Unit III

3 Estimation and Scheduling

3.1 : The Project Planning Process

Q.1 Explain the project planning process.

Ans. : • Project planning is an activity in which the project manager makes reasonable estimates of resources, cost and Schedule.

- In order to make these estimates, it is necessary to prepare a project plan and the beginning of the project and as the project proceeds this plan can be updated appropriately.
- Following figure represents various task sets that are associated with project planning process.

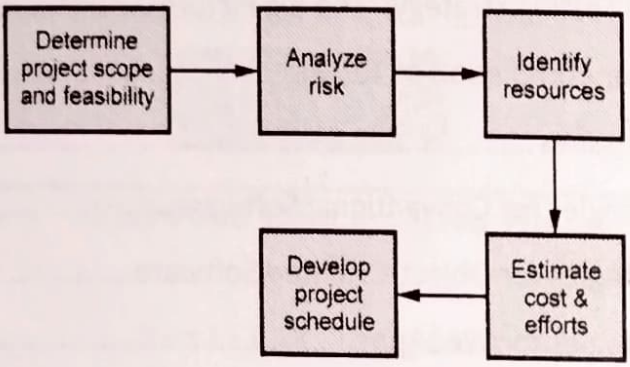


Fig. Q.1.1 Project planning process

Step 1 : This is an **initial stage** in which the **scope and feasibility** of the project is determined. This stage helps to identify the functions and features that can be delivered to the end user.

Step 2 : **Risks** are identified and analyzed.

Step 3 : The required **resources** such as human resources, environmental resources, and reusable software resources can be determined.

Step 4 : The **estimation for cost and efforts** is made. In this stage the problem is first decomposed, then using the size, function point or use cases the estimates are made.

Step 5 : Using the scheduling tools or task networks the **schedule** for the project is prepared.

3.2 : Defining Software Scope and Checking Feasibility

Q.2 Write a short note on - Software scope and its need.

Ans. : Software **scope describes** four things -

- The **function** and **features** that are to be delivered to end-users.
  - The **data** which can be input and output.
  - The **content** of the software that is presented to user.
  - Performance, constraints, reliability and interfaces that bounds the System.
- There are two ways by which the **scope can be defined** -
  1. A scope can be defined using the narrative description of the software obtained after communication with all stakeholders.
  2. Scope can be defined as a set of use cases developed by the end users.
- In the scope description, various **functions** are described. These functions are **evaluated** and **refined** to provide more details before the estimation of the project.
- For **performance consideration**, processing and response time requirements are analyzed.
- The constraints identify the limitations placed on the software by external hardware or any other existing system.

Q.3 What are the four dimensions of software feasibility ?

Ans. : Following are the four dimensions of software feasibility. To ensure the feasibility of the software project the set of questions based on these dimension has to be answered. Those are as given below -

1. Technology

- Is a project technically feasible ?
- Is it within the state of art ?

- Are the defects to be reduced to a level that satisfies the application's need ?

2. Finance

- Is it financially feasible ?
- Is the development cost completed at a cost of software organization, its client, or market affordable ?
- Are the defects to be reduced to a level that satisfies the application's need ?

3. Time

- Will the project's time to market beat the competition ?

4. Resource

- Does the organization have the resources needed to succeed ?
- **Putnam and Meyers** suggests that scoping is not enough. Once scope is understood, and and **feasibility** have been identified the next task is estimation of the **Resources** required to accomplish the software development effort.

3.3 : Resources Management

Q.4 What is the need for defining a software scope ? What are the categories of software engineering resources ?

☞ **[SPPU : Dec.-17,May-18, End Sem, Marks 7]**

Ans. : Need for defining software scope : Refer Q.2.

Categories of software engineering resources :

- There are three major **categories** of software engineering resources -
  1. Peoples
  2. Reusable software components
  3. Development environment i.e. hardware and software tools.

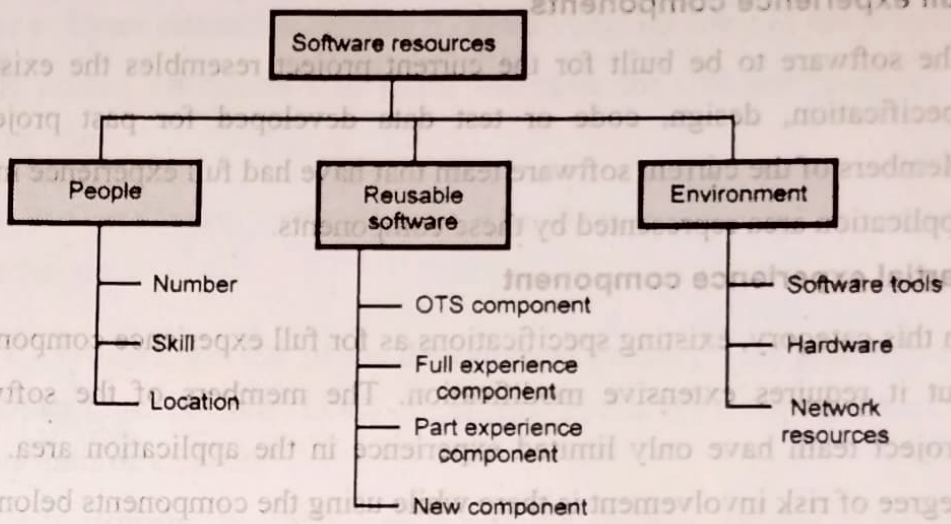


Fig. Q.4.1 Project resources

(1) Human resources :

- Both **organizational position** (such as manager, senior software engineer and so on) and the **specialty** (database telecommunication, client/server etc) are specified for a relatively **small project**. A single individual may perform all software engineering tasks, by consulting with the specialist whenever needed.
- For **larger projects,** software team may be geographically at different locations. Hence, the location of each human resource is specified. The number of people required for a software project can be determined only after an estimate of **development effort** (e.g. person-months) is made.

(2) Reusable software resources :

There are **four reusable software resource categories** that should be considered at **planning** -

1. Off-the-shelf components (OTS Components)

Existing software acquired from a third party or has been developed internally for a past project.

2. Full experience components

The software to be built for the current project resembles the existing specification, design, code or test data developed for past projects. Members of the current software team that have had full experience in the application area represented by these components.

3. Partial experience component

In this category, existing specifications as for full experience components but it requires extensive modification. The members of the software project team have only limited experience in the application area. Fair degree of risk involvement is there while using the components belonging to this category.

4. New components

According to need and specification of the current project the software components can be built by the software team.

(3) Environmental resources :

The Software Engineering Environment (SEE) incorporates hardware and software.

Hardware provides a platform for using the tools required to produce work products. These work products are the outcome of good software Engineering practice.

3.4 : Software Project Estimation

Q.5 What is need of project estimation ? What are the steps while estimation of software ? ☞ **[SPPU : May-19, Dec.-19, End Sem, Marks 8]**

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.

3.5 : Decomposition Techniques

Q.6 What is LOC-based estimation ? Explain with a suitable example.

☞ **[SPPU : Dec.-11, June-22,Marks 8]**

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

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

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

- 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.7 Explain the FP based estimation technique.

☞ **[SPPU : May-18, Dec.-22, End Sem, Marks 6, June-22, Marks 5]**

OR How to calculate FP and how it is used in estimation of software project ?

☞ **[SPPU : Dec.-19, End Sem, 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.6.

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 (F_i).
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.8 Compare the Lines of Code (LOC) and Function Point (FP) based estimation techniques with suitable example.

☞ [SPPU : May-19, Dec 22, End Sem, Marks 8]

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.6.

Example of Function Point Based Metrics : Refer Q.7.

Q.9 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 :

- 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.10 What is project decomposition ? What are the work tasks for communication process using process decomposition ?

☞ **[SPPU : Dec.-15, End Sem, Marks 9, June 22, Marks 8]**

Ans. : Project decomposition is the process of dividing components of a large project into smaller, more manageable portions, often called deliverables. This process helps project managers assign tasks more easily and assists in time management and workflow. As team members complete tasks associated with their deliverables, they complete steps within the decomposed workflow, working toward the eventual completion of an entire project.

Work task for communication process :

- Various **factors that influence** the tasks sets are -
  1. Size of project
  2. Project development staff
  3. Number of user of that project
  4. Application longetivity
  5. Complexity of application
  6. Performance constraints
  7. Use of technologies
- **Task set example :** Consider the concept development type of the project. Various tasks sets in this type of project are -
  - **Defining scope :** This task is for defining the scope, goal or objective of the project.
  - **Planning :** It includes the estimate for schedule, cost and people for completing the desired concept.
  - **Evaluation of technology risks :** It evaluates the risk associated with the technology used in the project.
  - **Concept implementation :** It includes the concept representation in the same manner as expected by the end user.

Q.11 Discuss with suitable example - Process based estimation.

☞ [SPPU : June-22, Marks 5]

Ans. : • 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.

For 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	Risk analysis	Engineering		Construction release		CE	TOTAL
TASK				Analysis	Design	Code	Test		
BUSINESS FUNCTION									
User interface and control facilities (UICF)				0.50	2.75	0.50	4.50	NA	8.25
2D graphics analysis (2DGA)				0.75	4.50	0.75	2.00	NA	8.00
3D Geometric analysis function (3DGA)				0.50	4.50	1.00	3.50	NA	9.5
Database Management (DBM)				0.50	3.25	1.00	1.00	NA	5.75
Computer graphics display				0.50	3.25	0.70	1.50	NA	5.95
Facility (CGDF)									

Peripheral control function (PCF)				0.25	2.00	0.50	1.50	NA	4.25
Design Analysis Modules (DAM)				0.50	2.00	0.50	2.00	NA	5.00
Total	**0.25**	**0.25**	**0.25**	**3.50**	**22.25**	**4.95**	**16.00**		**48**
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

3.6 : Empirical Estimation Models

Q.12 Explain COCOMO II model.

☞ **[SPPU : Dec.-11, Marks 8, Dec.-16, End Sem, Marks 5]**

Ans. : COCOMO II is applied for modern software development practices addressed for the projects in 1990's and 2000's.

The sub-models of COCOMO II model are -

1. Application composition model

- For estimating the efforts required for the prototyping projects and the projects in which the existing software components are used application-composition model is introduced.

- The estimation in this model is based on the number of application points. The application points are similar to the object points.
- This estimation is based on the level of difficulty of object points. Boehm has suggested the object point productivity in the following manner.

Developers experience and capability	Very low	Low	Nominal	High	Very high
CASE maturity	Very low	Low	Nominal	High	Very high
Productivity (NOP/Month)	4	7	13	25	50

- Effort computation in application-composition model can be done as follows -

$$PM = (NAP^{(1-\%reuse/100)}) / PROD$$

where

PM means effort required in terms of person-months.

NAP means number of application points required.

% reuse indicates the amount of reused components in the project. These reusable components can be screens, reports or the modules used in previous projects.

PROD is the object point productivity. These values are given in the above table.

2. An early design model

- This model is used in the early stage of the project development. That is after gathering the user requirements and before the project development actually starts, this model is used. Hence approximate cost estimation can be made in this model.
- The estimation can be made based on the functional points.
- In early stage of development different ways of implementing user requirements can be estimated.

- The effort estimation (in terms of person month) in this model can b made using the following formula :

$$\text{Effort} = A \times \text{size}^{B} \times M$$

where

Boehm has proposed the value of coefficient **A = 2.94**.

Size should be in terms of Kilo source lines of code i.e. KSLOC.

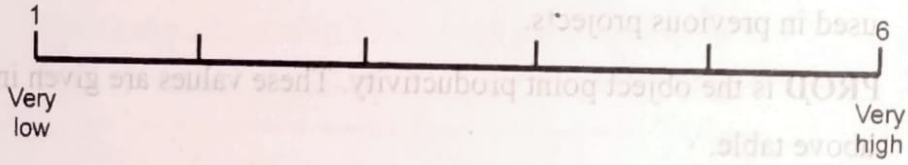
The lines of code can be computed with the help of function point.

The value of **B** is varying from 1.1 to 1.24 and depends upon the project.

M is based on the characteristics such as

1. Product reliability and complexity (RCPX)
2. Reuse required (RUSE)
3. Platform difficulty (PDIF)
4. Personnel capability (PERS)
5. Personnel experience (PREX)
6. Schedule (SCED)
7. Support facilities (FCIL)

These characteristics values can be computed on the following scale -



- Hence the effort estimation can be given as

$$PM = 2.94 \times \text{size}^{B} \times M$$

$$M = RUSE \times PDIF \times PERS \times PREX \times SCED \times FCIL$$

3. A reuse model

- This model considers the systems that have significant amount of code which is reused from the earlier software systems. The estimation made in reuse model is nothing but the efforts required to integrated the reused models into the new systems.

- There are two types of reusable codes : black box code and white box code. The black box code is a kind of code which is simply integrated with the new system without modifying it. The white box code is a kind of code that has to be modified to some extent before integrating it with the new system, and then only it can work correctly.
- There is third category of code which is used in reuse model and it is the code which can be generated automatically. In this form of reuse the standard templates are integrated in the generator. To these generators, the system model is given as input from which some additional information about the system is taken and the code can be generated using the templates.
- The efforts required for automatically the generated code is

$$PM = (ASLOC \times AT / 100) / ATPROD$$

where

AT is percentage of automatically generated code.

ATPROD is the productivity of engineers in estimating such code

- Sometimes in the reuse model some white box code is used along with the newly developed code. The size estimate of newly developed code is equivalent to the reused code. Following formula is used to calculate the effort in such a case -

$$ESLOC = ASLOC \times (1 - AT / 100) \times AAM$$

where

ESLOC means equivalent number of lines of new source code.

ASLOC means the source lines of code in the component that has to be adapted.

AAM is adaptation Adjustment multiplier. This factor is used to take into the efforts required to reuse the code.

4. Post architecture model

- This model is a detailed model used to compute the efforts. The basic formula used in this model is

$$\text{Effort} = A \times \text{Size}^{B} \times M$$

- In this model efforts should be estimated more accurately. In the above formula **A** is the amount of code. This code size estimate is made with the help of three components -

1. The estimate about **new** lines of code that is added in the program.
2. Equivalent number of Source Lines Of Code (ESLOC) used in **reuse model**.
3. Due to changes in requirements the lines of code get modified. The estimate of amount of **code** being **modified**.

The exponent term **B** has three possible values that are related to the levels of project complexity. The values of B are continuous rather than discrete. It depends upon the five scale factors. These scale factors vary from very low to extra high (i.e. from 5 to 0).

- These factors are -

Scale factor for component B	Description
Precedentedness	This factor is for previous experience of organisation. Very low means no previous experience and high means the organisation knows the application domain.
Development flexibility	Flexibility in development process. Very low means the typical process is used. Extra high means client is responsible for defining the process goals.
Architecture / risk resolution	Amount of risk that is allowed to carry out. Very low means little risk analysis is permitted and extra high means high risk analysis is made.
Team cohesion	Represents the working environment of the team. Very low cohesion means poor communication or interaction between the team members and extra high means there is no communication problem

	and team can work in a good spirit.
Process maturity	This factor affects the process maturity of the organisation. This value can be computed using Capacity Maturity Model (CMM) questionnaire, for computing the estimates CMM maturity level can be subtracted from 5.

- Add up all these rating and then whatever value you get, divide it by 100. Then add the resultant value to 1.01 to get the exponent value.
- This model makes use of 17 cost attributes instead of seven. These attributes are used to adjust initial estimate.

Cost attribute	Type of attribute	Purpose
RELY	Product	System reliability that is required.
CPLX	Product	Complexity of system modules
DATA	Product	Size of the data used from database
DOCU	Product	Some amount of documentation used
RUSE	Product	Percentage of reusable components
TIME	Computer	Amount of time required for execution
PVOL	Computer	Volatility of development platform.
STOR	Computer	Memory constraint
ACAP	Personnel	Project analyst's capability to analyse the project.
PCAP	Personnel	Programmer capability
PCON	Personnel	Personnel continuity
PEXP	Personnel	Programmer's experience in project domain.
LTEX	Personnel	Experience of languages and tools that are used.

AEXP	Personal	Analyst's experience in project domain.
TOOL	Project	Use of software tools
SCED	Project	Project schedule compression.
SITE	Project	Quality of inter-site and multi-site working.

Q.13 Explain COCOMO model for project estimation with suitable example

☞ [SPPU : May-16, Dec.-17, 18, End Sem, Marks 7, Dec.-22, Marks 9]

Ans. : Refer Q.12.

3.7 : Object Point based Estimation

Q.14 Discuss with suitable example - Object Point (OP) based estimation. ☞ [SPPU : June-22, Marks 5]

Ans. : Lorenz and Kidd suggested the estimation technique for object oriented project. The steps for this estimation technique are as follows :

1. Using requirement model, develop use cases and determine the count. As the project proceeds, the **number of use cases** may get changed.
2. Using the requirement model, determine the **number of key classes**.
3. Categorize the type of interface and develop the **multiplier**. For example

Interface Type	Multiplier
No GUI	2.0
Text based interface	2.25
Graphical User Interface	2.5
Complex GUI	3.0

Multiply the key classes by these multipliers to obtain the number of **support classes.**

4. Now multiply the key and support classes by average number of work units per class. Approximately there are 15-20 person-days per class.

3.8 : Project Scheduling

Q.15 What is project scheduling ? What are the basic principles of project scheduling ? [SPPU : Dec.-16, 22, End Sem. Marks 6, Dec.-19, June-22, End Sem, Marks 5]

Ans. : Definition : Software project scheduling can be defined as an activity that distributes the estimated effort across the planned project duration by allocating the effort to specific software engineering tasks.

1. **Compartmentalization :** The project must be compartmentalized into a number of manageable activities, actions and tasks. To accomplish compartmentalization, both the product and the process are decomposed.
2. **Interdependency :** The interdependency of each compartmentalized activity, action, or task must be determined. Some tasks must occur in sequence while others can occur in parallel.
3. **Time allocation :** Each task to be scheduled must be allocated some number of work units. In addition, each task must be assigned a start date and a completion date that are a function of the inter dependencies and whether work will be conducted on a full-time or part-time basis.
4. **Effort validation :** As time allocation occurs, the project manager must ensure that no more than the allocated number of people have been scheduled at any given time.
5. **Defined responsibilities :** Every task that is scheduled should be assigned to a specific team member.
6. **Defined outcomes :** Every task that is scheduled should have a defined outcome. For software projects, the outcome is normally a work product or a part of work product.
7. **Defined milestones :** Every task or group of tasks should be associated with a project milestone. A milestone is accomplished when one or more work products has been reviewed for quality.

3.9 : Defining a Task Set for the Software Project

Q.16 Define task set for the software project. Also enlist the various types of projects.

Ans. : • **Definition of task set :** The task set is a collection of software engineering **work tasks**, **milestones**, and **work products** that must be accomplished to complete particular project.

- Every process model consists of various tasks sets. Using these tasks sets the software team define, develop or ultimately support computer software.
- There is no single task that is appropriate for all the projects but for developing large, complex projects the set of tasks are required. Hence every effective software process should define a collection of task sets depending upon the type of the project.
- Using tasks sets the **high quality software** can be developed and any unnecessary work can be avoided during software development.
- The number of tasks sets will vary depending upon the type of the project. Various **types of projects** are enlisted below -

1. **Concept Development project :** These are the projects in which new business ideas or the applications based on new technologies are to be developed.
2. **New application development project :** These projects are developed for satisfying a specific customer need.
3. **Application upgradation project :** These are kind of projects in which existing software application needs a major change. This change can be for performance improvement, or modifications within the modules and interfaces.
4. **Application maintenance project :** These are kind of projects that correct, adapt or extend the existing software applications.
5. **Reengineering projects :** These are the projects in which the legacy systems are rebuilt partly or completely.

3.10 : Task Network

Q.17 What is a task network in project scheduling ? Explain with an example.

☞ [SPPU : May-11, 18, 19, End Sem, Marks 6, Dec.-17, End Sem, Marks 8]

OR What is a task network ? How it is used in scheduling of software project ? ☞ [SPPU : May -11, Dec.-13, Marks 8]

Ans. : • The task is a small unit of work.

- The task network or an activity network is a graphical representation, with :

  (1) Nodes corresponding to activities.

  (2) Tasks or activities are linked if there is a dependency between them.

  (3) The task network for the product development is as shown in Fig. Q.17.1.

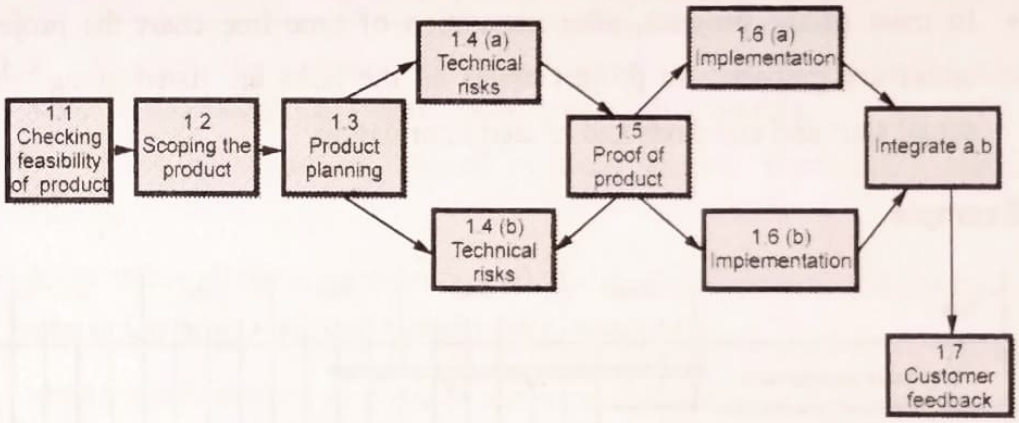


Fig. Q.17.1 Task network

- 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.

3.11 : Scheduling with Time Line Chart

Q.18 What is time line chart ? How it is used in scheduling of software project ? ☞ [SPPU : Dec.-12, Marks 8, Dec.-16, End Sem, Marks 5, May -18, End Sem, Marks 6

Ans. : • In software project scheduling the timeline chart is created. The **purpose** of timeline chart is to emphasize the **scope of individual task**. Hence set of tasks are given as input to the time line chart.

- The time line chart is also called as **Gantt chart**.
- The time line chart can be developed for entire project or it can be developed for individual functions.
- In time line chart
  1) All the tasks are listed at the leftmost column.
  2) The horizontal bars indicate the time required by the corresponding task.
  3) When multiple horizontal bars occur at the same time on the calendar, then that means concurrency can be applied for performing the tasks.
  4) The diamonds indicate the milestones.
- In most of the projects, after generation of time line chart the project tables are prepared. In project tables all the tasks are listed along with actual start and end dates and related information.

Example

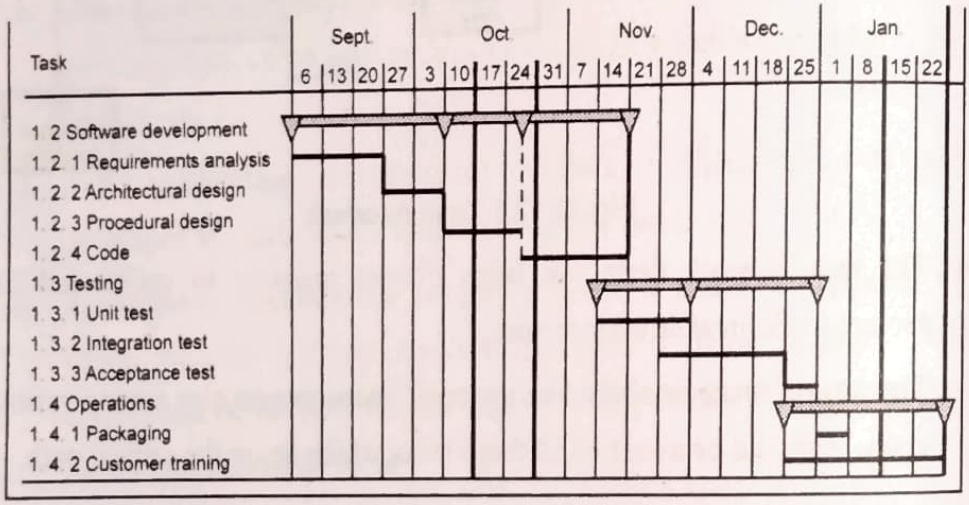


Fig. Q.18.1 Time line chart

3.12 : Schedule Tracking Tools

Q.19 Write short note on - DART.

Ans. : • This tool enables you to track the changes to record being made by users.

- Many organizations deploy the Daily activity reporting and tracking tool that monitors the progress of the software project.
- DART collects the activity data and keep track of the work flow.
- DART can compute the set of business queries which are also called as indicators that can be used to keep track of the performance.
- The aim of DART is to empower software project stakeholders in their daily discussion regarding the performance of the project.

3.8 : Preparing Requirement Traceability Matrix

Q.20 Explain the concept of requirement traceability matrix. Also specify the advantages of requirement traceability matrix.

Ans. : • Requirements are tracked by Requirements Traceability Matrix (RTM).

- RTM traces all the requirements from design, development, and testing.
- The traceability matrix is typically a worksheet that contains the requirements with its all possible test scenarios and cases and their current state, i.e. if they have been passed or failed. This would help the testing team to understand the level of testing activities done for the specific product.

Parameters to be included in requirement traceability matrix

(1) Requirement ID

(2) Requirement type and description

(3) Test cases

(4) Test case status

Example

Test case ID	Test case description
TC=001	Login with invalid username but valid password
TC=002	Login with valid username but invalid password
TC=003	Login with valid user name and valid password

Requirement ID	Requirement Description	Testcase ID	Status
101	Login the website	TC=001	Pass
		TC=002	Pass
		TC=003	Pass

Advantages of requirements traceability matrix

1. It confirms 100 % test coverage.
2. It highlights any requirements missing or document inconsistencies.
3. It shows the overall defects or execution status with a focus on business requirements.
4. It helps in analysing or estimating the impact on the QA team's work with respect to revisiting or re-working on the test cases.

END...

Unit IV

4 Design Engineering

Part I : Design Concepts

4.1 : Introduction to Design

Q.1 Explain the term – 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.

Q.2 What are the characteristics of a well formed design class ?

☞ [SPPU : Dec.-12, Marks 6]

Ans. :

1. The good design should **implement all the requirements** specified by the customer. Even if there are some implicit requirements of the software product then those requirements should also be implemented by the software design.
2. The design should be **simple** enough so that the software can be understood easily by the developers, testers and customers.
3. The design should be **comprehensive**. That means it should provide complete picture of the software.

4.2 : The Design Process

Q.3 Elaborate software design process

Ans. : • Software design is an **iterative process** in which the requirements are translated into the blueprint of the software. Initially the software is represented at high level of abstraction, but during the subsequent iterations data, functional and behavioural requirements are traced in more detail.

- Thus refinement made during each iteration leads to design representations at much lower level of abstraction. Throughout the software design process the quality of the software is assessed by considering certain characteristics of the software design.

4.3 : Software Quality Guidelines and Attributes

Q.4 Explain the quality attributes, considered in software design.

☞ [SPPU : Dec.-12, Marks 8, Dec.-16, End Sem, Marks 5]

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.5 What are the design quality guidelines ? ☞ **[SPPU : May-14, Marks 8]**

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.6 What are the software design quality attribute and guidelines ? ☞ **[SPPU : Dec.-19, End Sem, Marks 7]**

Ans. : Refer Q.4 and Q.5.

4.4 : Design Concepts

Q.7 Explain the following design concepts. i) Refinement ii) Modularity iii) Architecture. ☞ **[SPPU May-11, Marks 6, May-13, Marks 8]**

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.**
- 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.
- **Meyer** defines five criteria that enable us to evaluate a design method with respect to its ability to define an effective modular system :

1. **Modular decomposability** : A design method provides a systematic mechanism for decomposing the problem into sub-problems. This reduces the complexity of the problem and the modularity can be achieved.
2. **Modular composability** : A design method enables existing design components to be assembled into a new system.
3. **Modular understandability** : A module can be understood as a standalone unit. It will be easier to build and easier to change.

4. **Modular continuity** : Small changes to the system requirements result in changes to individual modules, rather than system-wide changes.

5. **Modular protection** : An aberrant condition occurs within a module and its effects are constrained within the module.

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.8 Differentiate between abstraction and refinement.

☞ [SPPU : Dec.-12, June 22, Marks 4]

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.9 List the design concepts. Explain refinement and refactoring. Give the importance of Refactoring in improving the quality of software.

☞ [SPPU : June-22, Marks 9]

Ans. : Design concepts are -

i) Abstraction

ii) Modularity

iii) Architecture

iv) Refinement

v) Pattern

vi) Information Hiding

vii) Functional Independence

viii) Refactoring

Refinement : Refer Q.7(i).

Refactoring : Refer Q.10.

Q.10 What do you understand by refactoring ? Give the importance of refactoring in improving quality of software.

☞ [SPPU : May-15, End Sem, Marks 6, Dec.-18, End Sem, Oct.-19, In Sem, Marks 5]

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".

Benefits of refactoring are -

- 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.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, End Sem, Marks 5, Aug.-17, In Sem, Marks 3]

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.

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.

iii) Architecture : Refer Q.7(iii).

iv) Refinement : Refer Q.7(ii).

Q.16 What is the relationship between modularity and functional dependence ? ☞ [SPPU : Dec.-18, End Sem, Marks 5]

Ans. : • 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.**

Q.17 Explain about various design concepts considered during design.

☞ **[SPPU : April-15, In Sem, Marks 6, Aug.-17, In Sem, Marks 3]**

Ans. : Refer Q.7 and Q.13.

Q.18 Why modularity is important in software projects ?

Ans. : Modularity is important in software projects because of following reasons :

(1) Modularity reduces complexity and helps in easier implementation.

(2) The parallel development of different parts of the system is possible due to modular design.

(3) Changes made during testing and maintenance become manageable and they do not affect the other modules.

Q.19 Differentiate between : Abstraction and modularization.

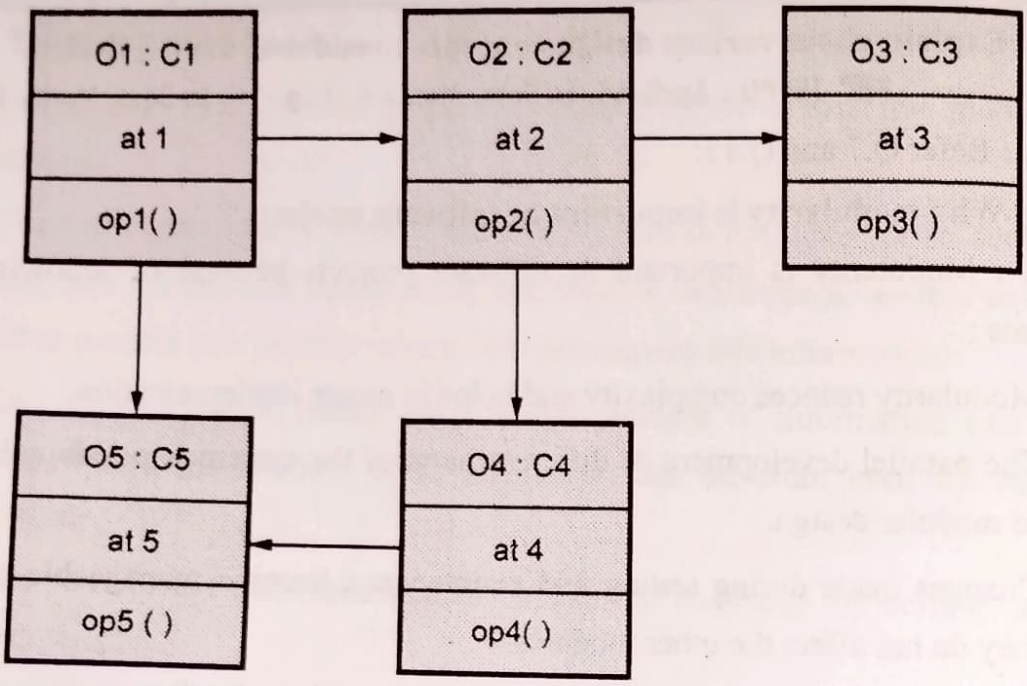
Ans. : Abstraction is a software design concept which is applied to refine software solution. In the process of abstraction only necessary information is abstracted form requirement analysis to create the software solution in broad terms. While moving through different levels of abstraction the procedural and data abstraction are prepared. During abstraction the procedural and data objects are prepared.

Modularity is a design concept in which requirements are divided into separately named and addresable components called as modules. Modularity in design reduces complexity and helps in easier implementation.

4.5 : Object-Oriented Design Concept

Q.20 Explain object oriented design.

Ans. : Object is an important element of an Object Oriented Design (OOD). Various objects and their interaction with each other objects is graphically represented in object oriented design. An object is an entity which can be obtained from its belonging class and it posses attributes and operations. The typical presentation of object oriented system is depicted by following Fig. Q.20.1.



4.6 : Design Classes

Q.21 Explain the types of design classes. ☞ **[SPPU : Dec.-13, Marks 6]**

Ans. : Design classes are defined as the classes that describe some elements of problem domain, focus on various aspects of problem from user's point of view.

The goals of design classes is :

1. To refine the analysis classes by providing the detail design, so that further implementation can be done easily.
2. To create new set of classes for implementing the infrastructure of the software.

There are five different types of design classes.

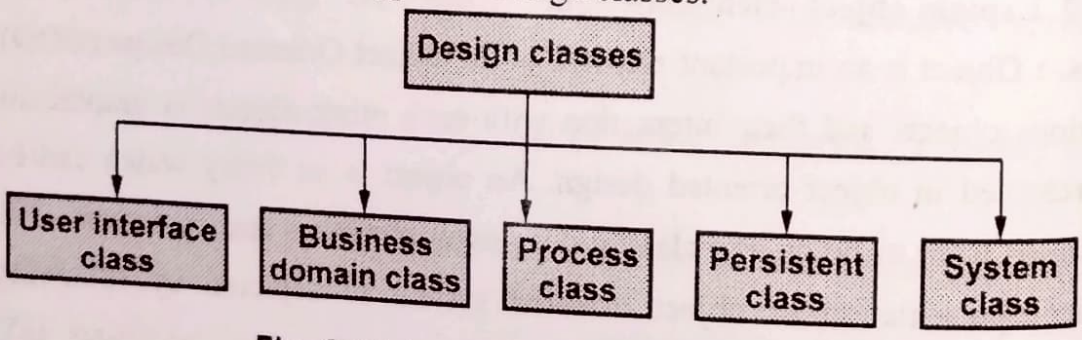


Fig. Q.21.1 Types of design classes

1. **User interface class :** The user interface class defines all the abstractions that are necessary for Human Computer Interface(HCI). The user interface classes is basically a **visual representation** of the HCI.
2. **Business domain class :** Business domain classes are the refinement of analysis classes. These classes identify the **attributes and services** that are needed to implement some elements of **business domain.**
3. **Process class :** Process class implement lower level business abstractions used by the business domain.
4. **Persistent class :** These classes represent the data store such as databases which will be retained as it is even after the execution of the software.
5. **System class :** These classes are responsible for software management and control functions that are used for system operation.

4.7 : The Design Model

Q.22 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.22.1 the **dashed line** shows the boundary between analysis and design model. (See Fig. Q.22.1 on next page.)
- 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.

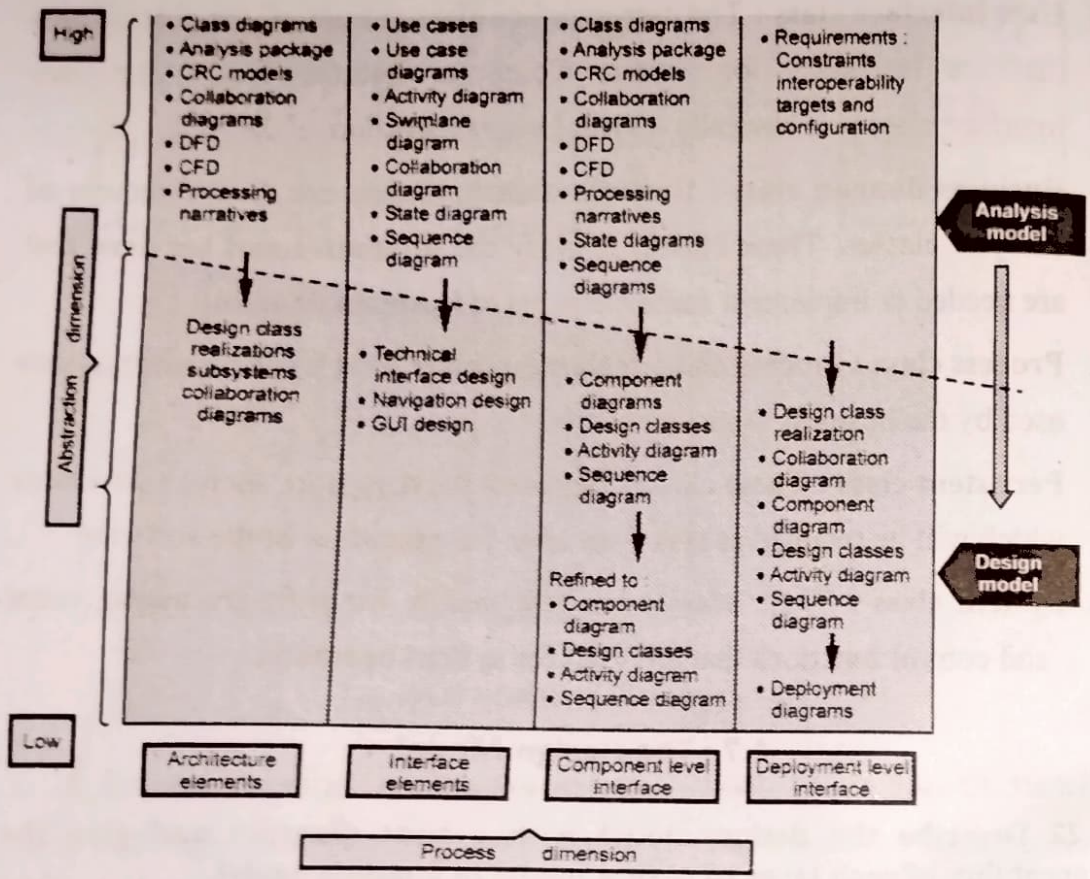


Fig. Q.22.1 Dimension of design model

- 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.

Q.23 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 -

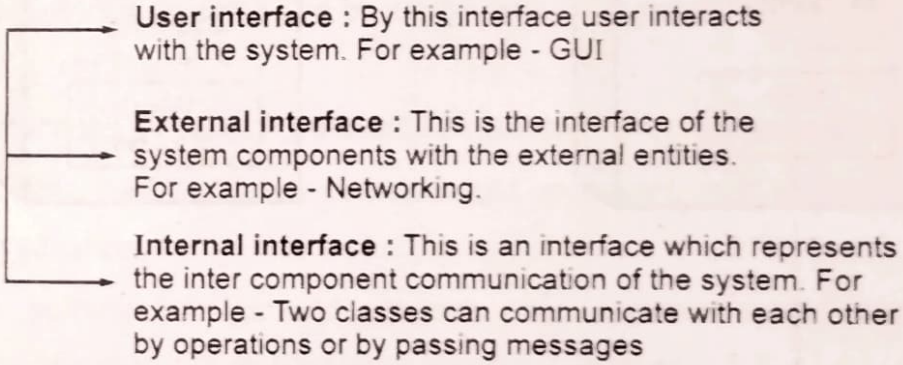


Fig. Q.23.1

(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.23.2 represents that component **order** makes use of another component **form**.

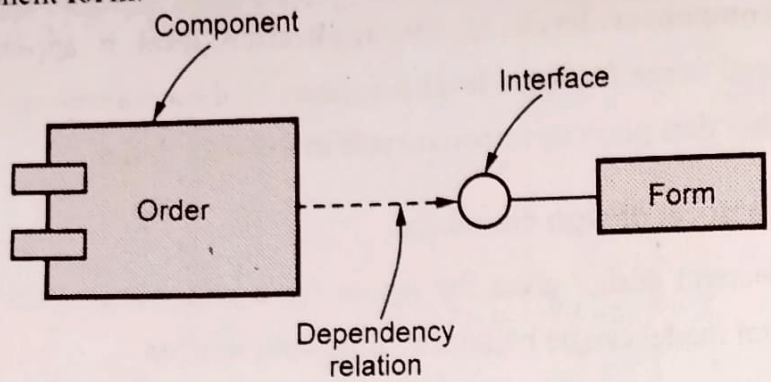


Fig. Q.23.2 Components

- The **order** is dependent upon the component form. These two objects can be interfaced with each other.

(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.

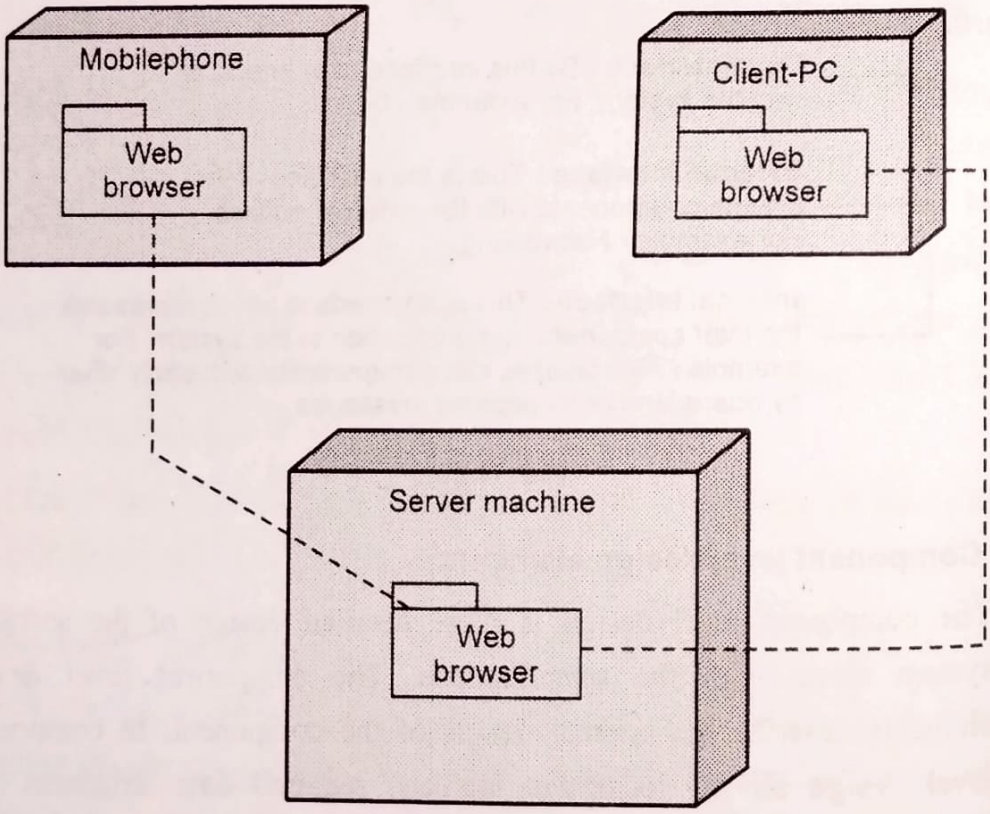


Fig. Q.23.3 Deployment diagram

For example web browsers may work in mobile phoned or they may run on client PC or can execute on server machines.

Q.24 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 :

1) **Data structure :** At the program component level, the data structure is associated with algorithms

2) **Databases :** At the application level data model is converted to databases.

3) **Datawarehouse :** At the business level, the collection of information stored in various databases is identified and Datawarehouse is formed.

Guideline for data design :

1) The systematic analysis principles applied to function and behaviour should also be applied to data.

2) All the data structures and operations to be performed should be identified.

3) A data dictionary should be established and used to define both data and program design.

4) Low level data decisions should be deferred until late in the design.

5) A library of useful data structures and operations must be developed well in advance.

(6) A software design and programming language should support the specification and realization of abstract data types.

Q.25 What are the deployment level design elements ?

☞ **[SPPU : Dec.-11, Marks 4]**

Ans. : Refer Q.23(5).

Q.26 Enlist the golden rules of user interface design.

☞ **[SPPU : Dec.-22, Marks 6]**

Ans. : 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.

1. Place the user in control

- Following are the design principles that allow user to control the system :

i) **Define the interaction modes in such a way that user will be restricted from doing the unnecessary actions.**

ii) **The interaction should be flexible :** The user interaction should be flexible. For example in the Microsoft power point slide show the slide transition is possible using mouse clicks as well as using keyboard. Such flexibility allows any user to operate the system as per his comfort.

iii) **Provide the facility of 'undo' or 'interruption' in user interaction**

iv) **Allow user to customize the interaction :** It is observed that in while handling user interface certain actions need to be done repeatedly. It saves the time if these actions are collected in a Macro.

v) **Hide technical details from the user :** This feature is essential for a casual user. User should not be aware of system commands, operating system functions or file management functions.

vi) **The objects appearing on the screen should be interactive with the user.**

2. Reduce the user's memory load

- Following are the principles suggested by Mandel to reduce the memory load of the user.

1. **Do not force the user to have short term memory :** 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.

3. Make the interface consistent

- The user interface should be consistent. This consistency can be maintained at three levels such as,
  - The visual information (all screen layouts) should be consistent throughout and it should be as per the design standards.
  - There should be limited set of input holding the non conflicting information.
  - The information flow transiting from one task to another should be consistent.

4.8 : Component-Level Design

Q.27 Describe notations used for deployment diagram. Describe the importance of deployment diagram. ☞ [SPPU : Dec.-22, Marks 9]

Ans. : Notations :

Node :

- A node is a physical element that exists at run time and represents a computational resource. It generally has at least some memory and some processing capability.
- Graphically the node is represented as cube.
- **Example -**

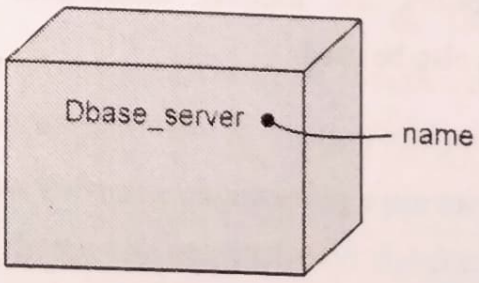


Fig. Q.27.1 Node

Components :

Components are nothing but the physical packaging of the logical elements such as classes, interfaces and collaborations.

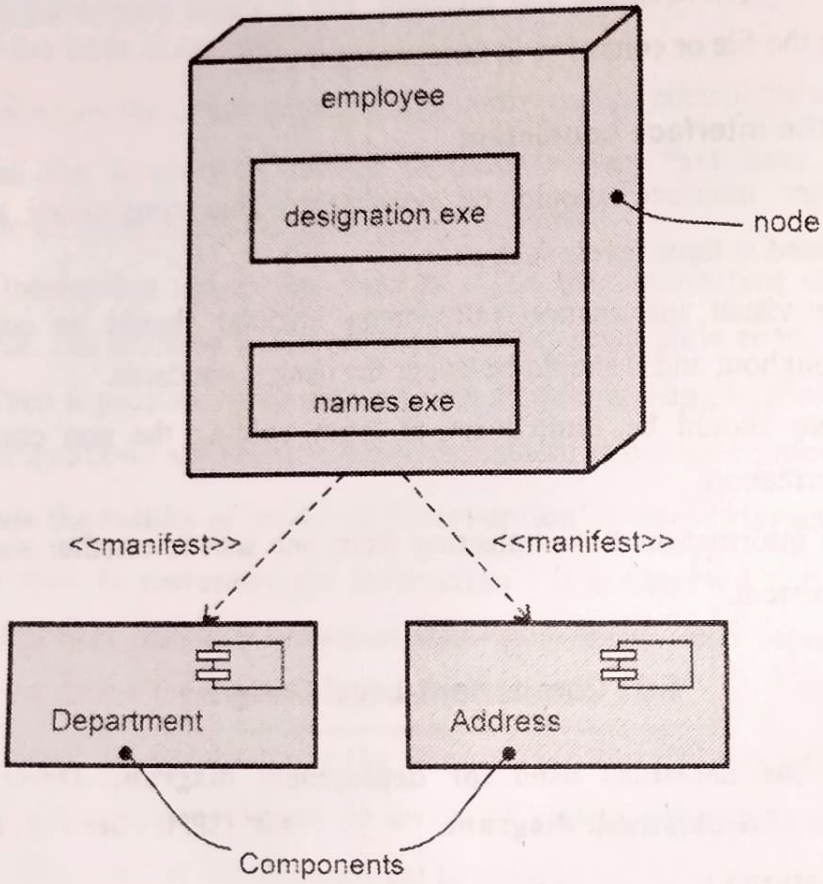


Fig. Q.27.2 Nodes and components

Connections :

- The association is the most common relationship used by the nodes among themselves. The association is actually a physical relationship such as Ethernet connection, shared bus or RS -232 cables.
- For distant processors the indirect connections such as satellite communication can also be used.

- **Example -**

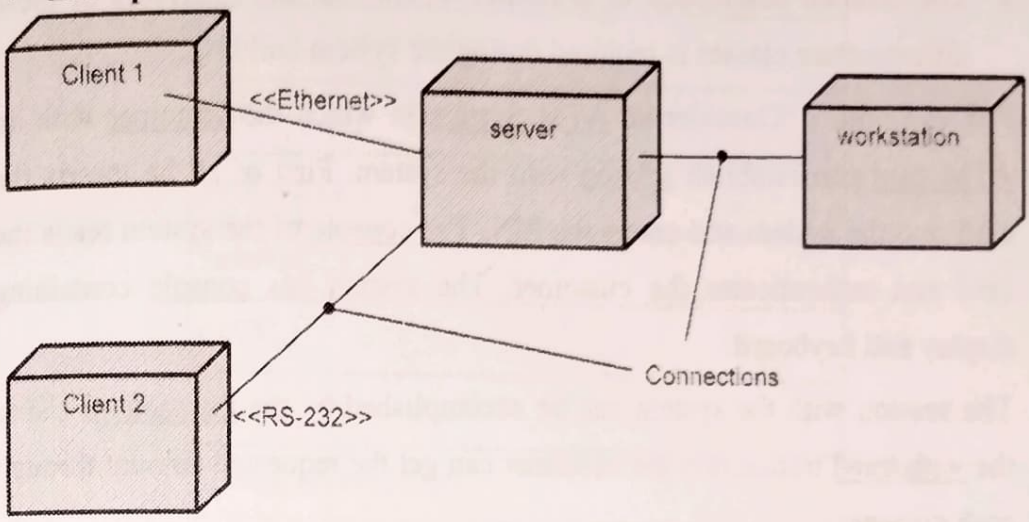


Fig. Q.27.3 Connections

Importance : Following applications of deployment diagram illustrates its importance.

1) The deployment diagram is used to represent the deployment of software elements to various hardware components.
2) This diagram illustrates the runtime processing of hardware.
3) It is used to model the network and hardware topology of a system.
4) It is also used to model distributed networks and systems.
5) The deployment diagram is used for modelling the embedded system.

4.9 : Class Based Design

Q.28 Explain class based modeling with suitable example.

☞ **[SPPU : April-16, In Sem, Marks 4, May-16, End Sem, Marks 7]**

OR Explain object oriented view of component level design with suitable example. ☞ **[SPPU : Dec.-22, Marks 6]**

Ans. : • Component is represented as a part of architectural model. It collects the information about the system as a part of analysis model.

- If **object oriented software engineering approach** is adopted then the component level design will emphasize on elaboration of analysis classes and refinement of **infrastructure classes.**

- The detailed description of attributes, operations and interfaces of these infrastructure classes is required during the system building.

For example : "Consider an ATM System in which the customer with an ATM card can establish session with the system. First of all he inserts the card into the system and enters the PIN. The console of the system reads the card and authenticates the customer. The system has console containing display and keyboard.

The session with the system can be accomplished by the transaction. Using the withdrawl transaction the customer can get the requested amount through cash dispenser.

In above problem statement underlined words denote objects of the system. Hence -

1) ATM system is controller object.
2) Individual components can be
   - Card reader
   - Console
   - Cash dispenser
3) The initiating object is customer.
4) The controlling objects are
   - Session
   - Transaction

Various objects can then be as shown below.

ATM System	
int	id
Char	Bankname[20]
Char	BankAdd[20]
int	State
int	Switch_ON

int Switch_OFF
int Customer_State
run ( )
Switchon ( )
Switchoff ( )
getID ( )
getplace ( )
Startup_system ( )
Shutdown_system ( )

Card reader
readCard ( )
ejectCard ( )
keepCard ( )

Console
Char mesage
display ( )
readPIN ( )
readMenu ( )
readAmt ( )

Cash dispenser
int money
Check_initial_amt ( )
dispense_amp ( )

Customer
name
PIN
ACC_ID
Address
enter_PIN( )
select_choice ( )

Q.29 Explain the guidelines of component level design.

[SPPU : Dec.-17, End Sem, Dec.-18, End Sem, Marks 5, June 22, Marks 9, Dec.-22, Marks 6]

Ans. : • **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.

4.10 : Conduction Component Level Design

Q.30 Enlist the steps that are applied for component level design.

Ans. : Following is a set of steps that are applied for component level design -

1. Identify all the design classes from the problem domain.
2. Identify all the design classes from the infrastructure domain.
3. Detail out all the design classes which are not the reusable components.
   a. When components communicate or collaborate then represent their communication using messages.
   b. Identify interfaces for each component.
   c. Define the attributes of the classes by specifying the data types and deciding the data structures.
   d. Specify all the operations by describing the processing flow within each operation.

4. Specify the databases and files and identify the classes that are involved in handling them
5. Describe the behavioural representation for the class.
6. Design the deployment diagram fro providing additional implementation.
7. Factor the component level design.
8. Find out alternative design solutions as well.

4.11 : Component Level Design for Web Apps

Q.31 Write short note on - Functional design at the component level.

Ans. : • Modern web applications provide various **useful functionalities** such as :

i) Processing of contents and navigation in **dynamic fashion**.

ii) Performing **computations or data processing operations** dynamically.

iii) **Efficient query processing** and access to database.

iv) Establishing **data interfaces** with external corporate systems.

- To achieve these functionalities, it is necessary to design and construct the **functional components** of web applications.
- The Web contents and above mentioned functionalities are combined to create functional architecture.
- A **functional architecture** is a representation of the functional domain of the WebApplications and describes the key functional components in the WebApplications and how these components interact with each other.

Part II : Architectural Design

4.12 : Software Architecture

Q.32 Explain software architecture as a concept of software design.

☞ [SPPU : May-12, Marks 4]

OR What do you mean by software architecture ? Explain with suitable example.

☞ [SPPU : April-16, In Sem, Marks 4]

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.

There are three reasons why the software architecture is so important ?

1. Software architecture gives the representation of the computer based system that is to be built. Using this system model even the stakeholders can take active part in the software development process. This helps in clear specification/understanding of requirements.
2. Some early design decisions can be taken using software architecture and hence system performance and operations remain under control.
3. The software architecture gives a clear cut idea about the computer based system which is to be built.

4.13 : Architectural Styles

Q.33 What are different architectural styles exist for software ? Explain any one software architecture in detail.

☞ [SPPU : April-16, In Sem, Marks 4, June-22, Marks 8]

Ans. : The commonly used architectural styles are

1. Data centered architectures.
2. Data flow architectures.
3. Call and return architectures.
4. Object oriented architectures.
5. Layered architectures.

Q.34 Explain in detail data-centered architectural style.

☞ [SPPU : May-18, End Sem, Dec.-22, Marks 5, June-22, Marks 8]

Ans. : In this architecture the data store lies at the centre 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.

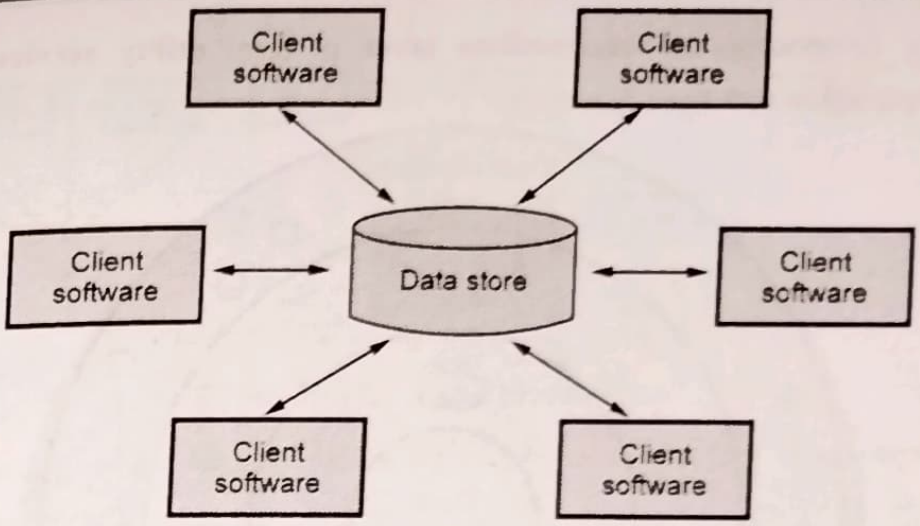


Fig. Q.34.1 Data centered architecture

Data centered architecture posses 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.

Q.35 Explain layered architecture style with neat diagrams.

☞ [SPPU : Dec.-18, 22, Marks 5, June-22, Marks 8]

Ans. : • The layered architecture is composed of different layers. **Each layer** is intended to perform **specific operations** so machine instruction set can be generated. Various components in each layer perform specific operations.

- The **outer layer** is responsible for performing the **user interface** operations while the components in the inner layer perform operating system interfaces.

- The components in **intermediate layer** perform **utility services** and **application software** functions.

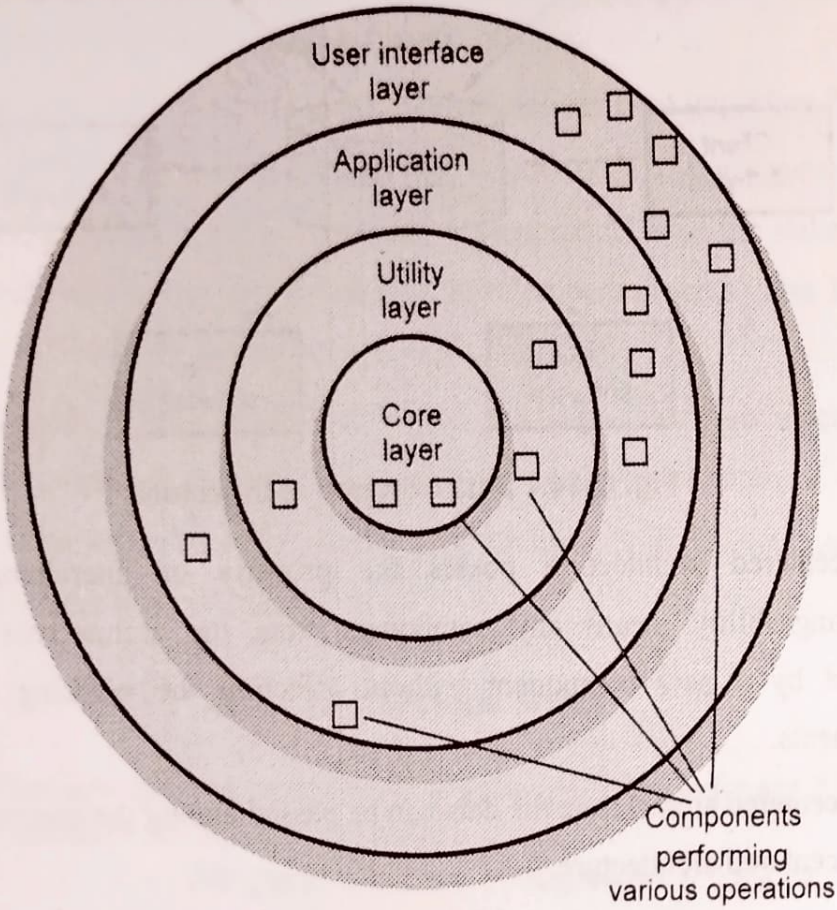


Fig. Q.35.1 Layer architecture of component

Q.36 Explain data centred and layered architectures with neat diagrams.

☞ [SPPU : Dec.-11, Marks 8]

Ans. : Data centered architecture : Refer Q.34.

Layered architecture : Refer Q.35.

Q.37 Explain data flow architecture style with neat diagram.

☞ [SPPU : Dec.-11, 22, May-12, Dec.-14, Oct.-19, In Sem, Marks 5]

Ans. : In this architecture series of transformations are applied to produce the output data. The set of components called filters are connected by pipes to transform the data from one component to another. These filters work independently without a bothering about the working of neighbouring filter.

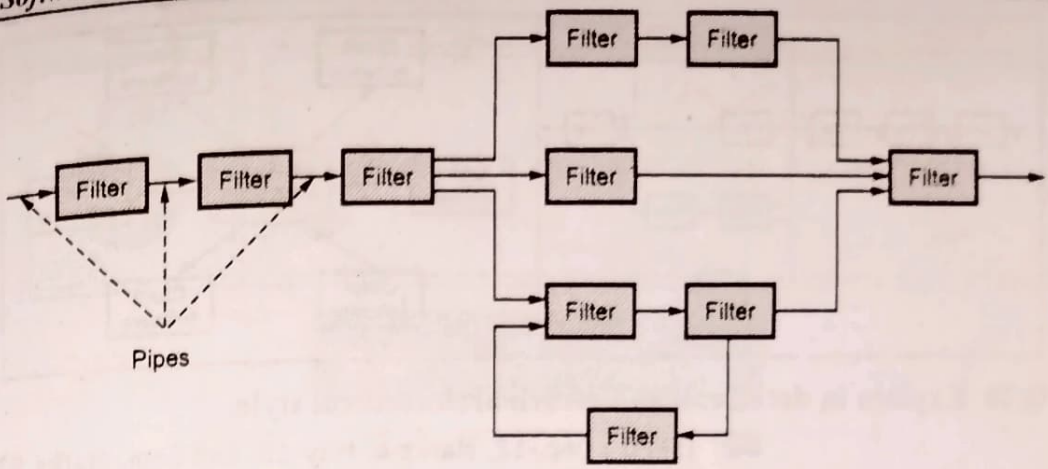


Fig. Q.37.1 Pipes and filters

If the data flow degenerates into a single line of transforms, it is termed as batch sequential.

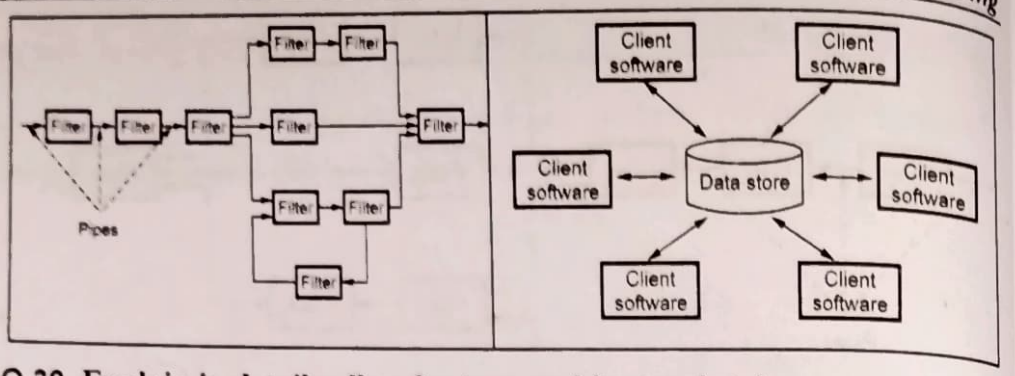
Fig. Q.37.2 Batch sequential

In this pattern the transformation is applied on the batch of data.

Q.38 What is the difference between data flow architecture and data centered architecture ?

Ans. :

Data flow architecture	Data centered architecture
In this architecture series of transformations are applied to produce output data.	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.
In this architecture the set of components are called filters and are connected by pipes to transform data from one component to another.	In data centered architecture the components are the database elements such as tables and queries. And the communication among these components takes place with the help of relationship.
The filters normally work independently without bothering about working of neighboring filter.	The components of this architecture perform with the central repository without affecting the working of other components.



Q.39 Explain in detail call and return architectural style.

[SPPU : Dec.-12, Marks 4, May-19, End Sem, Marks 5]

Ans. : • The program structure can be easily modified or scaled. The program structure is organized into modules within the program. In this architecture how modules call each other. The program structure decomposes the function into control hierarchy where a main program invokes number of program components.

- In this architecture the hierarchical control for call and return is represented.

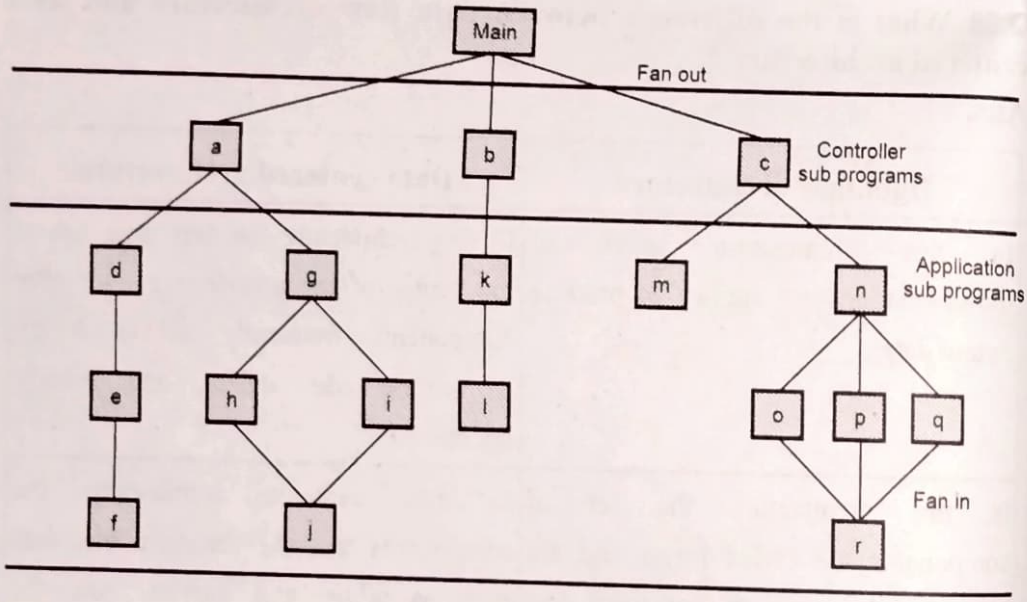


Fig. Q.39.1 Call and return architecture

Q.40 Explain any 3 architecture styles. **[SPPU : May-12, Marks 6]**

Ans. : Refer Q.35, Q.36 and Q.39.

END...

Unit V

5 Risks and Configuration Management

Part I : Risk Management

5.1 : Introduction to Risk Management

Q.1 Define software risk in detail. What are different types of risk ?

☞ [SPPU : Dec.-22, Marks 6]

Ans. : Software risk :

- **Definition of risk :** The risk denotes the uncertainty that may occur in the choices due to past actions and risk is something which causes heavy losses.
- **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.

Q.2 What are the types of risks ? Explain in brief.

☞ [SPPU : May-14, Marks 6]

Ans. :

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.

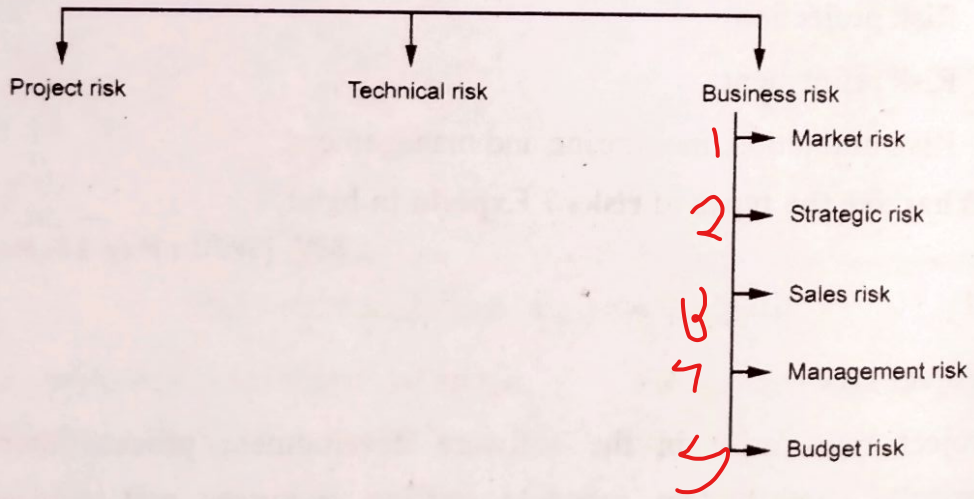


Fig. Q.2.1 Categorization of risk

5.2 : Software Risks

Q.3 Explain various risk associated with software project. How they are managed ? [SPPU : Dec.-17, End Sem, Marks 6]

Ans. : Various risks associated with software project : Refer Q.2.

Management strategies :

Reactive and proactive risk strategies are the approaches used for managing the risks.

Reactive risk strategy

only managing the risk when it occurs

- Reactive risk management is a risk management strategy in which when project gets into trouble then only **corrective action** is taken. But when such risks can not be managed and new risks come up one after the other, the software team flies into action in an attempt to correct problems rapidly. These activities are called **"firefighting" activities**.
- Resources are utilized to manage such risks. And if still the risks do not get managed then project is in danger.
- In this strategy no preventive care is taken about the risks. They are handled only on their occurrences.
- This is an older approach of risk management.

Proactive risk strategy

- Proactive risk management strategy begins before the technical activity by **considering the probable risk**.
- In this strategy potential risks are identified first then their probability and impact is analyzed. Such risks are then specified according to their priorities (i.e. high priority risks should be managed first!). Finally the software team prepares a plan for managing these risks.
- The objective of this strategy is to **avoid the risks** (*prevention is better than cure!!!*). But it is not possible to avoid all the risks, hence team prepares the risk management plan in such a manner that risk controlling can done efficiently.

- This is an **intelligent strategy** for risk management and now a day it is used by most of the IT industries.

5.3 : Risk Identification

Q.4 What is risk identification ? What are various categories of risks ?

☞ [SPPU : May-18, Dec.-18, 19, 22, End Sem, Marks 6, 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.

Efforts taken to Know the Risk for the project to minimize the future loses

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.

Categories of risks : Refer Q.3.

Q.5 Explain risk identification and assessment process for software project. ☞ [SPPU : Dec.-13, 19, End Sem, Marks 6]

Ans. : Risk identification : Refer Q.4.

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.

5.4 : Risk Projection

Q.6 How risk projection is carried out using risk table ?

☞ [SPPU : May-11, Marks 8, May-19, End Sem, Marks 5]

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.

5.5 : Risk Refinement

Q.7 Explain risk projection and risk refinement in detail.

☞ [SPPU : June-22, Marks 8]

Ans. : Risk projection : Refer Q.6.

Risk refinement : Risk refinement is a process of specifying the risk in more detail. The risk refinement can be represented using **CTC format** suggested by **D.P.Gluch.**

The CTC stands for *condition-transition-consequence.* The condition is first stated and then based on this condition sub conditions can be derived. Then determine the effects of these sub conditions in order to refine the risk. This refinement helps in exposing the underlying risks. This approach makes it easier for the project manager to analyze the risk in greater detail.

5.6 : Risk Mitigation, Monitoring and Management

Q.8 What is risk mitigation, monitoring and management (RMMM) ? Write a note on it.

☞ [SPPU : May-12, Marks 10, Dec.-19, Marks 5, Dec.-22, Marks 9, May-14, Marks 5]

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.

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.

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.

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.

5.7 : The RMMM Plan

Q.9 Explain RMMM plan with suitable example.

☞ [SPPU : May-18, Dec.-18, 22, End Sem, Marks 6, June-22, Marks 5]

Ans. : • 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.

Q.10 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. : 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.

Part II : Software Configuration Management

5.8 : Software Configuration Management

Q.11 What do you understand by Software Configuration Management(SCM) ? Discuss the importance of SCM.

☞ [SPPU : May-19, End Sem, Marks 8, Dec.-22, Marks 9]

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

- During the development of software change must be managed and controlled in order to improve quality and reduce error.
- Hence Software Configuration Management is a **quality assurance activity** that is applied throughout the software process.
- The **origins of changes** that are requested for software are -
  1. New business or market positions cause the changes in the requirements. Due to which the changes need to occur.
  2. New stakeholders may require some changes in the existing requirements.
  3. Due to business growth or project extension, it is essential to makes changes in the project.
  4. Sometimes due to schedule or budge constraints, changes are must in project.
- The software configuration management is concerned with managing evolving software systems.
- Software configuration management is a set of tracking and control activities that begin when a software development project begins and terminates when the software is taken out of operation.

Q.12 What are the elements that exist when an effective SCM system is implemented ? Discuss each briefly.

☞ **[SPPU : Dec.-17, End Sem, Marks 6]**

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

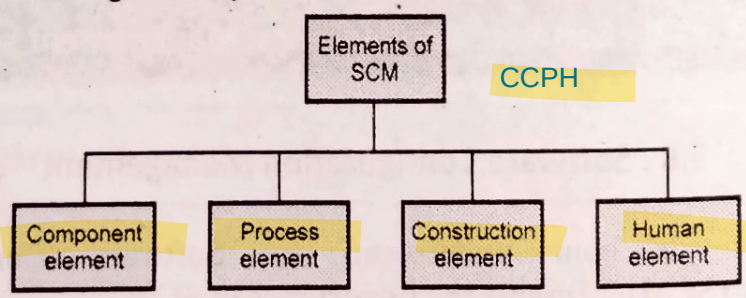


Fig. Q.12.1 Elements of SCM

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.13 Explain software configuration items in detail.

Ans. : • Software configuration item is basically the **information** that is created as a part of software engineering process.

- **Examples** of Software Configuration Items are
  - Computer programs
    - Source programs
    - Executable programs
  - Documents describing the programs
    - Technical manual
    - Users manual
  - Data
    - Program components or functions
    - External data
    - File structure
- For each type of item, there may be a large number of different individual items produced. For instance there may be many documents for a software specification such as project plan, quality plan, test plan, design documents, programs, test reports, review reports.
- These SCI or items will be produced during the project, stored, retrieved, changed, stored again, and so on.

- Each configuration item must have a unique name, and a description specification which distinguishes it from other items of the same type.

5.9 : The SCM Repository

Q.14 What is software SCM repository ? Explain the features of tool s supporting SCM repository. ☞ [SPPU : Dec.-17, 18, End Sem, Marks 6

Ans. : SCM repository :

- Software repository is a **collection of information** accessed by softwar engineers to make appropriate changes in it.
- This repository is handled using all the modern database managemen 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** : It establishes additional information about when, where, why the changes are made.

Q.15 Explain the SCM repository in detail. What are the advantages of SCM repository ?

☞ [SPPU : May-18, Dec.-22, Marks 6]

Ans. : Refer Q.14.

5.10 : The SCM Process

Q.16 What is Software Configuration Management (SCM)? Explain the change control mechanism in SCM.

☞ [SPPU : May-16, Dec.-18, End Sem, Marks 6, Dec.-19, June-22, Marks 5]

Ans. : SCM : Refer Q.14.

Change control mechanism : The change control process is shown by following Fig. Q.16.1 (See Fig. Q.16.1 on Next page)

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.

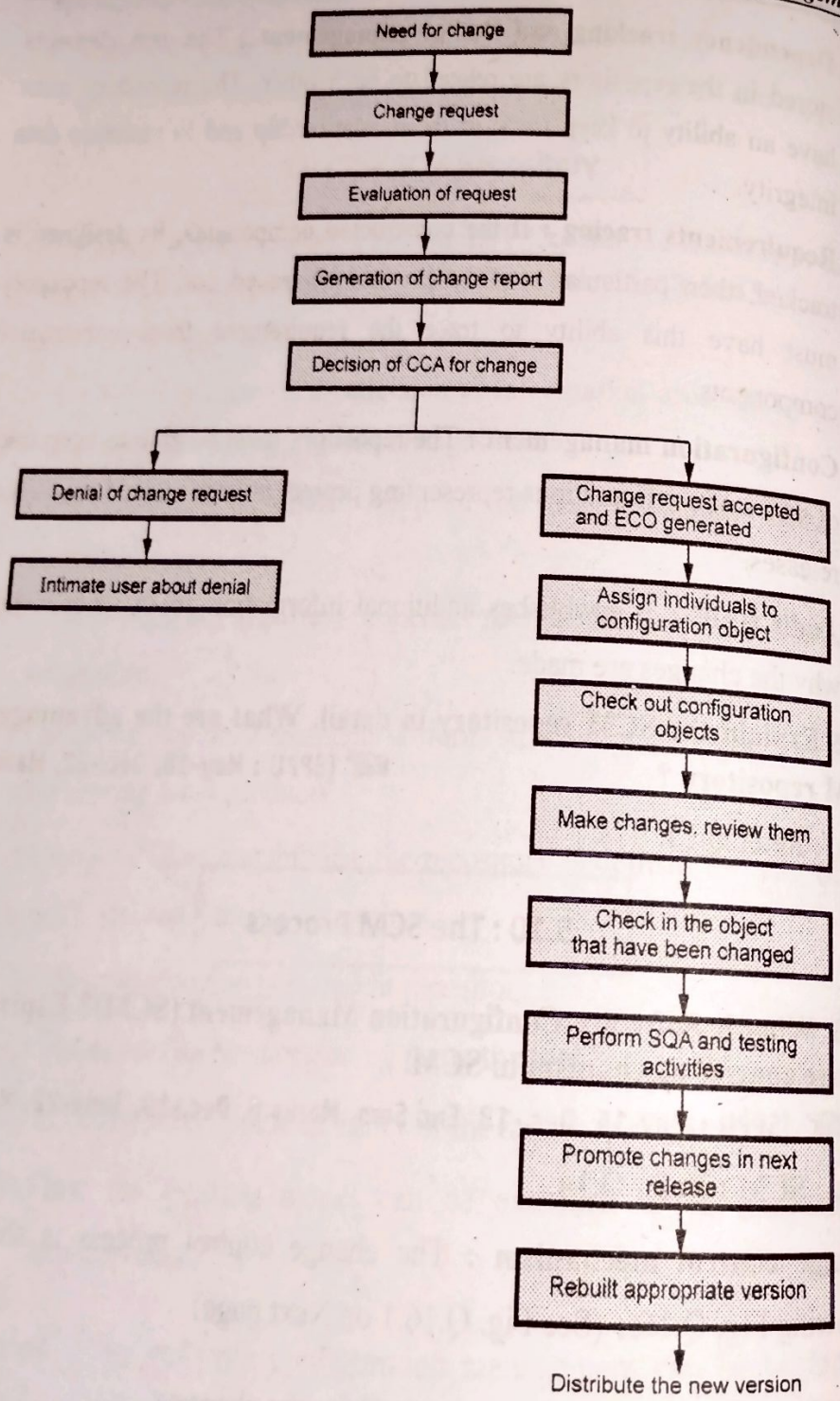


Fig. Q.16.1 Change control process

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.

Q.17 What are the layers of SCM process ? Explain each in detail.

☞ **[SPPU : May-18, 19, Dec.-19, End Sem, Marks 6, June-22, Marks 5]**

Ans. : Layers of SCM process are as follows -

1. Configuration identification : 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.

2. Change control : Refer Q.16.

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.17.1.
- Each version of software system is a collection of software configuration items.

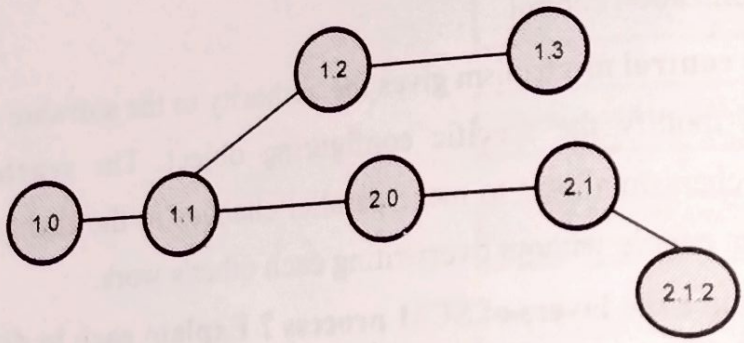


Fig. Q.17.1 Version numbering in evolution graph

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.

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.

5.11 : Configuration Management for any Suitable Software System

Q.18 Explain domain issues associated with web applications from configuration management point of view.

Ans. :

1. Content :

- The typical web application contains text, graphics, scripts, audio/video files, forms, active web pages and so on.
- There is a **challenge** to **organize** these contents into **configurable objects** and establish **control mechanism** for these objects.

2. People :

- Many times the contents for web apps are developed by the people having no software engineering background. Such people are completely unaware of need for configuration management.
- As a result changes are accommodated in the web app in an uncontrolled manner.

3. Scalability :

- The web app grow significantly with interconnections with information subsystems.
- As size and complexity grows the small changes are difficult to incorporate or these changes may lead to problematic situations.
- Hence as the scalability of web apps grows, the configuration control mechanism need to be rigorously applied.

4. Politics :

- There can be issues related to -
  1. Responsibility for quality control of the web site before publishing the web site.
  2. Responsibility about making changes.
  3. Responsibility for accuracy of information on web page.
  4. Responsibility for cost changes.

People who handle the above mentioned responsibilities adopt configuration management process for web apps.

END...

Unit VI

6 Software Testing

6.1 : A Strategic Approach to Software Testing

Q.1 What is testing ?

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

Q.2 What is importance of testing practices ? What are the principles of testing practices ? ☞ [SPPU : May-14, Marks 8]

Ans. : Concept of testing : Refer Q.1.

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 :

Every software engineer must apply following testing principles while performing the software testing.

1. All tests should be traceable to customer requirements.
2. Tests should be planned long before testing begins.

3. The Pareto principle can be applied to software testing : 80 % of all errors uncovered during testing will likely be traceable to 20 % of all program modules.
4. Testing should begin "in the small" and progress toward testing "in the large".
5. Exhaustive testing is not possible.
6. To be most effective, testing should be conducted by an independent third party.

Q.3 What are the main objectives of software testing and what are the principles of software testing ?

☞ [SPPU : May-16, End Sem, Marks 9,Dec.-19, End Sem, Marks 5]

Ans. : Objectives :

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.

Principles : Refer Q.2.

Q.4 What is software testing ? Explain the software testing strategy for software development.

☞ [SPPU : Dec.-13,18, May-19, Marks 8, May-18, End Sem, Marks 7]

Ans. : Software testing : Refer Q.1.

Software Testing strategy : Refer Q.2.

6.2 : Verification and Validation

Q.5 Differentiate between verification and validation.

☞ [SPPU : Dec.-13, May-16, 18, End Sem, 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.

Q.6 Explain phases in verification and validation model with suitable diagram. ☞ [SPPU : June-22, Marks 7]

Ans. : • This model is V-shaped hence is the name. It is just like waterfall model in which each phase must be completed before the next phase begins.

- This model contains various stages of software development along with various types of testing that can be conducted at each stage of development. The test plans serve as a link between the development stages and various tests.
- As shown by this model at each stage specific testing can be carried out. Refer Fig. Q.6.1.

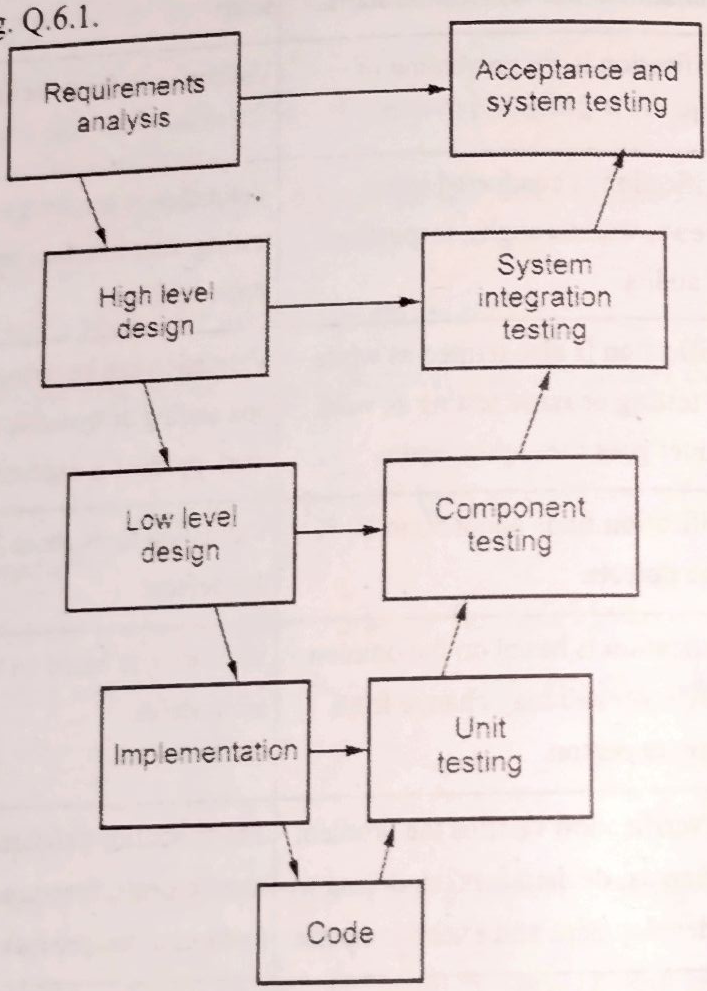


Fig. Q.6.1

Requirements analysis : As per the waterfall model of software development process model the V model begins with requirements gathering and analysis. The Software Requirements Specification (SRS) is created during this phase. The acceptance and system testing is conducted in which the focus is on meeting of functionalities specified in the requirements gathering.

High level design : During this phase the system architecture is designed. It provides the overview of platform used, system, product, and service or process. An integration test plan is prepared and testing is conducted to test the pieces of the software systems to ensure about the ability to work together.

Low level design : During this phase actual software components are designed. The relationship with other components are defined at this level. Hence component testing is conducted for this phase.

Implementation : All coding takes place at this phase. The unit testing is carries out to test every source code module. The creation and review of various test cases are carried out in this phase.

6.3 : Organizing for Software Testing

Q.7 Explain the process of organizing for software testing.

Ans. : • Testing needs to be conducted from the beginning of the software development process.

- It is desired that the software developer should test the individual unit of software system for its proper functioning. This is called unit testing. That means, there must be an involvement of software developers for unit testing activities.
- In many cases, the software developers also perform the integration testing. Thus the complete software architecture is been tested by the software developers during development phases.
- Then the Independent Test Group (ITG) gets involved in the software testing activity.
- The role of ITG is to remove other inherent problems and conflicts associated with the software system.
- The software developers and the ITG work together to remove all possible errors from the software system and a thorough testing must be performed.

- While testing is conducted, the developer must be available to correct errors that are uncovered.
- The ITG is part of the software development project team in the sense that it becomes involvedduring analysis and design and stays involved throughout a large project.
- In many cases, the ITG reports to **software quality assurance organization** so that the independent testing can be performed.

6.4 : Software Testing Strategy - The Big Picture

Q.8 Explain STLC (Software Testing Life Cycle)in detail.

☞ [SPPU : June-22,Marks 7,Dec.-22, Marks 6]

OR Explain the strategic approach to software testing.

☞ [SPPU : Dec.-12, Marks 6]

Ans. : Software Testing Life Cycle(STLC) is a sequence of activities conducted during the testing process. Following are the phases in every STLC model.

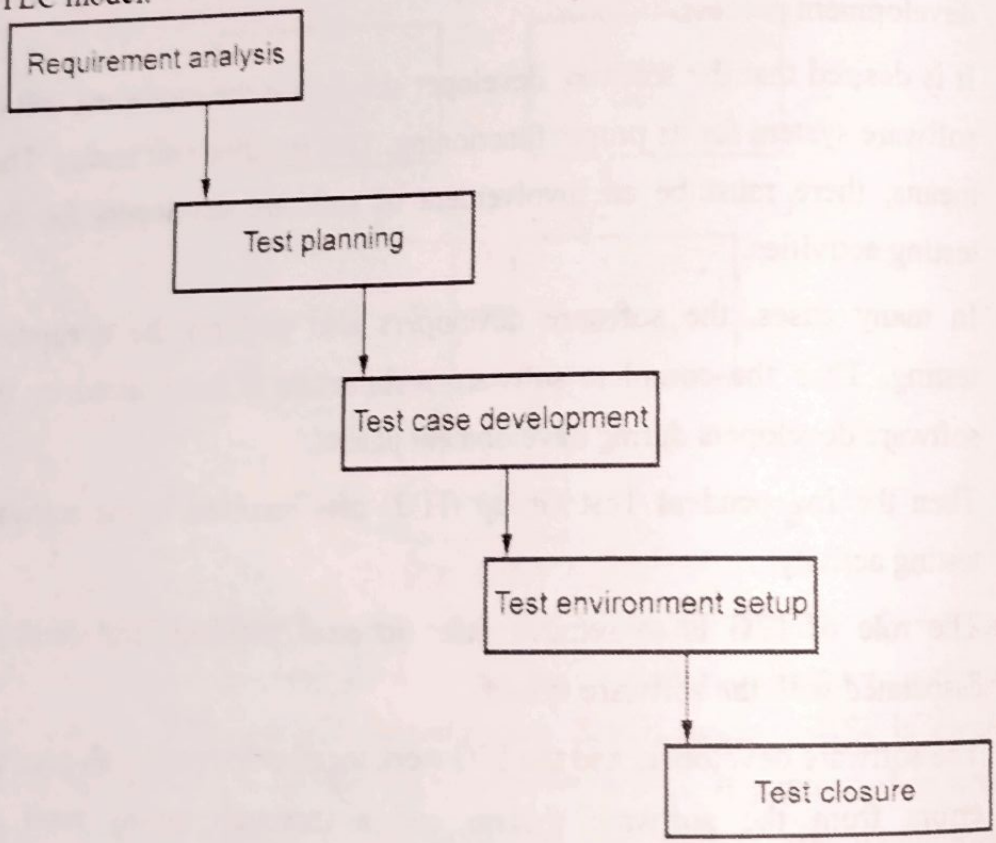


Fig. Q.8.1 Software Testing Life Cycle(STLC)

(1) Requirements analysis

- In this phase it is identified that the requirements are testable or not.
- If the requirements are not testable then test team communicates to various stakeholders such as business analyst, clients, technical lead, and System architect so that mitigation strategy can be planned.

(2) Test planning

- It is a first phase of test process.
- In this phase senior QA manager determines the test plan. In this phase, manager of the testing team calculates estimated effort and cost for the testing work.
- Activities like resource planning, determining roles and responsibilities, tool selection (if automation), training requirements, etc., carried out in this phase

(3) Test case development or design

- Once the test planning phase gets completed, the test case development phase starts.
- In this phase, the test cases are created and the test data is identified.

(4) Test environment setup

- Test Environment Setup decides the software and hardware conditions under which a work product is tested.
- The testing team is not involved in setting up the testing environment, its senior developers who create it.

(5) Test execution

- In this phase, the testing team starts case development and execution activity.
- The testing team writes down the detailed test cases, also prepares the test data if required.

- The prepared test cases are reviewed by peer members of the team or Quality Assurance.

(6) Test cycle closure

- The test cycle closure is a completion of test execution in which test cycle closure report is prepared.
- The test cycle closure report includes all the documentation related to software design, development, testing results, and defect reports.
- This is the last stage of STLC in which the process of testing is analyzed.

6.5 : Criteria for Completion of Testing

Q.9 What are entry and exit criteria for completion of testing ?

Ans. : Entry criteria

- Software testing should start early in the Software Development Life Cycle. This helps to capture and eliminate defects in the early stages of SDLC i.e. requirement gathering and design phases.
- An early start to testing helps to reduce the number of defects and ultimately the rework cost in the end.
- **Definition :** Entry criteria for testing can be defined as "Specific conditions or on-going activities that must be present before a process can begin." The Software Testing Life Cycle (STLC) specifies the entry criteria required during each testing phase.
- It also defines the time interval or the expected amount of lead-time to make the entry criteria item available to the process.
- Following are the inputs necessary for the entry criteria –
  - The requirements documents
  - Complete understanding of the application flow
  - Test plan

Exit criteria

- **Definition :** It can be defined as "The specific conditions or on-going activities that should be fulfilled before completing the software testing life cycle."
- The exit criteria can identify the intermediate deliverables.
- The following exit criteria should be considered for completion of a testing phase :
  - Ensuring all critical Test Cases are passed
  - Achieving complete Functional Coverage
  - Identifying and fixing all the high-priority defects
- The output achieved through exit criteria are –
  - Test summary report
  - Test logs

6.6 : Strategic Issues

Q.10 Enlist and explain strategic issues of testing

Ans. : • ***Before the testing starts, Specify product requirements appropriately*** - Certain quality characteristics of the software such as maintainability, portability and usability should be specified in order to obtain the unambiguous test results.

- ***Specify testing objectives clearly*** - Testing objectives such as effectiveness, mean time to failure and cost of defects should be stated clearly in the test plan.
- ***Identify categories of users for the software and their role with reference to software system*** - Use cases describe the interactions among different class of users and thereby testing can focus on the actual use of the product.
- ***Develop a test plan that emphasizes rapid cycle testing*** - Test plan is an important document which helps the tester to perform rapid cycle testing (2 percent of project effort).

- ***Build robust software which can be useful for testing*** - The software should be capable of detecting certain classes of errors. Moreover, the software design should allow automated testing and regression testing.
- ***Use effective formal reviews before actual testing process begins*** - Formal technical reviews need to be conducted to uncover errors. The effective technical reviews conducted before testing, reduce significant amount of testing efforts.

6.7 : Test Strategies for Conventional Software

Q.11 What is unit testing ? Explain the unit testing process.

[SPPU : Dec.-12, 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.

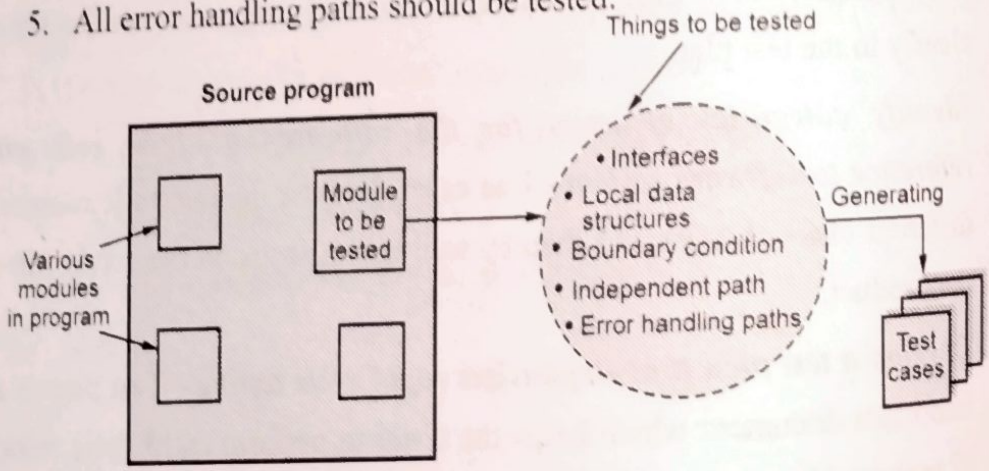


Fig. Q.11.1 Unit testing

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.11.2.

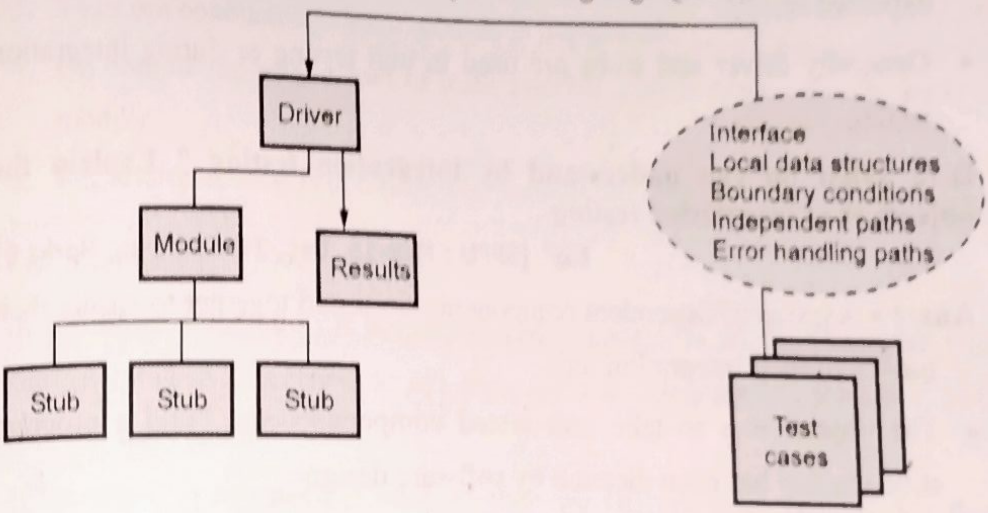


Fig. Q.11.2 Unit testing environment

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.

Q.12 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.

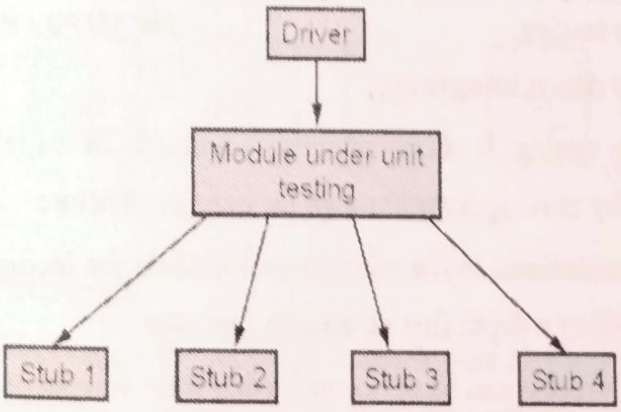


Fig. Q.12.1 Driver and stub

- 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.

Q.13 What do you understand by integration testing ? Explain the objectives of integration testing .

☞ [SPPU : May-16, Dec.-19, End Sem, 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.14 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. : 1) Top down integration

- 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.

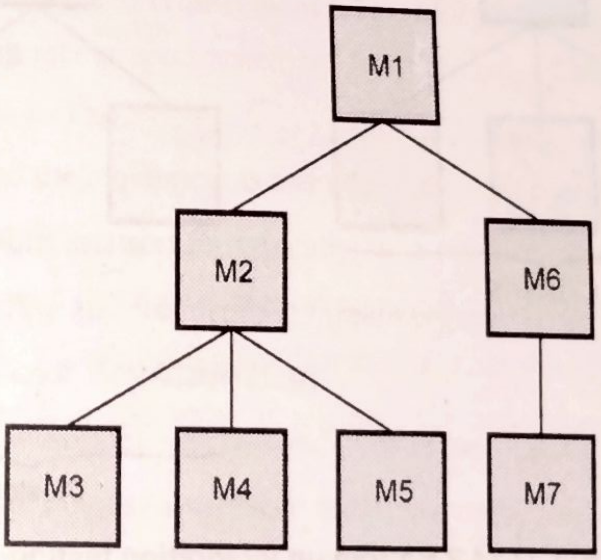


Fig. Q.14.1 Program structure

2) Bottom up integration

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 :

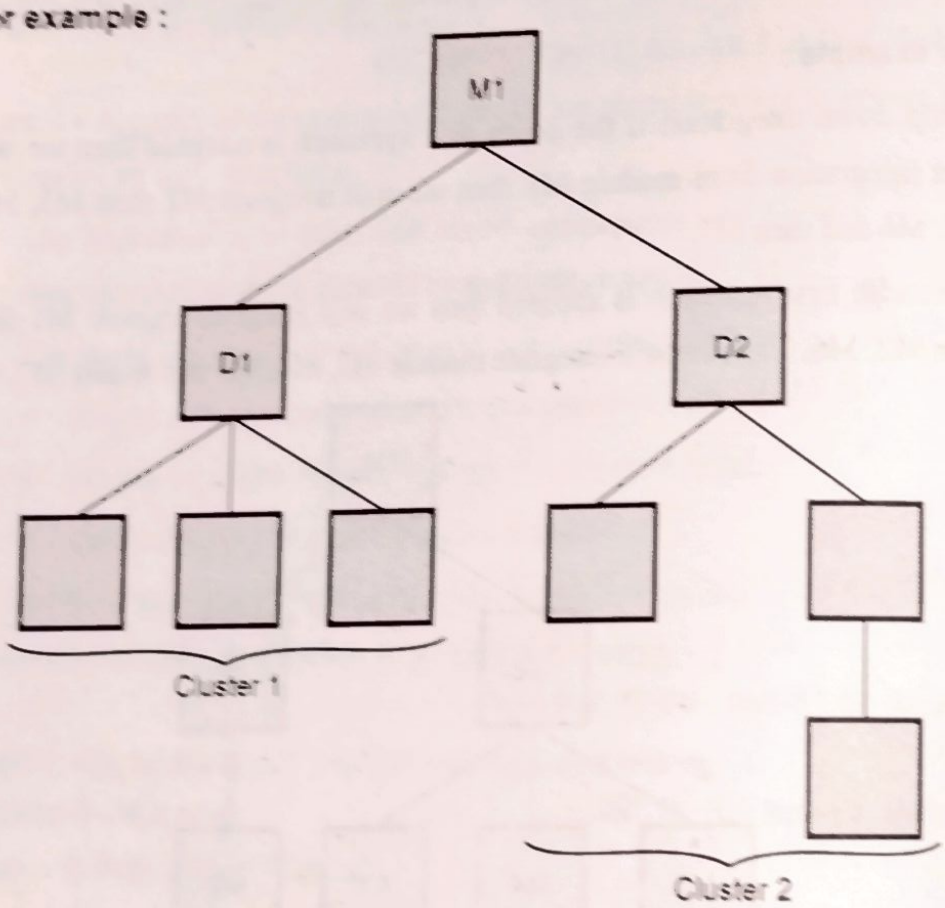


Fig. Q.14.2 Bottom up integration testing

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.15 Explain - Unit testing and integration testing.

☞ [SPPU : June-22, Marks 5]

Ans. : Unit testing : Refer Q.11.

Integration testing : Refer Q.13.

Q.16 Explain bottom up testing with its advantages.

☞ [SPPU : Dec.-22, Marks 5]

Ans. : Refer Q.14.

Q.17 What is system testing ? Explain any two system testing strategies.

☞ [SPPU : May-11, Marks 8, Dec.-12, Marks 6, Dec.-17, Marks 8, Dec.-22, Marks 9]

Ans. : System testing :

The system test is a series of tests conducted to fully the computer based system.

Various types of system tests are :

1. **Recovery testing**
2. **Security testing**
3. **Stress testing**
4. **Performance testing**

The main focus of such testing is to test

- System functions and performance.
- System reliability and recoverability (recovery test).
- System installation (installation test).
- System behaviour in the special conditions (stress test).
- System user operations (acceptance test/alpha test).
- Hardware and software integration and collaboration.
- Integration of external software and the system.

Types :

(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.

6.8 : Test Strategies for Object-Oriented Software

Q.18 Explain object oriented testing. ☞ [SPPU : May-12, Marks 3]

Ans. : • The primary objective of testing for object oriented software is to **uncover as much errors as** possible with manageable amount of efforts in realistic time span.

- The object oriented testing strategy is identical to conventional testing strategy. The strategy is start **testing in small** and work outward to **testing in large**.
- The basic unit of testing is **class** that contains attributes and operations. These classes **integrate** to form object oriented architecture. These are called **collaborating classes.**

Q.19 How object oriented testing differs from conventional testing strategies ? ☞ [SPPU : May-13, Marks 6]

Ans. : • The testing strategies applied to conventional software are based on **unit testing, integration testing, validation testing** and **system testing.** But in object oriented testing, the unit testing loses some of its meaning and integration testing changes significantly.

- The object oriented testing strategy is also based on the principle - "test in small" and then work "in the large".
  1. **Testing in the small** involves **class attributes and operations;** the **main focus** is on communication and collaboration within the **class.**
  2. **Testing in the large** involves a series of regression tests to uncover errors due to **communication and collaboration among classes**

Finally, the system as a whole is tested to detect errors in fulfilling requirements

6.9 : Test Strategies for WebApps

Q.20 Enlist the principles of testing for WebApps.

Ans. : Testing strategy suggests to use following basic **principles of software testing -**

1. The **content model** must be reviewed in order to **uncover the errors.**
2. The **interfaces model** is reviewed to ensure all the **use cases.**
3. The **design model** is reviewed to uncover **navigation errors.**
4. **User interface** must be tested to remove the **navigation errors.**
5. For selected function components unit testing is done.
6. Navigation must be tested for the complete web architecture.
7. The web application is tested in different environmental configuration.
8. **Security tests** must be conducted to exploit **vulnerabilities** of the web application.
9. **Performance** of the web application must be tested using **performance testing.**
10. Finally controlled and monitored group of users must test the web application.

The web application **evolves continuously** and hence it is an on-going activity.

6.10 : Validation Testing

Q.21 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.

2) Acceptance testing : Refer Q.22.

3) 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 -
  1. Software components already translated into code are integrated into a "build". The "build" can be data files, libraries, reusable modules or program components.
  2. A series of tests are designed to expose errors from build so that the "build" performs its functioning correctly.
  3. The "build" is integrated with the other builds and the entire product is smoke tested daily.

Q.22 Discuss in detail - Acceptance testing

☞ [SPPU : May-13, Marks 4, June-22, Marks 5]

Ans. : The acceptance testing is a kind of testing conducted to ensure that the software works correctly in the **user work environment**.

The acceptance testing can be conducted over a period of weeks or months.

The types of acceptance testing are :

1. **Alpha test** - The alpha testing is a testing in which the version of complete software is tested by the customer under the supervision of developer. This testing is performed at developer's site. The software is used in natural setting in presence of developer. This test is conducted in controlled environment.
2. **Beta test** - The beta testing is a testing in which the version of software is tested by the customer without the developer being present. This testing is performed at customer's site. As there is no presence of developer during testing, it is not controlled by developer. The end user records the problems and report them to developer. The developer then makes appropriate modification.

Q.23 Write note on - Alpha and beta testing.

☞ [SPPU : Dec.-22, Marks 8]

Ans. : Refer Q.22.

Q.24 Differentiate between alpha and beta testing.

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

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.

6.11 : Validation-Test Criteria

Q.25 Explain validation test criteria.

Ans. : After each validation testing one of the two possible conditions exists :

1) The function or performance characteristics conforms to specification and is accepted. or
2) A deviation from specification is represented. In other words the list of requirements that does not get fulfilled by the system is represented.

Q.26 Explain - Tools for automated testing and features.

☞ [SPPU : June-22, Marks 5]

Ans. : 1. Test manager : This tool is used to manage the program tests. It checks the test data and verifies the expected results. For example JUnit is a tool containing Java Classes.

2. Test data generator : This tool generates the test data for the program which is under testing. The test data can be chosen form the existing databases of the system.

3. Oracle : This tool is for generating the expected test results.

4. File comparator : Previous test results of the program are compared with the current test results and the differences between them are highlighted by this tool.

5. Report generator : Using this tool the reports can be generated for the test result.

6. Dynamic analyzer : This tool analyzes the number of times each programming statement is executed.

7. Simulators : There are various types of simulating programs used for various purposes. For instance : The user-interface simulator simulates various multiple user interfaces. The target simulate simulates the machine intended for program execution.

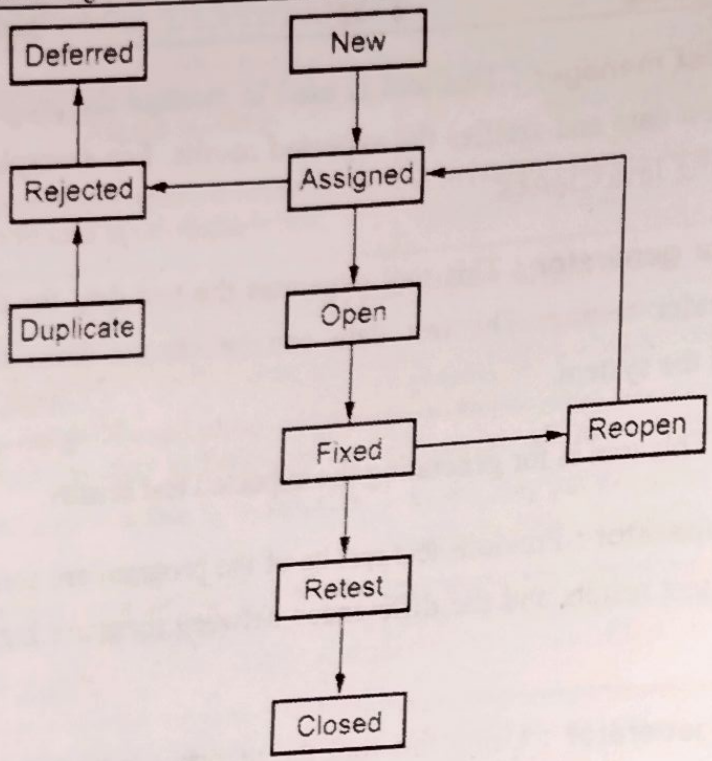
Large amount of time and efforts are required to create the testing workbench. Hence its use is preferred when the large system has to be developed.

Some examples of testing tools are CLIF, JMeter, JUnit, DBMonster.

Q.27 Explain - Defect life cycle. ☞ [SPPU : June-22, Marks 5]

Ans. : Definition : Defect life cycle is a cycle of defect in which defect goes through different stages during its lifetime.

The defect life cycle starts as a soon as new defect is found by a tester and comes to an end when tester closes that defect assuring that it won't get reproduced again.



1) **New :** This is the first state of a defect in the defect life cycle. When any new defect is found, it falls in a 'New' state and validations and testing are performed on this defect in the later stages of the defect life cycle.

2) **Assigned :** In this stage, a newly created defect is assigned to the development team for working on the defect. This is assigned by the project lead or the manager of the testing team to a developer.

3) **Open :** Here, the developer starts the process of analyzing the defect and works on fixing it, if required. If the developer feels that the defect is not appropriate then it may get transferred to any of the below four states namely Duplicate, Deferred, or Rejected.

4) **Fixed :** When the developer finishes the task of fixing a defect by making the required changes then he can mark the status of the defect as 'Fixed'.

5) **Retest :** At this point, the tester starts the task of working on the retesting of the defect to verify if the defect is fixed accurately by the developer as per the requirements or not.

6) **Reopen :** If any issue still persists in the defect then it will be assigned to the developer again for testing and the status of the defect gets changed to 'Reopen'.

7) **Rejected :** If the defect is not considered as a genuine defect by the developer then it is marked as 'Rejected' by the developer.

8) **Duplicate :** If the developer finds the defect as same as any other defect or if the concept of the defect matches with any other defect then the status of the defect is changed to 'Duplicate' by the developer.

9) **Deferred :** If the developer feels that the defect is not of very important priority and it can get fixed in the next releases or so in such a case, he can change the status of the defect as 'Deferred'.

10) **Closed :** When the defect does not exist any longer then the tester changes the status of the defect to 'Closed'.

6.12 : Types of Testing

Q.28 Explain - Black box testing and white box testing.

☞ [SPPU : June-22, Marks 5]

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.

6.13 : White Box Testing

Q.29 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.

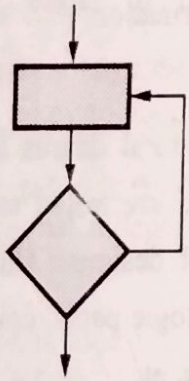


Fig. Q.29.1 Simple loop

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.

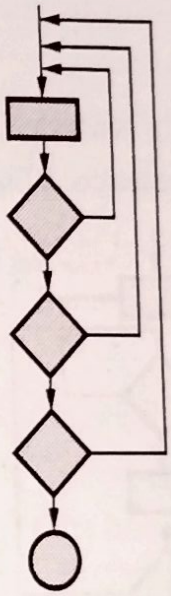


Fig. Q.29.2 Nested loops

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.

3. Concatenated loops :

The concatenated loops can be tested in the same manner as simple loop tests. (Refer Fig. Q.29.3)

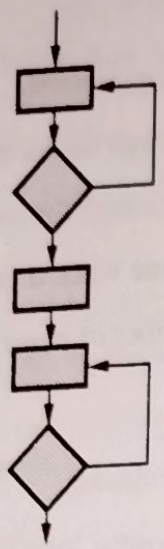


Fig. Q.29.3 Concatenated loops

4. Unstructured loops :

The testing cannot be effectively conducted for unstructured loops. Hence these types of loops needs to be redesigned. (Refer Fig. Q.29.4)

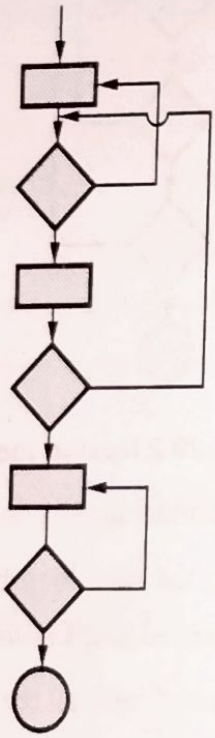


Fig. Q.29.4 Unstructured loops

Q.30 What is basis path testing ? What is cyclomatic complexity ? How is it determined for a flow graph ? Illustrate with an example.

☞ [SPPU : May-12, 14, 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 –

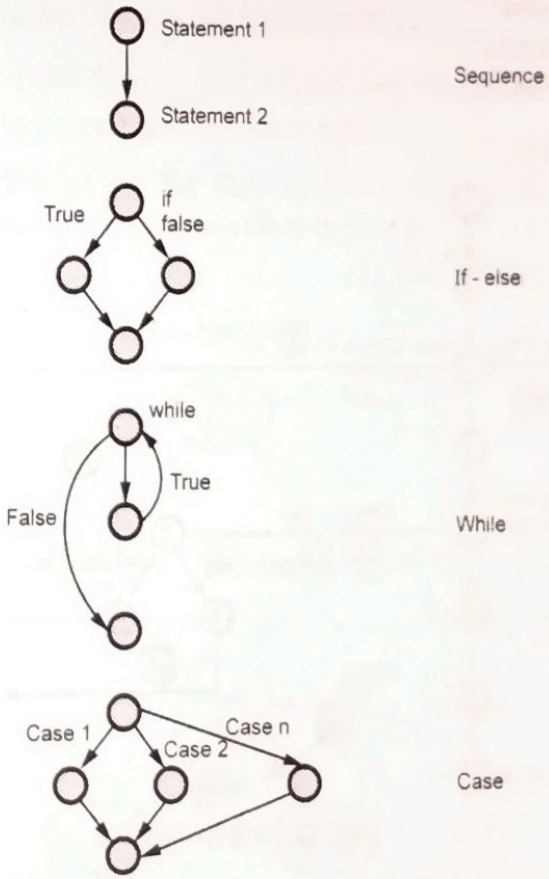


Fig. Q.30.1

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

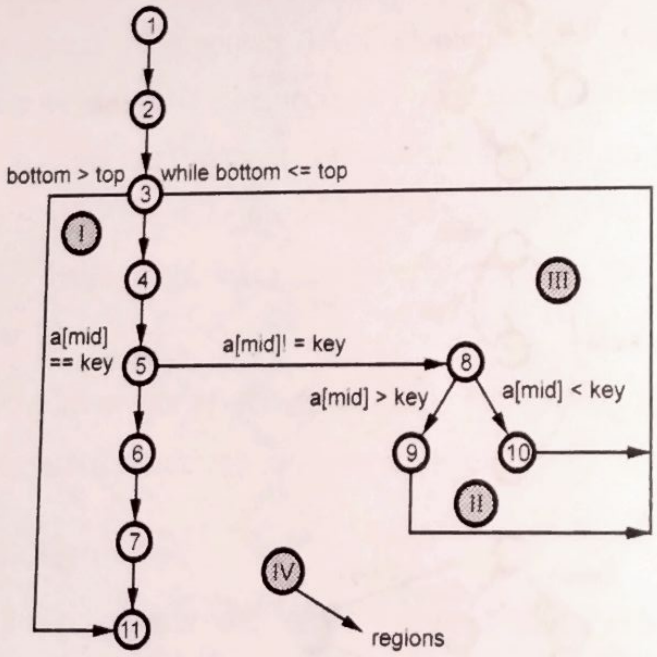


Fig. Q.30.2

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

$$= 2 + 2 = \mathbf{4}$$

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.

Q.31 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 -

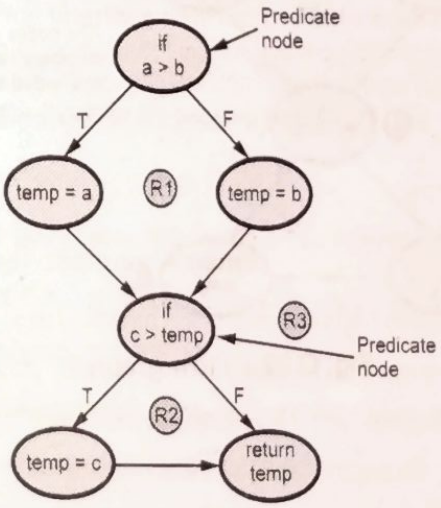


Fig. Q.31.1 Flow graph for finding maximum of three numbers

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.**

Q.32 Explain the graph matrix and loop testing methods.

☞ **[SPPU : May-13, Marks 8]**

Ans. : Graph matrix :

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.

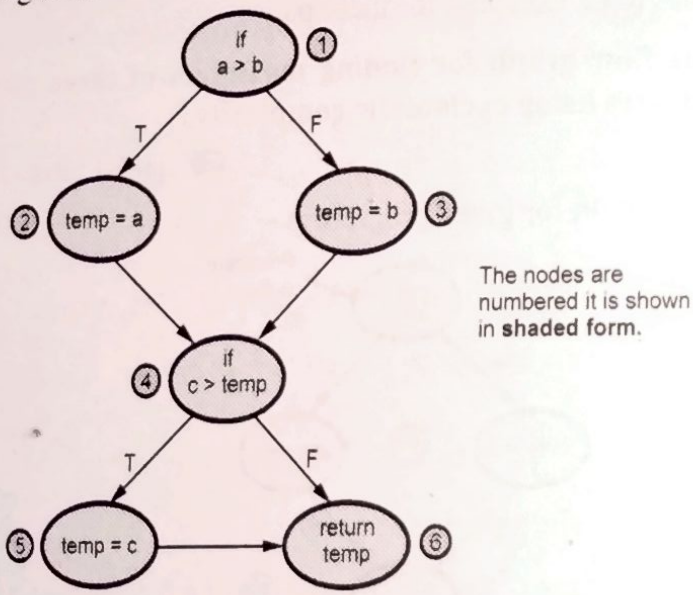


Fig. Q.32.1 Flow graph

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 : Refer Q.29 (ii).

Q.33 What do you mean by white box testing ?

☞ [SPPU : Dec.-16, End Sem, Marks 7]

Ans. : Basic concept : Refer Q.28 (ii).

White Box testing method : Refer Q.29.

6.14 : Black Box Testing

Q.34 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.

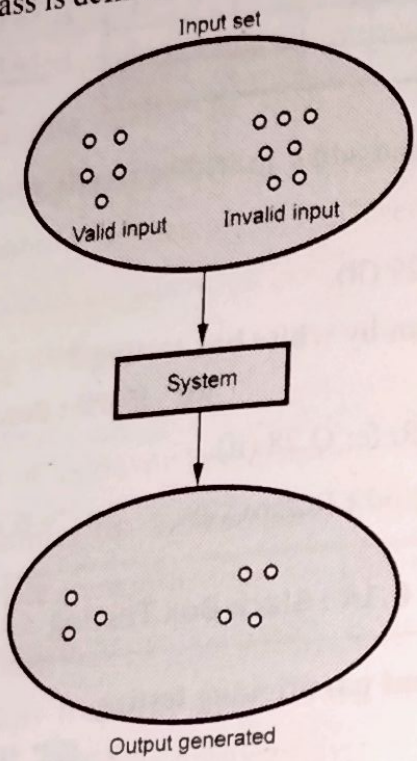


Fig. Q.34.1

6.15 : Comparison between Black Box Testing and White Box Testing

Q.35 Differentiate between white box and black box testing.

☞ [SPPU : Dec.-12, Marks 4, Dec.-22, Marks 6]

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.

END...

JUNE-2022 [5869] - 278

Solved Paper

Course 2019

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

Instructions to the candidates :

1) *Solve Q.1 or Q.2, Q.3 or Q.4, Q.5 or Q.6, Q.7 or Q.8.*

2) *Neat diagrams must be drawn wherever necessary.*

3) *Assume suitable data, if necessary.*

Q.1 a) *Design and discuss the project decomposition and work task communication process.* **(Refer Q.10 of Chapter - 3)** [8]

b) *Discuss any 2 of the following with suitable example :*

i) *FP-based estimation* **(Refer Q.7 of Chapter - 3)**

ii) *Object Point (OP)-based estimation.* **(Refer Q.14 of Chapter - 3)**

iii) *Process-based estimation.* **(Refer Q.11 of Chapter - 3)** [10]

OR

Q.2 a) *Describe any two software size estimation techniques.* **(Refer Q.6 and Q.7 of Chapter - 3)** [8]

b) *Discuss any 2 of the following with suitable example :*

i) *Problem-based estimation* **(Refer Q.7 of Chapter - 3)**

ii) *LOC-based estimation* **(Refer Q.6 of Chapter - 3)**

iii) *Project scheduling and basic principles of project scheduling* **(Refer Q.15 of Chapter - 3)** [10]

Q.3 a) *List the design concepts. Explain refinement and refactoring. Give the importance of refactoring in improving the quality of software.* **(Refer Q.9 of Chapter - 4)** [9]

b) *List the different architectural styles. Explain any two in detail.* **(Refer Q.33, Q.34 and Q.35 of Chapter - 4)** [8]

OR

Q.4 a) *Enlist and explain component level design steps in detail.* **(Refer Q.29 of Chapter - 4)** [9]

b) *Differentiate between following :* [8]

i) Cohesion and coupling in context of software design. How are these useful for good design of a system ? **(Refer Q.14 of Chapter - 4)**

ii) Abstraction and refinement **(Refer Q.8 of Chapter - 4)**

Q.5 a) *Explain risk identification process. What are the different categories of risks ?* **(Refer Q.4 of Chapter - 5)** [8]

b) *Write short note :* [10]

i) Layers of SCM process **(Refer Q.20 of Chapter - 5)**

ii) RMMM plan **(Refer Q.9 of Chapter - 5)**

OR

Q.6 a) *Explain risk projection and risk refinement in detail.* **(Refer Q.7 of Chapter - 5)** [8]

b) *Explain the change control mechanism in SCM.* **(Refer Q.19 of Chapter - 5)** [10]

Q.7 a) *Explain STLC (Software Testing Life Cycle).* **(Refer Q.8 of Chapter - 6)** [7]

b) *Explain the following :*

i) Unit testing and integration testing **(Refer Q.15 of Chapter - 6)**

ii) White box testing and black box testing **(Refer Q.28 of Chapter - 6)** [10]

OR

Q.8 a) *Explain phases in verification and validation model with suitable diagram.* **(Refer Q.6 of Chapter - 6)** [7]

b) *Discuss any 2 of the following in detail.*

i) Acceptance testing **(Refer Q.22 of Chapter - 6)**

ii) Tools for automated testing and feature. **(Refer Q.26 of Chapter - 6)**

iii) Defect life cycle. **(Refer Q.27 of Chapter - 6)** [10]

DECEMBER-2022 [5925] - 262 Solved Paper

Course 2019

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

Q.1 a) *Explain object oriented view of component level design with suitable example.* **(Refer Q.28 of Chapter - 3)** [6]

b) *Explain FP based estimation technique.* [6]

(Refer Q.7 of Chapter - 3)

c) *What is project scheduling ? What are the basic principles of project scheduling ?* **(Refer Q.15 of Chapter - 3)**

OR

Q.2 a) *Explain COCOMO model for project estimation with suitable example.* **(Refer Q.13 of Chapter - 3)** [9]

b) *How LOC and FP used during project estimation ? Explain both estimation techniques with suitable example.* [10]

(Refer Q.8 of Chapter - 3)

Q.3 a) *Explain guidelines for component level design.* [6]

(Refer Q.29 of Chapter - 4)

b) *Enlist the golden rules of user interface design.* [6]

(Refer Q.26 of Chapter - 4)

c) *Explain layered system architecture with neat diagram.* [5]

(Refer Q.35 of Chapter - 4)

OR

Q.4 a) *Describe notations used for deployment diagram. Describe the importance of deployment diagram.* **(Refer Q.27 of Chapter - 4)** [9]

b) *Explain the following architectural styles with merits/demerits :*

i) Data-centered architecture **(Refer Q.34 of Chapter - 4)**

ii) Data-flow architecture **(Refer Q.37 of Chapter - 4)** [8]

Q.5 a) *What is risk identification ? What are different categories of risk ?* **(Refer Q.4 of Chapter - 5)** [6]

b) *Define software risk in detail. What are different types of software risk ?* **(Refer Q.1 of Chapter - 5)** [6]

c) *What are the advantages of SCM repository ? Explain functions performed by SCM repository.* **(Refer Q.15 of Chapter - 5)** [6]

OR

Q.6 a) *What is Software Configuration Management (SCM).* **(Refer Q.11 of Chapter - 5)** [6]

b) *What is RMMM ? Write short note on it ?* **(Refer Q.9 of Chapter - 5)** [9]

Q.7 a) *What are difference between white box testing and black box testing.* **(Refer Q.35 of Chapter - 6)** [6]

b) *Explain the software testing life cycle in detail.* **(Refer Q.8 of Chapter - 6)** [6]

c) *Explain bottom-up testing with its advantages.* **(Refer Q.16 of Chapter - 6)** [5]

OR

Q.8 a) *What is system testing ? Explain any three types system testing.* **(Refer Q.17 of Chapter - 6)** [9]

b) *Write note on alpha and beta testing.* **(Refer Q.23 of Chapter - 6)** [8]

END...