MAHARASHTRA STATE BOARD OF TECHNICAL EDUCATION
(Autonomous)
(ISO/IEC - 27001 - 2005 Certified)
SUMMER-16 EXAMINATION
Model Answer

Subject Code: 17513 **Subject Name: Software Engineering**

Important Instructions to examiners:

1) The answers should be examined by key words and not as word-to-word as given in the model answer scheme.

2) The model answer and the answer written by candidate may vary but the examiner may try to assess the understanding level of the candidate.

3) The language errors such as grammatical, spelling errors should not be given more Importance (Not applicable for subject English and Communication Skills).

4) While assessing figures, examiner may give credit for principal components indicated in the Figure. The figures drawn by candidate and model answer may vary. The examiner may give Credit for any equivalent figure drawn.

5) Credits may be given step wise for numerical problems. In some cases, the assumed Constant values may vary and there may be some difference in the candidate's answers and model answer.

6) In case of some questions credit may be given by judgment on part of examiner of relevant answer based on candidate's understanding.

7) For programming language papers, credit may be given to any other program based on Equivalent concept.

Marks

1. A) Attempt any three of the following: **12**

i) Define Software. State three characteristics of software.
(Definitions - 1 mark; characteristics any 3 points - 3 marks)

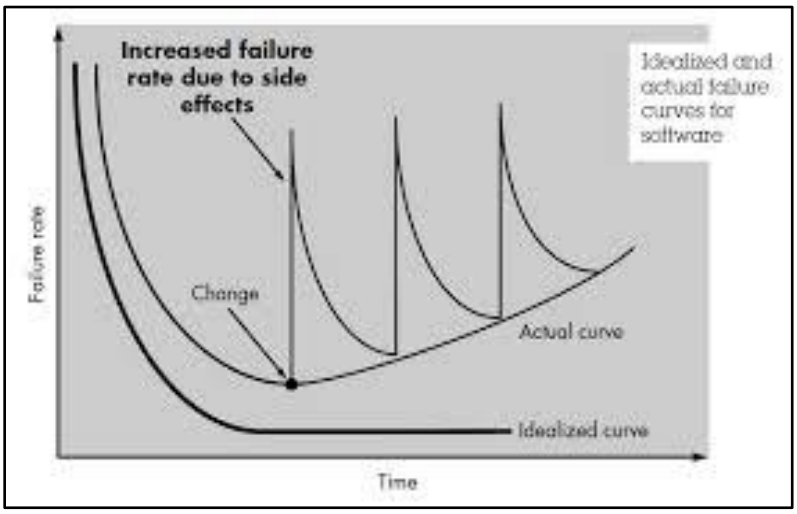
Ans:

Computer Software is the product that the software professionals build and then support over the long term. Software is written to handle an Input – Process – Output system to achieve predetermined goals. Software is logical rather than a physical system element.

Software has following characteristics:

1. Software is developed or engineered; it is not manufactured in the classical sense.
2. Software doesn't "wear out" like hardware and it is not degradable over a period.
3. Although the industry is moving toward component – based construction, most software continues to be custom built.

Failure curve for Software:



ii) What is software coding? State three principles of code validation.
(Software coding - 1 mark; any 3 principles of code validations - 3 marks)

Ans:

The Software coding activity encompasses a set of coding and testing tasks that lead to operational software that is ready for delivery to the customer or the end user. Even the software development process has undergone a radical change over the years.

In modern Software engineering work, coding may be

1. The direct creation of source code using a programming language. (e.g., Java).
2. Automatic generation of source code using an intermediates design like representation of the components to be built.
3. Automatic generation of executable code using 4GL language (e.g., Visual , C++)
   Validation testing that assesses whether requirements have been met for the complete system (or software increment)

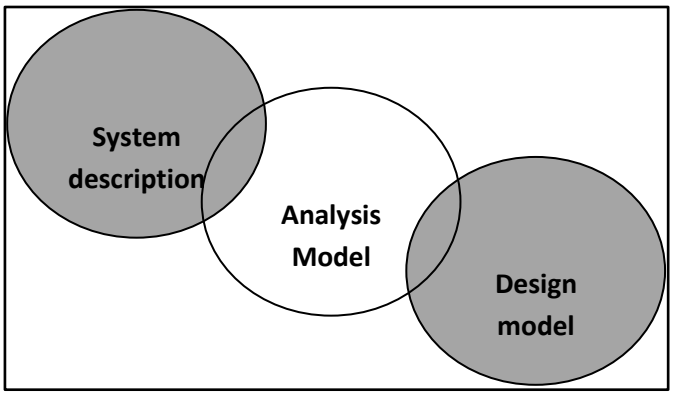
Validation Principles are:

1. Conduct a code walkthrough when appropriate.
2. Perform unit tests and correct errors you've uncovered.
3. Refactor the code.

iii) Describe the terms: Analysis Modeling and Design Modeling.
(Description of analysis Modeling - 2 marks; Design modeling - 2 marks)

Ans:

The analysis model and requirements specification provide a means for assessing quality once the software is built.
Requirements analysis results in the specification of software's operational characteristics.



The analysis model as a bridge between the system description and the design model.

Objectives

Analysis model must achieve three primary objectives:

1. Describe customer needs
2. Establish a basis for software design
3. Define a set of requirements that can be validated once the software is built.

Design Modeling:

Software design is an iterative process through which requirements are translated into a "blueprint" for constructing the software. The design is representation at a high level of abstraction – data, functional, and behavioral requirements. As design iterations occur, subsequent refinement leads to design representations at much lower levels of abstraction.

There are three characteristics that serve as a guide for the evaluation of a good design:

1. The design must implement all the explicit requirements contained in the requirements model, and it must accommodate all the implicit requirements desired by stakeholders.
2. The design must be a readable, understandable guide for those who generate code and for those who test and subsequently support the software.
3. The design should provide a complete picture of the software, addressing the data, functional and behavioral domains for implementation.

iv) Differentiate between Prescriptive Process Model and Agile Process Model (any four points).
(Any 4 Points of differentiation - 4 marks)

Ans:

Sr. No	Prescriptive Process Model	Agile Process Model
1.	Product Oriented process. Process and technology are crucial	People oriented process. Favors people over technology
2.	A traditional approach for software product development	It is an recent approach for Project Management
3.	Traditional and modern approaches using generic process framework activities with medium to large cycle-time	Cycle-time reduction is most important
4.	Focus is on tasks, tools such as estimating, scheduling, tracking and control	Model focuses on modularity, iterative, time bound, parsimony, adaptive, incremental convergent, collaborative approach
5.	Models include Waterfall, Incremental, Prototype, RAD and spiral	Agile process model uses the concept of Extreme Programming

B) Attempt any one of the following. 6

i) Describe the layered technology approach of Software Engineering.
(4 Points of layered technology approach of Software Engineering - 4 marks)

Ans:

Software engineering is a layered technology. The layers of software engineering as shown in the above diagram are:-

1. A Quality Focus:

Any engineering approach (including software engineering) must rest on an organizational commitment to quality. Total quality management, six sigma and similar philosophies foster a continuous process improvement culture, and it is this culture that ultimately leads to the development of increasingly more effective approaches to software engineering. The bedrock that supports software engineering is a quality focus.

2. Process Layer:

The foundation for software engineering is the process layer. Software Engineering process is the glue that holds the technology layers together and enables rational and timely development of computer software. Process defines a framework that must be established for effective delivery of software engineering technology. The software process forms the basis for management control of software projects and establishes the context in which technical methods are applied, works products (models, documents, data, reports, forms etc.) are produced, milestones are established, quantity is ensured and change is properly managed.

3. Methods:

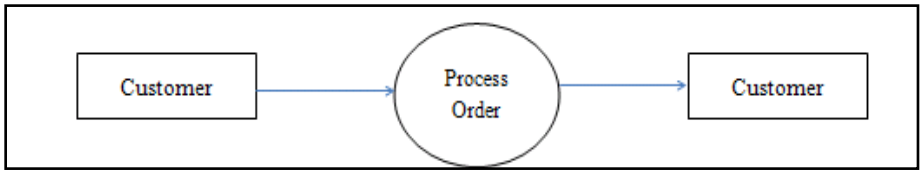
Software Engineering methods provide the technical "how to's" for building software. Methods encompass a broad array of tasks that include communication, requirements analysis, design modeling, program construction, testing and support.

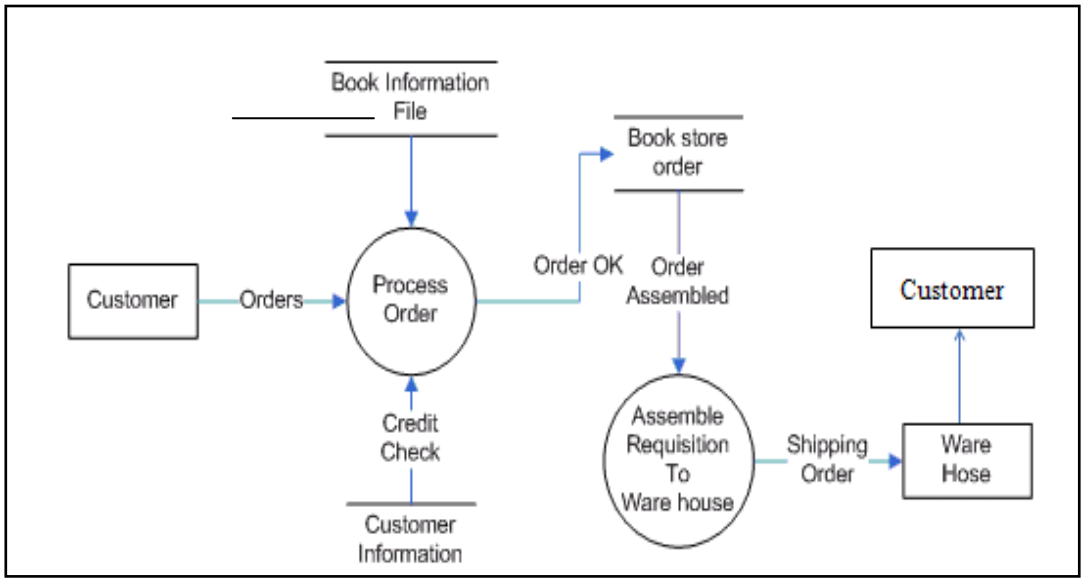
4. Tools:

Software Engineering tools provide automated or semi-automated support for the process and the methods. When tools are integrated so that information created by one tool can be used by another, a system for the support of software development, called computer–aided software engineering is established.

ii) Draw a data flow diagram level 0 and 1 for a Book publishing House.
(Level 0 DFD - 1 mark; Level 1 DFD - 3 marks)

Ans:





2. Attempt any four of the following: 16

(a) Define the terms software process, software product, software work product and software engineering.
(1 mark each)

Ans:

Software Process:
A software process provides the framework from which a comprehensive plan for software development can be established. A number of different tasks sets-tasks; milestones, work products, and quality assurance points enable the framework activities to be adapted to the characteristics of the software projects and the requirements of the project.

Software Product:
Before a project can be planned, product objectives and scope should be established, alternative solutions should be considered, and technical and management constrains should be identified. Without this information, it is impossible to define reasonable and accurate estimates of the cost, an effective assignment of risk, a realistic breakdown of project tasks, or a manageable product schedule that provides a meaningful indication of progress. The developers must examine the product and the problem intended to solve at the very beginning of the project. For this reason, the scope of the product must be established and bounded.

Software Work Product:
Each framework activity is populated by a set of software engineering actions- a collection of related tasks that produces a major software engineering work product (e.g. design is a SE action). Each action is populated with individual work tasks that accomplish some part of the work implied by the action.

Software Engineering:
Software Engineering is defined as a discipline that addresses the following aspects of the software and its development. They are:

1. Economic : Cost, Benefits, and Returns on Investment (ROI).
2. Design : Ease of development and ensuring delivery of customer requirements.
3. Maintenance : Ease of effecting changes and modifications.
4. Implementation: Ease of installation, Demonstration, and implementation of software by the customer and users.

(b) What is SRS? Explain importance of SRS.
(SRS - 2 marks; importance of SRS - 2 marks)

Ans:

A **Software requirements specification (SRS),** a requirements specification for a software system, is a description of the behavior of a system to be developed and may include a set of use cases that describe interactions the users will have with the software. In addition it also contains non-functional requirements. Non-functional requirements impose constraints on the design or implementation (such as performance engineering requirements, quality standards, or design constraints).

Software requirements specification establishes the basis for agreement between customers and contractors or suppliers (in market-driven projects, these roles may be played by the marketing and development divisions) on what the software product is to do as well as what it is not expected to do. Software requirements specification permits a rigorous assessment of requirements before

design can begin and reduces later redesign. It should also provide a realistic basis for estimating product costs, risks, and schedules.

The software requirements specification document enlists enough and necessary requirements that are required for the project development. To derive the requirements we need to have clear and thorough understanding of the products to be developed or being developed. This is achieved and refined with detailed and continuous communications with the project team and customer till the completion of the software.

(c) What is domain analysis? Explain with suitable examples
(Description of domain analysis - 2 marks; example - 2 marks)

Ans:

Software domain analysis is the identification, analysis and specification of common requirements from a specific application domain, typically for reuse in multiple projects within that application domain.

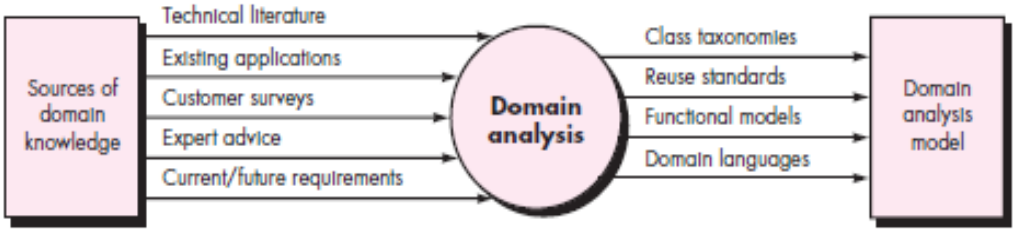
Object-oriented domain analysis is the identification, analysis and specification of common, reusable capabilities within a specific application domain, in terms of common objects, classes, subassemblies, and frameworks.

Example of Domain

The specific application domain can range from avionics to banking, from multimedia video game to software embedded within medical devices.

Goal

To find or create analysis classes and/or common functions those are broadly applicable, so that they may be reused.



Input and Output for Domain Analysis

The role of domain analyst is to discover and define reusable analysis patterns, analysis classes and related information that may be used by many people working on similar but not necessarily the same applications.

(d) Describe the relationship between systems engineering and software engineering.
(System engineering - 2 marks; software engineering - 2 marks)

Ans:

i. Software Engineering

Software engineering deals with designing and developing software of the highest quality. A software engineer does analyzing, designing, developing and testing software. Software engineers carry out software engineering projects, which usually have a standard software life cycle. For example, the Water Fall Software Life cycle will include an analysis phase, design phase, development phase, testing and verification phase and finally the implementation phase. Analysis phase looks at the problem to be solved or the opportunities to be seized by developing the software. Sometimes, a separate business analyst carries out this phase. However, in small companies, software engineers may do this task. Design phase involves producing the design documents such as UML diagrams and ER diagrams depicting the overall structure of the software to be developed and its components. Development phase involves programming or coding using a certain programming environment. Testing phase deals with verifying that software is bug free and also satisfies all the customer requirements. Finally, the completed software is implemented at the customer site (sometimes by a separate implementation engineer). In recent years, there has been a rapid growth of other software development methodologies in order to further improve the efficiency of the software engineering process. For example, Agile methods focus on incremental development with very short development cycles. Software Engineering profession is a highly rated job because of its very high salary range.

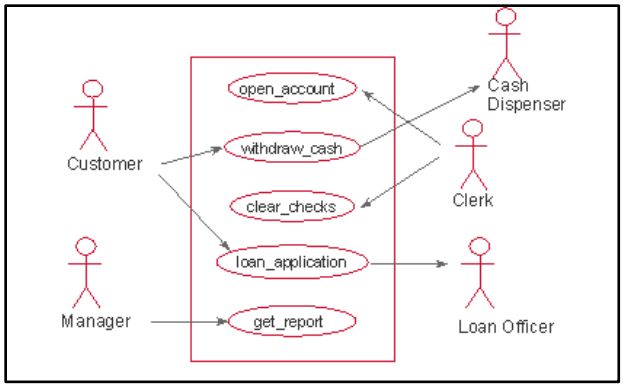
ii. System Engineering

System Engineering is the sub discipline of engineering which deals with the overall management of engineering projects during their life cycle (focusing more on physical aspects). It deals with logistics, team coordination, automatic machinery control, work processes and similar tools. Most of the times, System Engineering overlaps with the concepts of industrial engineering, control engineering, organizational and project management and even software engineering. System Engineering is identified as an interdisciplinary engineering field due to this reason. System Engineer may carry out system designing, developing requirements, verifying requirements, system testing and other engineering studies.

(e) Draw a use case diagram for Bank Management System.
(4 marks for appropriate Use case diagram)
[NOTE: any relevant diagram shall be considered**]**

Ans:

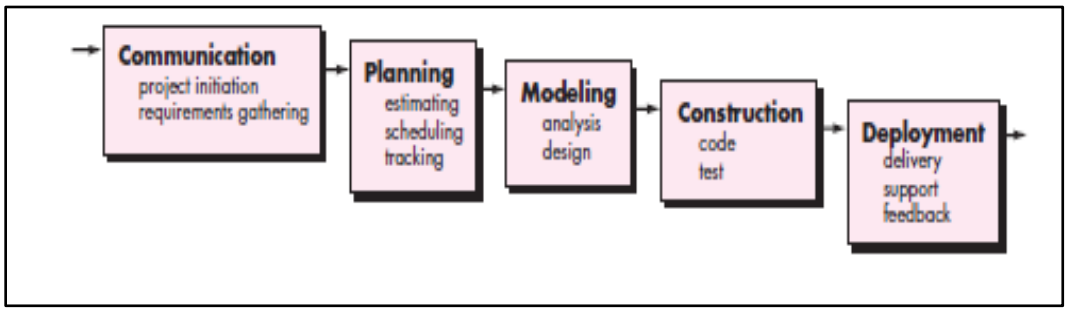
Use case Diagram for a Bank Management System



(f) What is waterfall model? State the practical situations in which it can be used.

(Description - 2 marks; Diagram - 1 mark; Use - 1 mark)

Ans:



There are times when the requirements of a problem are reasonably well understood – when work flows from communication through deployment in a reasonably linear fashion. The waterfall model is a traditional method, sometimes called the classic life cycle, suggests a systematic, sequential approach to software development that begins with customer specification of requirements and progresses through planning, modeling, construction and deployment, culminating in on-going support of the completed software.

This is one of the initial models. As the figure implies stages are cascaded and shall be developed one after the other. In other words one stage should be completed before the other begins. Hence, when all the requirements are elicited by the customer, analyzed for completeness and consistency, documented as per requirements, the development and design activities commence.

This model presents a high level view and suggests to the developer the sequence of events they should expect to encounter. This model is used to prescribe software development activities in variety of contexts. It is the basis for software deliverables. Associated with each activity are milestones and outcomes, for managers to monitor.

One of the main needs of this model is the user's explicit prescription of complete requirements at the start of development. For developers it is useful to layout what they need to do at the initial stages. Its simplicity makes it easy to explain to customers who may not be aware of software development process. It makes explicit with intermediate products to begin at every stage of development.

One of the biggest limitation is it does not reflect the way code is really developed. Problem is well understood but software is developed with great deal of iteration. Often this is a solution to a problem which was not solved earlier and hence software developers shall have extensive experience to develop such application; as neither the user nor the developers are aware of the key factors affecting the desired outcome and the time needed. Hence at times the software development process may remain uncontrolled.

Today software work is fast paced and subject to a never-ending stream of changes in features, functions and information content. Waterfall model is inappropriate for such work. This model is useful in situation where the requirements are fixed and work proceeds to completion in a linear manner.

3. Attempt any four of the following: 16

(a) State and explain any four types of software.
(Any four types - 4 marks; 1 mark for each type)

Ans:

System software: System software is a collection of programs written to service utilities, process complex, but determinate, information structures.
Real-time software: Software that monitors/analyzes/controls real-world events as they occur is called real time.
Business software: Business information processing is the largest single software application area. Discrete "systems" (e.g., payroll, accounts receivable/payable, inventory) have evolved into management information system (MIS) software that accesses one or more large databases containing business information.

Engineering and scientific software: Engineering and scientific software have been characterized by "number crunching" algorithms. Applications range from astronomy to volcanology, from automotive stress analysis to space shuttle orbital dynamics, and from molecular biology to automated manufacturing.
Embedded software: Intelligent products have become commonplace in nearly every consumer and industrial market. Embedded software resides in read-only memory and is used to control products and systems for the consumer and industrial markets.
Personal computer software: The personal computer software market has burgeoned over the past two decades. Word processing, spreadsheets, computer graphics, multimedia, entertainment, database management, personal and business financial applications, external network, and database access are only a few of hundreds of applications.
Web-based software: The Web pages retrieved by a browser are software that incorporates executable instructions (e.g., CGI, HTML, Perl, or Java), and data.
Artificial intelligence software: Artificial intelligence (AI) software makes use of non-numerical algorithms to solve complex problems that are not amenable to computation or straightforward analysis.

(b) What is Requirements Elicitation? What are the problems faced in eliciting requirements?

(Explanation of requirement elicitation - 1 mark; any three problems - 1 mark for each problem)

Ans:

Elicitation
Elicitation means to draw out the truth or reply from anybody. In relation with requirement engineering, elicitation is a task that helps the customer to define what is required. To know the objectives of the system to be developed is a critical job.
The problems faced in eliciting requirement are:

a. **Problem of Scope:**
The boundary of the system is ill-defined or the customers/users specify unnecessary technical detail that may confuse, rather than clarify, overall system objectives.

b. **Problem of understanding:**
The customers/users are not completely sure of what is needed, have a poor understanding of the capabilities and limitations of their computing environment, don‘t have full understanding of the problem domain, have trouble communicating needs to the system engineer, omit information that is believed to be ―obvious, specify requirements that conflict with the needs of other customer/users or specify requirements that ambiguous or un-testable.

c. **Problems of volatility:**
Volatility means change from one state to another. The customer‘s requirement may change time to time.

(c) Explain the importance of SRS.
(Explanation - 4 marks)

Ans:

The importance of SRS documents are:

Establish the basis for agreement: SRS helps in establishing agreement between the customers and the suppliers on what the software product is to do. The complete description of the functions to be performed by the software specified in the SRS will assist the potential users to determine if the software specified meets their needs or how the software must be modified to meet their needs. Reduce the development effort. The preparation of the SRS forces the various concerned groups in the customer's organization to consider rigorously all of the requirements before design begins and reduces later redesign, recoding, and retesting. Careful review of the requirements in the SRS can reveal omissions, misunderstandings, and inconsistencies early ip the development cycle when these problems are easier to correct.

Provide a basis for estimating costs and schedules. The description of the product to be developed as given in the SRS is a realistic basis for estimating project costs and can be used to obtain approval for bids or price estimates. .

Provide a baseline for validation and verification. Organizations can develop their validation and Verification plans much more productively from a good SRS. As a part of the development contract, the SRS provides a baseline against which compliance can be measured.

Facilitate transfer. The SRS makes it easier to transfer the software product to new users or new machines. Customers thus find it easier to transfer the software to other parts of their organization, and suppliers find it easier to transfer it to new customers.

Serve as a basis for enhancement. Because the-SRS discusses the product but not the project that developed it, the SRS serves as a basis for later enhancement of the finished product. The SRS may need to be altered, but it does provide a foundation for continued production evaluation.

(d) What is Data Modeling? Explain the terms cardinality and modality.
(Data Modeling - 2 marks; Cardinality - 1 mark; Modality - 1 mark)

Ans:

Data modeling answers a set of specific questions that are relevant to any data processing application by making use of the entity relationship diagram (ERD). The ERD enables a software engineer to identify data objects and their relationships using a graphical notation.

Cardinality:

1. Cardinality specifies number of occurrences of one object related to number of

occurrences of another object.

2. That is the maximum number of object relationship is represented by cardinality.
3. Shows different relationship like one to one, one to many and many to many.
4. E.g. many employees occupy one room.

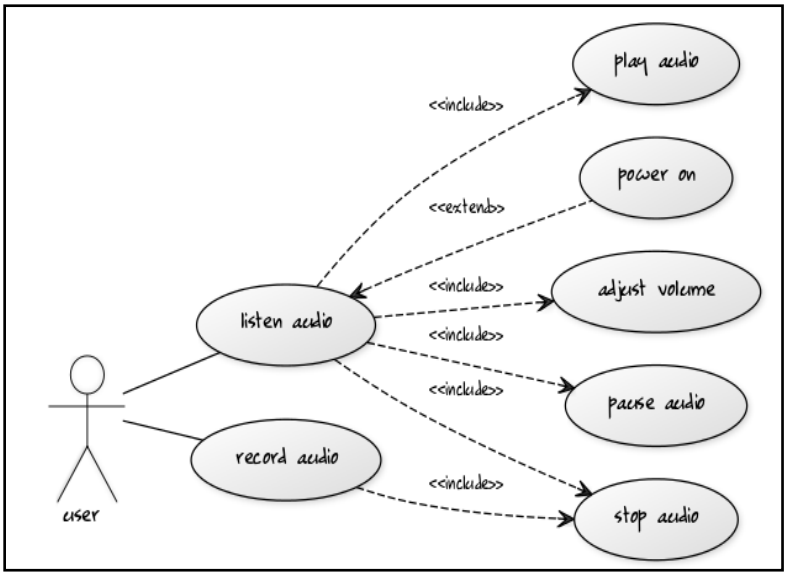
Modality:

1. A modality of relationship is zero if occurrence of relationship is optional and modality of relationship is 1 if occurrence of relationship is mandatory (i.e. compulsory).
2. The modality specifies the minimum number of relationship.
3. Shows maximum 1 to minimum or compulsory 1.
4. E.g. exactly one (maximum 1 and minimum 1) room is occupied by zero or many (maximum many and minimum 0) employees.

(e) Draw a use case diagram for a music system.

(4 marks for Correct Use case Diagram; any relevant diagram shall be considered)

Ans:



4. A) Attempt any three of the following: **12**

i) What aspects of the software are tested in Unit Testing?
(Any four points; each point - 1 mark)

Ans:

Interface: The module interface is tested to ensure that information properly flows into and out of the program unit under test. Tests of data flow across a module interface are required before any other test is initiated. If data do not enter and exit properly, all other tests are moot.
Local data structure: The local data structure is examined to ensure that data stored temporarily maintains its integrity during all steps in an algorithm's execution.
Boundary conditions: Boundary conditions are tested to ensure that the module operates properly at boundaries established to limit or restrict processing.
Independent paths: All independent paths (basis paths) through the control structure are exercised to ensure that all statements in a module have been executed at least once.
Error-handling paths: All error handling paths are tested so as to ensure that the error are handled properly without causing system failure and the appropriate message is shown to user.

ii) State any four basic principles to be followed for project scheduling.
(4 principles - 4 marks; 1 mark per principle)

Ans:

Basic principles software project scheduling:
Compartmentalization: The project must be compartmentalized into a number of manageable activities and tasks. To accomplish compartmentalization, both the product and the process are decomposed.
I**nterdependency:** The interdependency of each compartmentalized activity or task must be determined. Some tasks must occur in sequence while others can occur in parallel. Some activities cannot commence until the work product produced by another is available. Other activities can occur independently.
Time allocation: Each task to be scheduled must be allocated some number of work units (e.g., person-days of effort). In addition, each task must be assigned a start date and a completion date that are a function of the interdependencies and whether work will be conducted on a full-time or part-time basis.
Effort validation: Every project has a defined number of staff members. As time allocation occurs, the project manager must ensure that no more than the allocated number of people has been scheduled at any given time.

Defined responsibilities: Every task that is scheduled should be assigned to a specific team member. Defined outcomes: Every task that is scheduled should have a defined outcome.
Defined milestones: Every task or group of tasks should be associated with a project milestone. Program evaluation and review technique (PERT) and critical path method (CPM) are two project scheduling Methods that can be applied to software development.

iii) Define the terms: Software Reliability and Software Availability.
(Software Reliability - 2 marks; Software Availability - 2 marks)

Ans:

Reliability: - Software Reliability is probability of failure free operation of a system. It is calculated as follows
MTBF = MTTF + MTTR
where The acronyms MTTF and MTTR are mean-time-to-failure and mean-time-to- repair
MTBF=Mean Time Between Failure
Availability:-Software availability is the probability that a program is operating according to requirements at a given point in time and is defined as
Availability = [MTTF/(MTTF + MTTR)]X100%
The MTBF reliability measure is equally sensitive to MTTF and MTTR. The availability measure is somewhat more sensitive to MTTR, an indirect measure of the maintainability of software.

iv) Compare Alpha Testing and Beta Testing.

(Any 4 Points each - 1 mark; any other relevant points should be consider)

Ans:

ALPHA TESTING	BETA TESTING
Alpha Testing Conducted at Developer Site by End user.	Beta Testing is conducted at User site by End user.
Alpha Testing is Conducted in Control Environment as Developer is present	Beta Testing is conducted in Un-Environment as Developer is Absent
Less Chances of Finding an error as Developer usually guides user.	More Chances of Finding an error as Developer can use system in any way
It is kind of mock up testing	The system is tested as Real application
Error/Problem may be solved in quick time if possible.	The user has to send difficulties to the developer who then corrects it.
Short process.	Lengthy Process

B) Attempt any one of the following **6**

i) What are the activities involved in SCM?
(List of activities - 1 mark; Description - 1 mark each)

Ans:

Software Configuration Management Activities are:

Identification of change

To control and manage configuration items, each must be named and managed using an object-oriented approach

- Basic objects are created by software engineers during analysis, design, coding, or testing
- Aggregate objects are collections of basic objects and other aggregate objects
- An entity-relationship (E-R) diagram can be used to show the interrelationships among the objects

Version Control

- Combines procedures and tools to manage the different versions of configuration objects created during the software process
- An entity is composed of objects at the same revision level
- A variant is a different set of objects at the same revision level and coexists with other variants
- A new version is defined when major changes have been made to one or more objects

Change Control

- Change request is submitted and evaluated to assess technical merit and impact on the other configuration objects and budget
- Change report contains the results of the evaluation
- Change control authority (CCA) makes the final decision on the status and priority of the change based on the change report.

Software Configuration Audit

- A software configuration audit complements the formal technical review by assessing a configuration object for characteristics that are generally not considered during review. The audit asks and answers the questions such as:
- Has the change specified in the ECO been made? Have any additional modifications been incorporated?
- Has a formal technical review been conducted to assess technical correctness?

Status Reporting

Configuration status reporting (sometimes called status accounting) is an SCM task that answers the following questions:

1. What happened?
2. Who did it?
3. When did it happen?
4. What else will be affected?

ii) What is Software Quality Assurance? What are the activities carried out in SQA?

(Explanation of SQA - 2 marks; any four Activities of SQA - 4 marks)

Ans:

Software quality assurance is composed of a variety of tasks associated with two different constituencies - the software engineers who do technical work and an SQA group that has responsibility for quality assurance planning, oversight, record keeping, analysis, and reporting. Software engineers address quality (and perform quality assurance and quality control activities) by applying solid technical methods and measures, conducting formal technical reviews, and performing well-planned software testing.

Activities of SQA

1) **Prepare an SQA plan for a project.** The plan is developed during project planning and is reviewed by all interested parties. Quality assurance activities performed by the software engineering team and the SQA group are governed by the plan. The plan identifies
   - evaluations to be performed
   - audits and reviews to be performed
   - standards that are applicable to the project
   - procedures for error reporting and tracking
   - documents to be produced by the SQA group
   - amount of feedback provided to the software project team

2) **Participate in the development of the project's software process description.** The software team selects a process for the work to be performed. The SQA group reviews the process description for compliance with organizational policy, internal software standards, externally imposed standards (e.g., ISO-9001), and other parts of the software project plan.
3) **Review software engineering activities to verify compliance with the defined software process.** The SQA group identifies, documents, and tracks deviations from the process and verifies that corrections have been made.
4) **Audits designated software work products to verify compliance with those defined as part of the software process.** The SQA group reviews selected work products; identifies,

documents, and tracks deviations; verifies that corrections have been made; and periodically reports the results of its work to the project manager.

5) **Ensure that deviations in software work and work products are documented and handled according to a documented procedure.** Deviations may be encountered in the project plan, process description, applicable standards, or technical work products.
6) **Records any noncompliance and reports to senior management.** Noncompliance items are tracked until they are resolved.

5. Attempt any two of the following: 16

a) What is software deployment? State the principles to be followed while preparing to deliver the software increment.
(Software deployment - 3 marks; Principles - 5 marks (1 mark each))

Ans:

Software Deployment: Deployment occurs as each software increment is presented to the customer and encompasses delivery, support, and feedback. As modern software process models are evolutionary or incremental in nature, deployment happens not once, but a number of times as software moves toward completion. Each delivery cycle provides the customer and end users with an operational software increment that provides usable functions and features. Each support cycle provides documentation and human assistance for all functions and features introduced during all deployment cycles to date. Each feedback cycle provides the software team with important guidance that result in modifications to the functions, features, and approach taken for the next increment. The delivery of a software increment represents an important milestone for any software project.

Deployment Principles:

1. **Principle 1: Customer expectations for the software must be managed:** It always happens that customer wants more than he has started earlier as his requirements. It may be the case that customer gets disappointed, even after getting all his requirements satisfied. Hence at time of delivery developer must have skills to manage customer‘s expectations.
2. **Principle 2: A complete delivery package should be assembled and tested*:*** A CD-ROM or other media containing all executable software, support data files, support documents, and other relevant information should be assembled and thoroughly beta-tested with actual users. All installation scripts and other operational features should be thoroughly exercised in as many different computing configurations (i.e., hardware, operating systems, peripheral devices, networking arrangements) as possible.
3. **Principle 3: A support regime must be established before the software is delivered:** An end user expects responsiveness and accurate information when a question or problem arises. If support is ad hoc, or worse, nonexistent, the customer will become dissatisfied immediately. Support should be planned, support materials should be prepared, and appropriate

recordkeeping mechanisms should be established so that the software team can conduct a categorical assessment of the kinds of support requested.

4. **Principle 4: Appropriate instructional materials must be provided to end users*:*** Appropriate training aids (if required) should be developed; troubleshooting guidelines should be provided, and when necessary, a "what's different about this software increment" description should be published
5. **Principle 5: Buggy software should be fixed first, delivered later*:*** In incremental type of software, software organizations may deliver some defective software to the customer by giving assurance that the defects will be removed in next increment. This is a mistake. Buggy software should not be delivered.

b) What is project scheduling and tracking? State four reasons why project deadlines cannot be met.

(Project scheduling and tracking - 4 marks; 4 reasons - 1 mark each)

Ans:

Software project scheduling is an activity that distributes estimated effort across the planned project duration by allocating the effort to specific software engineering tasks.

During early stages of project planning a macroscopic schedule is developed. This type of schedule identifies all major process framework activities and the product functions to which they are applied. As the project proceeds each entry on the macroscopic schedule is refined into a detailed schedule. Here specific software tasks are identified and scheduled. Scheduling of software engineering projects can be viewed from two different perspectives .In the first, an end-date for release of a computer based system has already been established. The second view assumes that rough chronological bounds have been discussed and end-date is set by the software engineering organization.

Tracking: - can be accomplished in different ways:

- Conducting periodic project status meetings in which each team member reports progress and problems.
- Evaluating the results of all reviews conducted throughout the software engineering process.
- Determining whether formal project milestones have been accomplished by the scheduled date.
- Comparing actual start-date to planned start-date for each project task.
- Meeting informally with practitioners to obtain their subjective assessment t of progress to date and problems on the horizon
- Use earned value analysis to assess progress quantitatively

Although there are many reasons why software is delivered late, most can be traced to one or more of the following root causes:

1. An unrealistic deadline established by someone outside the software development group and forced on managers and practitioners within the group.
2. Changing customer requirements that are not reflected in schedule changes.
3. An honest underestimate of the amount of effort and/or the number of resources that will be required to do the job.
4. Predictable and/or unpredictable risks that were not considered when the project commenced.
5. Technical difficulties that could not have been foreseen in advance.
6. Human difficulties that could not have been foreseen in advance.
7. Miscommunication among project staff that results in delays.
8. A failure by project management to recognize that the project is falling behind schedule and a lack of action to correct the problem.

OR

- Insufficient front-end planning
- Unrealistic project plan
- Project scope underestimated
- Customer/management changes
- Insufficient contingency planning
- Inability to track progress
- Inability to track problem early
- Insufficient number of checkpoints
- Staffing problems
- Technical complexity
- Sinking team spirit
- Unqualified resources
- Product is unknown to customer
- Requirements late
- Requirements of insufficient quality
- (Specifications of) delivered software late.
- (Specifications of) delivered software of insufficient quality

c) What is CMMI? State two objectives of CMMI. Briefly explain the CMMI maturity levels.
(Definition - 2 marks; Objectives - 2 marks; maturity levels - 4 marks; Diagram optional)
[Note: Any other relevant objectives shall be considered. **]**

Ans:

Definition- Capability Maturity Model Integration (CMMI) is a process improvement approach that helps organizations improves their performance.

OR

CMMI (Capability Maturity Model Integration) is a proven industry framework to improve product quality and development efficiency for both hardware and software

Objectives of CMMI:

Specific Objectives

- Establish Estimates
- Develop a Project Plan
- Obtain Commitment to the Plan

Generic Objectives:

- Achieve Specific Goals
- Institutionalize a Managed Process
- Institutionalize a Defined Process
- Institutionalize a Quantitatively Managed Process
- Institutionalize an Optimizing Process

CMMI maturity levels:

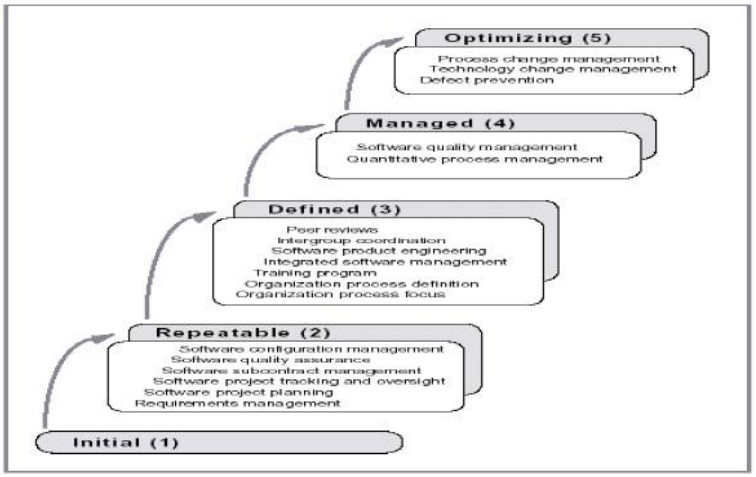
Level 1: Initial. The software process is characterized as ad hoc and occasionally even chaotic. Few processes are defined, and success depends on individual effort.

Level 2: Repeatable. Basic project management processes are established to track cost, schedule, and functionality. The necessary process discipline is in place to repeat earlier successes on projects with similar applications.

Level 3: Defined. The software process for both management and engineering activities is documented, standardized, and integrated into an organization wide software process. All projects use a documented and approved version of the organization's process for developing and supporting software. This level includes all characteristics defined for level

Level 4: Managed. Detailed measures of the software process and product quality are collected. Both the software process and products are quantitatively understood and controlled using detailed measures. This level includes all characteristics defined for level

Level 5: Optimizing. Continuous process improvement is enabled by quantitative feedback from the process and from testing innovative ideas and technologies. This level includes all characteristics defined for level 4.



OR

Level 0: Incomplete the process area (e.g., requirements management) is either not performed or does not achieve all goals and objectives defined by the CMMI for level 1 capability for the process area.

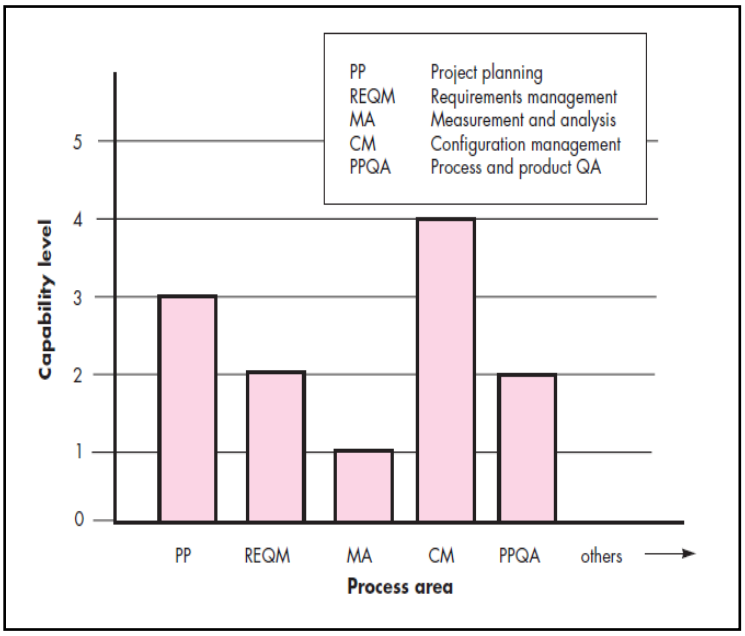
Level 1: Performed all of the specific goals of the process area (as defined by the CMMI) have been satisfied. Work tasks required to produce defined work products are being conducted.

Level 2: Managed all capability level 1 criteria have been satisfied. In addition, all work associated with the process area conforms to an organizationally defined policy; all people doing the work have access to adequate resources to get the job done; stakeholders are actively involved in the process area as required; all work tasks and work products are "monitored, controlled, and reviewed; and are evaluated for adherence to the process description"

Level 3: Defined all capability level 2 criteria have been achieved. In addition, the process is "tailored from the organization's set of standard processes according to the organization's tailoring guidelines, and contributes work products, measures, and other process-improvement information to the organizational process assets"

Level 4: Quantitatively managed all capability level 3 criteria have been achieved. In addition, the process area is controlled and improved using measurement and quantitative assessment. "Quantitative objectives for quality and process performance are established and used as criteria in managing the process"

Level 5: Optimized all capability level 4 criteria have been achieved. In addition, the process area is adapted and optimized using quantitative (statistical) means to meet changing customer needs and to continually improve the efficacy of the process area under consideration.



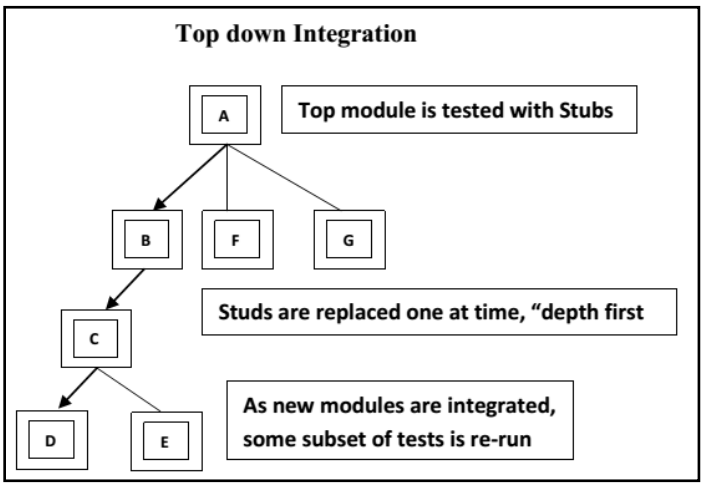
6. Attempt any four of the following: **16**

a) Compare top-down and bottom-up approach used for integration testing.
(1 mark for each point of comparison; any 4 points shall be considered)

Ans:

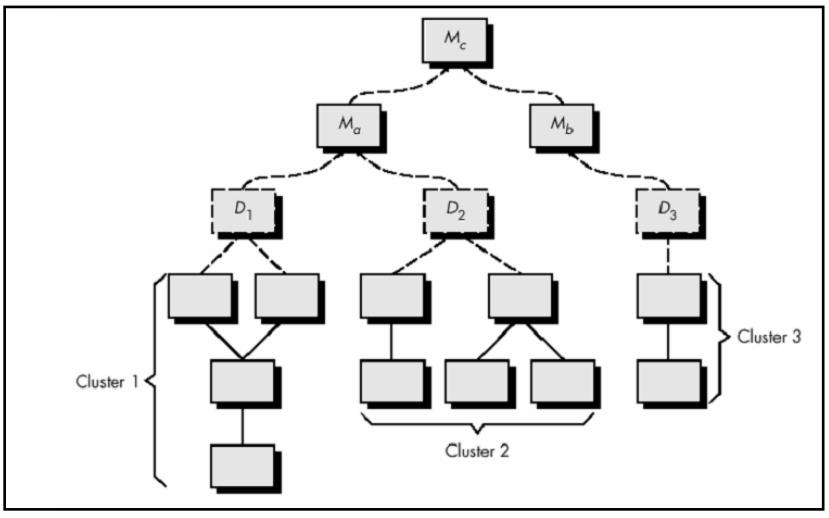
i) Top- down integration

- Main control module used as a test driver and stubs are substitutes for components directly subordinate to it.
- Subordinate stubs are replaced one at a time with real components (following the depth-first or breadth-first approach).
- Tests are conducted as each component is integrated.
- On completion of each set of tests and other stub is replaced with a real component.
- Regression testing may be used to ensure that new errors not introduced.



ii) Bottom- up integration

- Low level components are combined in clusters that perform specific software function.
- A driver (control program) is written to coordinate test case input and output.
- The cluster is tested.
- Drivers are removed and clusters are combined moving upward in the program structure.



OR

Top- Down integration	Bottom –Up Integration
• This is incremental approach to construction of the software architecture.	• Bottom up integration begins with sub modules and atomic checking
• Modules are integrated by moving downward through the control hierarchy, beginning with the main control module (main Program).	• Low- level components are combined into clusters that perform a specific software sub function
• Drivers are not required for test cases	• Drivers are required for test cases.
• Depth-first integration integrates all components on a major control path of the program structure.	• Different clusters are formed for the testing.

b) Describe different debugging strategies.
(List - 1 mark; Description - 1 mark each)

Ans:

There are three Different Debugging Strategies are available as follows: -

(1) Brute Force,
(2) Backtracking, and
(3) Cause Elimination.

1. **Brute Force:** This category of debugging is probably the most common and least efficient method for isolating the cause of a software error. Brute force debugging methods are applied when all else fails. Using a "let the computer find the error" philosophy, memory dumps are taken, run-time traces are invoked, and the program is loaded with WRITE statements. In the morass of information that is produced a clue is found that can lead us to the cause of an error. Although the mass of information produced may ultimately lead to success, it more frequently leads to wasted effort and time. Thought must be expended first.

2. **Backtracking:** It is a fairly common debugging strategy that can be used successfully in small programs. Beginning at the site where a symptom has been uncovered, the source code is traced backward (manually) until the site of the cause is found. Unfortunately, as the number of source lines increases, the number of potential backward paths may become unmanageably large.

3. **Cause Eliminatio**n: It is manifested by induction or deduction and introduces the concept of binary partitioning. Data related to the error occurrence are organized to isolate potential causes. A "cause hypothesis" is devised and the aforementioned data are used to prove or disprove the hypothesis. Alternatively, a list of all possible causes is developed and tests are conducted to eliminate each. If initial tests indicate that a particular cause hypothesis shows promise, data are refined in an attempt to isolate the bug.

c) What is software risk? Explain types of software risks.
(Software risk - 1 mark; types - 3 marks)

Ans:

Software Risk: - A software risk is anything which can cause a delay in software or stops the progress of a system or even terminates the software project.

There are two basic types of risks:

(a) Generic Risk Generic Risk is the general purpose possible threat to every software product.

(b) Product Specific Risk Product Specific risk can be find out only by those with a clear understanding of the technology going to be used for that project, the people and the environment that is particular to the software that is to be built.

Different Categories of Risks: -

(a) Project Risk: - Threaten the project plan. That is if project risk become real it is likely that project schedule will slip and that costs will increase. Project risks identity potential budgetary, schedule, personnel, resource, customer and requirement problem.

(b) Technical Risk: - Threaten the quality and timeliness of the software to be produced. If technical risk becomes real, implementation may become difficult or impossible.

(c) Business Risk: - Threaten the viability of the software to be built. Business risk often jeopardizes the product or the project.

(d) Market Risk:- Building product that no one wants

(e) Strategic Risk: - Building a product that no longer fits into the business strategy

(f) Management Risk: - Building a product that the sale force doesn't understand.

(g) Budget Risk: - Losing budgetary or personnel commitment

OR

a. ***Performance risk*** **-** the degree of uncertainty that the product will meet its requirements and be fit for its intended use.

b. ***Cost risk*** **-** the degree of uncertainty that the project budget will be maintained.

c. ***Support risk*** **-** the degree of uncertainty that the resultant software will be easy to correct, adapt, and enhance.

d. ***Schedule risk*** **-** the degree of uncertainty that the project schedule will be maintained and that the product will be delivered on time.

d) List different ways in which the project schedule can be tracked.
(Any two ways - 2 marks each)

Ans:

Project schedule can be tracked by using different scheduling tools and techniques.

1. **PERT and CPM:** Program evaluation and review technique (PERT) and the critical path method (CPM) are two project scheduling methods that can be applied to software development. Both techniques are driven by information already developed in earlier project planning activities: estimates of effort, a decomposition of the product function, the selection of the appropriate process model and task set, and decomposition of the tasks that are selected. Both PERT and CPM provide quantitative tools that allow you to (1) determine the critical path—the chain of tasks that determines the duration of the project, (2) establish "most likely" time estimates for individual tasks by applying statistical models, and (3) calculate "boundary times" that define a time "window" for a particular task.

2. **Time-Line Charts or Gantt chart**: A time-line chart can be developed for the entire project or separate charts can be developed for each project function or for each individual working on the project. All project tasks (for concept scoping) are listed in the left-hand column. The horizontal bars indicate the duration of each task. When multiple bars occur at the same time on the calendar, task concurrency is implied. The diamonds indicate milestones.

e) Compare software verification and software validation.
(Any 4 points - 1 mark each)

Ans:

- **Verification:**
  1. Verification is a set of activities which ensures that software correctly implements a specific function.
  2. Verification evaluates plans, documents, code, requirements and specifications etc.
  3. Verification takes place before validation.
  4. The input of verification could be check list, review of meetings, plans, walkthroughs, etc.
- **Validation:**
  1. Validation is different set of activities which ensures that the software has been built is traceable to customer.
  2. Validation evaluates product itself.
  3. Validation is next step to verification.
  4. An input of validation is actual testing of product.

OR

Verification	Validation
It answers the questions like: Am I building the product right?	It answers the question like: Am I building the right product?
Verification is a static practice of verifying documents, design, code and program.	Validation is a dynamic mechanism of validating and testing the actual product.
It does not involve executing the code.	It always involves executing the code.
It is human based checking of documents and files.	It is computer based execution of program
Verification uses methods like inspections, reviews, walkthroughs, and Desk-checking etc	Validation uses methods like black box (functional) testing, gray box testing, and white box (structural) testing etc.
Verification is to check whether the software conforms to specifications	Validation is to check whether software meets the customer expectations and requirements.
It can catch errors that validation cannot catch.	It can catch errors that verification cannot catch.

It is low level exercise.	It is High Level Exercise.
Target is requirements specification, application and software architecture, high level, complete design, and database design etc.	Target is actual product-a unit, a module, a bent of integrated modules, and effective final product.
It generally comes before validation.	It generally follows after verification.