideally the components in the system must be least dependent on each other and every component must perform only one task at particular instance.
- Hence it is said that the system must have high cohesiveness and low coupling.

3.3 : Design Model

Q.14 Describe the design model with a neat diagram and give the traceability of each layer of design model to analysis model.

☞ [SPPU : Dec.-11, Marks 8]

Ans. :

- The **process dimension** denotes that the design model evolutes due to various software tasks that get executed as the part of software process.
- The **abstract dimension** represents level of details as each element of analysis model is transformed into design equivalent.
- In following Fig. Q.14.1 the **dashed line** shows the boundary between analysis and design model.
- In both the analysis and design models the same UML diagrams are used but in analysis model the UML diagrams are abstract and in design model these diagrams are refined and elaborated. Moreover in design model the implementation specific details are provided.

- Along the horizontal axis various elements such as architecture element, interface element, component level elements and deployment level elements are given. It is not necessary that these elements have to be developed in sequential manner. First of all the preliminary architecture design occurs then interface design and component level design occur in parallel. The deployment level design ends up after the completions of complete design model.

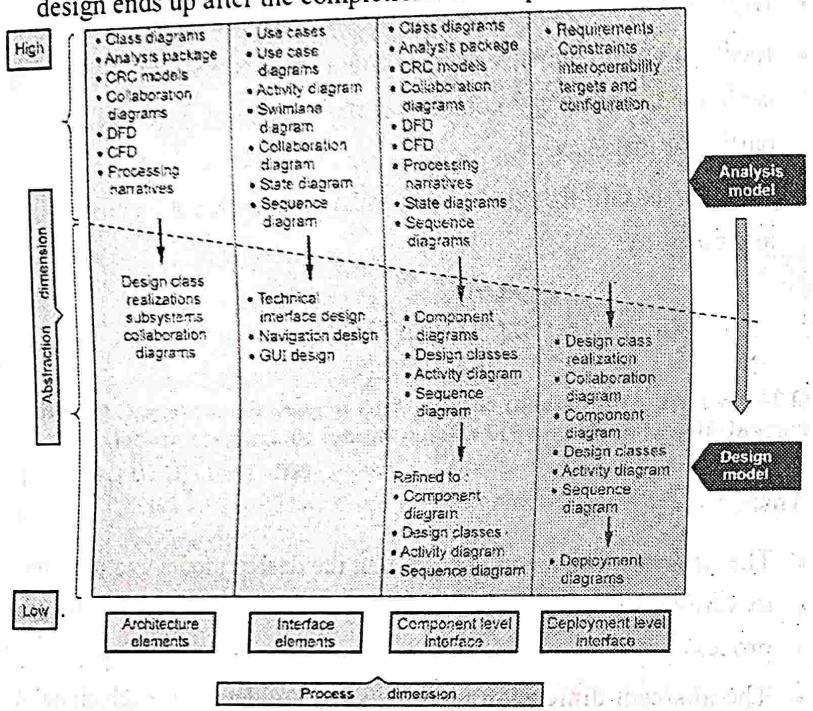


Fig. Q.14.1 Dimension of design model

Q.15 Explain the elements of design model.

☞ **[SPPU : May-13, Marks 8]**

Ans. : Following are the elements of design model -

1. Data Design Elements

- The data design represents the **high level of abstraction.**
- This data represented at data design level is **refined gradually** for implementing the computer based system.

- The data has great impact on the architecture of software systems. Hence structure of data is very important factor in software design.
- Data appears in the form of **data structures and algorithms** at the program **component level.**
- At the **application level** it appears as the **database** and at the **business level** it appears as **data warehouse and data mining.**
- Thus data plays an important role in software design.

2. Architectural Design Elements

The architectural design gives the layout for overall view of the software. Architectural model can be built using following sources -

- Data flow models or class diagrams
- Information obtained from application domain
- Architectural patterns and styles.

3. Interface Design Elements

- Interface Design represents the **detailed design** of the software system. In interface design how **information flows** from one component to other component of the system is depicted. Typically there are three types of interfaces –

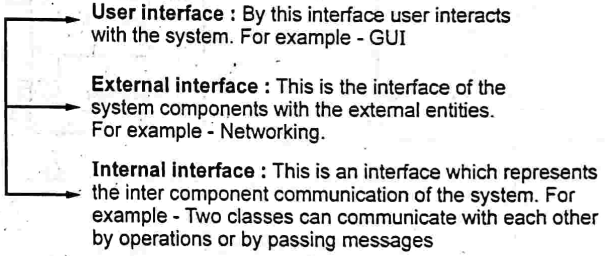


Fig. Q.15.1

4. Component Level Design Elements

- The component level design is more detailed design of the software system along with the specifications.
- The component level design elements describe the internal details of the component.
- In component level design all the local data objects, required data structures and **algorithmic details** and **procedural details** are exposed.

- Fig. Q.15.2 represents that component **order** makes use of another component **form.**
- The **order** is dependent upon the component form. These two objects can be interfaced with each other.

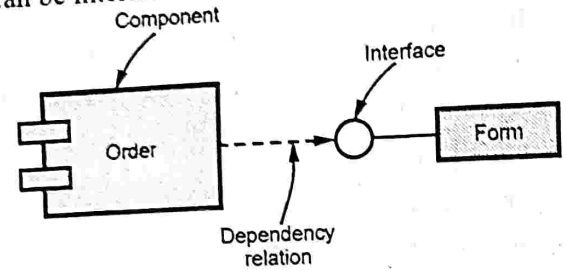


Fig. Q.15.2 Components

5. Deployment Level Design Elements

- The deployment level design elements indicate how software functions and software subsystems are assigned to the physical computing environment of the software product.
- For example web browsers may work in mobile phoned or they may run on client PC or can execute on server machines.

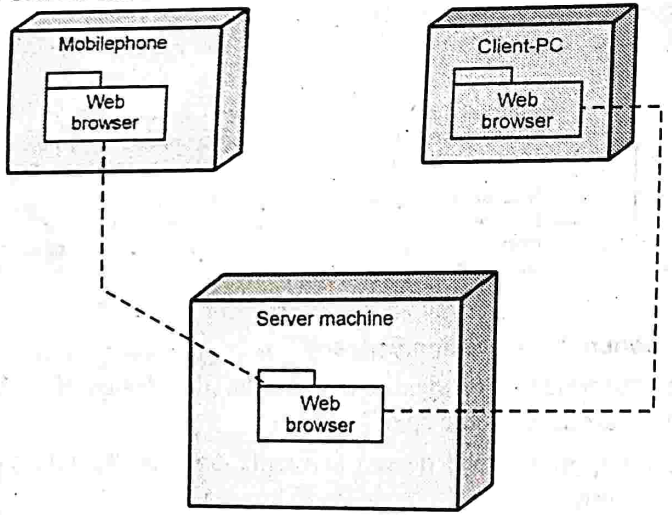


Fig. Q.15.3 Deployment diagram

Q.16 What are the elements in data design ? What are the guidelines for the data design ? ☞ **[SPPU : May-14, Marks 8]**

Ans. : Elements of Data Design :

- **Data Structure :** At the program component level, the data structure is associated with algorithms
- **Databases :** At the application level data model is converted to databases.
- **Datawarehouse :** At the business level, the collection of information stored in various databases is identified and Datawarehouse is formed.

Guideline for Data Design :

- The systematic analysis principles applied to function and behaviour should also be applied to data.
- All the data structures and operations to be performed should be identified.
- A data dictionary should be established and used to define both data and program design.
- Low level data decisions should be deferred until late in the design.
- A library of useful data structures and operations must be developed well in advance.
- A software design and programming language should support the specification and realization of abstract data types.

Q.17 What are the deployment level design elements ?

☞ **[SPPU : Dec.-11, Marks 4]**

Ans. : Refer Q.15 (5).

3.4 : Pattern - based Software Design

Q.18 Explain the template used for representing design pattern.

Ans. : The design pattern can be used using following design pattern template.

Template for Design Pattern	
Name :	The name of the design pattern is given initially. The name must be short and expressive.
Purpose :	The purpose of the design pattern is given in this section.
Known-as :	If any synonym for the pattern is existing then it should be mentioned.
Motivation :	It denotes nature of the problem along with supporting example.
Applicability :	It describes the design situation in which the pattern is applicable.
Structure :	It describes the classes that are required to implement the problem
Participants :	It includes the responsibilities of the classes that are used in the pattern.
Collaborations :	It describes all the collaborating classes that share some responsibilities.
Consequences :	It describe various design forces of the pattern.
Cross References :	If any related design patterns exist then those are described under this section.

3.5 : Architectural Design

Q.19 Explain why it is necessary to design the system architecture. What are the system factors affected by system architecture ? Explain.

Ans. :

- **Architectural design** is a specialized activity in which using specific architectural styles, by considering the system's structure and by understanding the properties of system components a large complex system is built.

- The person who is responsible to design such a system is called **software architect** in software engineering.
- The architectural design gives the layout of the system that is to be built. In short the **program structure** is created during architectural design.
- In architectural design process -
  - i) Basic structural framework is established.
  - ii) Major components of the system are identified.
  - iii) Communication between these components is established.

Factors Affected by System Architecture

1. **Business importance :** The software system which has to be built should have some business importance. Such a system should remain important for a long span of time.
2. **System age :** If the system becomes very old then it becomes very difficult to modify the architecture of such a system. Because changing such system architecture means disturbing previous functionalities of that architecture.
3. **System structure :** The modularization in the system structure is more preferred. If the system is built into then modules then it becomes easy to modify such system as per the requirements.
4. **Hardware procurement policies :** Sometime company policies may get changed for the hardware. For instance : if a company decides to built the system on servers instead of mainframe, then it becomes very expensive to change such hardware support.

3.6 : Design Decisions

Q.20 What is architectural design ? Explain architectural design decisions.

Ans. : Architectural Design : Refer Q.19.

Architectural Design Decisions :

- While building the software architecture many important decisions can be made about

1. Is it possible to create template for the system being developed ?
2. How to distribute the system uniformly across all the processors ?
3. What will be the appropriate style or pattern that needs to be chosen for the building the software architect ?
4. What is the purpose of the system development ?
5. How to make the subsystem component into the operation ?
6. What will be the structural decomposition of system in various modules ?

3.7 : Views

Q.21 Explain the 4+1 view model with suitable diagram

Ans. : There is **4+1 view model** of software architecture. These views are -

1. **Logical View :** The logical view represents the key objects or object classes. The system requirements can be correlated to the logical view of the system.
2. **Process View :** This view represents the run time interacting processes of the system.
3. **Development View :** This view represents how software is decomposed into components that are implemented by a single developer or a development team.
4. **Physical View :** This view shows how the system hardware and software components are distributed across the processors in the system. This view is used by system engineers for planning the software development activity.

- These views can be related using use cases or scenarios.

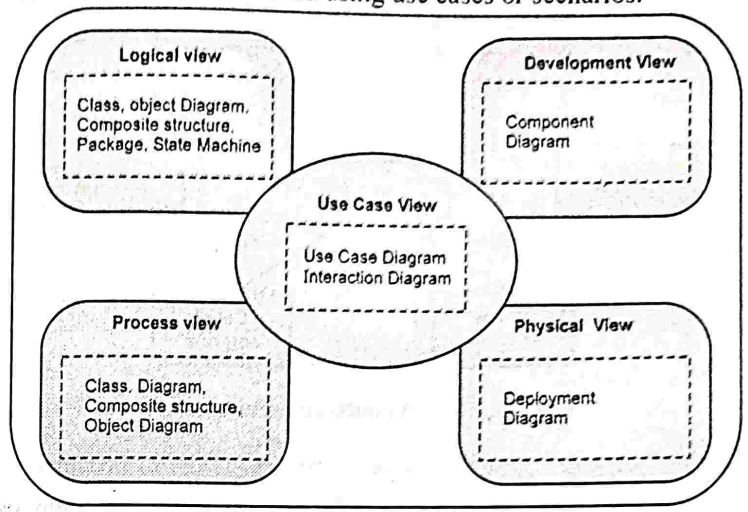


Fig. Q.21.1

3.8 : Patterns

Q.22 What is software architecture ? Explain data centered and object oriented architectural style of the system.

☞ **[SPPU : June-22, Marks 9]**

Ans. : Software Architecture is a structure of systems which consists of **various components**, externally visible properties of these components and the **inter-relationship** among these components.

1) Data Centered Architecture style

- In this architecture the data store lies at the center of the architecture and other components frequently access it by performing add, delete and modify operations. The client software requests for the data to central repository. Sometime the client software accesses the data from the central repository without any change in data or without any change in actions of software actions.

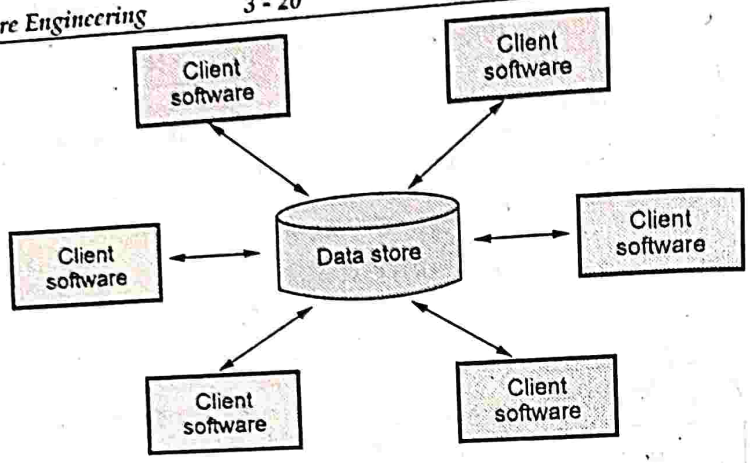


Fig. Q.22.1 Data centered architecture

- Data centered architecture possess the property of interchangeability. Interchangeability means any component from the architecture can be replaced by a new component without affecting the working of other components.
- In data centered architecture the data can be passed among the components.
- In data centered architecture,

  Components are : Database elements such as tables, queries.

  Communication are : By relationships

  Constraints are : Client software has to request central data store for information.

2) Object oriented architecture style

- In this architecture the system is decomposed into number of interacting objects.
- These objects encapsulate data and the corresponding operations that must be applied to manipulate the data.
- The object oriented decomposition is concerned with identifying objects classes, their attributes and the corresponding operations. There is some control models used to co-ordinate the object operations.

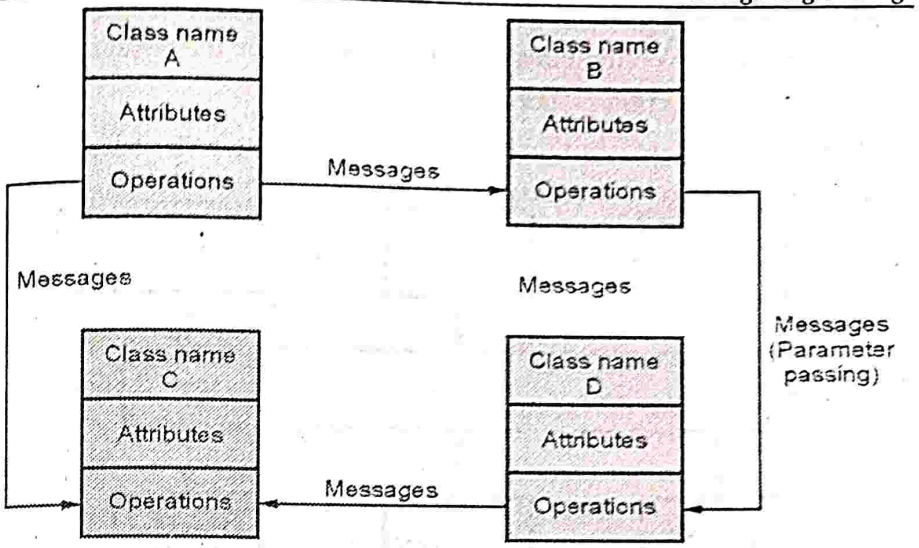


Fig. Q.22.2 Object oriented architecture

Q.23 Explain client server model of software architecture

Ans. : The model in which server contains the set of services and client can use those services is called the client-server model.

- In client-server model the client should know the names of the servers and the services offered by them. However, it is not necessary that the server should know the client.

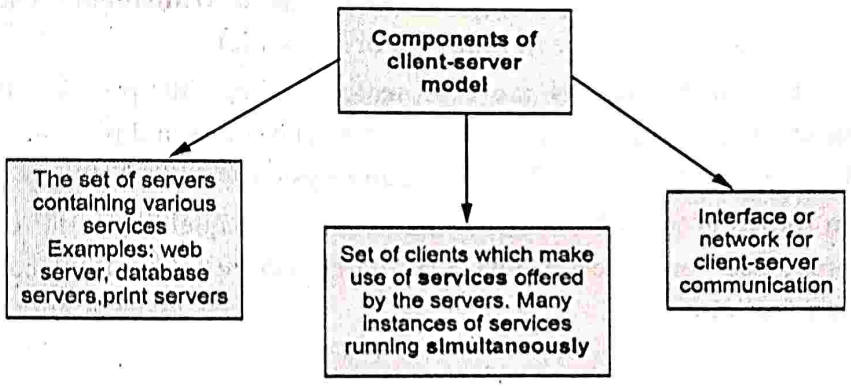


Fig. Q.23.1 Components of client-server model

- In client-server communication, the client makes the request to the server and the server responds the client.
- The client has to wait until the server responds it. The typical client-server model is as shown below.

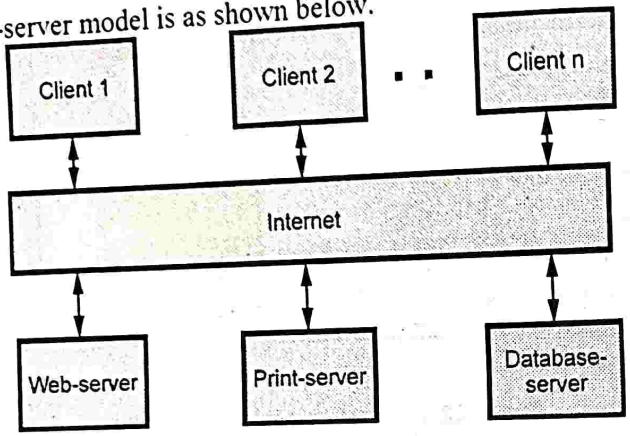


Fig. Q.23.2 Client-server model

Q.24 Explain pipe and filter pattern

Ans. :

- In function oriented pipelining, the functional transformations can be applied on the inputs for getting the output.
- Each processing step is implemented as a **transform**. These transforms can execute in sequence or in parallel.
- When transformations are represented as separate process, this model is called pipeline model or filter. The filters and pipe are the terms that are used in UNIX Operating System.
- In pipeline model if transformations are sequential with data processed in batches then this architecture is called batch **sequential model.**

For example :

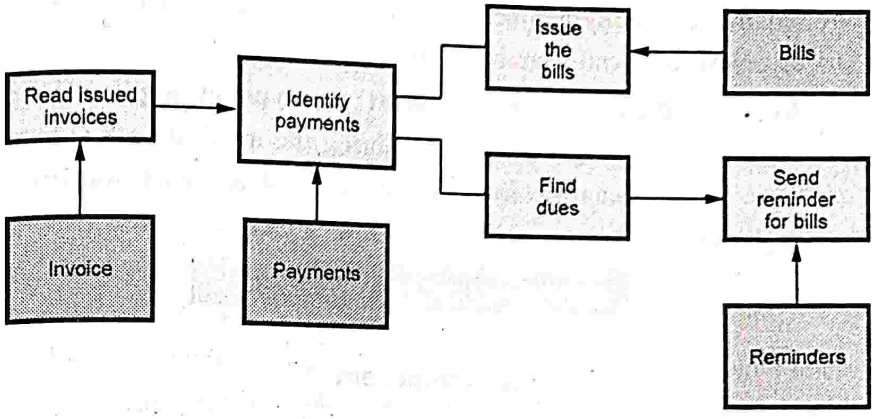


Fig. Q.24.1 Invoice processing system

3.9 : Application Architectures

Q.25 What is application architecture? Also mention uses of it.

Ans. : The **generic application architecture** is an architecture for the software systems that can meet various commonly found requirements in business logic.

Use of Generic Application Architecture

The generic application architecture can be used as

1. **The starting point for architectural design process :** The initial design of the generic application can be used to understand the purpose of the system getting developed.
2. **For organizing the work of development team :** The generic application architecture specifies the stable structural features of the system. Using these features it is possible to analyze all the important tasks to be performed while building the system.
3. **For identifying the reusable components :** The generic structure helps in identifying the components that can be reused during system implementation.

4. **For deciding the design checklist :** The check list for requirements or system components can be prepared with the help of generic application architecture.
5. **As vocabulary for talking about the type of application :** Various concepts get introduced during the application building, these concepts can be termed by the vocabulary found in generic application architecture.

Part II : Component Level Design

3.10 : Component

Q.26 What is component ?

Ans. :

- "Component is a modular, deployable and replaceable part of system that encapsulates the implementation and exposes the set of interfaces".
- Components are the part of software architecture and hence they are important factors in achieving the objectives and requirements of system to be built.
- Components can communicate and collaborate with other components.
- Design models can be prepared using object oriented views and conventional views.

3.11 : Designing Class based components

Q.27 List and explain the design principles

Ans. : There are four design principles that are used during the component level design. These principles are -

1. **The Open-closed Principle :** This principle states that - A module or a component should be open for extension and should be closed for modification. A designer should design a component in such a

manner that some functionalities can added to it if required but in doing so there should not be any change in the internal design of the component itself.

2. **The Liskov Substitution Principle :** This principle states that - subclasses should be substitutable for their base classes. This principle is proposed by **Barbara Liskov**. A component that contains the base class and if that component works properly then the same component should work properly even if the derived class of that base class is used by the component as a substitute.
3. **Dependency Inversion Principle :** The components should be dependant on the other abstract components and not on the concrete component. This is because if some component has to be extended then it becomes easy to enhance it using other abstract component instead of concrete component.
4. **The interface Segregation Principle :** Many client specific interfaces are better than one general purpose interface. According to this principle, designer should create specialized interfaces for each major category. So that the clients can access the interfaces based on the category. If a general purpose interface is created then multiple clients may try to access it for different operations at the same time.

Q.28 What are elements of design model ? What are the elements of architectural design ? Explain design principles ?

☞ **[SPPU : June-22, Marks 9]**

Ans. : Elements of Design Model : Refer Q.15.

Elements of Architectural Design :

1. **Components :** Components are independent units of software that perform specific functionality. The components are then combined to other components for building a system.

2. **Connectors :** Connectors are the links used for connecting the components together. For example - call, pipes.
3. **Interfaces :** Using Interfaces the components can communicate with each other. Interfaces define the methods that components use to exchange data and perform actions.
4. **Architectural patterns :** Architectural patterns are well-established solutions to common software architecture problems. For example - repository model, layered model, client server model.
5. **Architectural styles :** Architectural styles are sets of principles and guidelines that inform the overall structure of a software system. For example - Peer-to-Peer architecture.
6. **Deployment architecture :** Deployment architecture is the process of planning and designing how a software system will be deployed on hardware and software infrastructure.

- **Design Principles :** Refer Q.27

3.12 : Conducting Component-Level Design

Q.29 Explain the guidelines of component level design

Ans. :

- **Ambler** suggested following guideline for conducting component level design -
- **Components :** Components are the part of architectural model. The naming conventions should be established for components. The component names should be specified from the problem domain. These names should be meaningful.
- **Interfaces :** Interfaces serve important role in communication and collaboration. The interface should be drawn as a circle with a solid line connecting it to another element. This indicates that the

element to which it is connected offers the interface. The interface should flow from left side of component. Only important interfaces must be drawn.

- **Dependencies and interfaces :** The dependencies must be shown from left to right. The inheritance should be shown from bottom to top i.e. from derived to base class. The component interdependencies should be represented via interfaces.

3.13 : User Interface Design

Q.30 What is the necessity of a good user interface ? What are the rules to keep in mind while designing a user interface ?

☞ [SPPU : May-12, Marks 6]

Ans. : While designing for user interface the very first step is to identify **user, tasks** and **environmental techniques**. Based on user tasks, **user scenarios** are prepared. Such user scenarios help to design user interface objects and corresponding actions.

The Golden Rules

Thao Mandel has proposed three golden rules for user interface design -

1. Place the user in control
2. Reduce the user's memory load
3. Make the interface consistent.

Q.31 What are the design principles for reducing the user's memory load in user interface design ?

☞ [SPPU : Dec.-12, Marks 6, May-14, Marks 8]

Ans. :

1. **Do not force the user to have short term memory :** When user is involved in complex tasks then he has to remember more. The user interface should be such that the user need not have to remember

past actions and results. And this can be achieved by providing proper visual interface.

2. **Establish meaningful defaults :** Meaningful default options should be available to the user. For example in the Microsoft word the user can set the default font size as per his choice. Not only this, there should be some reset option available to the user in order to get back the original values.
3. **Use intuitive shortcuts :** For quick handling of the system such short cuts are required in the user interface. For example control+s key is for saving the file or control+o is for opening the file. Hence use of meaningful mnemonics is necessary while designing an interface.
4. **The visual layout of the interface should be realistic :** When certain aspect/feature of the system needs to be highlighted then use of proper visual layout helps a casual user to handle the system with ease. For example in an online purchase system if pictures of Master card/Visa card are given then it guides the user to understand the payment mode of online purchasing.
5. **Disclose the information gradually :** There should not be bombarding of information on user. It should be presented to the user in a systematic manner. For instance one can organize the information in hierarchical manner and narrate it to the user. At the top level of such hierarchical structure the information should be in abstract form and at the bottom level more detailed information can be given.

Q.32 Explain the user interface design issues.

☞ **[SPPU : May-11, Marks 6]**

Ans. : 1) System Response Time : Any user hate the delayed response time.

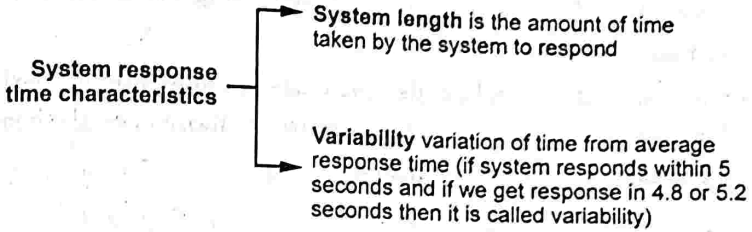


Fig. Q.32.1

2) **Help facilities :** This is the most essential criteria for any user interface this makes the system more interactive. The help can be **on-line help** or it can be in the form of **user manual**. Even some software provide the help in the **assistance** form i.e. the question may be asked by the user and the answer given by the system will serve as a help to the user.
3) **Error handling :** Errors and warning cause the frustrations to the user. Sometimes the error messages are in very vague form. For example : Application ABC has encountered error 1033. Now such type of error messages do not direct the user about what went wrong.

   Thus error messages should be so effective that they should bring qualitative interaction between the user and the system.
4) **Menu and command labelling :** There are typically two ways by which the user handles the system i.e. **keyboard** and **mouse**. The system can be handled using commands by the power user(i.e. knowledgeable and frequent users) whereas any ordinary user makes use of GUI and he prefers not to use any command as such.

5) Application accessibility :

- Accessibility guidelines are used for while developing the applications. These guidelines are mostly used by the web applications. The guideline assist in designing the interfaces used within the application.
- The accessibility guideline also provides the guideline for assistive technology that addresses the needs of those visual, hearing, speech and mobility impairments.

6) Internationalization

- There are situations in which the user interface is developed for localised purpose and for local language. If the same interface is required in other side of the country then existing interface is used with more or less modifications.
- The real challenge is to develop **globalised** software. That means the user interfaces should be designed to accommodate a generic core functionality. This functionality can be used by any user belonging to any country.
- The **Unicode standard** has been developed to handle the challenges of managing the natural languages with lots of characters and symbols.

Q.33 Explain the user interface design steps.

☞ **[SPPU : Dec.-12, Marks 8]**

Ans. : Following are the commonly used interface design steps -

1. During the interface analysis step **define** interface **objects** and corresponding **actions** or operations.
2. **Define** the major **events** in the interface. These events depict the user actions. Finally **model** these events.
3. Analyse how the **interface** will look like from user's point of view.
4. **Identify** how the **user understands** the interface with the information provided along with it.

- While designing the interface software engineer communicates the user and according to his thinking about the interface, software engineer draws the sketches. Get it approved from the user and then work on defining objects and corresponding actions.
- While designing the interface the designer has to follow -
  - Golden rules
  - Model the interface
  - Analyse the working environment

Q.34 Explain guidelines for component level design and principles for user interface design. ☞ **[SPPU : June-22, Marks 9]**

Ans. : Guidelines for Component Level Design : Refer Q.29.

Principles for User Interface Design : Refer Q.30 and Q.31.

END ...

Unit IV

4 Project Planning, Management and Estimation

Part I : Project Planning

4.1 : Project Initiation

Q.1 Explain project management life cycle.

Ans. :

- Project is defined as a temporary endeavor undertaken to create a unique product, service, or result.
- Projects can be large or small and involve one person or thousands of people. They can be done in one day or take years to complete. Examples of projects are :
  1. School implements a portal for tracking the progress of the students
  2. A pharmaceutical company launches a new drug.
  3. Mobile company introduces a new device.
- Project management is the application of knowledge, skills, tools and techniques to project activities to meet the project requirements.
- The project management activity passes through various life cycle activities. These activities are as follows
  - Initiation, planning, execution, monitoring and controlling and closing. These activities are described as below :

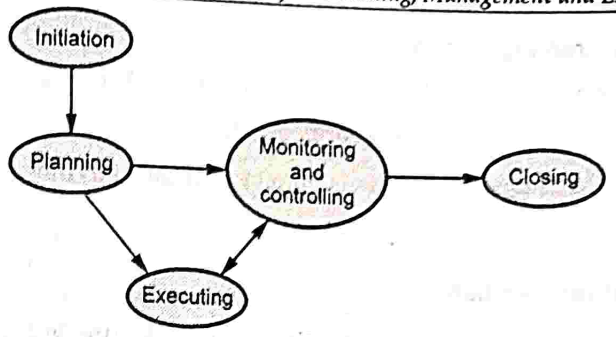


Fig. Q.1.1 Project management life cycle

Q.2 Explain four major sections of project management plan.

[SPPU : Dec.-18, Marks 5]

Ans. : Project management plan

The four major sections of project management plan are -

1) Project summary section :

- It gives high level overview of the project.
- It gives start date and end date of the project, project leader, contract at customer end, project objectives, milestones, deliverables, major commitments with customer.
- The assumptions made are explicitly listed in this section.

2) Project planning section :

- This section lists the outputs of executing various project planning procedures.
- It includes description of development processes being used, requirement change management processes, effort and scheduling estimates.
- It also specifies the environment and tools required for execution of project.
- The quality plan and the risk management plan are also given this section.

3) Project tracking section :

- This section defines the measurements to be taken. It also specifies the system to be used for recording data.
- This section describes various project tracking activities, frequency and nature of progress reporting.

4) Project team section :

- This section specifies the project team and structure of project team.
- It also specifies the roles and responsibilities of team members.

4.2 : Planning Scope Management

Q.3 Explain project scope management with an example.

☞ [SPPU : Oct.-18, Marks 5]

Ans. : Definition of project scope management : Project scope management includes the processes involved in defining and controlling what work is or is not included in a project.

Planning scope management process

- Planning scope management is an activity of planning how a scope will be managed throughout the life of the project.
- After reviewing the project management plan, project charter, enterprise environmental factors, and organizational process assets, the project team uses expert judgment and meetings to develop two important outputs: the **scope management plan and the requirements management plan.**

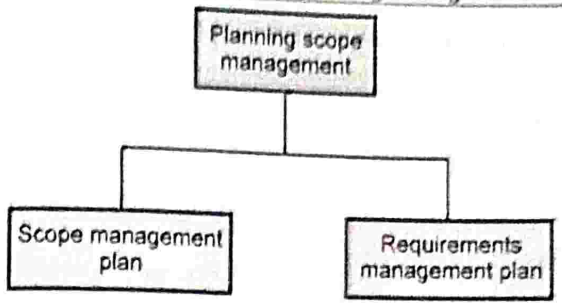


Fig. Q.3.1 Planning scope management

(1) Scope management plan : The scope management plan can be formal or informal depending upon the nature of the project. The scope management plan contains following information -

- **How to write detailed scope statement :** This aspect provides the information about some template that demonstrate the writing of detailed scope statement.
- **How to create work break down structure :** This section provides suggestions and resources for creating the work breakdown structure.
- **How to maintain work break down structure :** Initially the WBS often changes , but the scope management plan provides the guideline to maintain the work break down structure and how to get it approved.
- **How to obtain the acceptance about the deliverables :** The acceptance criteria is defined in this section which is important aspect for making the project payments.
- **How to handle request about change of project scope :** This section of scope management plan define how to handle changes in the project scope. This handling is using the organizational guideline about approval of changes in the scope.

(2) Requirement management plan : The requirement management plan documents how project requirements will be analyzed, documented and managed. This plan contains following information -

- How to plan, track and report requirements activities
- How to prioritize risks
- How to trace and capture requirements

4.3 : Creating the Work Breakdown Structure

Q.4 Explain work breakdown structure with an example.

☞ [SPPU : Aug.-17, Dec.-17, Marks 5]

Ans. : Definition of Work Breakdown Structure(WBS) : The work breakdown structure is a deliverable oriented grouping of the work involved in the project. The purpose of WBS is to define the scope of the project.

- The work break down structure is created after collecting the requirements and defining the scope.
- The project scope management plan, scope statement, requirements documentation, enterprise environmental factors, and organizational process assets are the **primary inputs** for creating a WBS.
- The WBS is created using **decomposition technique**. That means, project is decomposed into manageable chunks.
- There are two types of **outputs** of the process of WBS and those are - i) the scope baseline and ii) updated project documents. **The scope baseline** contains approved project scope statements.
- **Example of WBS :** *Fig. Q.4.1* represents the sample work breakdown structure(WBS) created for the task **Website building.** This structure is similar to organizational chart.

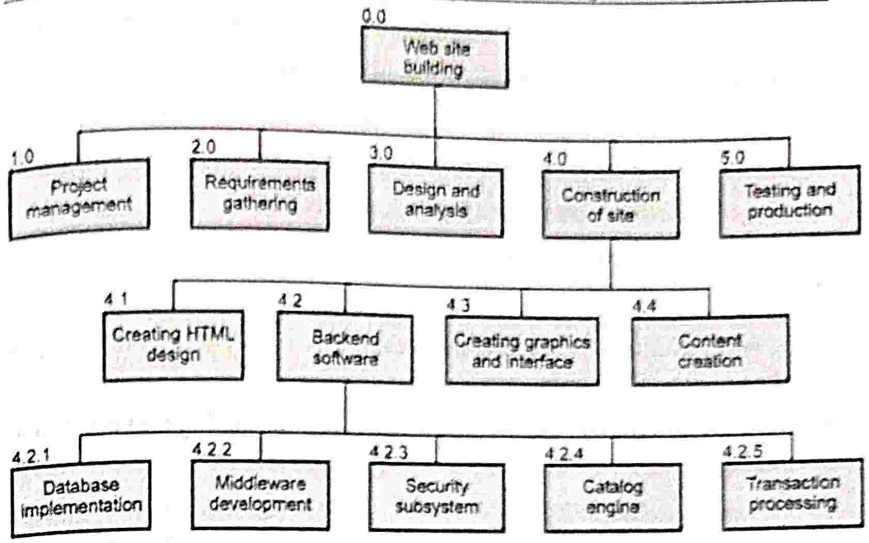


Fig.Q.4.1 Sample work break down structure (WBS) in chart form

Q.5 What is work breakdown structure ? How is it related with scope management and explain 8/80 rule.

☞ [SPPU : June-22, Marks 8]

Ans. : Work Breakdown Structure : Refer Q.4.

- The 8/80 rule in project management states that the lowest level of work should be no less than 8 hours and not more than 80 hours.

Q.6 What are the advantages of work breakdown structure ?

Ans. :

1. WBS forces the team to **create detailed step.** It helps to clarify ambiguities, bring out assumptions, narrow the scope of the project, and raise critical issues.
2. It lays groundwork for **budget and schedule.**
3. The work breakdown structure provides for a level of detail that makes it easier for the project manager to hold his team members **accountable and responsible** for the completion of their tasks.
4. The work breakdown structure allows for the allocation of **time and cost estimates** for all of the specific work packages.

5. The work breakdown structure can also be utilized in the **identification and minimization of the risks.**
6. The work breakdown structure facilitates the effective identification of the skills sets required to complete the work and there by helps in **increase of productivity.**

Q.7 Write short note on - WBS dictionary

Ans. :

- A WBS dictionary is a document that provides detailed information about each WBS item.
- In WBS dictionary is the work/activities defined for every work package.
- For each WBS element, the WBS Dictionary may include a brief definition of the scope or statement of work, defined deliverable(s), a list of associated activities, and a list of milestones.
- It can also include information such as resources required, an estimate of cost, contract information, quality requirements, and technical references to facilitate performance of the work .
- The WBS dictionary is also an important part of the **scope baseline.**
- When the stakeholder read the detailed description in WBS dictionary, they get better understanding about the **scope of the work.**
- **Sample WBS Dictionary :**

Most organizations use Microsoft Word to create and maintain a WBS dictionary.

Online Shopping System
Work Package ID : XXX1234
Work Package Name : Items Display Screen
Work Package Description : Create GUI that will list out various items available for online purchase. The cost and available quantity should also be displayed along with the items lists.

Assigned To : Mr. XYZ	Department: I.T.
Date Assigned:	Due Date :
Estimated Cost:	Accounting Code : abc123

- WBS dictionary provides a detailed information about each element of the Work Breakdown Structure.
- It provides the team with the information they need to produce quality deliverables that meet project requirements and organization standards.

4.4 : Scheduling

Q.8 What are different processes in project time management ?

☞ [SPPU : May-18, Marks 5]

Ans. : Definition of project time management : Project time management involves the processes required to ensure timely completion of a project.

- The activities involved in Project Time management are enlisted below :

1. **Activity definition :** This is a step in which specific activity-task are identified. Team members perform these activities and project deliverables are produced.
2. **Activity sequencing :** All the correlated project activities are identified and documented in this step.
3. **Activity resource estimation :** This is a step of estimating how many resources a project team should use to perform project activities.
4. **Activity duration estimation :** In this step, the number of work periods that are needed to complete individual activities are identified.
5. **Schedule development :** Various tasks conducted in this step

are - analyzing activity sequences, activity resource estimates, and activity duration estimates to create the project schedule.

6. **Schedule control :** This is a step of controlling and managing the changes to the project schedule.

Q.9 Explain the method of estimating activity duration.

Ans. :

- After defining activities and determining their sequence, the next step in time management is duration estimating.
- **Definition of activity duration :** The activity duration can be defined as the actual amount of time worked on an activity plus elapsed time.
- **Definition of effort :** Effort is the number of workdays or working hours required to complete the task.
- Effort and activity duration are two distinct entities.
- For estimating activity duration various inputs are used. This includes - activity lists, schedule management plan, activity resource requirements, project scope statements and so on.
- The outcome of estimating activity duration process are :
  i) Project duration estinates and
  ii) Updated project document. People doing the work should help create estimates, and an expert should review them.
- There are various types of estimates produced during this process. Out of which the most common type of estimate is - **three point estimate.**
- The three point estimate is an estimate that includes an
  1) Optimistic,
  2) Most likely, and
  3) Pessimistic estimate,

  such as
  1) Three weeks for the optimistic,

2) Four weeks for the most likely, and
3) Five weeks for the pessimistic estimate.

Q.10 Calculate activity expected time and variance for given problem.

☞ [SPPU : June-22, Marks 9]

Activity ID	Optimistic time (t_o)	Most likely time (t_m)	Pessimistic time (t_p)
Job 1	1	3	5
Job 2	2	6	9
Job 3	2	3	5
Job 4	5	8	10
Job 5	11	15	20
Job 6	2	5	8
Job 7	3	3	3
Job 8	2	4	6

Ans. : Formula used for calculating

1) Expected time $= \frac{t_o + 4t_m + t_p}{6}$

2) Variance $= \sigma^2 = \left[\frac{t_p - t_o}{6}\right]$

The values of expected time and variance for each activity are calculated using above formula and enlisted in the following table.

Activity ID	Optimistic time(t_o)	Most likely time(t_m)	Pessimistic time(T_p)	Expected time	Variance
Job 1	1	3	5	3	0.45
Job 2	2	6	9	5.83	1.37

Job 3	2	3	5	3.16	0.25
Job 4	5	8	10	7.83	0.69
Job 5	11	15	20	15.16	2.25
Job 6	2	5	8	5	1
Job 7	3	3	3	3	0
Job 8	2	4	6	4	0.44

Q.11 What is COCOMO II ? What areas does COCOMO II address ? ☞ [SPPU : June-22, Marks 9]

Ans. : This model is developed in 1981 by **Barry Boehm** to give an estimate of the number of man-months it will take to develop a software product. COCOMO predicts the efforts and schedule of a software product based on **size** of the software. COCOMO stands for "COnstructive COst MOdel".

COCOMO has three different models that reflect the complexity -

1) **Basic model :** The basic COCOMO model estimates the software development effort using only **Lines of Code**. Various equations in this model are -

$$E = a_b (KLOC)^{b_b}$$

$$D = c_b (E)^{d_b}$$

$$P = E/D$$

Where E is the **effort** applied in person-months.

D is the development time in chronological months.

KLOC means kilo line of code for the project.

P is total number of persons required to accomplish the project.

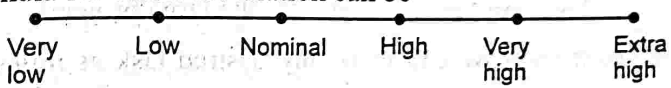
The coefficients a_b, b_b, c_b, d_b, for three modes are as given below.

Software projects	a_b	b_b	c_b	d_b
Organic	2.4	1.05	2.5	0.38
Semi-detached	3.0	1.12	2.5	0.35
Embedded	3.6	1.20	2.5	0.32

Table Q.11.1

2) **Intermediate model :** This is an extension of Basic COCOMO model. This estimation model makes use of set of "Cost driver attributes" to compute the cost of software.

The cost driver attributes get a 6-point scale ranging from "very low" to "extra high". These ratings can be viewed as



The formula for effort calculation can be -

$$E = a_i (KLOC)^{b_i} \text{ EAF person-months}$$

The values for a_i and b_i for various class of software projects are -

Software project	a_i	b_i
Organic	3.2	1.05
Semi-detached	3.0	1.12
Embedded	2.8	1.20

Table Q.11.2

The duration and person estimate is same as in basic COCOMO model. i.e.

$$D = c_b (E)^{d_b} \text{ months months}$$

i.e. use values of c_b and d_b coefficients that are in Table Q.11.1.

$$P = E/D \text{ persons}$$

3) Detailed COCOMO model : The detailed model uses the same equations for estimation as the intermediate model. But detailed model can estimate the effort (E), duration (D) and persons (P) of each of development phases, subsystems, modules.

The experimentation with different development strategies is allowed in this model.

Using the detailed cost drivers, an estimate is determined for each phase of the lifecycle.

Q.12 Explain the adding of milestones to Gantt chart.

Ans. :

- Milestones are significant event on the project. This event has zero duration.
- Normally project developers prefer to meet the milestones during the project development, especially for the large projects.
- On the Gantt chart we can mark any desired task as milestone by entering its duration as zero.
- Following is a SMART criteria that is to be followed while developing milestone.
  - **Specific :** The milestone should target a specific area of improvement or answer a specific need.
  - **Measurable :** The milestone must be quantifiable, or at least allow for measurable progress.
  - **Assignable :** The milestone should be realistic, based on available resources and existing constraints.
  - **Realistic :** The milestone should align with other business objectives to be considered worthwhile.
  - **Time-framed :** The milestone should have definite deadline

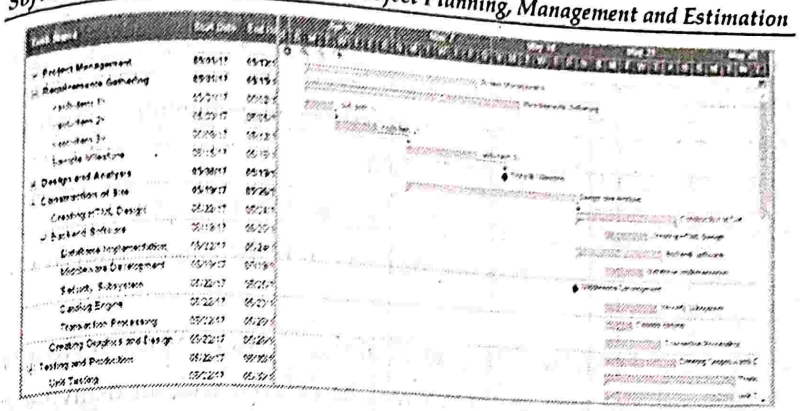


Fig. Q.12.1

Q.13 Give the advantages and disadvantages of Gantt chart.

Ans. : Advantages

Various advantages of using Gantt chart are as given below :

1. The Gantt chart provides the standard format for displaying the planned and actual schedule information.
2. They are easy to create and understand.
3. The progress of each individual task can be tracked.
4. Milestone for particular task can be set and developer can get acknowledged with it.

Disadvantages

1. The relationship or dependencies among the task is not very clear using Gantt chart.
2. If large number of tasks are involved in the project, the Gnatt chart becomes extraordinarily complex.
3. The size bar does not indicate the amount of work, it simply shows the duration.
4. The cost of the project can not be depicted in the Gantt chart.

Q.14 What is critical path method ? Explain it with suitable example

Ans. : • Critical path method is a network diagramming technique used to predict total project duration.

- Critical path denotes the series of activities that determine the time during which the project is completed. It is the longest path in the network diagram.
- The critical path has least amount of slack or float. Slack or float is the amount of time an activity may be delayed without delaying a succeeding activity or the project finish date.
- A Network Diagram is a visual representation of a project's schedule. A network diagram in project management is useful for planning and tracking the project from beginning to finish. It represents a project's critical path as well as the scope for the project.
- Consider, some hypothetical tasks, duration and dependencies.

Task	Duration	Dependencies
T1	10	
T2	15	T1 (M1)
T3	15	T1 (M2)
T4	7	T1, T2 (M3)
T5	20	T1 (M1)
T6	5	T4, T5 (M4)
T7	4	T6 (M6)
T8	8	T7 (M5)
T9	10	T6, T8 (M7)
T10	12	T9 (M8)

- The T1, T2, ... represent various tasks. The milestones of various tasks is shown by M1, M2, ... The network diagram for these activities can be as shown below.

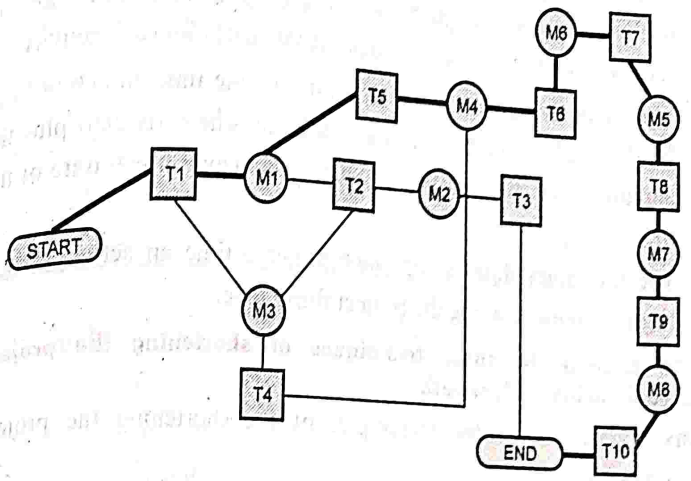


Fig. Q.14.1 Network diagram

- The minimum amount of time required by the project can be computing the longest possible path to reach from begin to end. This path is referred as the **critical path** in the project and the activities along this path are referred as the critical activities. In Fig. Q.14.1 critical path is shown by thick edge.

Q.15 Define the terms - Free slack, early start date, total slack, forward pass, backward pass, early finish.

Ans. : Free slack or free float is the amount of time an activity can be delayed without delaying the early start date of any immediately following activities.

- The **early start date** is the earliest possible time an activity can start based on the project network logic.
- **Total slack or total float** is the amount of time an activity can be delayed from its early start without delaying the planned project finish date.

- Project managers calculate free slack and total slack by doing a **forward and backward** pass through a network diagram.
- A **forward pass** determines the early start and early finish. A **backward pass** through the network diagram determines the late start and late finish dates for each activity dates for each activity.
- The **early finish date** is the earliest possible time an activity can finish based on the project network logic. The early start plus the duration of the first activity is equal to the **early finish date** of the first activity.
- The **late start date** is the latest possible time an activity might begin without delaying the project finish date.

Q.16 Explain the three techniques of shortening the project schedule using critical path.

Ans. : There are three techniques of for shortening the project schedule

1. By adding more resources or by changing the scope of the project the project duration can be shortened.
2. **Crashing :** This is an activity for obtaining the greatest amount of schedule compression for least incremental cost. For example - temporary worker could be hired to work in parallel with the other worker to speed up the task or A 2 week task with one person working 50 % could be shortened to 1 week if the person is assigned 100 %.
3. **Fast tacking :** This is a technique in which the activities are conducted in parallel or in overlapping manner instead of doing them in sequence. For example - instead of waiting for complete analysis , some part of coding can be started.

Q.17 Write short note on - PERT.

Ans. :

- PERT is a network analysis technique used to estimate project duration when there is a high degree of uncertainty about the individual activity duration estimates.

- It makes use of **Probabilistic Time Estimate**. This is a duration estimates based on using optimistic, most likely, and pessimistic estimates of activity durations, or a three-point estimate
- The formula used in PERT to calculate number of working days is based on three point estimate.
- The three point estimates are optimistic, most likely and pessimistic estimates.
- **Formula** is

PERT weighted average

$$= \frac{\text{Optimistic time} + 4 \times \text{Most likely time} + \text{Pessimistic time}}{6}$$

- **Example :** Given that

Optimistic time = 6 days
Most likely time = 9 days
Pessimistic time = 12 days

then PERT weighted average $= \frac{6 + (4 \times 9) + 12}{6} = 9$

- Thus it would be 9 days using PERT.
- The **main advantage** of PERT is that it attempts to address the risk associated with duration estimates.

Q.18 An R and D project has a list of task to be performed whose time estimates are given in the table as follows :

Activity	Activity name	Optimistic	Most likely	Pessimistic
1 - 2	A	4	6	8
1 - 3	B	2	3	10
1 - 4	C	6	8	16
2 - 4	D	1	2	3
3 - 4	E	6	7	8
3 - 5	F	6	7	14
4 - 6	G	3	5	7
4 - 7	H	4	11	12

5 - 7	I	2	4	6
6 - 7	J	2	9	10

Calculate expected time and variance. Draw project network diagram. Find critical path and find the probability that the project is completed in 19 days. Assume Z (1.34) = 0.4099.

☞ [SPPU : Aug.-17, Marks 10]

Ans. : The expected time and variance of each activity is calculated using following formula

1) Expected time $(t_c) = \frac{t_o + 4t_m + t_p}{6}$

2) Variance $\sigma^2 = \left(\frac{t_p - t_e}{6}\right)^2$

The following table shows t_e and σ^2 values for each activity

Activity	t_o	t_m	t_p	t_e	σ^2
1 - 2	4	6	8	6	0.444
1 - 3	2	3	10	4	1.777
1 - 4	6	8	16	9	2.777
2 - 4	1	2	3	2	0.111
3 - 4	6	7	8	7	0.111
3 - 5	6	7	14	8	1.777
4 - 6	3	5	7	5	0.444
4 - 7	4	11	12	10	1.777
5 - 7	2	4	6	4	0.444
6 - 7	2	9	10	8	1.777

The network diagram is as follows

The largest path is 1 - 3 - 4 - 6 - 8 = 24 days (t_e)

The probability of completing the project with in 19 (t_s) days is given by $P(Z < Z_0)$.

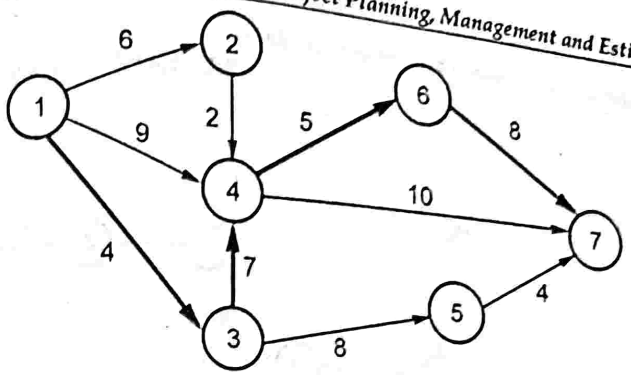


Fig. Q.18.1

To find Z_0, following formula is used

$$\frac{t_s - t_e}{\sqrt{\text{Sum of } \sigma \text{ in critical path}}} = \frac{19 - 24}{\sqrt{1.777 + 0.111 + 0.444 + 1.777}}$$

$$= -2.466$$

$$\text{As} = P(Z < Z_0) = 0.5 - Z\,(1.34)$$

$$= 0.5 - 0.4099 = 0.0901 = 9.01\ \%$$

Thus probability of completing R and D project in 19 days is **9.01 %**

Q.19 An assemble to made from two parts 'X' and 'Y' both parts must be turn a lathe. 'Y' must be polished and 'X' need not be polished. The sequence of activities together with their predecessors given below.

Activity	Description	Predecessor
A	Open work order	-
B	Get material 'X'	A
C	Get material 'Y'	A
D	'X' on lathe	B
E	'Y' on lathe	B, C
F	Polish 'Y'	E
G	Assemble 'X' and 'Y'	D, F
H	Pack	G

Draw the network diagram for above data. [SPPU : Oct.-18, Marks 5]

Ans. :

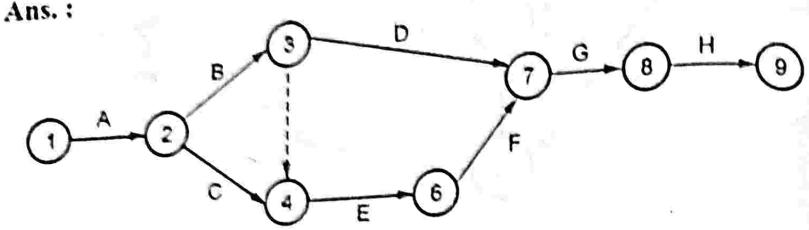


Fig. Q.19.1

Q.20 What is meant by network diagram ? What are different ways of drawing the network diagram ? [SPPU : May-18, Marks 5]

Ans. : Concept of network diagram : A network diagram is a graphical representation of a project and is composed of a series of connected arrows and boxes to describe the inter-relationship between the activities involved in the project. Boxes or nodes represent the activity description and arrows show the relationship among the activities.

Different ways of drawing network diagram :

There are two types of drawing network diagram -

1) Arrow diagram method :

- The arrow diagram is also called as **activity network diagram.**
- Arrows are used to represent activities associated with the project.
- The tail of the arrow represents the start of the activity and the head represents the finish.
- The length of the arrow represents the duration of the activity.
- Each arrow connects two boxes called nodes. The node represents the start or end of the activity. The starting node of the activity is called **i-node** and the end node of the activity is called as **j-node.**

- The chart can only show a "finish to start" logical relationship

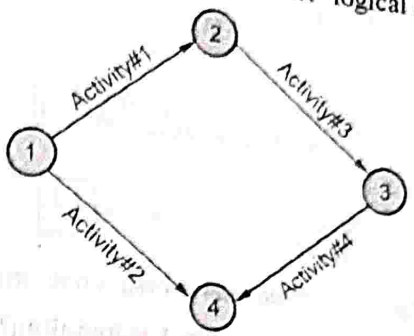


Fig. Q.20.1

2) Precedence diagram method :

- In this method the activities are represented by boxes or nodes.
- The arrows connecting them represent the relationship between the activities.
- Depending upon the precedence of the activity, the arrows represent different relationships.
- Following are the four ways of developing a diagram -
  - **Start to Finish :** This dependency requires only one activity to start before the other can be finished.
  - **Finish to Start :** The activity has to be finished before starting the next activity.
  - **Start to Start :** Both the activities can start together.
  - **Finish to Finish :** Both the activities need to be finished together.
- The duration for completion of the particular activity can be mentioned over the arrow connecting the activity to its successor.

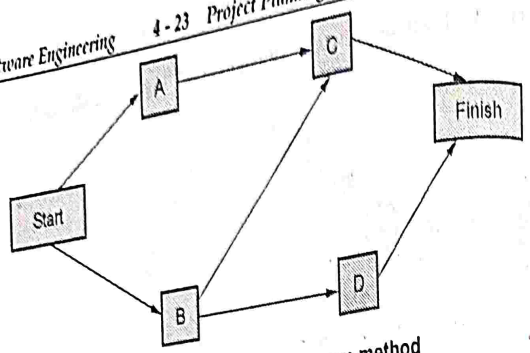


Fig. Q.20.2 Precedence diagram method

Q.21 What is task network in project scheduling? Explain with an example. [SPPU : May-19, Marks 8]

Ans. :

- The task is a small unit of work.
- The task network or an activity network is a graphical representation, with :
  - Nodes corresponding to activities.
  - Tasks or activities are linked if there is a dependency between them.
  - The task network for the product development is as shown in Fig. Q.21.1.

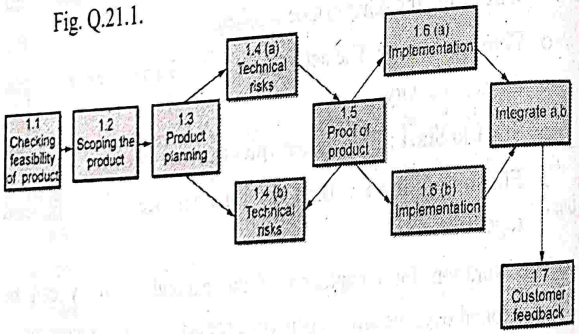


Fig. Q.21.1 Task network

- The task network definition helps project manager to understand the project work breakdown structure.

- The project manager should be aware of interdependencies among various tasks. It should be aware of all those tasks which lie on the critical path.

Part II : Project Management

4.5 : The Management Spectrum

Q.22 Explain the role of people, product and process in project management. [SPPU : May-11, Marks 6, May-13,15,19, Marks 8]

Ans. : 1. The people :

- **People factor** is an important issue in software industry. There is a strong need for motivated and highly skilled people for developing the software product. The Software Engineering Institute (SEI) has developed the People Management Capability Maturity Model (PM-CMM)
- By using PM-CMM model software organizations become capable for undertaking complex applications which ultimately attracts or motivates the talented people.
- Following are some key practice areas for software people -
  - Recruitment
  - Selection
  - Performance management
  - Training compensation
  - Career development
  - Organization and work design
  - Culture development.

2. The product :

- Before planning the project three important tasks need to be done -
  (1) Product objectives and scope must be established.
  (2) Alternative solutions should be considered.
  (3) and management constraints must be identified.
- The software developer and customer must communicate with each other in order to define the objectives and scope of the product.
- This is done as the first step in requirement gathering and analysis.
- The scope of the project identifies primary data, functions and behaviour of the product.
- After establishing the objectives and scope of the product the alternative solutions are considered.
- Finally, the constraints imposed by - delivery deadline or budgetary restrictions, personal availability can be identified.

3. The process

- The software process provides the framework from which the software development plan can be established.
- There are various framework activities that needs to be carried out during the software development process. These activities can be of varying size and complexities.
- Different task sets-tasks, milestones, work products and quality assurance points enable framework activities to adapt the software requirements and certain characteristics of software project.
- Finally, umbrella activities such as software quality assurance (SQA) and Software Configuration Management (SCM) are conducted. These umbrella activities depend upon the framework activities.

4.6 : People

Q.23 What are the categories of stakeholders ? What are the characteristics of effective project manager ?

☞ [SPPU : May-14, Dec.-19, Marks 8]

Ans. : Categories of stakeholders :

- Stakeholders mean persons who will be affected by the system. Following are the categories of the stakeholders -

1. Senior manager

- These are the persons who define the business issues and the decisions taken by have significant influence on the success of the project.

2. Project manager

- The project manager performs various tasks such as planning, motivating, organizing and controlling the software practitioners who are involved in software building.

3. Practitioners

- The practitioners are the entities having sufficient technical skills required for engineering the product or application.

4. Customers

- These stakeholders specify the requirements of the project and are interested only in outcome of the project.

5. End-users

- The end-users interact with the software product once get released.
- The team should be organized in such a way that skills and abilities of each person can be utilized at the most and this job is done by Team leader.

Characteristics of effective project manager :

- **Motivation**

  This ability is necessary to encourage technical people to do their work with the fullest of their ability.

- **Problem solving**

  During project management, the project manager must diagnose the technical and organizational issues in such a manner that systematic and effective solution can be obtained. He/She must take lessons from the previous projects and must remain flexible to change the direction of the if initial problem solving methods get failed.

- **Organization**

  This is an ability to arrange the existing processes so that intial concept can be translated into the final working product.

- **Innovation**

  Innovation or ideas mean encouraging people to create and to feel creative.

- **Managerial identity**

  The effective project manager must take the charge of complete project and must have full control over the project development process.

- **Achievement**

  The productivity of the project team can be optimized by awarding the team members for their good work and by not punishing them on the controlled risk.

- **Influence and team building**

  An effective project manager should be able to "read" the people in his team, he should understand their needs and must remain in controlled in stressful situations.

Q.24 List the four P's of software project management spectrum. Explain how "the people" factor contributes towards the success of software project. ☞ [SPPU : Dec.-16,22, Marks 5]

Ans. : In the software project management spectrum the four P's are -

1) People
2) Product
3) Process
4) Project

People factor : Refer Q.22 (i).

4.7 : Product

Q.25 Explain the term problem decomposition in detail. ☞ [SPPU : Dec.-22, Marks 9]

Ans. : Problem decomposition means partitioning or elaborating the problems.

- During the project scope no attempt is made to decompose the problem fully. The decomposition that is normally applied in based on two major areas -
  1. **Functionalities** that must be delivered.
  2. The **process** that will be used to deliver it.
- Normally, "**Divide and Conquer**" strategy is applied to partition the product into smaller pieces which can be managed easily as project planning begins.
- The software functions described in the scope are evaluated and refined to provide more details before the beginning of the estimation.
- As both cost and schedule are functionally oriented, some degree of decomposition is often useful.
- As the statement of scope evolves the first level of partitioning naturally occurs.

4.8 : Process

Q.26 Explain the term process decomposition in detail. [SPPU : Dec.-22, Marks 9]

Ans. :

- The software development team has a flexibility for choosing the software process model which is best suitable for the project. After decision of process model the software engineering tasks are decided.
- When a project manager raises a question "How do we accomplish the framework activities ?" then project decomposition commences. For example - a small project might require following work tasks for performing the communication activity -
  1. A list of **clarification issues** is to be prepared.
  2. For addressing the clarification issues conduct **meetings** with customers.
  3. The developer and customer together must prepare **statement of scope**.
  4. **Review** the statement of scope.
  5. Perform **modifications** in the statement of scope if required.
- Similarly for the large projects the work task list can be prepared which may contain some additional tasks.

4.9 : Project

Q.27 Enlist any five symptoms to indicate why the software project fail

Ans. :

1. Software developers do not understand the customer's need.
2. The scope of the project is not defined properly.
3. Change management is done poorly.

4. Business needs change very often.
5. The technological changes are quite often.
6. Users are not co-operative.
7. Unrealistic deadlines are set.

4.10 : The W5HH Principle

Q.28 Explain W5HHH principle. [SPPU : Dec.-22, Marks 5]

Ans. : The W5HH principle is nothing but the series of questions in the form of why, what, when, who, how and so on. Answers to these questions lead to definition of key project characteristics.

- **Why is the system being developed ?**

  Answer to this question assesses the validity of business reasons for the software work. It also justifies the expenditure of people, time and money.

- **What will be done ?**

  Answer to this question will help the software team member to identify the project tasks and milestones.

- **When will be done ?**

  The answer to this question will help to prepare the project schedule with identified project tasks and milestones.

- **Who is responsible for a function ?**

  By answering this question, the roles and responsibilities required to develop the system can be defined.

- **Where are they organisationally located ?**

  All the roles can not be defined within the software team itself. There are customers, users and other stakeholders holding some responsibilities.

- **How to do the job technically with proper management ?**

  Answer to this question will help to establish the technical strategy about the project.

- **How much of each resource required ?**

  This answer will be useful for deriving the project estimate or cost of the project.

4.11 : Metrics in the Process and Project Domains

Q.29 Explain in detail the software process and project metrics.

☞ [SPPU : May-15, Marks 8]

Ans. : Process metrics are collected across all projects and over long periods of time. Their intent is to provide indicators that lead to long term software process improvement.

- The project indicators enable a software project managers to (1) Asses the status of an ongoing project. (2) Track potential risks (3) Uncover problem areas before they go critical (4) Adjust work flow or tasks (5) Evaluate the project team's ability to control quality of software work products.
- In making improvements to any software system, there are three basic quality factors to consider : **Product, people and technology.**
- These three are the major determinants of software cost, schedule, productivity, and quality.
- The factor **people** includes hiring the best people you can find, motivating them to do the best job, and training them on the skills needed to perform their jobs effectively.
- The **technology** factor includes acquiring and installing tools that help automate. The technology also includes use of new software

languages to develop the desired quality product (e.g., Java, Visual C++, Oracle).

- The complexity of the factor **product** has great impact on quality and team performance.
- Out of these three components if one component is altered then it will impact other two factors. For example : If an organization makes use of new testing tool (**technology**) then the staff (**people**) must be trained to work with this tool. The organization must consider if this new tool helps in generating the desired **software product.**

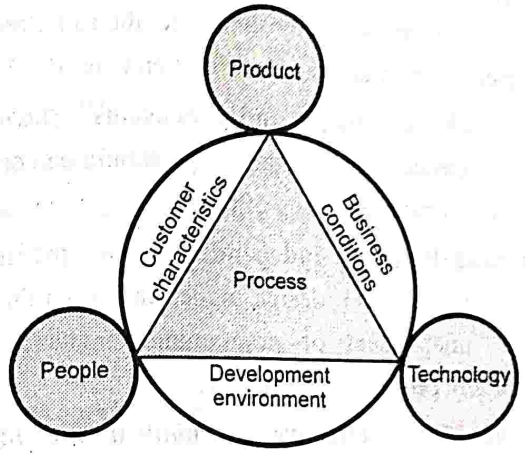


Fig. Q.29.1 Quality factors for Improvement

- This process triangle resides within the circle. This **circle** specifies the **environmental conditions** such as :
  - Customer characteristics(communication and collaboration between user and developer).
  - Business conditions(Organizational policies, Business rules)
  - Development environment (use of new technologies, use of automated tools).

4.12 : Software Measurement

Q.30 What are the attributes of effective software metrics ?

☞ [SPPU : May-14, Marks 4]

Ans. : The effective software metrics should have following attributes.

1. **Simple and computable** - The derivation of metric should be easy to compute and should not be a time consuming activity.
2. **Empirically and intuitively persuasive** - It should be immediate and can be derived based on observations.
3. **Consistent and objective** - The metric should produce unambiguous results. Anybody should get the same result by using these metrics when same set of information is used.
4. **Consistent in its use of units and dimensions** - The mathematical units and dimensions used for the metric should be consistent. And there should not be intermixing of units.
5. **Programming language independent** - The metric should be based on analysis model, design model and program structure. It should be independent of programing languages, syntax or semantic of any programing language.
6. **Metric should be effective mechanism for high quality feedback** - The metric should provide a way to produce high quality software product.

Q.31 What is objected oriented metric ?

☞ [SPPU : Dec 11., May-14, Marks 6]

Ans. :

- Use-cases also provide a software team with insight into software **scope and requirements**. But there are some difficulties in developing estimations using use cases due to the following reasons
  - There are **varieties of ways** by which use cases can be described. There is no unique format for them.

  - Use-cases represent an **external view** (User's view) of the software. These are also written at different levels of **abstraction**. That means some use cases may be written to expose only primary functions where as some other use cases may describe each function in more detail.
  - The **complexity** of the functions and features cannot be described by use cases.
  - The **complex behaviour** involved in many functions and features is also not described in use cases.
- In spite of these difficulties the use cases can be used for estimation, but only if they are considered within the context of **structured hierarchy** that use cases describe.

Smith argues that any level of structured hierarchy -

1. Can be described by at the most 10 Use-cases.
2. Each use-case would not contain more than 30 distinct scenarios.

- Therefore before using the use cases for the estimation -
  - The **level** within the structured hierarchy must be established.
  - The **average length** of each use case must be determined.
  - The **type of software** application for which estimation is calculated must be defined.
  - The **rough system architecture** is defined.
  - After establishing these characteristics, **empirical data** can be used to estimate LOC or FP for each use case. Then the **structured hierarchical data** can be used to compute the efforts required to develop the system.

- Following formula is useful to compute this estimation -

$$\text{LOC estimates} = N * LOC_{avg} + [(Sa / Sh - 1) + (Pa / Ph - 1)] * LOC_{adjust}$$

where N = Actual number of Use-cases

LOC_{avg} = Historical average LOC/Use-case for this type subsystem

LOC_{adjust} = Adjustment based on 'n %' of LOC_{avg}

n = Defined locally and represents the difference between this project and 'Average' project

Sa = Actual scenarios per use-case

Sh = Average scenarios per use-case for this type subsystem

Pa = Actual pages per use-case

Ph = Average pages per use-case for this type of subsystem

The above expression is used to develop a rough estimation of the number of LOC, based on the actual number of use-cases adjusted, by the number of scenarios and page length of the use-cases.

Q.32 Explain size oriented metric ? What data should we collect to derive size oriented metrics ?

☞ **[SPPU : Dec.-11,13, Marks 8, May-14, Marks 4]**

Ans. :

- Size oriented measure is derived by considering the size of software that has been produced.
- The organization builds a simple record of size measure for the software projects. It is built on past experiences of organizations.
- It is a direct measure of software.

- A simple set of size measure that can be developed is as given below :
  - Size = Kilo Lines of Code (KLOC)
  - Effort = Person / month
  - Productivity = KLOC / Person-month
  - Quality = Number of faults / KLOC
  - Cost = $/KLOC
  - Documentation = Pages of documentation / KLOC
- The size measure is based on the lines of code computation. The lines of code is defined as one line of text in a source file.
- While counting the lines of code the simplest standard is :
  - Don't count blank lines.
  - Don't count comments.
  - Count everything else.
- The size oriented measure is not universally accepted method.

Example

- Consider an ABC project with some important modules such as
  1. User interface and control facilities
  2. 2D graphics analysis
  3. 3D graphics analysis
  4. Database management
  5. Computer graphics display facility
  6. Peripheral control function
  7. Design analysis models
- Estimate the project in based on LOC

- For estimating the given application we consider each module as separate function and corresponding lines of code can be estimated in the following table as

Function	Estimated LOC
User Interface and Control Facilities(UICF)	2500
2D graphics analysis(2DGA)	5600
3D Geometric Analysis function(3DGA)	6450
Database Management(DBM)	3100
Computer Graphics Display Facility(CGDF)	4740
Peripheral Control Function(PCF)	2250
Design Analysis Modules (DAM)	7980
Total Estimation in LOC	**32620**

- Expected LOC for 3D Geometric analysis function based on three point estimation is -
  - Optimistic estimation 4700
  - Most likely estimation 6000
  - Pessimistic estimation 10000

$$S = [S_{opt} + (4 * S_m) + S_{pess}]/6$$

Expected value = $[4700 + (4 * 6000) + 10000] /6 \rightarrow 6450$

- A review of historical data indicates -
  1. Average productivity is 500 LOC per month
  2. Average labor cost is $6000 per month
- Then cost for lines of code can be estimated as

  cost/LOC = (6000/500) = $12
- By considering total estimated LOC as 32620
  - Total estimated project cost = (32620*12) = $391440

  - Total estimated project effort = (32620/500)
    = 65 Person-months

Q.33 Differentiate and explain size and function oriented metrics.

☞ [SPPU : May-11, Marks 6]

Ans. :

Sr. No.	LOC based metrics	Function point based metrics
1.	Size oriented software metrics is by considering the size of the software that has been produced in lines of code.	Function oriented metrics use a measure of functionality delivered by the software.
2.	For a LOC based metric the software organization maintains simple records in tabular form. The typical table entries are : Project name, LOC, effort, pages of documents, errors, defects, total number of people working on project.	Most widely used function oriented metric is the Function Point (FP) computation of the function point is based on characteristics of software's information domain and complexity.

Example of LOC based metrics : Refer Q.32

Example of function point based metrics : Refer Q.34

Q.34 How do you calculate FP and how it is used in estimation of a software project ?

☞ [SPPU : May-11, Marks 6,May-13, Dec.-13, Marks 8, Dec 19, Marks 5]

Ans. :

- The function point model is based on functionality of the delivered application.

- These are generally independent of the programming language used.
- Function points are derived using :
  - Countable measures of the software requirements domain
  - Assessments of the software complexity.

How to calculate function point ?

- The data for following **information domain characteristics** are collected :
  1. Number of user inputs - Each user input which provides distinct application data to the software is counted.
  2. Number of user outputs - Each user output that provides application data to the user is counted, e.g. screens, reports, error messages.
  3. Number of user inquiries - An on-line input that results in the generation of some immediate software response in the form of an output.
  4. Number of files - Each logical master file, i.e. a logical grouping of data that may be part of a database or a separate file.
  5. Number of external interfaces - All machine - readable interfaces that are used to transmit information to another system are counted.
- The organization needs to develop criteria which determine whether a particular entry is **simple, average or complex**. The weighting factors should be determined by observations or by experiments.

Domain characteristics	Count	Weighting factor			Count
		Simple	Average	Complex	
Number of user input		3	4	6	
Number of user output		4	5	7	
Number of user inquiries		3	4	6	
Number of files		7	10	15	
Number of external interfaces		5	7	10	
Count total					

- The count table can be computed with the help of above given table.

Example

FP focuses on information domain values rather than software functions. Thus we create a function point calculation table for ABC project, discussed in Q.32.

Info domain value	Opt.	Most likely	Pessimistic	Esti. value	Weight factor	FP
No.of inputs	25	28	32	28.1	4	112
No. of outputs	14	17	20	17	5	85
No. of inquries	17	23	30	23.1	5	116
No. of files	5	5	7	5.33	10	53

No. of external interfaces	2	2	3	2	7	15
Count total						**381**

- For this example we assume average complexity weighting factor.
- Each of the complexity weighting factor is estimated and the complexity adjustment factor is computed using the complexity factor table below. (Based on the 14 questions)

Sr. No.	Factor	Value (Fi).
1.	Back-up and recovery ?	4
2.	Data communication ?	2
3.	Distributed processing ?	0
4.	Performance critical ?	4
5.	Existing operational environment ?	3
6.	On-line data entry ?	4
7.	Input transactions over multiple screens?	5
8.	Online updates ?	3
9.	Information domain values complex ?	5
10	Internal processing complex?	5
11.	Code designed for reuse?	4
12.	Conversion / installation in design?	3
13.	Multiple installations ?	5
14.	Application designed for change ?	5

$\Sigma (F_i) \rightarrow 52$

The estimated number of adjusted FP is derived using the following formula : -

FP ESTIMATED = (FP COUNT TOTAL * [COMPLEXITY ADJUSTMENT FACTOR]

FP ESTIMATED = COUNT TOTAL* [0.65 + (0.01* ? $\Sigma(F_i)$)]

Complexity adjustment factor= [0.65 + (0.01 * 52)] = 1.17

FP ESTIMATED = (381 * 1.17) = 446

(Function point count adjusted with complexity adjustment factor)

- A review of historical data indicates -
  1. Average productivity is 6.5 FP/Person month
  2. Average labor cost is $6000 per month
- Calculations for cost per function point, total estimated project cost and total effort
  1. The cost per function point = (6000 / 6.5) = $923
  2. Total estimated project cost = (446 * 923) = $411658
  3. Total estimated effort = (446 / 6.5) = 69 Person-month.

Q.35 Explain information domain values (any 4).

☞ [SPPU : June-22, Marks 8]

Ans. : Refer Q.34.

Part III : Project Estimation

4.13 : Software Project Estimation

Q.36 What is the need of project estimation ? What are the steps of software estimation ? ☞ [SPPU : Dec 19, 22, Marks 6]

Ans. :

Need :

- The different parameters of a project that need to be estimated include - project size, effort required to complete the project, project duration and cost.

- Accurate estimation of these parameters is important, because it helps in quoting an appropriate project cost to the customer.
- The software estimation helps in understanding the scope of the project.
- The project estimation also forms the basis for resource planning and scheduling.

Steps for software estimation : The four basic steps in software estimation are :

- **Step 1 :** Estimate the size of the development product. This generally ends up in either Lines of Code (LOC) or Function Points (FP), but there are other possible units of measure.

  The size is estimated by comparing it with the existing systems of the same kind. The experts use it to predict the required size of various components of software and then add them to get the total size.
- **Step 2 :** Estimate the effort in person - months or person - hours. Effort estimation is used to help draft project plans and budgets in the early stages of the software development life cycle. This practice enables a project manager to accurately predict costs and allocate resources accordingly.
- **Step 3 :** Estimate the schedule in calendar months. Effective project scheduling leads to success of project, reduced cost and increased customer satisfaction.
- **Step 4 :** Estimate the project cost in dollars or in local currency.

4.14 : Decomposition Techniques

Q.37 What is software project estimation ? How use cases are used in estimation ?

☞ [SPPU : May-11, Marks 6]

Ans. : Software project estimation is a process of predicting the time, effort, cost and scope of the project.

Estimation with use case : Refer Q.31

Q.38 What do you mean by software project estimation ? Explain the process based estimation with an example.

☞ [SPPU : Dec.-12, Marks 8]

Ans. : Software project estimation : Refer Q.37

Process based estimation :

- This is the most commonly used estimation technique for estimating the project based on the processes used.
- In this technique, the process is decomposed into relatively size set of tasks and the effort required to accomplish each task is estimated.
- The estimation begins with identification of software functions obtained from the project scope. Then a series of software process activities must be performed for each function. The function and software process activities are presented in a tabular form. Then planner estimates the efforts for each software function.

Example -

In this technique for each of the business function software engineering tasks such as analysis, design, coding and testing is carried out. The estimated efforts for each of these functions are identified and their sum is obtained. In the following table, the sum of all the rows and columns is taken and the effort for each business function is estimated in percentage.

Activity	CC	Planning	Riskanalysis	Engineering		Construction release		CE	Total
				Analysis	Design	Code	Test		
TASK →									
Business function				0.50	2.75	0.50	4.50	NA	8.25
User interface and control facilities(UICF)				0.75	4.50	0.75	2.00	NA	8.00
2D graphics analysis(2DGA)				0.50	4.50	1.00	3.50	NA	9.5
3D Geometric analysis function(3DGA)				0.50	3.25	1.00	1.00	NA	5.75
Database Management (DBM)				0.50	3.25	0.70	1.50	NA	5.95
Computer graphics display				0.25	2.00	0.50	1.50	NA	4.25
Facility(CGDF)				0.50	2.00	0.50	2.00	NA	5.00
Peripheral control function(PCF)									
Design Analysis Modules (DAM)									
Total	0.25	0.25	0.25	3.50	22.25	4.95	16.00	48	

Activity	CC	Planning	Riskanalysis	Engineering		Construction release		CE	Total
				Analysis	Design	Code	Test		
TASK →									
Percentage effort	1 %	1 %	1 %	7 %	46 %	11 %	33 %		

CC means customer communication CE means customer evaluation

The percentage effort is calculated based on the total 48.

- For instance we get the total for the task **Design as 22.25**
- 22.25*100/48 = 46 % approximately
- Thus the percentage efforts for all the software engineering tasks are computed.
- Based on average labor cost of $6000 per month
  1. Total estimated project Effort = **48 persons-month**
  2. Total estimated project cost=(6000*48)=$288000

4.15 : Cost Estimation Tools and Techniques

Q.39 What are the three types of cost estimates ? Explain.

Ans. : 1. Rough order magnitude

- A Rough Order of Magnitude Estimate (ROM estimate) is an estimation of a project's level of effort and cost to complete.
- A ROM estimate takes place very early in a project's life cycle - during the project selection and approval period and prior to project initiation.
- The main purpose of the ROM estimate is to help in decision making. That is to make a decision on whether it makes sense to move forward with the project using the estimated level of time and cost.

- The timeframe for this type of estimate is often three or more years prior to project completion.
- A ROM estimate's accuracy is typically – 50 percent to + 100 percent. That means, the project's actual costs could be 50 percent below the ROM estimate or 100 percent above the estimate.

2. Budgetary estimate

- When funds are allotted to the project for some specific period, then budgetary estimate is made.
- Budgetary estimates are made one to two years prior to project completion
- The range of variance on a budgetary estimate can be from – 10 % to + 25 %. That means, the actual costs of project could be 10 percent less or 25 percent more than the budgetary estimate.

3. Definitive estimate

- The definitive estimate is the most accurate estimate for the amount of work and resources needed to complete the project.
- The definitive estimates are the estimates that the organization will commit to in order for the project to baseline, tactically manage the project.

4.16 : Typical Problems with IT Cost Estimates

Q.40 Explain typical problems with IT cost estimates.

☞ **[SPPU : Aug.-17, Dec.-22, Marks 5]**

Ans. : Various problems that occur during the cost estimation are as mentioned below :

1. Preparing cost estimates for large and complex project is very difficult and it requires significant amount of effort. For such

projects the estimates are made at various stages of the project. Many times estimation is made too early.

2. Many times people doing the estimation have very little experience of estimating a project. In these situation proper training is required for such people.
3. People are bias towards the underestimation. Hence project cost estimation need to be reviewed to make sure that such estimates are not biased.
4. Management desire for perfect and accurate budget. In this case project managers should take a leading role to develop good cost and schedule estimates. Project managers must negotiate with project sponsors to create realistic cost estimates.

END ...

Unit V

5 Software Quality and Testing

Part I : Quality Concepts

5.1 : Quality

Q.1 Explain the types of quality

Ans. : There are two kinds of quality

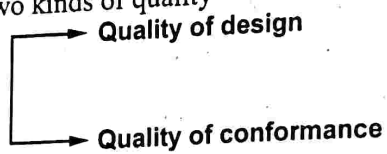


Fig. Q.1.1

- **Quality of design** is the characteristics of the item which is specified for the designer. For example if a temperature control system is designed, then it should display the temperature with Maximum limit of 100 degree centigrade. This is what the basic characteristic of that system is. And at the time of design of the product this issue must be focused.
- **Quality of conformance** is the degree to which the design specifications are followed during manufacturing. If the degree of conformance is **more** then it indicates **higher quality**.
- Thus in software development process **quality of design** is concerned towards requirements, specifications and design of the system and **quality of conformance** is concerned with implementation.

5.2 : Software Quality

Q.2 Discuss Garvin's eight quality dimensions.

☞ [SPPU : June-22, Marks 9, Dec.-22, Marks 8]

Ans. :

1) **Performance quality :** This attribute confirms that the software is comprised of all the contents, functions and features that are mentioned in the requirement model.
2) **Feature quality :** This attribute addresses that the software provides the features that may surprise and delight the end users.
3) **Reliability :** This attribute confirms that the software has the capability to work without failure.
4) **Conformance :** This attribute confirms that the software follows the design and coding standards.
5) **Durability :** The durability indicates that the software can be maintained(changed) or corrected(debugged) without any severe side effects.
6) **Serviceability :** Serviceability means, the software can be maintained or corrected within acceptable short period of time.
7) **Aesthetics :** Aesthetic software systems have some unique characteristics such as elegant look and feel, unique flow, and consistent user Interface.
8) **Perception :** Perception means being prejudice for vendors of the software products. That means if you have a negative impression for some vendor then even when he develops some better software product, the customer have least inclination towards it. On the other hand, if you have some good experience about some vendor about his software products, then still his products are accepted. This indeed influences the quality.

Q.3 List out ISO 9126 quality factors

☞ [SPPU : June-22, Marks 8]

Ans. : An ISO 9126 standard was developed to identify quality attributes for software systems. In this standard, a set of attributes of a software product by which its quality is described and evaluated. These attributes are as follows -

1) **Functionality:** A set of attributes that bear on the existence of a set of functions and their specified properties. The functions are those that satisfy stated or implied needs.
2) **Reliability :** A set of attributes that bear on the capability of software to maintain its level of performance under stated conditions for a stated period of time
3) **Usability :** A set of attributes that bear on the effort needed for use, and on the individual assessment of such use by a stated or implied set of users.
4) **Efficiency :** A set of attributes that bear on the relationship between the level of performance of the software and the amount of resources used, under stated conditions.
5) **Maintainability :** A set of attributes that bear on the effort needed to make specified modified modifications.
6) **Portability :** A set of attributes that bear on the ability of software to be transferred from on environment to another.

5.3 : Quality Metrics

Q.4 Explain in detail defect removal efficiency.

☞ [SPPU : May-14, Marks 4, Dec.-12, Marks 8]

Ans. : While developing the software project many work products such as SRS, design document, source code are being created. Along with these work products many errors may get generated. Project manager has to identify all these errors to bring quality software.

- Error tracking is a process of assessing the status of the software project.
- The software team performs the formal technical reviews to test the software developed. In this review various errors are identified and corrected. Any errors that remain uncovered and are found in later tasks are called defects.
- The defect removal efficiency can be defined as,

$$DRE = E/(E+D)$$

Where DRE is the defect removal efficiency,

E is the error

and D is defect.

- The DRE represents the effectiveness of quality assurance activities. The DRE also helps the project manager to assess the progress of software project as it gets developed through its scheduled work task.
- During error tracking activity following metrics are computed :
  1. Errors per requirements specification page : denoted by E_{req}
  2. Errors per component - design level : denoted by E_{design}
  3. Errors per component - code level : denoted by E_{code}
  4. DRE - requirement analysis
  5. DRE - architectural design
  6. DRE - component level design
  7. - coding

  The project manager calculates current values for E_{req}, E_{design}, and E_{code}.

  These values are then compared with past projects. If the current result differs more than 20 % from the average, then there may be cause for concern and investigation needs to be made in this regard.

- These error tracking metrics can also be used for better target review and testing resources.

Q.5 Explain any four quality measure of software. [SPPU : Dec.-13, Marks 8]

Ans. : Following are the measure of the software quality -

1. **Correctness :** Correctness is a degree to which the software produces the desired functionality. The correctness can be measured as,

   Correctness = Defects per KLOC

   Where defect means lack of conformance to requirements. Such defects are generally reported by the user of the program.

   Thus it is expected that the program must work correctly otherwise it is of no use.

2. **Integrity :** Integrity is basically an ability of the system to withstand against the attacks. Typically attacks are on programs, data and documents. There are two attributes that are associated with integrity: threat and security.

   Threat is the probability that specific type of attacks may occur. And **security** is the probability that the system will repel against the specific attack. Hence integrity can be measured as :

   $$\text{Integrity} = \sum((1 - \text{threat}) \times (1 - \text{security}))$$

   System integrity is now a days has got great importance.

3. **Usability :** Usability means **user friendliness** of the system or ability of the system that indicates the usefulness of the system. Following are the characteristics that are useful for measuring the usability. The user friendliness is measured using following four characteristics -
   - The time required to make the system efficient.
   - The skill required to learn the system

   - The net increase in productivity after regular use of the system.
   - The user attitude towards the system.

4. **Maintainability :**
   - Maintainability is an ability of the system to accommodate the corrections made after encountering errors, adapting the environment changes and adapt the changes made in the system in order to satisfy the user.
   - The metric used for maintainability is MTTC i.e. mean time to change.
   - The MTTC can be defined as the time required to analyze the change request, design an appropriate modification and implementationof those desired changes.
   - Lower the value of MTTC means the software is more maintainable.
   - Thus after sufficient understanding of software metrics and measurement we can look for managing the risks that arise during the software development activities.

Q.6 How do you measure software quality in terms of maintainability and integrity ? [SPPU : May-14, Marks 6]

Ans. : Refer Q.5.

5.4 : Software Quality Dilemma

Q.7 Explain the cost of software quality along with its components

Ans. : Definition : The cost of quality can be **defined** as the total cost required to obtain the quality in the product and to conduct the quality related activities.

- The cost of quality has various components such as :
  1. **Prevention cost -** This is the cost of quality required for conducting quality planning, formal technical reviews, test equipments and training.
  2. **Appraisal cost -** This is the cost of quality required for gaining the insight into the product. It includes the cost required for in-process and inter process inspection, maintenance and testing.

3. **Failure cost** - Failure cost means the cost required to remove the defects in the software product before delivering it to customer. There are two types of failure costs.

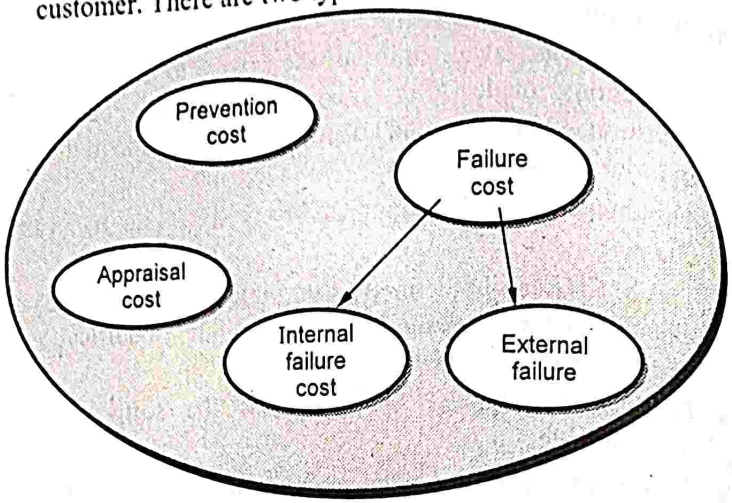


Fig. Q.7.1 Cost of quality

a. **Internal failure cost** - Internal failure is nothing but the cost of defects occurred in the product before delivering it to customer e.g. repair in networking, repairing of communication network.

b. **External failure cost** - External failure is the cost of defects occurred in the product after delivering it to the customer e.g. product repair/replace, complaint processing, warranty work.

5.5 : Achieving Software Quality

Q.8 Explain in detail the software quality control and software quality assurance

Ans. : Quality Control

- **Definition** : Quality control is a process in which activities are conducted in order to maintain the quality of product. These activities are series of inspections, reviews and tests used throughout the software process. These activities ensure whether each work product is satisfying the requirements imposed on it.

- While applying the quality control there should be a feedback loop to the process which generates the work product. With the help of such feedback we can tune the process if it does not satisfy the requirements. The feedback loop helps in minimizing the defects in the software product.
- The quality control activities can be fully automated or it can be completely manual or it can be a combination of automated tools and manual procedures.

Quality assurance

- **Definition of quality assurance** : It is planned and systematic pattern of activities necessary to provide a high degree of confidence in the quality of a product. It provides quality assessment of the quality control activities and determines the validity of the data or procedures for determining quality.
- The quality assurance consists of set of reporting and auditing functions.
- These functions are useful for assessing and controlling the effectiveness and completeness of quality control activities.
- The goal of quality assurance is to ensure the management of data which is important for product quality.

Q.9 Give the difference between quality assurance and quality control

Ans. :

Sr. No.	Quality assurance	Quality control
1.	It is **preventive technique.**	It is **corrective technique.**
2.	It is **proactive measure.**	It is **reactive measure.**
3.	It involves **full software development life** cycle.	It involves **testing phase** of software development cycle.

4.	It does not involve executing the program.	It always involves executing a program.
5.	In this technique deliverables are created.	In this technique the deliverables are verified.
6.	This technique is for managing the quality verification.	This technique is for managing the quality validation.

Part II : Software Testing

5.6 : Introduction to Software Testing

Q.10 Define the term - software testing

Ans. : Definition : **Software testing** is a critical element of software quality assurance and represents the ultimate review of specification, design and coding.

- The purpose of software testing is to ensure whether the software functions appear to be working according to specifications and performance requirements.

Q.11 What are objectives of testing ?

Ans. : According to Glen Myers the testing objectives are

1. Testing is a process of executing a program with the intend of finding an error.
2. A good test case is one that has high probability of finding an undiscovered error.
3. A successful test is one that uncovers an as-yet undiscovered error.

- The major testing objective is to design tests that systematically uncover types of errors with minimum time and effort.

5.7 : Principles of Testing

Q.12 Enumerate seven principles of testing.

☞ [SPPU : June-22, Marks 9]

Ans. :

- **Testing shows the presence of defects :** Software testing reduces the presence of defects. Testing can reduce the number of defects but not remove all defects.
- **Exhaustive testing is not possible :** It is the process of testing the functionality of the software in all possible inputs (valid or invalid) and pre-conditions is known as exhaustive testing. Exhaustive testing is impossible because the software can never test at every test case. If the software will test every test case then it will take more cost, effort, etc., which is impractical.
- **Early testing :** To find the defect in the software, early test activity shall be started. The defect detected in the early phases of SDLC will be very less expensive. For better performance of software, software testing will start at the initial phase.
- **Defect clustering :** In a project, a small number of modules can contain most of the defects. Pareto principle to software testing state that 80 % of software defect comes from 20 % of modules.
- **Pesticide paradox :** Repeating the same test cases, again and again, will not find new bugs. So it is necessary to review the test cases and add or update test cases to find new bugs.
- **Testing is context - dependent :** Different types of software need to perform different types of testing. For example - The testing of the e-commerce site is different from the testing of the computer game application.
- **Absence of errors fallacy :** If a built software is 99 % bug-free but it does not follow the user requirement then it is unusable. It is not

only necessary that software is 99 % bug-free but it is also mandatory to fulfill all the customer requirements.

Q.13 What are the main objectives of software testing and what are the principles of software testing ?
☞ [SPPU : Dec.-15, Marks 9, Dec.-19, Marks 5]

Ans. : Objectives : Refer Q.11.

Principles of testing : Refer Q.12.

Q.14 What is importance of testing practices ? What are the principles of testing practices ? ☞ [SPPU : May-14, Marks 8]

Ans. : Importance :

- Generally, testing is a process that requires more efforts than any other software engineering activity.
- Testing is a set of activities that can be planned in advance and conducted systematically.
- If it is conducted haphazardly, then only time will be wasted and more even worse errors may get introduced.
- This may lead to have many undetected errors in the system being developed. Hence performing testing by adopting systematic strategies is very much essential in during development of software.

Principles : Refer Q.12.

5.8 : Test Plan

Q.15 Explain the standard template for test plan

Ans. : Standard template for test plan

1. Introduction
   1.1 Scope
2. References
3. Test methodology and Strategy/Approach

4. Test criteria
   4.1 Entry criteria
   4.2 Exit criteria
   4.3 Suspension criteria
   4.4 Resumption criteria
5. Assumptions, dependencies and risks
   5.1 Assumption
   5.2 Dependencies
   5.3 Risk and risk management plans
6. Estimations
   6.1 Size estimate
   6.2 Effort estimate
   6.3 Schedule estimate
7. Test deliverables and milestones
8. Responsibilities
9. Resource requirement
   9.1 Hardware resources
   9.2 Software resources
   9.3 People resources
   9.4 Other resources
10. Training requirements
11. Defect logging and tracking process
12. Metrics plan
13. Product release criteria

5.9 : Test Case

Q.16 Explain the test case with example

Ans. : Definition : Test case be defined as series of steps executed on product using predefined set of input data, expecting to produce predefined set of outputs in the given environment.

- For creating the test case following things need to be identified.
  (1) The purpose of test : The intention for which the test is performed is specified.
  (2) Items to be tested.
  (3) Software and hardware environment setup.
  (4) Input data to be used for test case.
  (5) Steps to be followed to execute test.
  (6) Expected results that are considered to be correct result.
  (7) Actual result produced after performing tests.
  (8) Relationship of current test with other tests.

Example - Refer Q.17.

Q.17 Write important six test cases for the 'Login Form' of the facebook website.

Ans. :

Test case ID	Description	Test data	Expected result	Actual result	Status
TC01	Invalid username is entered.	Enter xyz as username.	It will prompt 'couldn't find your account' message.	It prompts couldn't find your account.	Pass
TC02	Valid username and invalid password is entered.	Enter 1111 as password.	It will display 'Wrong password' message.	It displays 'Wrong password' message.	Pass
TC03	Username field is left blank.	No data in user name field.	It will display 'enter username'.	It displays 'enter username'.	Pass

TC04	Valid username and no password is entered.	No data in password field.	It will display 'enter password'.	It displays 'enter password'.	Pass
TC05	Both user name and password field is left blank.	No data in user name and password field.	It will display'enter username /password'.	It displays 'enter username /password'.	Pass
TC06	Valid user name and password is entered.	User name = abc and password = abc1234.	It will display your account's facebook page.	It displays your account's faceb ook page.	Pass

5.10 : Types of Testing

Q.18 Explain black box testing and white box testing

Ans. : There are two general approaches for the software testing.

1. **Black box testing :** The black box testing is used to demonstrate that the software functions are operational. As the name suggests in black box testing it is tested whether the input is accepted properly and output is correctly produced.

   The major focus of black box testing is on functions, operations, external interfaces, external data and information.

2. **White box testing :** In white box testing the procedural details are closely examined. In this testing the internals of software are tested to make sure that they operate according to specifications and designs. Thus major focus of white box testing is on internal structures, logic paths, control flows, data flows, internal data structures, conditions, loops, etc.

Q.19 What do you mean by white box testing ? ☞ [SPPU : Dec.-16, Marks 7]

Ans. : Refer Q.18.

Q.20 What is basis path testing ? What is cyclomatic complexity ? How is it determined for a flow graph ? Illustrate with an example. ☞ [SPPU : May-12, Marks 8]

OR Basis path testing covers all statements in a program module. Justify with example. ☞ [SPPU : Dec.-12, Marks 8]

Ans. : Path testing is a structural testing strategy. This method is intended to exercise every independent execution path of a program atleast once.

Following are the steps that are carried out while performing path testing.

- **Step 1 :** Design the flow graph for the program or a component.
- **Step 2 :** Calculate the cyclomatic complexity.
- **Step 3 :** Select a basis set of path.
- **Step 4 :** Generate test cases for these paths.

Let us discuss each in detail.

- **Step 1 :** Design the flow graph for the program or a component.

  Flow graph is a graphical representation of logical control flow of the program. Such a graph consists of circle called a **flow graph node** which basically represents one or more procedural statements and arrow called as **edges** or **links** which basically represent control flow. In this flow graph the areas bounded by nodes and edges are called **regions**. Various notations used in flow graph are :

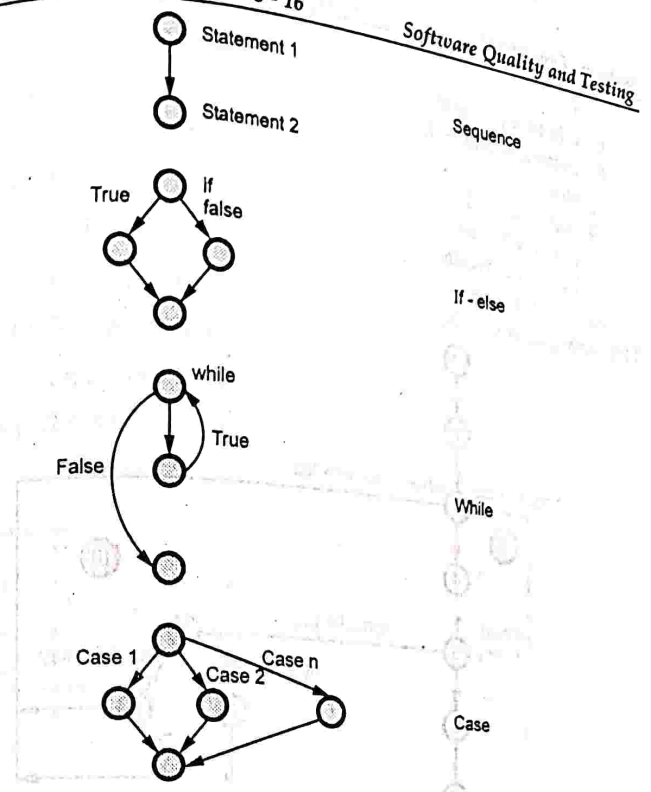


Fig. Q.20.1

For example : Following program is for searching a number using binary search method. Draw a flow graph for the same.

```
void search (int key, int n, int a [ ])
  {
    int mid;
1)  int bottom = 0;
2)  int top = n – 1;
3)  while (bottom <=top)
4)  { mid = (top + bottom) / 2;
5)  if (a [mid] = = key)
    {
6)  printf ("Element is present");
7)  return;
    } // end of if
    else
```

```
    {
8)  if (a [mid] < key)
9)  bottom = mid + 1;
    else
10) top = mid - 1;
    } // end of else
    } // end of while
11) } // end of search
```

The flow graph will be

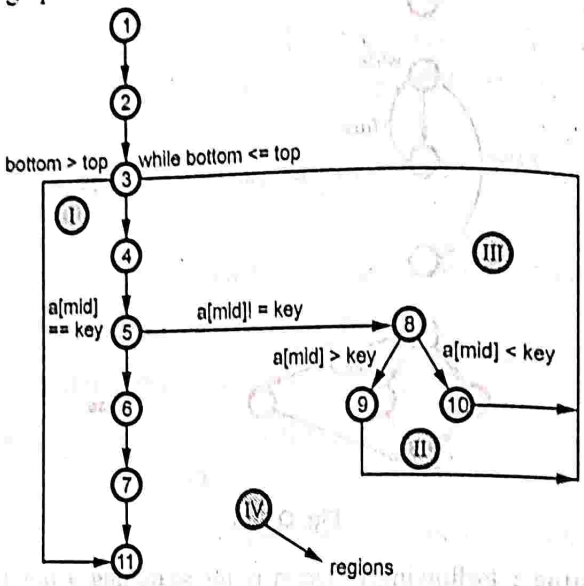


Fig. Q.20.2

- **Step 2 :** Calculate the cyclomatic complexity.

The cyclomatic complexity can be computed by three ways.

1) Cyclomatic complexity = Total number of regions in the flow graph = 4 (note that in above flow graph regions are given by shaded roman letters).
2) Cyclomatic complexity = E – N + 2 = 13 edges – 11 nodes + 2
   = 2 + 2 = 4
3) Cyclomatic complexity = P + 1 = 3 + 1 = 4. There are 3 predicate (decision making) nodes : Nodes 3, 5 and 8.

- **Step 3 :** Select a basis set of path

The basis paths are

Path 1 : 1, 2, 3, 4, 5, 6, 7, 11

Path 2 : 1, 2, 3, 11

Path 3 : 1, 2, 3, 4, 5, 8, 9, 3 ...

Path 4 : 1, 2, 3, 4, 5, 8, 10, 3 ...

- **Step 4 :** Generate test cases for these paths.

After computing cyclomatic complexity and finding independent basis paths, the test cases has to be executed for these paths. The format for test case is -

Preconditions :

Test case id	Test case name	Test case description	Test steps			Test case status (Pass /Fail)	Test priority	Defect severity
			Step	Expected	Actual			
1								
2								
:								
n								

The test case for binary search can be written as -

Precondition : There should be list of elements arranged in ascending order. The element to be searched from the list, its value should be entered and will be stored in variable 'key'.

Test case id	Test case name	Test case description	Test steps			Test case status	Test status (P/F)	Test priority	Defect severity
			Step	Expected	Actual				
1.	Validating the list boundary	Checking the bottom and top	Set bottom	Initially bottom < = top		Design			

		values for the list of elements	= 0 top = n – 1 check if bottom <= top by while loop. This condition defines the length of the list from which the key is searched.	will be true. But during iterations list's length will be reduced and if entire list gets scanned at one point (bottom >= top) will be reached then return to main.					
2.	Checking list element with key.	Checking if middle element of array is equal to key value	Set mid = (top + bottom) / 2Then compare if a[mid] is equal to key.	If a[mid] = key value then print message "Element is present" and return to main.		Design			
		if a[mid] is < key value if a[mid] is > key value	Set bottom = mid + 1 and then	The right sublist will be searched.		Design			

			goto "while" set top = mid +1 and go back to "while" loop	The left sublist will be searched for key element					

Q.21 Draw the flow graph for finding maximum of three numbers and derive the testcases using cyclomatic complexity.

☞ [SPPU : Dec.-11, Marks 8]

Ans. : The flow graph for given program is - (From Fig. Q.21.1)

Cyclomatic complexity = E – N + 2 = 7 – 6 + 2 = 3

Cyclomatic complexity = P + 1 = 2 + 1 = 3

Cyclomatic complexity = Regions encountered = 3

Hence cyclomatic complexity of given program is 3.

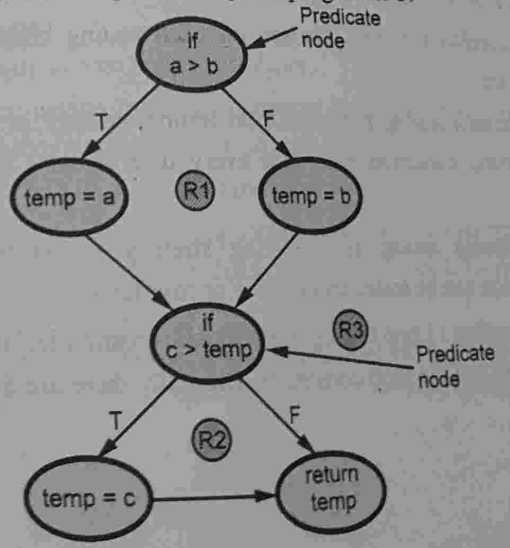


Fig. Q.21.1 Flow graph for finding maximum of three numbers

Q.22 What is cyclomatic complexity ? How is it determined for a flow graph ? Illustrate with example. [SPPU : May-14, Marks 8, May 19, Dec.-19, Marks 5]

Ans. : Refer Q.21.

Q.23 Explain the following : i) Condition testing ii) Loop testing. [SPPU : May-11, Marks 8]

Ans. : (i) Condition testing :

- To test the logical conditions in the program module the condition testing is used. This condition can be a Boolean condition or a relational expression.
- The condition is incorrect in following situations.
  - i) Boolean operator is incorrect, missing or extra.
  - ii) Boolean variable is incorrect.
  - iii) Boolean parenthesis may be missing, incorrect or extra.
  - iv) Error in relational operator.
  - v) Error in arithmetic expression.
- The *condition testing* focuses on each testing condition in the program.
- The *branch testing* is a condition testing strategy in which for a compound condition each and every true or false branches are tested.
- The *domain testing* is a testing strategy in which relational expression can be tested using three or four tests.

(ii) Loop testing : Loop testing is a white box testing technique which is used to test the loop constructs. Basically there are four types of loops.

1. **Simple loops :** The tests can be performed for n number of classes.

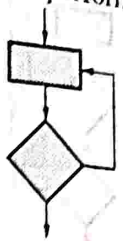


Fig. Q.23.1 Simple loop

where

i) $n = 0$ that means skip the loop completely.

ii) $n = 1$ that means one pass through the loop is tested.

iii) $n = 2$ that means two passes through the loop is tested.

iv) $n = m$ that means testing is done when there are m passes where $m < n$.

v) Perform the testing when number of passes are $n - 1$, n, $n + 1$.

2. **Nested loops :**

The nested loop can be tested as follows.

i) Testing begins from the innermost loop first. At the same time set all the other loops to minimum values.

ii) The simple loop test for innermost loop is done.

iii) Conduct the loop testing for the next loop by keeping the outer loops at minimum values and other nested loops at some specified value.

iv) This testing process is continued until all the loops have been tested.

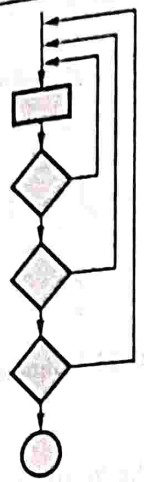


Fig. Q.23.2 Nested loops

3. **Concatenated loops :**

The concatenated loops can be tested in the same manner as simple loop tests. (Refer Fig. Q.23.3)

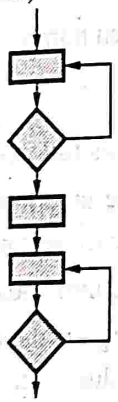


Fig. Q.23.3 Concatenated loops

4. **Unstructured loops :** The testing cannot be effectively conducted for unstructured loops. Hence these types of loops needs to be redesigned. (Refer Fig. Q.23.4)

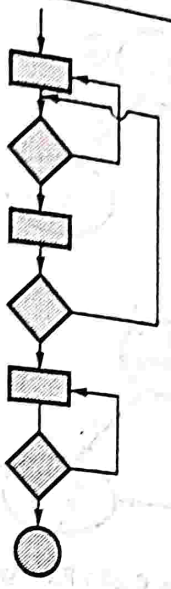


Fig. Q.23.4 Unstructured loops

Q.24 Explain the graph matrix and loop testing methods.

☞ [SPPU : May-13, Marks 8]

Ans. : Graph matrix method :

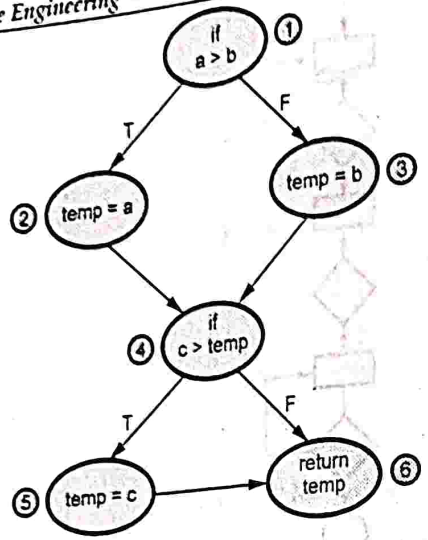
Definition : Graph matrix is a square matrix whose size is equal to number of nodes of the flow graph.

For example consider a flow graph -

The graph matrix will be

For computing the cyclomatic complexity. Following steps are adopted -

- **Step 1 :** Create a graph matrix. Mark the corresponding entry as 1 if node A is connected to node B.
- **Step 2 :** Count total number of 1's from each row and subtract 1 from each corresponding row.



The nodes are numbered it is shown in **shaded form.**

Fig. Q.24.1 Flow graph

- **Step 3 :** Add cyclometic complexity.

	1	2	3	4	5	6	
1		1	1				2 – 1 = 1
2				1			1 – 1 = 0
3				1			1 – 1 = 0
4					1	1	2 – 1 = 1
5						1	1 – 1 = 0
6							2 + 1 = 3

the results of each row and add 1 to it. The resultant value is the cyclomatic complexity.

Loop testing method : Refer Q.23.

Q.25 Explain equivalent partitioning testing.

☞ **[SPPU : Dec.-11, Marks 6]**

Ans. : It is a black box technique that divides the input domain into classes of data. From this data test cases can be derived.

- An ideal test case uncovers a class of errors that might require many arbitrary test cases to be executed before a general error is observed.

- In equivalence partitioning the equivalence classes are evaluated for given input condition. Equivalence class represents a set of valid or invalid states for input conditions.
- Equivalence class guidelines can be as given below :
  - If input condition specifies a range, one valid and two invalid equivalence classes are defined.
  - If an input condition requires a specific value, one valid and two invalid equivalence classes are defined.
  - If an input condition specifies a member of a set, one valid and one invalid equivalence class is defined.
  - If an input condition is Boolean, one valid and one invalid equivalence class is defined.

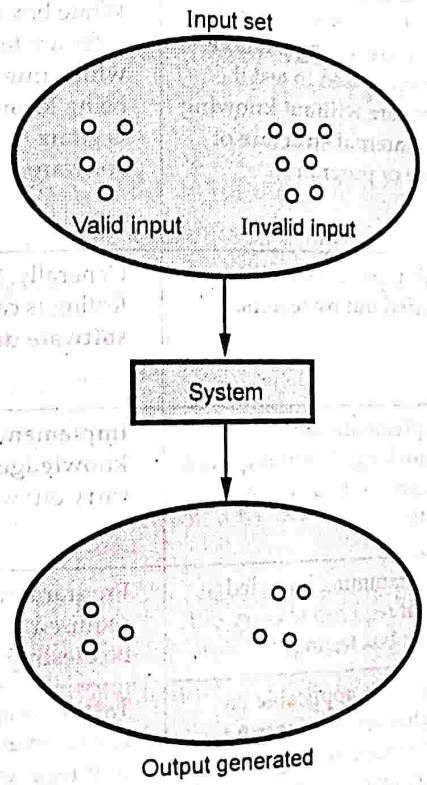


Fig. Q.25.1

For example :

Area code : Input condition, Boolean - The area code may or may not be present.

Input condition, range - Value defined between 200 and 700.

Password : Input condition, Boolean - A password may or may not be present.

Input condition, value - Seven character string.

Command : Input condition, set - Containing commands noted before.

Q.26 Differentiate between white box and black box testing.

☞ **[SPPU : Dec.-12, Marks 4]**

Ans. :

Sr. No.	Black box testing	White box testing
1.	Black box testing is the software testing method which is used to test the software **without knowing the internal structure** of code or program.	White box testing is the software testing method in which **internal structure** is being known to tester who is going to test the software.
2.	This type of testing is **carried out by testers.**	Generally, this type of testing is carried out by **software developers.**
3.	**Implementation knowledge is not required** to carry out black box testing.	**Implementation knowledge is required** to carry out white box testing.
4.	Programming knowledge is not required to carry out black box testing.	Programming knowledge is required to carry out white box testing
5.	Testing is applicable on **higher levels of testing** like system testing, acceptance testing.	Testing is applicable on **lower level of testing** like unit testing, integration testing

6.	Black box testing means functional test or **external testing.**	White box testing means structural test or **interior testing.**
7.	Black box testing can be started based on **requirement specifications documents.**	White box testing can be started based on **detail design documents.**
8.	The functional testing, behavior testing, **close box testing** is carried out under black box testing.	The structural testing, logic testing, path testing, loop testing, code coverage testing, open box testing is carried out under white box testing.

5.11 : Verification and Validation

Q.27 Explain in brief software verification and validation.

☞ **[SPPU : Oct.-18, Marks 5]**

Ans. :

- Verification refers to the set of activities that ensure that software correctly implements a specific function.
- Validation refers to a different set of activities that ensure that the software that has been built is traceable to customer requirements.
- According to Boehm
  - Verification : "Are we building the product right ?"
  - Validation : "Are we building the right product ?"
- Software testing is only one element of Software Quality Assurance (SQA).
- Quality must be built into the development process, you can't use testing to add quality after the fact.
- Verification and validation involve large number of software quality assurance activities such as -
  - Formal technical reviews
  - Quality and configuration audits

- Performance monitoring
- Feasibility study
- Documentation review
- Database review
- Algorithmic analysis
- Development testing
- Installation testing

Q.28 Differentiate between : Verification and validation.

☞ [SPPU : Dec.-15, Marks 4]

Ans. :

Sr. No.	Verification	Validation
1.	Verification refers to the set of activities that ensure software correctly implements the specific function.	Validation refers to the set of activities that ensure that the software that has been built is traceable to customer requirements.
2.	After a valid and complete specification the verification starts.	Validation begins as soon as project starts.
3.	Verification is for prevention of errors.	Validation is for detection of errors.
4.	Verification is conducted using reviews, walkthroughs, inspections and audits.	Validation is conducted using system testing, user interface testing and stress testing.
5.	Verification is also termed as white box testing or static testing as work product goes through reviews.	Validation can be termed as black box testing or dynamic testing as work product is executed.

6.	Verification finds about 50 to 60 % of the defects.	Validation finds about 20 to 30 % of the defects.
7.	Verification is based on the opinion of reviewer and may change from person to person.	Validation is based on the fact and is often stable.
8.	The verification verifies the problem statements, decisions taken during the development and execution paths.	The validation validates the requirements, functionalities and features of the product.
9.	Verification is about process, standard and guideline.	Validation is about the product.

5.12 : Testing Strategies

Q.29 What is unit testing ? Explain the unit testing process.

☞ [SPPU : Dec.-12, 22, Marks 8]

Ans. :

- In unit testing the individual components are tested independently to ensure their quality.
- The focus is to uncover the errors in design and implementation.
- The various tests that are conducted during the unit test are described as below.
  1. Module interfaces are tested for proper information flow in and out of the program.
  2. Local data are examined to ensure that integrity is maintained.
  3. Boundary conditions are tested to ensure that the module operates properly at boundaries established to limit or restrict processing.

4. All the basis (independent) paths are tested for ensuring that all statements in the module have been executed only once.
5. All error handling paths should be tested.

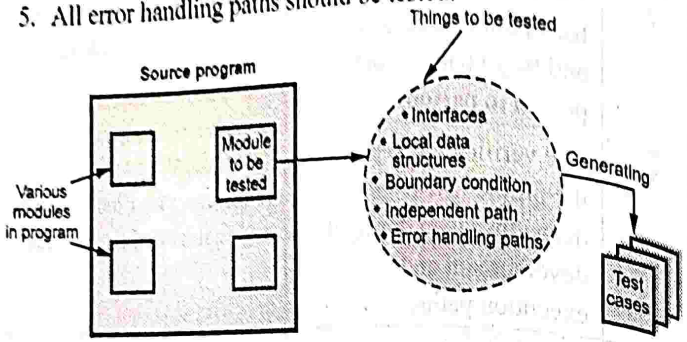


Fig. Q.29.1 Unit testing

6. Drivers and stub software need to be developed to test incomplete software. The "driver" is a program that accepts the test data and prints the relevant results. And the "stub" is a subprogram that uses the module interfaces and performs the minimal data manipulation if required. This is illustrated by following Fig. Q.29.2.
7. The unit testing is simplified when a component with high cohesion (with one function) is designed. In such a design the number of test cases are less and one can easily predict or uncover errors.

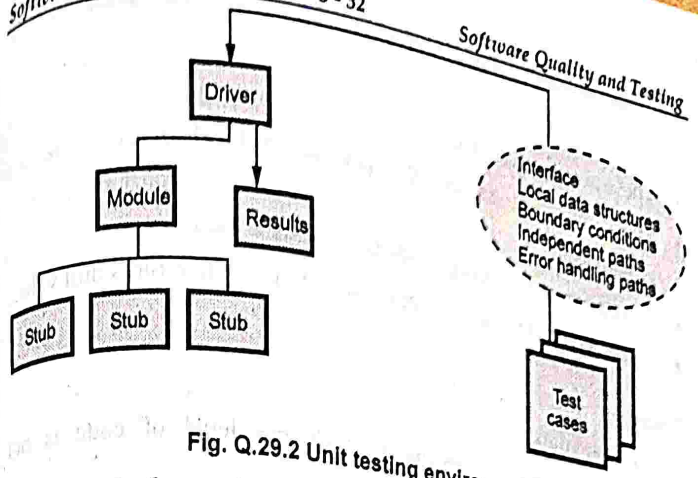


Fig. Q.29.2 Unit testing environment

Q.30 What is the need of stubs and drivers in software testing ?

☞ [SPPU : Dec.-13, Marks 4, Dec.-19, Marks 5]

Ans. :

- Stubs and drivers are two such elements used in software testing process, which act as a temporary replacement for a module.
- The driver and stubs are generally dummy codes or fake codes that help to simulate the behavior of existing code.
- The stubs and drives are specifically developed to meet the necessary requirements of the unavailable modules and are useful in getting expected results.
- Generally driver and stubs are used in unit testing or during integration testing.

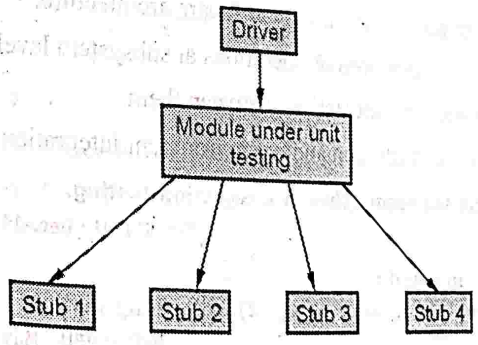


Fig. Q.30.1 Driver and stub

Stub

- These are dummy code.
- These are used in testing when lower level of code are not developed.
- To test the upper level of code the stubs are used.
- It just accepts the value from calling module and returns null value.

Driver

- These are dummy code.
- These are used in testing when upper level of code is not developed.
- To test the lower level code the drivers are used.
- Basically we call the driver as main function which calls other modules to form complete applications.

Q.31 What do you understand by integration testing ? Explain objectives of integration testing. ☞ **[SPPU : Dec.-19, Marks 6]**

Ans. :

- A group of dependent components are tested together to ensure their quality of their integration unit.
- The objective is to take unit tested components and build a program structure that has been dictated by software design.
- The focus of integration testing is to uncover errors in :
  - Design and construction of software architecture.
  - Integrated functions or operations at subsystem level.
  - Interfaces and interactions between them.
  - Resource integration and/or environment integration.

Q.32 Explain the approaches in integration testing.

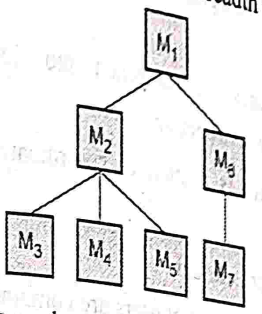
☞ **[SPPU : Dec.-11, 13,Marks 8]**

OR Explain in detail :
1) Top-down integration testing 2) Bottom-up testing.

☞ **[SPPU : May-14, Marks 8]**

Ans. : Top down Integration testing

- Top down testing is an incremental approach in which modules are integrated by moving down through the control structure.
- Modules subordinate to the main control module are incorporated into the system in either a depth first or breadth first manner.



- Integration process can be performed using following steps.

1. The main control module is used as a test driver and the stubs are substituted for all modules directly subordinate to the main control module.
2. Subordinate stubs are replaced one at a time with actual modules using either depth first or breadth first method.
3. Tests are conducted as each module is integrated.
4. On completion of each set of tests, another stub is replaced with the real module.
5. Regression testing is conducted to prevent the introduction of new errors.

For example :

- In top down integration if the depth first approach is adopted then we will start integration from module M1 then we will integrate M2 then M3, M4, M5, M6 and then M7.
- If breadth first approach is adopted then we will integrate module M1 first then M2, M6. Then we will integrate module M3, M4, M5 and finally M7.

2. **Bottom Up Integration testing**

- In bottom up integration the modules at the lowest levels are integrated at first, then integration is done by moving upward through the control structure.
- The bottom up integration process can be carried out using following steps.

1. Low-level modules are combined into clusters that perform a specific software subfunction.
2. A driver program is written to co-ordinate test case input and output.
3. The whole cluster is tested.
4. Drivers are removed and clusters are combined moving upward in the program structure.

For example :

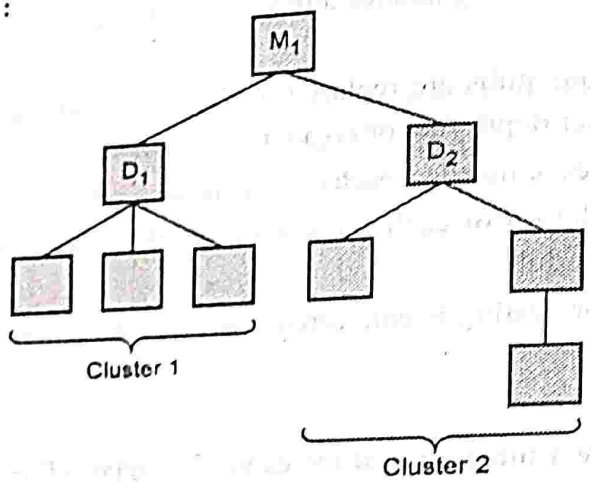


Fig. Q.32.1 Bottom up integration testing

- First components are collected together to form cluster 1 and cluster 2. Then each cluster is tested using a driver program. The clusters subordinate the driver module. After testing the driver is removed and clusters are directly interfaced to the modules.

Q.33 What is system testing ? Explain any two system testing strategies. [SPPU : May-11, Marks 8, Dec.-12, Marks 6]

Ans. : System testing : The system test is a series of tests conducted to fully the computer based system. The main focus of such testing is to test

1) System functions and performance.
2) System reliability and recoverability (recovery test).
3) System installation (installation test).
4) System behaviour in the special conditions (stress test).
5) System user operations (acceptance test / alpha test).
6) Hardware and software integration and collaboration.
7) Integration of external software and the system.

Types of system testing :

1) Recovery testing
   - Recovery testing is intended to check the system's ability to recover from failures.
   - In this type of testing the software is forced to fail and then it is verified whether the system recovers properly or not.
   - For automated recovery then reinitialization, checkpoint mechanisms, data recovery and restart are verified.

2) **Security testing**
   - Security testing verifies that system protection mechanism prevent improper penetration or data alteration.
   - It also verifies that protection mechanisms built into the system prevent intrusion such as unauthorized internal or external access or willful damage.
   - System design goal is to make the penetration attempt more costly than the value of the information that will be obtained.

Q.34 What do you mean by acceptance testing ?

☞ [SPPU : May-13, Marks 4]

Ans. : • Acceptance testing is a testing phase which is conducted normally after system testing by the customer or representative of customer.

- The **goal of acceptance testing** is to establish the confidence in the system.
- There two ways by which the acceptance testing is conducted, these are - 1. Alpha testing and 2. Beta testing.

Q.35 Differentiate between alpha and beta testing.

☞ [SPPU : Dec.-11, Marks 4, May 19, Dec 19, Marks 6]

Ans. :

Sr. No.	Alpha testing	Beta testing
1.	Performed at developer's site.	Performed at end user's site.
2.	Performed in controlled environment as developer is present.	Performed in uncontrolled environment as developer is not present.
3.	Less probability of finding of error as it is driven by developers.	High probability of finding of error as end user can use it the way he wants.
4.	It is not considered as live application	It is considered as live application.
5.	It is done during implementation phase of software.	It is done as pre-release of software.
6.	Less time consuming as developer can make necessary changes in given time.	More time consuming. As user has to report bugs if any via appropriate channel.

Q.36 Explain following testing types :
1) Validation testing 2) Acceptance testing
3) Smoke testing process.

☞ [SPPU : May-14, Marks 8]

Ans. : 1) Validation testing : The integrated software is tested based on requirements to ensure that the desired product is obtained.

- In validation testing the main focus is to uncover errors in
  - System input/output
  - System functions and information data
  - System interfaces with external parts
  - User interfaces
  - System behaviour and performance
- Software validation can be performed through a series of **black box tests.**
- After performing the validation tests there exists two conditions.
  1. The function or performance characteristics are **according** to the **specifications** and are accepted.
  2. The requirement specifications are derived and the **deficiency** list is created. The deficiencies then can be **resolved** by establishing the proper communication with the customer.
- Finally in validation testing a review is taken to ensure that all the elements of software configuration are developed as per requirements. This review is called configuration review or audit.

Acceptance testing : Refer Q.34.

Smoke testing :

The smoke testing is a kind of integration testing technique used for time critical projects wherein the project needs to be assessed on frequent basis.

Following activities need to be carried out in smoke testing -

- Software components already translated into code are integrated into a "build". The "build" can be data files, libraries, reusable modules or program components.
- A series of tests are designed to expose errors from build so that the "build" performs its functioning correctly.
- The "build" is integrated with the other builds and the entire product is smoke tested daily.
- **Smoke testing benefits**

i) Integration risk is minimized.

ii) The quality of the end product is improved.

iii) Error diagnosis and correction are simplified.

iv) Assessment of progress is easy.

5.13 : Defect Management

Q.37 Write a short note on - Defect management.

☞ [SPPU : May-19 Marks 4]

OR How defects are managed ? Explain.

☞ [SPPU : June-22, Marks 8]

Ans. : Defect management process can be carried out in various phases as - defect prevention, baseline delivery, defect discovery, defect resolution and process improvement.

1. **Defect prevention :** Defect prevention is the highest priority activity in the defect management process. This stage suggests to prevent the introduction of defects in the software product. Following are the steps taken for defect prevention -

   i) Identify the cause of the defect and reduce the occurrence of defect.

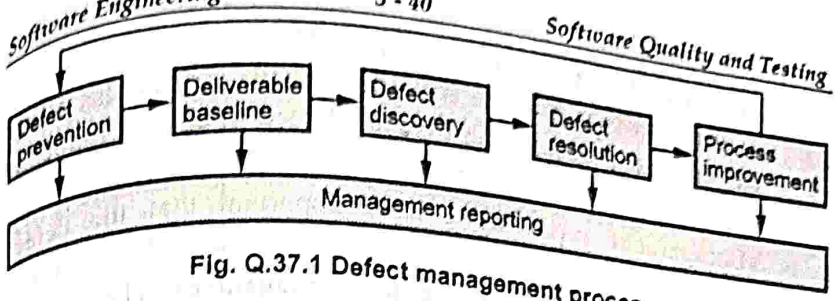


Fig. Q.37.1 Defect management process

   ii) There are some commonly occurring defects - such as coding defects or interface defects. Focus on these common causes of defects.

   iii) Identify critical risks, assess it and minimize the expected impact of the risks.

2. **Baseline delivery :** The baseline means the work product which is in **deliverable stage**. The deliverable is baselined when it reaches a predefined milestone in its development. If newly created product satisfies the acceptance criteria, then it is baselined. And only baselined product moves to the next stage.
3. **Defect discovery :** Defect discovery means defects are **brought to attention** to the developers. The defect must be **identified** before it gets a serious problem. As soon as defect gets identified, it must be reported to the authority team in order to resolve it.
4. **Defect resolution :** Once the developers have acknowledged a valid defect, the resolution process begins. The **defect resolution** must be done in **following steps -**
   1. **Determine** the **importance** of the defect.
   2. **Schedule and fix** the defect as per the order of its importance.
   3. **Notify it** to all the concern parties when defect gets resolved.
5. **Process improvement :** This is the most neglected step in defect management process. This step suggests that participants should **go back to the process** that originated the defect to

understand what caused the defect. Then they should go back to the validation process that should have caught the defect earlier in the process. Thus treat the defects as an opportunity to improve the software development activities.

6. **Management reporting :** It is important that the defect information must be analyzed and **communicated** to both the **project management and senior management**. The purpose of collecting such information is -
   1. To know the status of individual defect.
   2. To provide insight into the processes that need the improvement.
   3. To provide the strategic information to senior management to make important decisions.

5.14 : Defect Life Cycle

Q.38 Explain defect life cycle in detail.

Ans. : Defect life cycle is a cycle which demonstrates how a defects goes through various stages in its lifetime.

- The number of states that defect goes through varies from project to project.
- The defect life cycle is also called as bug life cycle.
- Following Fig. Q.38.1 shows various stages of defect life cycle -

1. **New :** This is an initial stage in which the defect is raised and posted for the first time.
2. **Assigned :** After the bug identification it is assigned to a developer for fixing. This state is called as assigned state.
3. **Active :** At this state, the developer starts analyzing the defect. The outcome of this stage can be -

   Deferred : If the defect is expected to be fixed in next subsequent stages.

   Rejected : If the defect is not genuine, then developer may reject it.

Duplicate : If the bug or defect appears twice then it is considered to be duplicated.

Not a Bug : If there is no change in the functionality of the application, then it is reported as no bug state.

4. **Test :** The defect is fixed and it is ready for testing.
5. **Retest :** At this stage the developer can retest the code.
6. **Reopen :** If the defect still remains even after fixing then it goes in reopen state. After this stage the defect traverse through the life cycle stages.

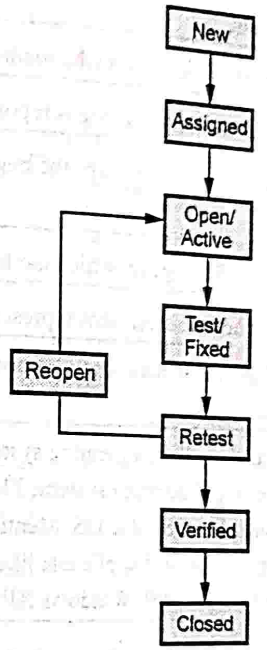


Fig. Q.38.1 Defect life cycle

7. **Verified :** After testing and retesting when the defect gets fixed, then its state is declared as verified.
8. **Closed :** When the defect no longer exists, then it enters in the closed state.

5.15 : Bug Reporting

Q.39 What is bug ? Give the template for bug report

Ans. : Definition : Software bug is an error, failure or fault in computer program or system that causes incorrect or unexpected results.

Template for bug report

Bug ID	The specific ID of the bug.
Bug name	The name of the bug that simply denotes what the bug is.
Reporter	Name of the person who identifies the bug.
Submit date	Date on which the bug is reported.
Description of bug	Detailed description of the bug.
URL	The URL page on which the bug occurs.
Screenshot	The actual screen shot representing the status of bug.
Platform	Mention the hardware platform where you found this bug.
Operating system	Mention all the operating systems where you found the bug. Operating systems like Windows, Linux, Unix, SunOS, Mac OS. Mention the different OS versions also if applicable like Windows NT, Windows 2000, Windows XP etc.
Browser	The bug may occur on particular web page. The name of the web browser in which this web page is opened.
Severity	Severity can be major, minor, critical, trivial, blocker
Assigned to	The name of the person who is responsible for handling the bug.

5.16 : Debugging

Q.40 Explain the debugging process.

☞ [SPPU : Dec.-13, Marks 6]

Ans. : Debugging is a process of removal of a defect. It occurs as a consequence of successful testing.

Debugging process starts with execution of test cases. The actual test results are compared with the expected results. The debugging process attempts to find the lack of correspondence between actual and expected results. The suspected causes are identified and additional tests or regression tests are performed to make the system to work as per requirement.

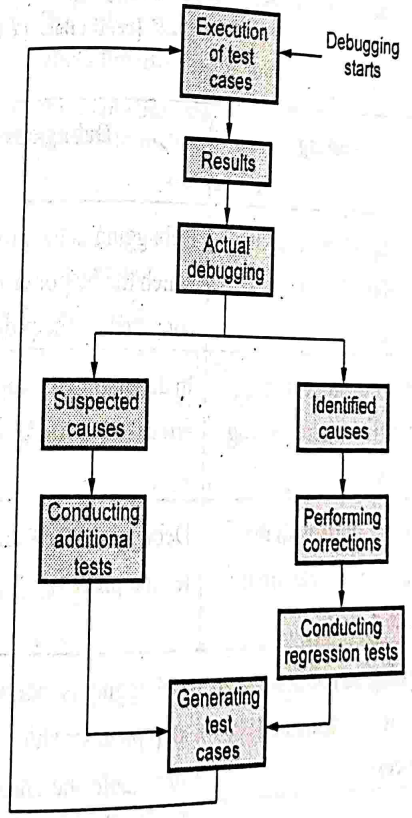


Fig. Q.40.1 Debugging process

Common approaches in debugging are :

Brute force method - The memory dumps and run-time traces are examined and program with write statements is loaded to obtain clues to error causes.

In this method "Let computer find the error" approach is used.

This is the least efficient method of debugging.

Backtracking method - This method is applicable to small programs.

In this method, the source code is examined by looking backwards from symptom to potential causes of errors.

Cause elimination method - This method uses binary partitioning to reduce the number of locations where errors can exist.

Q.41 Compare the software testing with debugging.

☞ [SPPU : Dec.-12,17, Marks 4]

Ans. :

Sr. No.	Testing	Debugging
1.	Testing is a process in which the bug is identified	Debugging is the process in which the bug or error is corrected by the programmer
2.	In testing process, it is identified where the bug occurs.	In debugging the root cause of error is identified.
3.	Testing starts with the execution results from the test cases.	Debugging starts after the testing process.
4.	Testing can be automated and it is a structured process.	Debugging is manual step by step process which is unreliable and unstructured.

END ...

Unit VI

6 Formal Methods Recent Trends in Software Engineering

6.1 : SCM

Q.1 What is software configuration management ?

Ans. : Definition : Software configuration management is a set of activities carried out for identifying, organizing and controlling changes throughout the lifecycle of computer software.

Q.2 What is configuration management repository ? Discuss role and features of SCM repository.

☞ [SPPU : May-18, Marks 8, June-22, Marks 9]

Ans. :

- Software configuration Items (SCI) are maintained in project repository or project library.
- The software repository is basically a database that acts as a center for both accumulation and storage for software engineering information.
- The software engineer interacts with repository using tools that are integrated within it.

Role of Project Repository

- Software repository is a collection of information accessed by software engineers to make appropriate changes in it.
- This repository is handled using all the modern database management functions.
- It must maintain important properties such as data integrity, sharing and integration.

- The repository must maintain uniform structure and format for software engineering work products.
- To achieve these capabilities, the repository is defined in terms of **meta-model**. The meta model determines –

i) How information is stored in repository ?

ii) How data can be accessed by using the tool ?

iii) How the security and integrity of the data is maintained ?

iv) How the existing model can be extended to accommodate new requirements ?

Features

The Software Configuration Management can be performed by interacting with repository. The repository must have toolset that provides support for following features -

1. **Versioning :** As project progresses various versions of individual work products may get created. The repository must be able to maintain all these versions and permit the developer to go back to previous versions during testing and debugging.
2. **Dependency Tracking and Change Management :** The data elements stored in the repository are related to each other. The repository must have an ability to keep track of these relationship and to maintain data integrity.
3. **Requirements Tracing :** If the constructed components, its designed is tracked, then particular requirement can be traced out. The repository must have this ability to trace the requirement from constructed components.
4. **Configuration Management :** The repository must be able to keep track of series of configurations representing project milestones and production releases.
5. **Audit Trails :** The audit trail is a set of audit log which is a set of records. The purpose of audit trail is to trace specific event or operation.

Q.3 What are the elements that exist when an effective SCM system is implemented ? Discuss each briefly.

Ans : • Susan Dart identified four important elements for software configuration management system. These elements are -

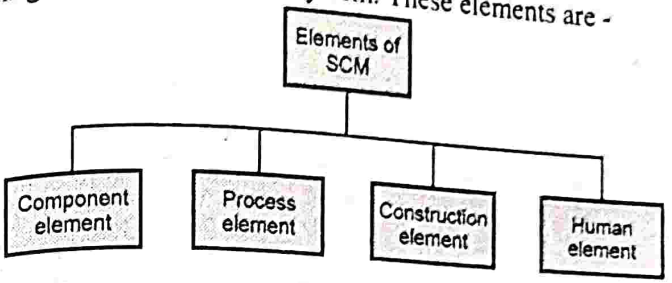


Fig. Q.3.1

1. **Component Elements :** It consists of collection of tools that are used for file management system. For example - Databases.
2. **Process Elements :** It consists of actions and tasks used during change management and use of software.
3. **Construction Elements :** It is a collection of tools that automate the construction of software.
4. **Human Elements :** It consists of set of tools that are used by software team to implement software configuration management.

Q.4 Which are the layers of SCM process ? Explain each in detail.

☞ **[SPPU : Dec.-19, 22, Marks 6]**

Ans. : Layers of SCM process are as follows -

1) **Configuration Identification :** Refer Q.5.

2) **Change Control :** Refer Q.6.

3) **Version Control :**

- Version is an instance of a system which is functionally distinct in some way from other system instances.
- Version control works to help manage different versions of configuration items during the development process.
- In practice the version needs an associated name for easy reference.

- Different versions of a system can be shown by an evolution graph as shown in Fig. Q.4.1.
- Each version of software system is a collection of software configuration items.

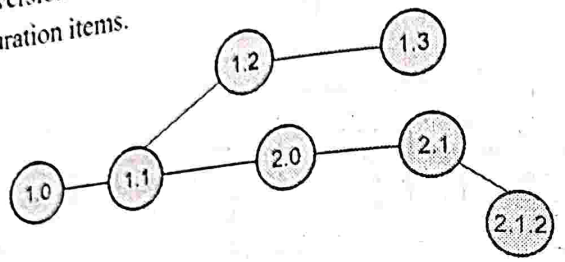


Fig. Q.4.1 Version numbering in evolution graph

4) Configuration Audit :

- In order to ensure that the change has been properly implemented or not two activities are carried out.
  1. Formal Technical Review (FTR)
  2. Software Configuration Audit.
- In Formal Technical Review, the correctness of configuration object is identified and corrected. It is conducted by technical reviewer.
- The software configuration audit assess the configuration object for the characteristics that are not reviewed in formal technical review. It is conducted by software quality assurance group.
- Following are some primary questions that are asked during configuration audit -
  1. Whether FTR is conducted to assess the technical correctness ?
  2. Whether or not the change specified by ECO has been made ?
  3. If additional changes need to be made or not ?
  4. Whether the software engineering standards are properly followed ?
  5. Do the attributes of configuration objects reflect the change ?

6. Whether all the SCI are updated properly ?
7. Whether the SCM process (object identification, change and version control, configuration audit and status reporting) are properly followed ?

The above questions can be asked as a part of formal technical review.

5) Status Reporting :

- The status reporting focuses on communication of changes to all people in an organization that involve with changes.
- During status reporting following type of questions were asked.
  1. **What happened ? :** What are the changes that are required ?
  2. **Who did it ? :** Who will be handling these changes ?
  3. **When did it happen ? :** The time at which these changes are arised.
  4. **What else will be affected ? :** The objects or part of the software that might be reflected due to these changes.

Q.5 What is configuration identification in SCM ?

☞ **[SPPU : June-22, Marks 9]**

Ans. : The software configuration items must be separately named and identified as object.

- These objects must be arranged using **object oriented approach.**
- There are two categories of objects - **Basic objects** and **aggregate objects.**
- The **basic object** is unit of information created during requirements analysis, design, coding or testing. For example basic object can be part of **source code.**
- Aggregate object is a collection of basic objects and other aggregate objects. For example SRS or data model can be aggregate object.

- Each object can be uniquely identified because it has got -

1. **Name :** The name of the object is nothing but the collection of characters (string) or some text. It is unique.
2. **Description :** For describing the object, the object description can be given. This description contains document, program or some other description such as project identifier or version information.
3. **List of resources :** The resources are the entities that are used for accessing, referencing and processing of objects. The data types and functionalities can serve as a resource.
4. **Realization or identification :** It is pointer to object.

- The configuration object identification can also consider relationships that exist between the named objects.
- If a change is made to one configuration object it is possible to determine which other configuration objects in the repository are affected by the change.
- Basically object evolve throughout the software process. During the process of object identification the evolution of objects along with its process must be identified.
- Major modifications in the object must be noted.

Q.6 Explain change control mechanism in SCM.

Ans. : •Changes in any software projects are vital. Sometimes, introducing small changes in the system may lead to big problems in product.

- Similarly, introducing some changes may enhance the capabilities of the system.
- According to **James Bach** too little changes may create some problems and too big changes may create another problems.
- For a large software engineering project, uncontrolled change creates lot of chaos. For managing such changes, human procedures or automated tools can be used.

- The change control process is shown by following Fig. Q.6.1.

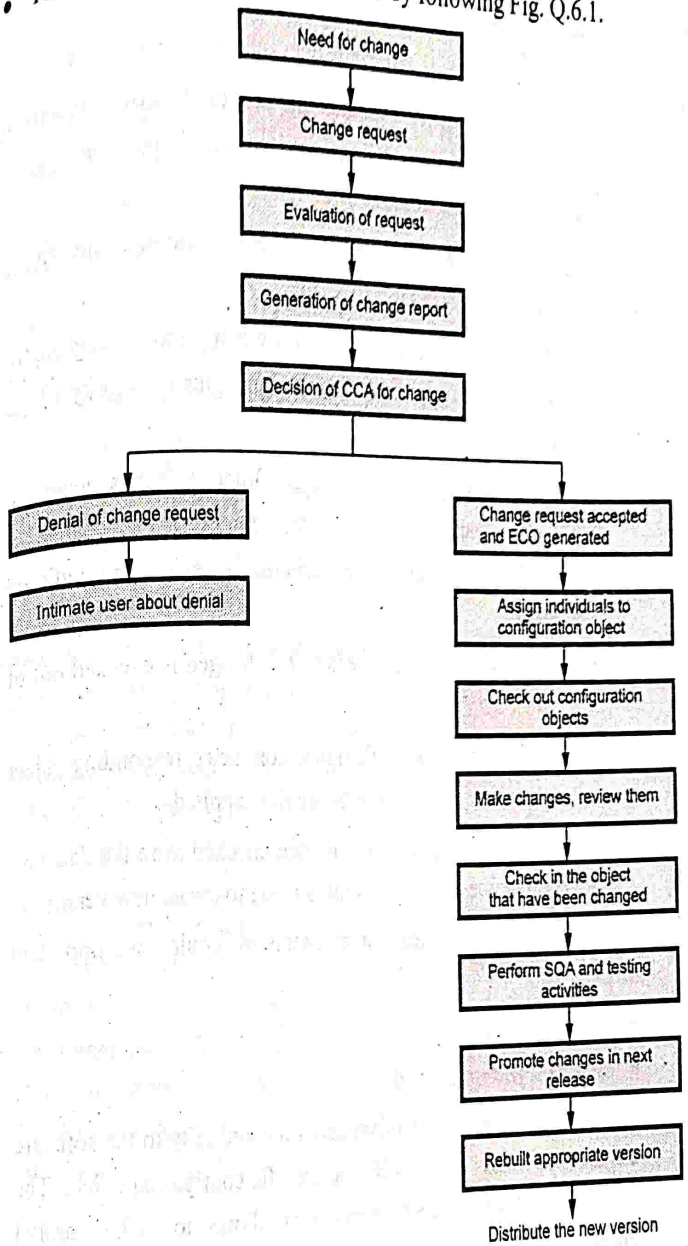


Fig. Q.6.1 Change control process

- **Step 1 :** First of all there arises a need for the change.
- **Step 2 :** The change request is then submitted by the user
- **Step 3 :** Developers evaluate this request to assess technical merits, potential side effects, and overall impact on system functions and cost of the project.
- **Step 4 :** A change report is then generated and presented to the Change Control Authority (CCA).
- **Step 5 :** The change control authority is a person or a group of people who makes a final decision on status or priority of the change.
- **Step 6 :** An Engineering Change Order (ECO) is generated when the change gets approved. In ECO the change is described, the restrictions and criteria for review and audit are mentioned.
- **Step 7 :** The object that needs to be changed is checked out of the project database.
- **Step 8 :** The changes are then made on the corresponding object and appropriate SQA activities are then applied.
- **Step 9 :** The changed object is then checked in to the database and appropriate version control is made to create new version.

The checked in and checked out mechanisms require two important elements -

- Access control
- Synchronization control.

The **access control** mechanism gives the authority to the software engineer to access and modify the specific configuring object. The **synchronization control** mechanism allows to make parallel changes or the changes made by two different people without overwriting each other's work.

6.2 : Risk Management

Q.7 What is software risk management ? What are the risks associated with software projects ? How do project managers manage such risks ? ☞ [SPPU : May-18, Marks 10]

Ans. : Risk Management :

- **Definition of risk management :** Risk management refers to the process of making decisions based on an evaluation of the factors that threats to the business.
- Various **activities** that are carried out for **risk management** are -
  1. Risk identification
  2. Risk projection
  3. Risk refinement
  4. Risk mitigation, monitoring and management.

Risks associated with software projects :

1. **Project risk :** Project risks arise in the software development process then they basically affect budget, schedule, staffing, resources, and requirements. When project risks become severe then the total cost of project gets increased.
2. **Technical risk :** These risks affect quality and timeliness of the project. If technical risks become reality then potential design implementation, interface, verification and maintenance problems gets created. Technical risks occur when problem becomes harder to solve.
3. **Business risk :** When feasibility of software product is in suspect then business risks occur. Business risks can be further categorized as,
   i) **Market risk -** When a quality software product is built but if there is no customer for this product then it is called market risk (i.e. no market for the product).
   ii) **Strategic risk -** When a product is built and if it is not following the company's business policies then such a product brings strategic risks.

iii) **Sales risk** - When a product is built but how to sell is not clear then such a situation brings sales risk.

iv) **Management risk** - When senior management or the responsible staff leaves the organization then management risk occurs.

v) **Budget risk** - Losing the overall budget of the project is called budget risk.

Another categorization of risk proposed by **Charette** is -

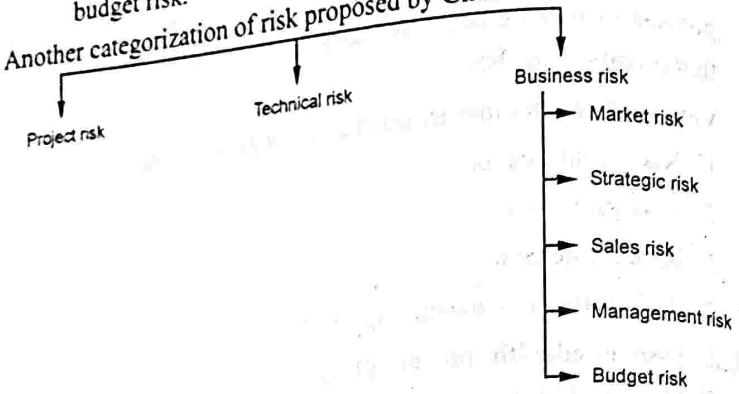


Fig. Q.7.1 Categorization of risk

Q.8 What is risk identification? What are different categories of risk ? ☞ [SPPU : Dec.-16, Marks 7]

Ans. : Risk Identification : Risk identification can be defined as the efforts taken to specify threats to the project plan. Risks identification can be done by identifying the known and predictable risks.

The risk identification is based on two approaches

1. **Generic risk identification** - It includes potential threat identification to software project.
2. **Product-specific risk identification** - It includes product specific threat identification by understanding people, technology and working environment in which the product gets built.

Normally the risk identification is done by the project manager who follows following steps -

- **Step 1 :** Preparation of risk item check list

The risk items can be identified using following known and predictable components

i) **Product size** - The risk items based on overall size of the software product is identified.

ii) **Business impact** - Risk items related to the marketplace or management can be predicted.

iii) **Customer characteristics** - Risks associated with customer-developer communication can be identified.

iv) **Process definition** - Risks that get raised with the definition of software process. This category exposes important risks items because whichever is the process definition made, is then followed by the whole team.

v) **Development environment** - The risks associated with the technology and tool being used for developing the product.

vi) **Staff size and experience** - Once the technology and tool related risks items are identified it is essential to identify the risk associated with sufficient highly experienced and skilled staff who will do the development.

vii) **Technology to be built** - complexity of the system should be understood and related risk items needs to be identified.

After preparing a risk item checklist a questionnaire is prepared. These set of questions should be answered and based on these answers the impact or seriousness of particular risk item can be judged.

- **Step 2 :** Creating risk components and drivers list.

The set of risk components and drivers list is prepared along with their probability of occurrence. Then their impact on the project can be analysed.

Let us understand which are the risk components and drivers.

Categories of Risk : Refer Q.7.

Q.9 Explain the risk identification and assessment process for a software project. ☞ [SPPU : May-13, Dec.-13, Marks 8]

Ans. : Risk Identification : Refer Q.8.

Risk Assessment Process : The best approach is to prepare a set of questions that can be answered by project managers in order to asses the overall project risks. These questions can be,

1. Will the project get proper support by the customer manager ?
2. Are the end-users committed to the software that has been produced ?
3. Is there a clear understanding of requirements ?
4. Is there an active involvement of the customer in requirement definition ?
5. Is that the expectations set for the product are realistic ?
6. Is project scope stable ?
7. Are there team members with required skills ?
8. Are project requirements stable ?
9. Does the technology used for the software is known to the developers ?
10. Is the size of team sufficient to develop the required product ?
11. Is that all the customers know the importance of the product/ requirements of the system to be built ?

Thus the number of negative answers to these questions represents the severity of the impact of the risk on overall project.

Q.10 How risk projection is carried out using risk table ? ☞ [SPPU : May-11, Marks 8]

Ans. :

1. Building the risk table is the simplest and most commonly used technique adopted by project managers in order to project the risks. The sample risk table is as given below -

Risk table

Risk	Category	Probability	Impact	RMMM
Is the skilled staff available.	Staff	50 %	Catastrophic	
Is that the team size sufficient.	Staff	62 %	Critical	
Have the staff received sufficient training.	Staff	25 %	Marginal	
Will technology meet the expectations.	Technology	30 %	Critical	
Is the software management tool available.	Environment	40 %	Negligible	

How much amount of reused software is required?	Project size	60 %	Marginal	
Will customer change the requirement?	Customer	20 %	Critical	

While building the risk table

- The project team first of all enlists all probable risks with the help of risk item checklist.
- Each risk is then categorized. As we know various categories of risk can be a) Project size b) Technology c) Customer d) Staff e) Business f) Developing environment.
- Probability of occurrence of each risk is then estimated by each team member individually.
- Then impact of each risk is assessed. While calculating the impact of each risk, each using the cost drivers each component of risk (performance, cost, support, and schedule) is assessed and it then averaged to quote the overall impact of particular risk.

2. After building this table it is then sorted by probability and impact. The high probability and high impact risks will be at the top of the table. And low probability and low impact risk will be at the bottom of the table. This arrangement of the table is called first-order prioritization.
3. Then the project manager goes through this first-order prioritized risk table and draws a horizontal line at some point in the table. This line is called cut off line. The risks table above the cut off line is now considered for further risk analysis.

4. The risk table below the cut off line is again sorted and a second-order prioritization is applied on this table.
5. The risk table above the cut-off line is having the risks with high probability and high impact and such risks should occupy the significant amount of management time.
6. All the risks that lie above the cut off line should be managed. Using Risk mitigation, monitoring and management plan the last column of the risk table is filled up.

Q.11 Write short note on : RMMM.

☞ **[SPPU : May-14, Marks 5, May-12, Dec.-17,18,22, Marks 10]**

Ans. : RMMM stands for risk mitigation, monitoring and management. There are three issues in strategy for handling the risk is

1. Risk avoidance 2. Risk monitoring 3. Risk management.

1. **Risk mitigation** : Risk mitigation means preventing the risks to occur(risk avoidance). Following are the steps to be taken for mitigating the risks.

1. Communicate with the concerned staff to find of probable risk.
2. Find out and eliminate all those causes that can create risk before the project starts.
3. Develop a policy in an organization which will help to continue the project even though some staff leaves the organization.
4. Everybody in the project team should be acquainted with the current development activity.
5. Maintain the corresponding documents in timely manner. This documentation should be strictly as per the standards set by the organization.
6. Conduct timely reviews in order to speed up the work.
7. For conducting every critical activity during software development, provide the additional staff if required.

2. Risk monitoring : In risk monitoring process following things must be monitored by the project manager,

1. The approach or the behaviour of the team members as pressure of project varies.
2. The degree in which the team performs with the spirit of "team-work".
3. The type of co-operation among the team members.
4. The types of problems that are occurring.
5. Availability of jobs within and outside the organization.

The project manager should monitor certain mitigation steps. For example.

If the current development activity is monitored continuously then everybody in the team will get acquainted with current development activity.

The objective of risk monitoring is,

1. To check whether the predicted risks really occur or not.
2. To ensure the steps defined to avoid the risk are applied properly or not.
3. To gather the information which can be useful for analyzing the risk.

3. Risk management

- Project manager performs this task when risk becomes a reality.
- If project manager is successful in applying the project mitigation effectively then it becomes very much easy to manage the risks.
- For example, consider a scenario that many people are leaving the organization then if sufficient additional staff is available, if current development activity is known to everybody in the team, if latest and systematic documentation is available then any 'new comer'

can easily understand current development activity. This will ultimately help in continuing the work without any interval.

Q.12 In recent year, university has computerized its examination system by using various software applications. Find out risk involved in implementation and administration you as software expert. Prepare RMMM Plan for the same.

☞ [SPPU : Dec.-17, Marks 6]

Ans. : Template for RMMM Plan :

- The RMMM plan is a document in which all the risk analysis activities are described.
- Sometimes project manager includes this document as a part of overall project plan.
- Sometimes specific RMMM plan is not created, however each risk can be described individually using risk information sheet.
- Typical template for RMMM plan or Risk information sheet can be,

Risk information sheet			
Project name <enter name of the project for which risks can be identified>			
Risk id <#>	**Date**<date at which risk is identified >	**Probability**<risk probability>	**Impact**<low/medium/high>
Origin<the person who has identified the risk>		**Assigned to** <who is responsible for mitigating the risk>	
Description<Description of risk identified>			
Refinement/Context<associated information for risk refinement>			
Mitigation/Monitoring<enter the mitigation/monitoring steps taken>			
Trigger/Contingency plan <if risk mitigation fails then the plan for handling the risk>			
Status<Running status that provides a history of what is being done for the risk and changes in the risk. Include the date the status entry was made>			

Approval<name and signature of person approving closure>.
Closing date<date>

- The risk information sheet can be maintained by database systems.
- After documenting the risks using either RMMM plan or Risk information sheet the risk mitigation, monitoring and analysis activities are stopped.

Risk involved in computerized examination system

Risks	Probability	Impact
Computer crash	70 %	Catastrophic
Late delivery	30 %	Critical
Technology will not meet expectations	20 %	Catastrophic
Change in requirement	25 %	Critical
Deviation from software engineering standards	10 %	Marginal
Poor comments in code	20 %	Negligible

Following is RMMM information for sample two risks mentioned above -

Risk : Computer Crash

Mitigation : The computer crash result in loss of data. Due to loss of data the online examination system can not be delivered properly. Such a system will not be acceptable by the customer.

Monitoring : While working on building of online examination system, the staff member should always be aware of the stability of computing environment. Any changes in stability of environment should be recognized and taken seriously.

Management : The lack of stable system is extremely hazardous to software development team. If such unstable system is found then development team should cease work on the system until the environment is made stable again.

Risk : Late Delivery

Mitigation : The cost associated with late delivery of online examination system is critical. This result in rescheduling of online examination system.

Monitoring : Realistic schedule must be established for monitoring the project status.

Management : If the project cannot be delivered on time then the university can not conduct the examination system properly. If it becomes apparent that the project will not be completed on time, the only course of action available would be to request an extension to the deadline.

6.3 : Emerging Software Engineering Trends

Q.13 Write short note on - Technology evolution.

☞ [SPPU : May-15, Dec.-14, 15 , Marks 6, May-16 ,Dec.-18, Marks 4]

Ans. : • Ray Kurzeweil argues that technological evolution is similar to the biological evolution but at a very faster rate. Normally in this evolution process one more capable method at one stage of evolutionary progress is used to create the next stage.

- The rapid technology evolution has posed three big questions and those are -
  - How rapidly does a technology evolve ?
  - How significant effects of evolution ?
  - How profound will be the resultant changes ?

- Kurzeweil suggests that within 20 years technology evolution will accelerate at an increasingly rapid speed and will lead to an era of non biological intelligence.
- By the year 2040 a combination of extreme computation, nanotechnology, massively high speed networks and robotics will create a very different era.

Q.14 Write short note on - Test driven development.

☞ **[SPPU : Dec.-14, May 15, Dec.-15 , Marks 6, Dec.-16 , June-22, Marks 5]**

Ans. : Test Driven software development is software development technique in which there occurs very short development cycle followed by repetitive tests.

- A Test First Design (TFD) approach is adopted in TDD. That means, the first step is always to add the test. If test is passed then add another test. But if test is failed then make some corrections in the code, run the automated test until all the tests get success.
- Following are the basic steps followed in TDD cycle.
  - **Add a Test :** In the test driven development the development cycle begins by writing a test. To write these tests, developer must understand the feature and requirements clearly.
  - **Run tests :** Execute the test for existing code. If the tests succeed then iteratively add new test.
  - **Write correcting code :** If the tests fail then correct the existing code and send it for testing. This will increase the confidence in the code.

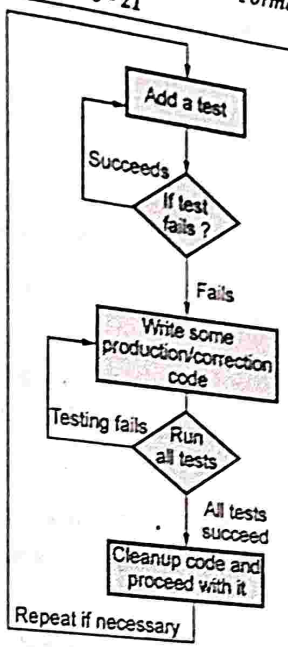


Fig. Q.14.1 Test driven development cycle

  - **Run automated tests :** If all the test cases pass in this manner then programmer can be confident that code meets all the tested requirements.
  - **Repeat**

    Starting with another new test case the cycle will be repeated to push forward the functionality.

    Finally the code can be cleaned up. By re-running the test cases the developer is re-factoring the code, testing is without affecting the its functionality.

Q.15 Write short note on - Collaborative development.

☞ **[SPPU : May-16 , Dec.-18, Marks 4; June-22, Marks 8]**

Ans. :

- Today, software engineers collaborate across the international boundaries, and every one of them shares the information.
- The challenges over next decade will be to develop methods and tools that facilitate the collaboration for software development.

- Number of success factors that lead to successful collaborative efforts are -

1. **Shared Goals :** The project goals must be clearly specified and all the stakeholders must understand and agree with them.
2. **Shared Culture :** The cultural difference should be clearly defined and proper communication and educational approach must be adopted.
3. **Shared Process :** Process is the basic unit of collaborative project. All the team members work on it to create a successful working system.
4. **Shared Responsibilities :** Every team member must recognize the needs of the customer and should work to give best possible solution.

Q.16 What are challenges of global software development ?
☞ [SPPU : Dec.-16, Marks 5, May-18, Marks 8]

Ans. : Following are some challenges faced by global software development

1. Teams involved in software development have **different cultural background**. Hence it create **problem in communication**.
2. Providing a training to the developers who speak or understand the **specific language** only can be a problematic.
3. Face to face meetings are difficult due to **remote locations** of teams. This leads to poor or broken communication among different teams.
4. Language barrier between client and vendors lead to communication and co-ordination issues.
5. Trust between two parties is always a matter of concern.
6. Due to **different time zones** of developers co-ordination among real time software modules is problematic.
7. There are chances of having **hidden cost** or **unexpected cost** in the software development process.

8. There is always a **threat of loss of control** in the development activities.
9. There can be some **legal issues** involved while **handling the new market**.

Q.17 What are different process trends in software engineering ?
☞ [SPPU : Dec.-17, Marks 6]

Ans. : The fundamental unit of business, organizational and cultural trends is process.

The **six process trends** suggested by Conradi and Fuggetta are as follows -

1. In a rapid software development the focus will be on short term goals that have product innovation.
2. Software engineers have a good sense that where the process is weak. Therefore the process change will be driven by this knowledge.
3. Automation software process technology is used only within those processes which will get benefited most by such automation.
4. Emphasis of process development will be on return of investment of software development activities.
5. As time passes.the software community will understand that the software development has a greater impact on sociology and anthropology.
6. New modes of learning will facilitate the effective software processes.

Q.18 Write short note on - i) Test driven development
ii) Collaborative development. ☞ [SPPU : Dec.-17, Marks 12]

Ans. :

i) Test driven development : Refer Q.14.

ii) Collaborative development : Refer Q.15.

6.4 : CASE

Q.19 Write short note on - CASE tools.

☞ [SPPU : May-15, Dec.-15 , Marks 6, Dec.-18, Marks 4]

Ans. :

- The Computer Aided Software Engineering (CASE) tools automate the project management activities, manage all the work products. The CASE tools assist to perform various activities such as analysis, design, coding and testing.
- Software engineers and project managers make use of CASE tools.
- The use of CASE tools in the software development process reduces the significant amount of efforts.
- CASE is used in conjunction with the process model that is chosen.
- CASE tools help software engineer to produce the high quality software efficiently.
- Use of CASE tool automates particular task of software development activity. But it is always useful to use a set of CASE tools instead of using a single CASE tool. If different CASE tools are not integrated then the data generated by one tool will be an input to other tools. This may also require a format conversions, as tools developed by different vendors may use different formatting. There are chances, that many tools do not allow exporting data and maintain data in proprietary formats.

Q.20 Explain the components of CASE.

Ans. : Various components of CASE tools are -

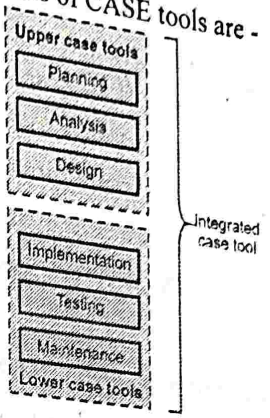


Fig. Q.20.1 Components of CASE tools

1. **Central Repository :**
   - The central repository contains the common, integrated and consistent information.
   - The central repository acts like a data dictionary.
   - It contains product specification, requirement documents, project management information and reports.
2. **Upper Case Tools :**
   - Uppercase tool focus on the planning, analysis phase and sometimes the design phase of the software development lifecycle.
3. **Lower Case Tools :**
   - Lower CASE Tool Computer-Aided Software Engineering (CASE) software tool that directly supports the implementation (programming) and integration tasks.
   - Lower CASE tools support database schema generation, program generation, implementation, testing, and configuration management.

4. Integrated CASE tools :

- These are type of tools that integrate both upper and lower CASE, for example making it possible to design a form and build the database to support it at the same time.
- An automated system development environment that provides numerous tools to create diagrams, forms and reports.
- It also offers analysis, reporting and code generation facilities and seamlessly shares and integrates data across and between tools.

Q.21 Write short note on - Workbenches In CASE.

Ans. : CASE workbench is a set of tools which supports a particular phase in the software process.

- These tools work together to provide the complete support to the software development activity.
- In the workbench common services are provided which are used by all the tools. Some kind of data integration is also supported.

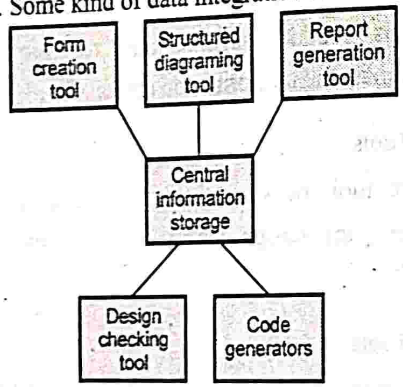


Fig. Q.21.1 Analysis and design workbench

- For example -**Analysis and Design Workbench** (Refer Fig. Q.21.1)
  - It support the generation of system models during design and analysis activities.

  - It provides graphical editors plus shared repository.
  - It may include code generators to create source code from design information.

Advantages

1. It is available at low cost.
2. There is productivity improvement due to support of workbenches.
3. It results in standardized documentation for software system.

Drawbacks

1. These systems are closed environments with tight integration with tools.
2. The import and export facilities for various types of data is limited.
3. It is difficult to adapt method for specific organizational need.

6.5 : Introduction to Agile Tools

Q.22 What is the objective of agile tools ?

Ans. :

- The **objective of agile development** tools is to help in one or more activities in agile development. Due to use of such tools the operational development software can be created rapidly.
- The agile tool set consists of automated support for project planning, use case development, and requirements gathering, code generation and testing.
- The project management tools focus on preparing 'Earned value charts' instead of preparing the Gnatt charts.
- The purpose of agile tools is to **enhance the environment.**
- Let us discuss **two commonly used agile tools:**

Q.23 Write short note on - Jira. [SPPU : Dec.-22, Marks 4]

Ans. :

- JIRA is an agile tool developed by a company named Atlassian.
- This tool is used for **bug tracking, issue tracking** and project management.
- The name of this tool is inherited from the Japanese word "Gojira"
- JIRA is written in **Java**.
- The **main features** of Jira for agile software development are the functionality to plan development iterations, the iteration reports and the bug tracking functionality.
- The tool is basically used for project management activities. It is also used to track issues bugs related to the software and mobile app.

JIRA Issues

JIRA issues can be used to track bug in the project. The Issue type displays all types of items that can be created and tracked via JIRA.

JIRA Components

Components are sub-sections of a project. They are used to group issues within a project into smaller parts. Using the Components the projects can be broken down into features, teams, modules, sub-projects and so on.

Using components we can generate reports, collect statistics, and display it on dashboards and so on.

Reports in JIRA

- In agile, the progress of the project can be represented with the help of **Burndown Chart**.

- Apart from Burn down types of reports available in JIRA like Sprint Report, Version Report, Control Chart, Cumulative flow diagram and so on. Using these various types of reports the progress of the project can be tracked.

Q.24 Write short note on - Kanban. [SPPU : Dec.-22, Marks 4]

Ans. :

- Kanban is a new technique for managing an agile software development process in a highly efficient way.
- It makes use of the **Just In Time(JIT)** approach.
- In Kanban technique, the "Work picture" can be created by using **visual information**.
- The board above shows a situation where the developers and analysts are tracking the agile software development process. The software development phases are represented in which the number in each column indicate the number of items to be analysed. Each phase is then sub divided into two more columns - **Doing** and **Done**. The tasks that are under **Done** columns will be then promoted to the next phase.

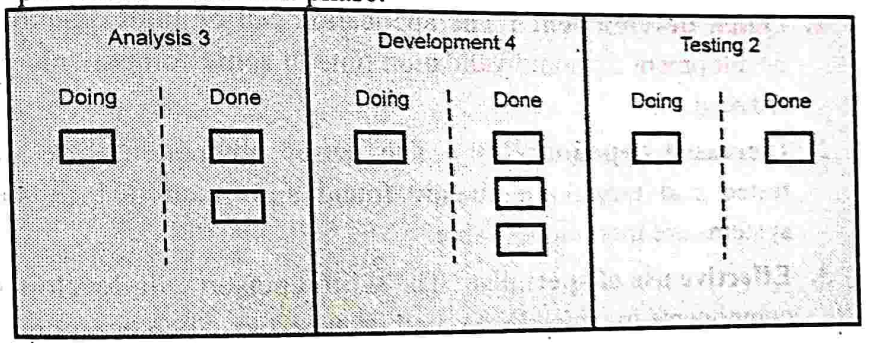


Fig. Q.24.1

Q.25 What is software reuse ? Give its benefits and drawbacks. [SPPU : June-22, Marks 9]

Ans. : Software reuse

- The software engineering is adopting software reuse approach. In this approach, the instead of developing original software from scratch, existing components can be used so that the software development can be quick and at low cost.
- There are three ways by which the **software reuse** is possible -
  1. **Application system reuse** - The whole application can be reused without making much changes in it. In this type of reuse the common architecture of the application is used as it is but the reused component is tailored for specific customer.
  2. **Component reuse** - The components of sub-systems are reused.
  3. **Object and function reuse** - The object class, some functionality of the object is reused. This reuse is based on the common libraries.

Benefits and drawbacks of software reuse

Various **benefits** of software reuse are -

1. **Quick development :** The application can be built quickly as development time and validation time of some components gets reduced.
2. **Increased dependability :** The reused components are well tested and errors are already found and fixed. Hence such systems are more dependable.
3. **Effective use of specialist :** The skillful person will develop the components in which he has gained expertise. Such components can be encapsulated in a library and this library can be reused as per the need.
4. **Standard compliance :** Certain standards can be implemented in the form of components. These components can be reused as per the need.

Various **drawbacks** of software reuse are -

1. Finding, understanding and adapting the software components is very difficult.
2. There is a need for creating and maintaining the library of reused components.
3. The maintenance cost is increased.

END ...

June - 2022 [END SEM] [5869] - 290 | Solved Paper

Course 2019

Time : $2\frac{1}{2}$ Hours] [Maximum Marks : 70

Instructions to the candidates :

1) Answer Q1 or Q2, Q3 or Q4, Q5 or Q6, Q7 or Q8.

2) Neat diagrams must be drawn wherever necessary.

3) Figures to the right side indicate full marks.

4) Assume suitable data, if necessary.

Q.1 (a) *What is software architecture ? Explain data centered and object oriented architectural style of the system.*

(Refer Q.22 of Chapter - 3) **[9]**

(b) *Explain guidelines for component level design and principles for user interface design.*

(Refer Q.34 of Chapter - 3) **[9]**

OR

Q.2 (a) *What are elements of design model ? What are the elements of architectural design ? Explain design principles ?* **(Refer Q.28 of Chapter - 3)** **[9]**

(b) *Explain the following fundamental software design concepts : i) Abstraction ii) Architecture iii) Patterns* **(Refer Q.7 of Chapter - 3)** **[9]**

Q.3 (a) *What is work breakdown structure ? How is it related with scope management and explain 8/80 rule.*

(Refer Q.5 of Chapter - 4) **[8]**

(b) *Calculate activity expected time and variance for given problem.* **(Refer Q.10 of Chapter - 4)** **[9]**

Activity ID	Optimistic Time (t_o)	Most Likely Time (t_m)	Pessimistic Time (t_p)
Job 1	1	3	5
Job 2	2	6	9
Job 3	2	3	5
Job 4	5	8	10
Job 5	11	15	20
Job 6	2	5	8
Job 7	3	3	3
Job 8	2	4	6

OR

Q.4 (a) *What is COCOMO II ? What areas does COCOMO II address ?* **(Refer Q.11 of Chapter - 4)** **[9]**

(b) *Explain information domain values (any 4).*

(Refer Q.35 of Chapter - 4) **[8]**

Q.5 (a) *Discuss Garvin's eight quality dimensions.*

(Refer Q.2 of Chapter - 5) **[8]**

(b) *List out ISO 9126 quality factors.*

(Refer Q.3 of Chapter - 5) **[9]**

OR

Q.6 (a) *Enumerate seven principles of testing.*

(Refer Q.12 of Chapter - 5) **[9]**

(b) *How defects are managed ? Explain.*

(Refer Q.37 of Chapter - 5) **[8]**

Q.7 **(a)** *What is software SCM repository ? Explain the features of tool set supporting SCM repository.*

(Refer Q.2 of Chapter - 6) [9]

(b) *What is configuration identification in SCM ?*

(Refer Q.5 of Chapter - 6) [9]

OR

Q.8 **(a)** *What is software reuse ? Explain benefits and drawbacks of software reuse.* **(Refer Q.25 of Chapter - 6)** [9]

(b) *Write short note on :*

i) Test Driven Development (TDD)

(Refer Q.14 of Chapter - 6)

ii) Collaborative development

(Refer Q.15 of Chapter - 6) [9]

December - 2022 [End Sem] [5925]-273 | Solved Paper

Course 2019

Time : $2\frac{1}{2}$ Hours] **[Maximum Marks : 70**

Q.1 **a)** *What is software architecture ? Explain data flow and layered architectural style of the system.* [9]

Ans. : Software architecture : The software architecture of a system represents the design decisions related to overall system structure and behavior.

Data flow :

- The data flow diagrams depict the information flow and the transforms that are applied on the data as it moves from input to output.

- The symbols that are used in data flow diagrams are -

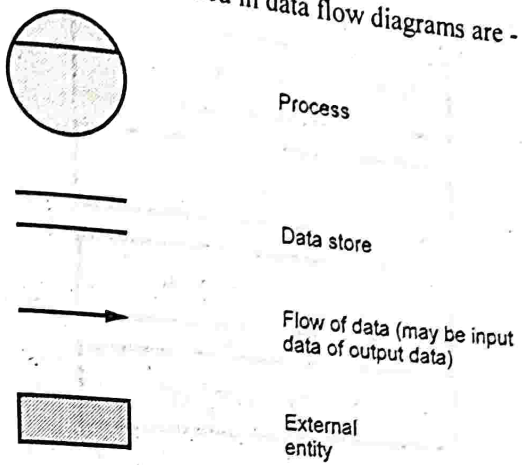


Fig. 1

- The data flow diagrams are used to represent the system at any level of abstraction.
- The DFD can be partitioned into levels that represent increase in information flow and detailed functionality.
- A level 0 DFD is called as 'fundamental system model' or 'context model'. In the context model the entire software system is represented by a bubble with input and output indicated by incoming and outgoing arrows.
- Each process shown in level 1 represents the sub functions of overall system.
- The number of levels in DFD can be increased until every process represents the basic functionality.
- As the number of levels gets increased in the DFD, each bubble gets refined.

Layered architecture :

- The model in which the system is organised into layers is called 'layered model'. This model is also called as **abstract machine model.**

- Fig. 2 shows the simple layered model.

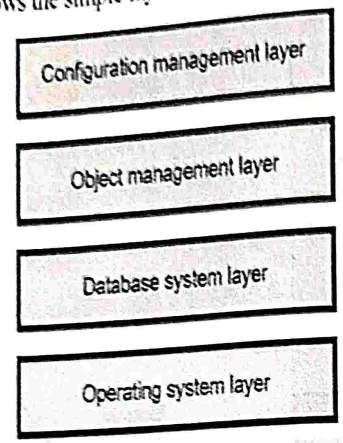


Fig. 2 Simple layered model

- Using the layered model the developments may be done in increments. After developing some layers some information is available to the user. Single layer can be replaced by other equivalent layer.

  For example : ISO-OSI reference model.

b) *What is functional independence ? Differentiate between coupling functional independence and cohesion functional independence.* [9]

Ans. : Functional independence :

- The functional independence can be achieved by developing the functional modules with single-minded approach.
- By using functional independence functions may be compartmentalized and interfaces are simplified.
- Independent modules are easier to maintain with reduced error propagation.
- Functional independence is a key to good design and design is the key to software quality.

- The major benefit of functional independence is in achieving effective modularity.
- The functional independence is assessed using two qualitative criteria - **Cohesion** and **coupling.**

Difference between cohesion and coupling :

Refer Q.12 of Chapter - 3

OR

Q.2 **a)** *What are the characteristics of a good design ? Explain software quality guidelines and attributes of software design.* **(Refer Q.2 and Q.3 of Chapter - 3)** [9]

b) *Explain design concepts :*

i) Pattern **(Refer Q.7 (iii) of Chapter - 3)**

ii) Information Hiding

iii) Modularity **(Refer Q.6 (ii) of Chapter - 3)** [9]

Ans. : (ii) Information hiding :

- Information hiding is one of the important property of effective modular design.
- The term information hiding means the modules are designed in such a way that information contained in one module cannot be accessible to the other module (the module which does not require this information).
- Due to information hiding only limited amount of information can be passed to other module or to any local data structure used by other module.
- The advantage of information hiding is basically in testing and maintenance.
- Due to information hiding some data and procedures of one module can be hidden from another module. This ultimately avoids introduction of errors module from one module to another. Similarly one can make changes in the desired module without affecting the other module.

Q.3 a) *Explain : i) 4P's of project management*

(Refer Q.24 of Chapter - 4)

ii) Software Project Estimation.

(Refer Q.36 of Chapter - 4) **[9]**

b) *What is decomposition technique ? Explain decomposition of problem and decomposition of process.*

(Refer Q.25 and Q.26 of Chapter - 4) **[9]**

OR

Q.4 a) *Explain Boehm's W^5HH principle.*

(Refer Q.28 of Chapter - 4) **[5]**

b) *What is the difference between PERT and CPM, state their application. What is the importance of critical path in a project ?* **[9]**

Ans. :

Sr. No.	PERT	CPM
1.	PERT stands for Project evaluation and Review Technique.	CPM stands for Critical Path Method.
2.	**Application :** It is a project management technique when the time required to finish the project is not certain.	**Application :** CPM is a statistical algorithm which is used when there is certain start and end time for a project.
3.	PERT is a probabilistic Model.	CPM is a deterministic Model.
4.	The main objective of PERT is to minimize the time required for completion of the project.	The main focus is to on a trade off between cost and time with effective cost cutting.
5.	It is event oriented technique.	It is activity oriented technique.

- **Importance of critical path :** Critical path method in project management helps project managers in two ways – (i) How long it will take to complete the project and (ii) What are the critical tasks that must be completed before starting other dependent tasks.

c) *Explain typical problems with IT cost estimates.*

(Refer Q.40 of Chapter - 4) **[4]**

Q.5 a) *Explain McCall's quality factors.* **[9]**

Ans. : • McCall, Richards and Walters have proposed that software quality factors can be classified into three categories which can be illustrated by **McCall's triangular model** for quality factor.

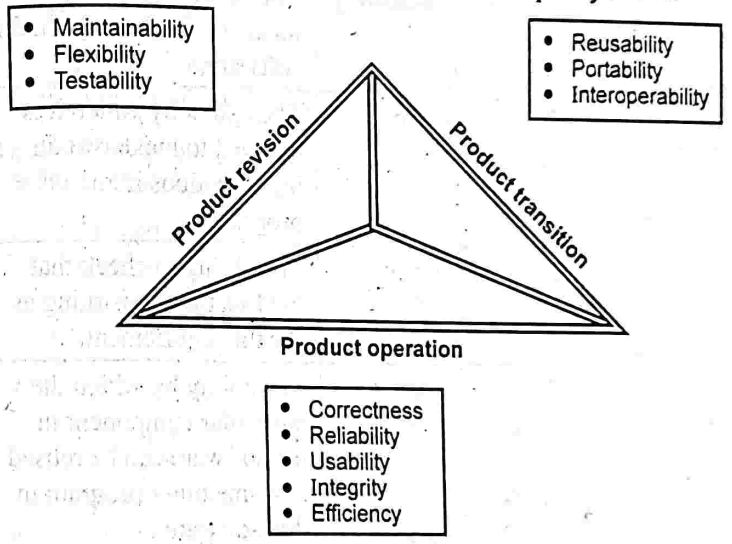


Fig. 3 McCall's triangular model (Three Dimensions)

- These factors can be described as below.

Operational characteristics	1.	Correctness	The ability to fulfil the specification and customers requirements.
	2.	Reliability	The degree by which the software should work as per the requirement precision

	3.	Usability	The ability to prepare the valid input and characteristics interpret the correct output.
	4.	Efficiency	The measure of computing resources and time required by the program to perform.
	5.	Integrity	This is the controlling ability by which unauthorized access to the system can be prevented.
Adaptability to new environment	6.	Maintainability	The ability required to locate or fix the bugs in the software.
	7.	Flexibility	The extent by which it is allowed to make certain modifications in the program.
	8.	Testability	The ability to check that the function is working as per the requirement.
Ability toundergo change	9.	Reusability	The ability by which the particular component in the software can be reused by some other program in that software.
	10.	Portability	The ability of the software to work properly even if the environment gets changed(i.e. change in hardware or software).
	11	Interoperability	The ability of the system to work with other system.

b) *Discuss Garvin's eight quality dimensions.*

(Refer Q.2 of Chapter - 5) **[8]**

OR

Q.6 a) *Explain unit testing ? Which testing scheme is suitable to remove conflict of interest ?* **[9]**

Ans. : Unit testing : Refer Q.29 of Chapter - 5.

- The role of Independent Test Group (ITG) is to remove the conflict of interest inherent when the builder is testing his or her own product. Hence independent testing scheme is used to remove the conflict of interest.

b) *How do you justify the statement "quality is a complex and multifaceted concept".* **[8]**

Ans. : Quality is complex and multifaceted concept because it focuses on three main dimensions –

(1) Work : The quality is concerned with the performance of various activities or tasks required for building up a system.

(2) People : The quality is also concerned about the management of human resources efficiently and effectively.

(3) Operations : The operations are nothing but the activities of the production life cycle. The operations are interlinked with the work and people. Quality is an important aspect of the operations that are performed during the project life cycle.

Q.7 a) *Explain any Four layers of SCM process in detail.*

(Refer Q.4 of Chapter - 6) **[8]**

b) *Explain CASE taxonomy.* **[9]**

Ans. : • The taxonomy of CASE tools is as given below.

1) Business process engineering tools :

This tool is used to model the business information flow. It represents business data objects, their relationships and how data objects flow between different business areas within a company.

2) Project planning tools :

These tools help to plan and schedule projects. Examples are PERT and CPM. The objective of this tool is finding parallelism and eliminating bottlenecks in the projects.

3) Requirements tracing tools :

The objective of these tools is to provide a systematic approach to isolate customer requirements and then to trace these requirements in each stage of development.

4) Metrics and management tools :

These tools assist to capture specific metrics that provide an overall measure of quality. These tools are indetended to focus on process and product characteristics. For example "defects per function point", "LOC/person-month".

5) Documentation tools :

Most of the software development organizations spend lot of time in developing the documents. For this reason the documentation tools provide a way to develop documents efficiently. For example - Word processors that give templates for the organization process documents.

6) System software tools :

These tools provide services to network system software, object management and distributed component support. For example - e-mail, bulletin boards and www access.

7) Quality assurance tools :

These are actually metrics tools that audit source code to insure compliance with language standards.

8) Database management tool :

It provides consistent interfaces for the project for all data, in particular the configuration objects are primary repository elements.

9) Software configuration management tools :

It assists with identification, version control, change control, auditing and status accounting.

10) Interface design and development tools :

These tools are used in developing user interface. It includes various components such as menu, icons, buttons, scrolling mechanisms etc. For example - JAVA, Visual Studio.

11) Programming tools :

The programming tool category include the programs that support most of the conventional programming languages. For example - compilers, debuggers, editors, database query languages.

12) Web development tools :

These tools help in developing the web based applications. The various components of these tools are text, graphics, form, scripts and applets.

OR

Q.8 a) *Explain in brief risk mitigation, monitoring and management.* **(Refer Q.11 of Chapter - 6)** [9]

b) *Write short note on :*

i) JIRA **(Refer Q.23 of Chapter - 6)**

ii) Kanban **(Refer Q.24 of Chapter - 6)** [8]

END ...