V2V EDTECH LLP

Online Coaching at an Affordable Price.

OUR SERVICES:

- Diploma in All Branches, All Subjects
- Degree in All Branches, All Subjects
- BSCIT / CS
- Professional Courses

☎ +91 93260 50669

🌐 v2vedtech.com

▶ V2V EdTech LLP

v2vedtech

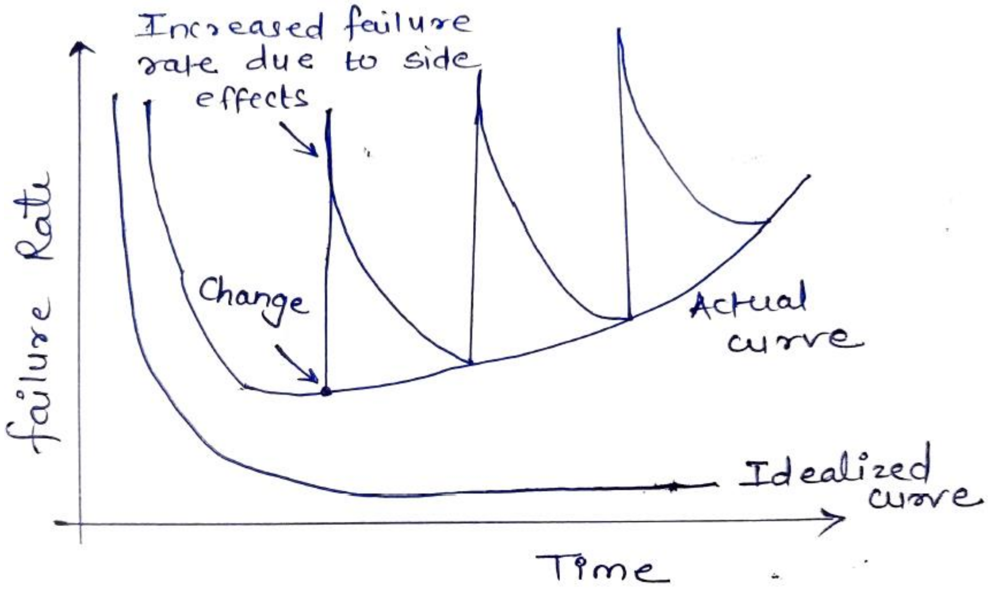
WINTER – 2023 EXAMINATION
Model Answer – Only for the Use of RAC Assessors

Subject Name: Software Engineering **Subject Code:** 22413

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 judgement 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.
8) As per the policy decision of Maharashtra State Government, teaching in English/Marathi and Bilingual (English + Marathi) medium is introduced at first year of AICTE diploma Programme from academic year 2021-2022. Hence if the students in first year (first and second semesters) write answers in Marathi or bilingual language (English +Marathi), the Examiner shall consider the same and assess the answer based on matching of concepts with model answer.

Q. No.	Sub-QN	Answer	Marking Scheme
1		**Attempt any FIVE of the following:**	**10 M**
	a)	**Define software. Draw the failure curve for software.**	**2 M**
	Ans	**Software** is: 1. Instructions (computer programs) that when executed provide desired features, function, and performance; 2. Data structures that enable the programs to adequately manipulate information, and 3. Descriptive information (documents) in both hard copy and virtual forms that describes the operation and use of the programs.	Correct definition- 1 M and diagram- 1 M

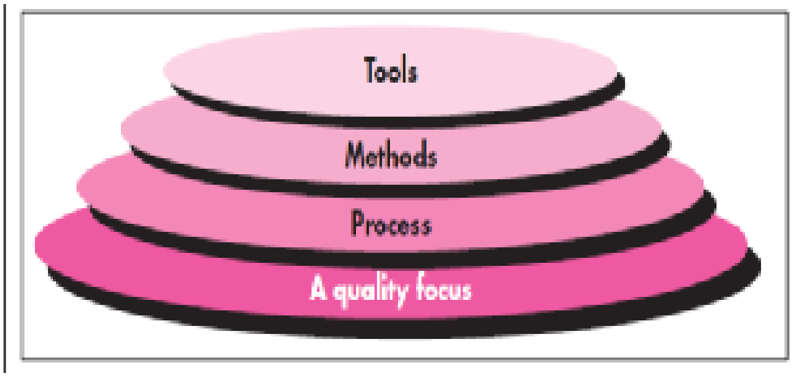
		 fig:- Failure curves for software	
	b)	**State two characteristics of software.**	**2 M**
	Ans	**Characteristics of software:** 1) Software is developed or engineered; it is not manufactured in the classical sense. 2) Software doesn't "wear out." But it does deteriorate! 3) Although the industry is moving toward component-based construction, most software continues to be custom built.	Any two correct Characteristics 1 M each
	c)	**State need of Software Requirement Specification (SRS).**	**2 M**
	Ans	**The need of SRS document is to provide:** 1) A detailed overview of software product, its parameters, and goals. 2) The description regarding the project's target audience and its user interface hardware and software requirements. 3) How clients, team and audience see the product and its functionality.	Any two points stating need of SRS- 2 M
	d)	**Define risk and list any two type of risk.**	**2 M**
	Ans	Risk is uncertain events associated with future events which have a probability of occurrence but it may or may not occur and if occurs it brings loss to the project. Following are the types of risk: 1) Generic risk 2) Product specific risk **OR**	

		1) Schedule/Time -related/Delivery Related Planning risks 2) Budget/Financial risks 3) Operational/Procedural Risks 4) Technical/Functional Performance Risks 5) Other Unavoidable Risks	
	e)	**Name two cost estimation approaches.**	**2 M**
	Ans	1) Heuristic Estimation Approach 2) Analytical Estimation Approach 3) Empirical Estimation Approach	Any two techniques-1 M each
	f)	**Define software quality control and quality assurance.**	**2 M**
	Ans	**Software quality control:** It is a procedure that focuses on fulfilling the quality requested. QC aims to identify and fix defects. It is a method to verify the quality-Validation. It always involves executing a program and hence it’s a Corrective technique. **Software quality assurance:** Quality assurance consists of the auditing and reporting functions of management. The goal of quality assurance is to provide management with the data necessary to be informed about product quality, thereby gaining insight and confidence that product quality is meeting its goals. It's a Preventive technique.	1 M for each definition
	g)	**List the phases of software quality assurance.**	**2 M**
	Ans	**Following are the phases of Software Quality Assurance:** i. Prepares an SQA plan for a project. ii. Participates in the development of the project’s software process description. iii. Reviews software engineering activities to verify compliance with the defined software process. iv. Audits designated software work products to verify compliance with those defined as part of the software process. v. Ensures that deviations in software work and work products are documented and handled according to a documented procedure. vi. Records any noncompliance and reports to senior management.	Any 2 phases-each 1 M
2.		**Attempt any <u>THREE</u> of the following:**	**12 M**

a) **Explain software engineering as layered technology approach.** **4 M**

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

Correct Diagram -1 M, explanation - 3 M



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 build 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.

b) **Explain the notations used for preparing a data flow diagram.** **4 M**

Ans

Symbol	Notation	Use
External entity	▭	External entities are objects outside the system
Process	⬭	A process receives input data and process output data with different form.
Data Flow	→	Dataflow is path for data to move from one end to another.
Data store	═	Data stores are repositories for data.

Circle: A circle (bubble) shows a process that transforms data inputs into data outputs.
Data Flow: A curved line shows the flow of data into or out of a process or data store.
Data Store: A set of parallel lines shows a place for the collection of data items. A data store indicates that the data is stored which can be used at a later stage or by the other processes in a different order. The data store can have an element or group of elements.
Source or Sink: Source or Sink is an external entity and acts as a source of system inputs or sink of system outputs.

Correct symbols with explanation - 1 M each

c) **Explain following elements of management spectrum :**
i) People
ii) Process
iii) Product
iv) Project

4 M

Ans The management Spectrum: 4p’s Effective software project management focuses on the four P’s: people, product, process, and project.

Explanation each element of management spectrum – 1 M

i) The People:

1. The “people factor” is so important that the Software Engineering Institute has developed a People Capability Maturity Model (People CMM) to continually improve its ability to attract, develop, motivate, organize, and retain the workforce needed to accomplish its strategic business objectives.
2. The people capability maturity model defines the following key practice areas for software people:

   a. Staffing
   b. communication and coordination
   c. work environment
   d. performance management

e. Training, compensation, competency analysis and development, career development, workgroup development, team/culture development and others.

3. Organizations that achieve high levels of People-CMM maturity have higher likelihood of implementing effective software project management practices.

ii) The Product:

1. Before a project can be planned, product objectives and scope should be established, alternative solutions should be considered and technical and management constraints should be identified.

2. Without this information, it is impossible to define reasonable (and accurate) estimates of the cost, an effective assessment of risk, a realistic breakdown of project tasks, or a manageable project schedule that provides a meaningful indication of progress.

3. Objectives identify the overall goals for the product (from the stakeholders' points of view) without considering how these goals will be achieved.

4. Scope identifies the primary data, functions, and behaviors that characterize the product

5. The alternatives enable managers and practitioners to select a "best" approach, given the constraints imposed by delivery deadlines, budgetary restrictions, personnel availability, technical interfaces, and other factors.

iii) The Process:

1. A software process provides the framework from which a comprehensive plan for software development can be established.

2. A small number of framework activities are applicable to all software projects, regardless of their size or complexity.

3. A number of different task sets—tasks, milestones, work products, and quality assurance points enable the framework activities to be adapted to the characteristics of the software project and the requirements of the project team.

4. Finally, umbrella activities—such as software quality assurance, software configuration management, and measurement occur throughout the process.

iv) The Project:

1. To manage complexity, we conduct planned and controlled software projects.

2. The success rate for present-day software projects may have improved but our project failure rate remains much higher than it should be.

3. To avoid project failure, a software project manager and the software engineers who build the product must avoid a set of common warning signs, understand the critical success factors that lead to good project management, and develop a common-sense approach for planning, monitoring, and controlling the project.

	d)	**Explain four basic principles of software project scheduling.**	**4 M**
	Ans	**Basic principles software project scheduling:** **1)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. **2)Interdependency:** 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. **3)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. **4)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. **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. **7)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. **8)Defined outcomes:** Every task that is scheduled should have a defined outcome for software projects such as a work product or part of a work product – Work products are often combined in deliverables.	Any four correct principles -1 M each
3.		**Attempt any THREE of the following:**	**12 M**
	a)	**Distinguish between perspective process model and agile process model.**	**4 M**

Ans

Sr No	Parameter	Perspective Process Model	Agile Process Model
1	**Quality**	It changes from analysis>Design>Code>Test	It focuses on all aspects of SDLC at any given time
2	**Quality control**	Detection & fixing during system and regression testing at the last phase of project	Early detection and fixing in each sprint followed by stabilization
3	**Continual improvement**	Lesson learned from previous release implemented in next release.	Lesson learned from Previous sprint will be implemented d in the next sprint.
4	**Risk**	No risk identification. firefighting during testing phase	Early identification and mitigation in every sprint
5	**Postmortem/retrospection**	After Every place	After Every sprint in Retrospection meeting
6	**Customer Feedback**	At the end of project	At the end of every sprint

Any four points each point 1 M

b) **Describe any four principles of communication for software engineering.** **4 M**

Ans

1M for one principle, Any four principles with description

Principle 1 Listen:

- Try to focus on the speaker‘s words, rather than formulating your response to those words.
- Ask for clarification if something is unclear but avoid constant interruptions.
- Never become contentious in your words or actions (e.g., rolling your eyes or shaking your head) as a person is talking.

Principle 2 Prepare before you communicate:

- Spend the time to understand the problem before you meet with others. If necessary, perform some research to understand business domain.
- If you have responsibility for conducting a meeting, prepare an agenda in advance of the meeting.

Principle 3 someone should facilitate the activity:

- Every communication meeting should have a leader (a facilitator)
- To keep the conversation moving in a productive direction,
- To mediate any conflict that does occur, and
- To ensure that other principles are followed.

Principle 4 Face-to-face communication is best:

- It usually works better when some other representation of the relevant information is present.
- For example, a participant may create a drawing /document that serves as a focus for discussion.

Principle 5 Take notes and document decisions:
Someone participating in the communication should serve as a recorder and write down all important points and decisions.

Principle 6 Strive for collaboration:

- Collaboration occurs when the collective knowledge of members of the team is used to describe product or system functions or features.
- Each small collaboration builds trust among team members and creates a common goal for the team.

Principle 7 Stay focused; modularize your discussion:

- The more people involved in any communication, the more likely that discussion will bounce from one topic to the next.
- The facilitator should keep the conversation modular; leaving one topic only after it has been resolved.

Principle 8 If something is unclear, draw a picture:

- Verbal communication goes only so far.
- A sketch or drawing can often provide clarity when words fail to do the job.

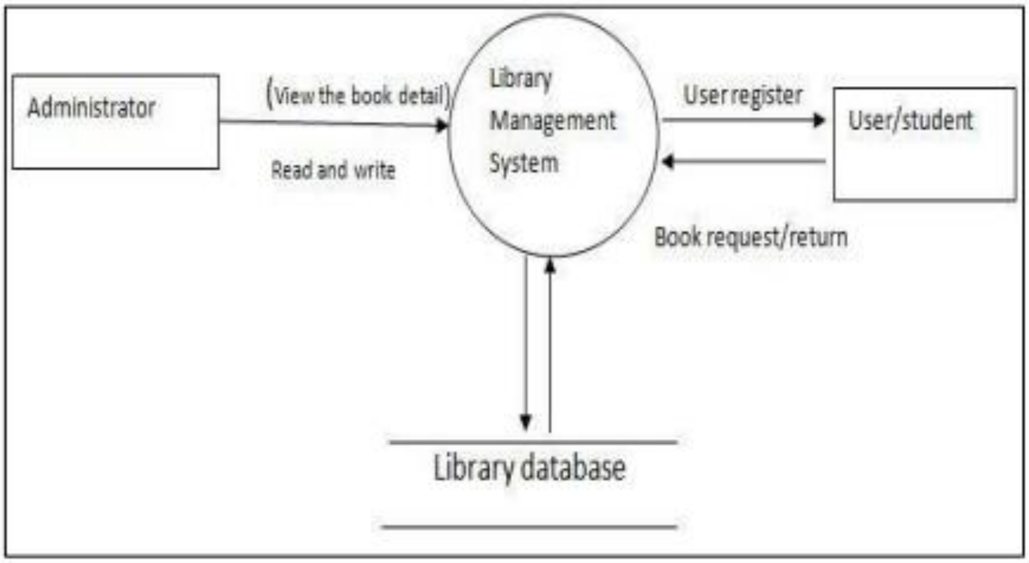
Principle 9

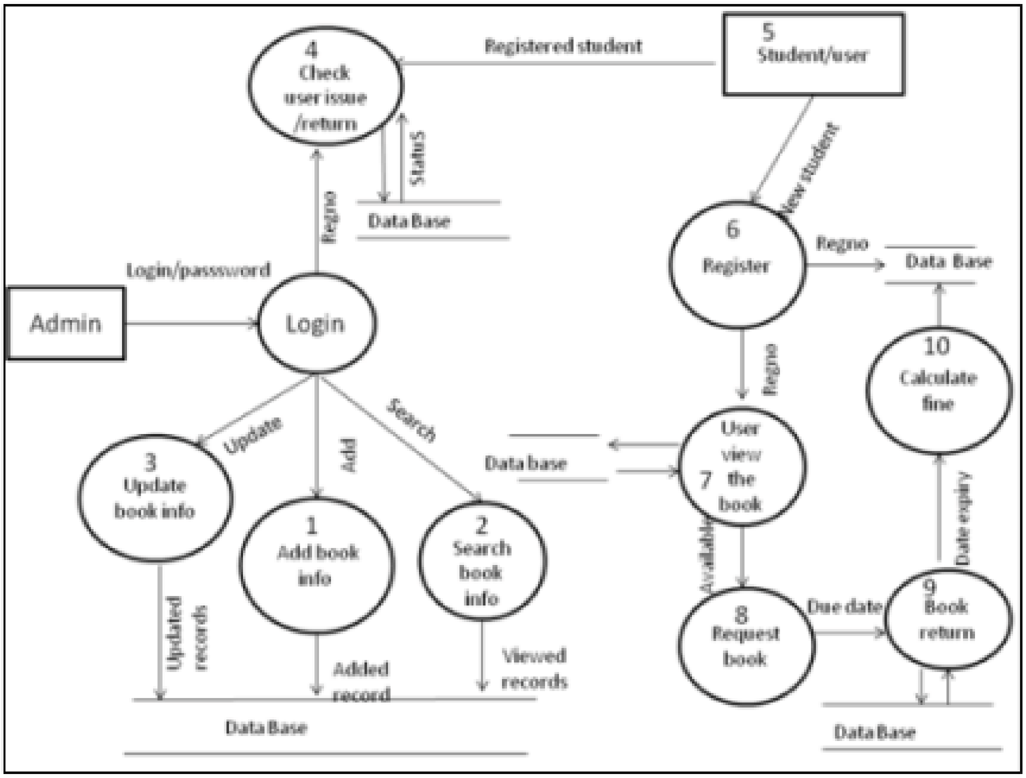
(a) Once you agree to something, move on.

(b) If you can’t agree to something, move on.

(c) If a feature or function is unclear and cannot be clarified at the moment, move on.

- The people who participate in communication should recognize that many topics require discussion and that moving on is sometimes the

		best way to achieve communication agility. **Principle 10 Negotiation is not a contest or a game: It works best when both parties win.** • There are many instances in which you and other stakeholders must negotiate functions and features, priorities, and delivery dates. If the team has collaborated well, all parties have a common goal. Still, negotiation will demand compromise from all parties.	
	c)	**Draw DFD0 and DFDI diagram for library management system.**	**4 M**
	Ans	 Administrator (View the book detail) Read and write Library Management System User register User/student Book request/return Library database **DFD Level 0 for Library Management System**	Level 0 2M Level 1 2M

		 4 Check user issue /return Registered student 5 Student/user New student Status Regno Data Base 6 Register Regno Data Base Login/password Admin Login 10 Calculate fine Update Add Search Regno Data base User view 7 the book 3 Update book info 1 Add book info 2 Search book info Available Date expiry 8 Request book Due date 9 Book return Updated records Added record Viewed records Data Base Data Base **DFD Level 1 for Library Management System**	
	d)	**Explain line of code metrics for size estimation.**	**4 M**
	Ans	**Line of code metrics for size estimation:** LOC counts the total number of lines of source code in a project. The units of LOC are: KLOC- Thousand lines of code NLOC- non-comment lines of code KDSI- Thousands of delivered source instruction. 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. Parameters to count LOC: 1. count only executable lines. 2. count executable lines plus data definitions. 3. count executable lines, data definitions and comments.	Correct Explanation 4 M

		4. count physical lines on input screen. Consider the following example for counting LOC: KCSI: thousands of changed source instructions. KSSI: thousands shipped source instructions. First Release of Product Y KCSI = KSSI = 50 KLOC Defects/KCSI = 2.0 Total number of defects = 2.0 × 50 = 100 Second Release, KCSI = 20 KSSI = 50+ 20 (new and changed lines of code) -4 (assuming 20% are changed lines of code) = 66 Defect/KCSI = 1.8 (assuming 10% improvement over the first release). Total number of additional defects = 1.8 x 20 = 36. Third Release, KCSI=30 KSSI 66+30 (new and changed lines of code) -6 (assuming 20% of changed lines of code) = 90. Targeted number of additional defects (no more than previous release) = 36. Defect rate target for the new and changed lines of code: 36/30= 1.2 defects/KCSI or lower.	
4.		**Attempt any THREE of the following:**	**12 M**
	a)	**Describe extreme programming with proper diagram.**	**4 M**
	Ans	Extreme programming is a lightweight, efficient, low-risk, flexible, predictable, scientific, and fun way to develop software. eXtreme Programming (XP) was conceived and developed to address the specific needs of software development by small teams in the face of vague and changing requirements. Extreme Programming is one of the Agile software development methodologies. It provides values and principles to guide the team's behavior. The team is expected to self-organize. Extreme Programming provides	1 M for Diagram and 3 M for explanation

specific core practices.

Where

- Each practice is simple and self-complete.
- Combination of practices produces more complex and emergent behavior.

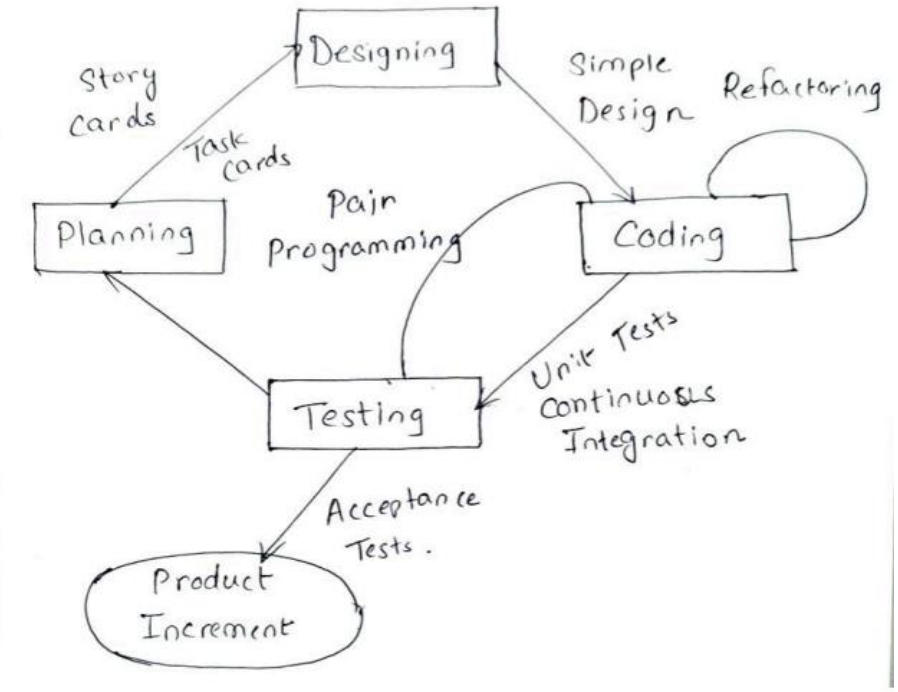


Fig: Extreme Programming

Extreme Programming is based on the following values-

- Communication
- Simplicity
- Feedback
- Courage
- Respect

Extreme Programming involves-

- Writing unit tests before programming and keeping all of the tests running at all times. The unit tests are automated and eliminate defects early, thus reducing the costs.
- Starting with a simple design just enough to code the features at hand and redesigning when required.

- Programming in pairs (called pair programming), with two programmers at one screen, taking turns to use the keyboard. While one of them is at the keyboard, the other constantly reviews and provides inputs.
- Integrating and testing the whole system several times a day.
- Putting a minimal working system into production quickly and upgrading it whenever required.
- Keeping the customer involved all the time and obtaining constant feedback. Iterating facilitates the accommodating changes as the software evolves with the changing requirements.

Extreme Programming solves the following problems often faced in

the software development projects-

- Slipped schedules: Short and achievable development cycles ensure timely deliveries.
- Cancelled projects: Focus on continuous customer involvement.

  ensures transparency with the customer and immediate resolution of

  any issues.
- Costs incurred in changes: Extensive and ongoing testing makes sure the changes do not break the existing functionality. A running working system always ensures sufficient time for accommodating changes such that the current operations are not affected.
- Production and post-delivery defects: Emphasis is on the unit tests to

  detect and fix the defects early.
- Misunderstanding the business and/or domain: Making the customer

  a part of the team ensures constant communication and clarifications.
- Business changes: Changes are inevitable and are accommodated at any point of time.
- Staff turnover: Intensive team collaboration ensures enthusiasm and goodwill. Cohesion of multi-disciplines fosters the team spirit.

  Extreme Programming takes the effective principles and practices to

  extreme levels.

  Extreme Programming

		• Code reviews are effective as the code is reviewed all the time. • Testing is effective as there is continuous regression and testing. • Design is effective as everybody needs to do refactoring daily. • Integration testing is important as integrate and test several times a day. • Short iterations are effective as the planning game for release planning and iteration planning.	
	b)	**Enlist core principles of software engineering practice.**	**4 M**
	Ans	1 **Core Principles of software engineering practice are:** 1.The Reason It All Exists A software system exists for one reason: to provide value to its users. All decisions should be made with this in mind. 2.Keep it simple stupid. All design should be as simple as possible, but no simpler. This facilitates having a more easily understood and easily maintained system. 3.Maintain the vision. A clear vision is essential to the success of a software project. 4. What You Produce, Others Will Consume. Always specify, design, and implement by keeping in mind that someone else will have to understand what you are doing. 5.Be open to the future. A system with a long lifetime has more value. Systems must be ready to adapt changes. 6. Plan ahead for reuse. The reuse of code and designs has a major benefit of using object-oriented technologies. 7. Think! Placing clear, complete thought before action almost always produces better results.	List of all core principles- 4 M
	c)	**Describe RMMM strategy with example.**	**4 M**
	Ans	**RMMM Plan** ➢ It is a part of the software development plan or a separate document. ➢ The RMMM plan documents all work executed as a part of risk analysis and used by the project manager as a part of the overall project plan.	1 M for introduction to risk and 3 M for RMMM plan example

- The risk mitigation and monitoring start after the project is started and the documentation of RMMM is completed.

Risk: Computer Crash

Mitigation:

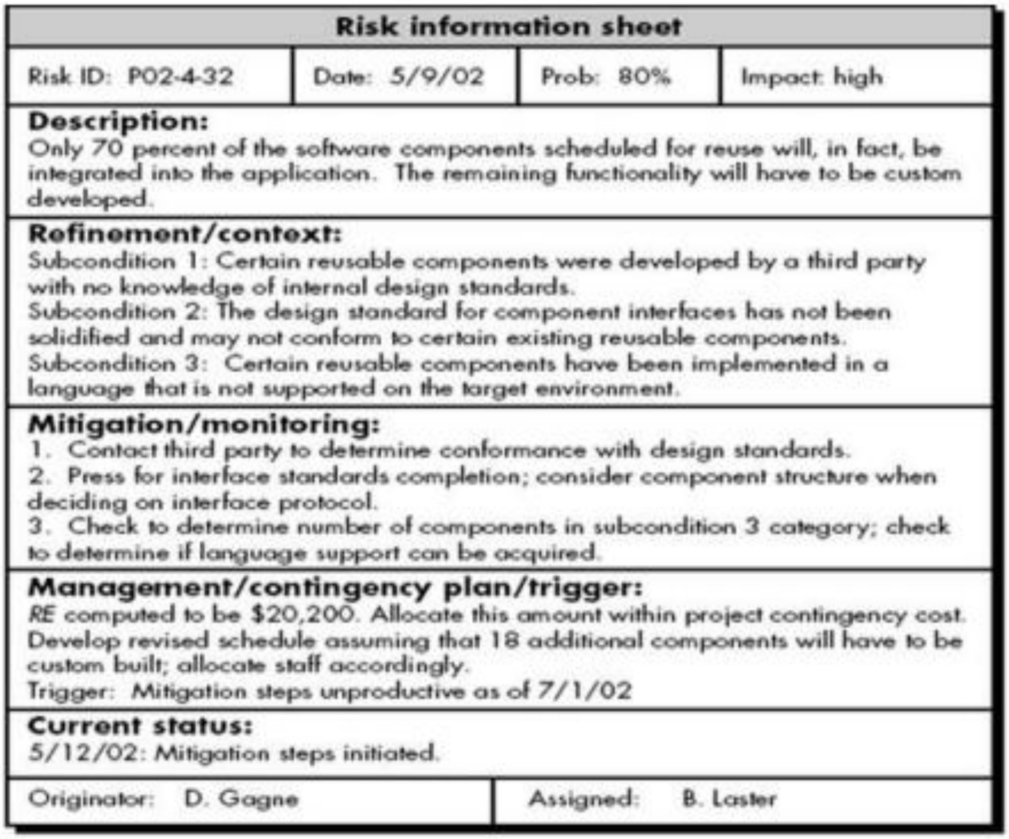
The cost associated with a computer crash resulting in the loss of data is crucial. A computer crash itself is not crucial, but rather the loss of data. A loss of data will result in not being able to deliver the product to the customer. This will result in not receiving a letter of acceptance from the customer. Without the letter of acceptance, the group will receive a failing grade for the course. As a result, the organization is taking steps to make multiple backup copies of the software in development and all documentation associated with it, in multiple locations. ·

Monitoring:

When working on the product or documentation, the staff members should always be aware of the stability of the computing environment they're working in. Any changes in the stability of the environment should be recognized and taken seriously. ·

Management:

The lack of a stable-computing environment is extremely hazardous to a software development team. In the event that the computing environment is found unstable, the development team should cease work on that system until the environment is made stable again or should move to a system that is stable and continue working there.



Risk information sheet

Risk ID: P02-4-32	Date: 5/9/02	Prob: 80%	Impact: high

Description:
Only 70 percent of the software components scheduled for reuse will, in fact, be integrated into the application. The remaining functionality will have to be custom developed.

Refinement/context:
Subcondition 1: Certain reusable components were developed by a third party with no knowledge of internal design standards.
Subcondition 2: The design standard for component interfaces has not been solidified and may not conform to certain existing reusable components.
Subcondition 3: Certain reusable components have been implemented in a language that is not supported on the target environment.

Mitigation/monitoring:
1. Contact third party to determine conformance with design standards.
2. Press for interface standards completion; consider component structure when deciding on interface protocol.
3. Check to determine number of components in subcondition 3 category; check to determine if language support can be acquired.

Management/contingency plan/trigger:
RE computed to be $20,200. Allocate this amount within project contingency cost. Develop revised schedule assuming that 18 additional components will have to be custom built; allocate staff accordingly.
Trigger: Mitigation steps unproductive as of 7/1/02

Current status:
5/12/02: Mitigation steps initiated.

Originator: D. Gagne	Assigned: B. Laster

d) Describe following project cost estimation approaches — **4 M**

i) Heuristic
ii) Empirical

Ans **Project cost estimation approaches**

2 M for Heuristic and 2 M for Empirical

i) Heuristic

Heuristic techniques assume that the relationships among the different project parameters can be modeled using suitable mathematical expressions. Once the basic (independent) parameters are known, the other (dependent) parameters can be easily determined by substituting the value of the basic parameters in the mathematical expression. Different heuristic estimation models can be divided into the following two classes: single variable model and the multi variable model.

Single variable estimation models provide a means to estimate the desired characteristics of a problem, using some previously estimated basic (independent) characteristic of the software product such as its size. A single variable estimation model takes the following form:

$$\text{Estimated Parameter} = c_1 * e_1^{d}{}_1$$

In the above expression, e is the characteristic of the software which has already been estimated (independent variable). Estimated Parameter is the dependent parameter to be estimated. The dependent parameters to be estimated could be effort, project duration, staff size, etc. c1 and d1 are constants. The values of the constants c1 and d1 are usually determined using data collected from past projects (historical data). The basic COCOMO model is an example of single variable cost estimation model.

A multivariable cost estimation model takes the following form:

$$\text{Estimated Resource} = c_1 * e_1^{d_1} + c_2 * e_2^{d_2} + \ldots$$

Where e1, e2, … are the basic (independent) characteristics of the software already estimated, and c1, c2, d1, d2, … are constants.

ii) **Empirical**

Empirical estimation is a technique or model in which empirically derived formulae are used for predicting the data that are a required and essential part of the software project planning step.

These techniques are usually based on the data that is collected previously from a project and based on some guesses, prior experience with the development of similar types of projects, and assumptions.

It uses the size of the software to estimate the effort.

In this technique, an educated guess of project parameters is made. Hence, these models are based on common sense. However, as there are many activities involved in empirical estimation techniques, this technique is formalized.

e) **Explain GANTT chart and it's application for project tracking with an example.** **4 M**

Ans **GANTT Charts**

Description and Example- 3 M
Application- 1 M

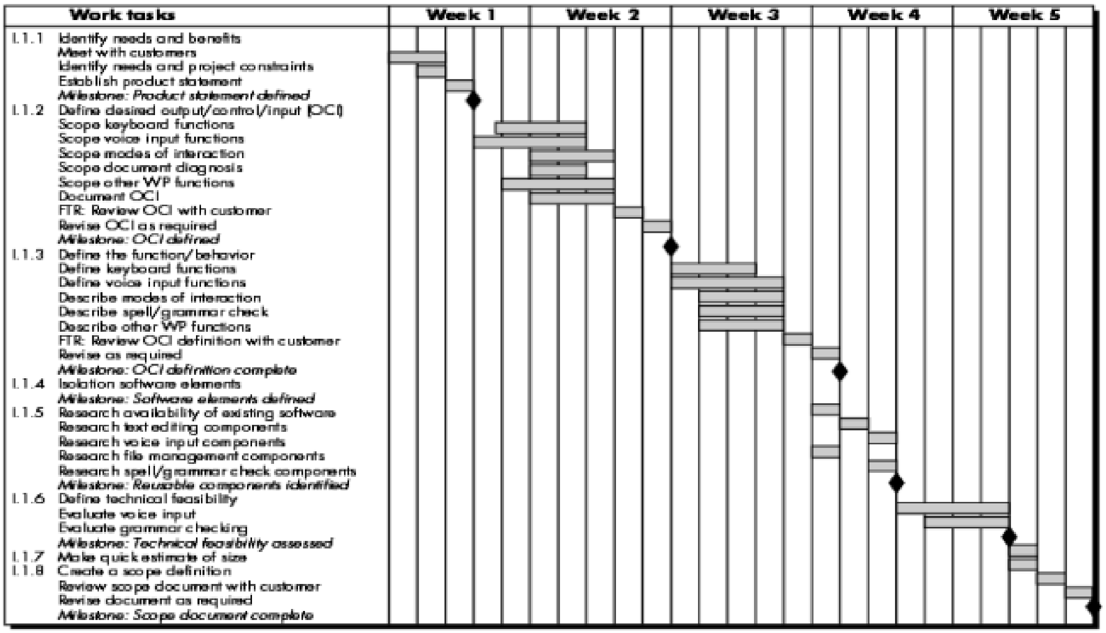
When creating software project schedule, we begin with a set of tasks. If automated tools are used, the work breakdown is input as a task network or task outline. Effort, duration and start date are then input for each task, in addition, tasks may be assigned to specific individuals.

Because of this input, a time-line chart, also called a Gantt chart is generated. A timeline chart can be developed for the entire project.

The figure below depicts a part of a software project schedule that emphasizes scoping task for a word-processing (WP) software product.

All project tasks 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 diamond indicates milestones.

Once the information necessary for the generation of a time-line chart has been input, many software project scheduling tools produce project tables – a tabular listing of all project tasks, their planned and actual start and end dates, and a variety of related information. Used in conjunction with the time-line chart, project tables enable to track progress.



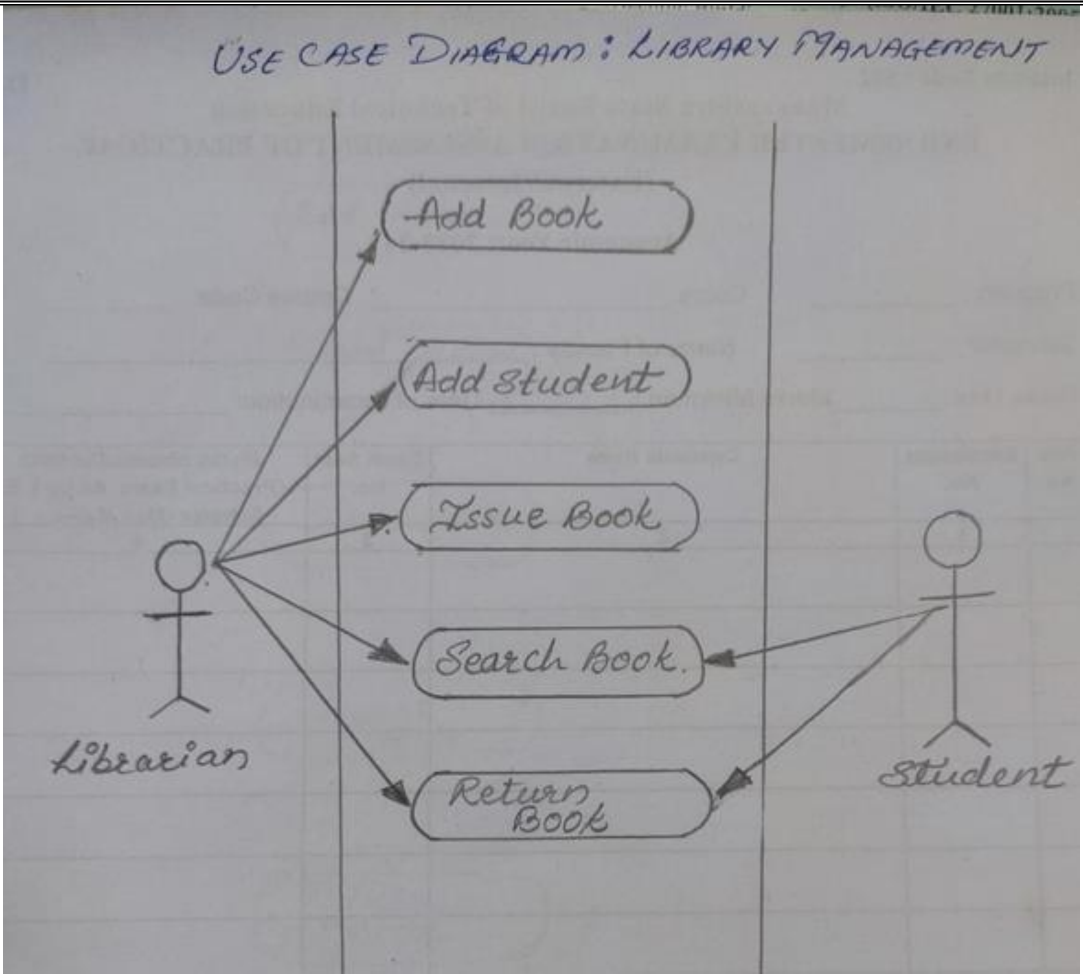
Application of Gantt Chart

- The sheer simplicity and ease-of-access of all relevant information make Gantt charts an ideal choice for teams to use for organizing their schedules. Due to this, Gantt charts are widely used in project management, IT, and development teams.

Apart from them, marketing, engineering, product launch, manufacturing teams can also use Gantt charts to get an overview of how things are rolling on the work front.

5.		**Attempt any <u>TWO</u> of the following:**	**12 M**
	a)	**Sketch use-case diagram for library management with minimum four use-cases and two actors.**	**6 M**

Ans



Use case diagram : 4 Valid Use Cases – 4 M ,

2 valid actors : 2 M = 6 M

b) **Differentiate between white box and black box testing. (Any six point)** **6 M**

Ans

Sr. No.	Black Box Testing	White Box Testing
1	It is a way of software testing in which the internal structure or the program or the code is hidden and nothing is known about it.	It is a way of testing the software in which the tester has knowledge about the internal structure or the code or the program of the software.
2	Implementation of code is not needed for black box testing.	Code implementation is necessary for white box testing.
3	It is mostly done by software testers.	It is mostly done by software developers.
4	No knowledge of implementation is needed.	Knowledge of implementation is required.

Differences between white box and black box testing any valid six points = 6 M, 1 M each

5	It can be referred to as outer or external software testing.	It is the inner or the internal software testing.
6	It is a functional test of the software.	It is a structural test of the software.
7	It is also called as closed box/ opaque box testing.	It is also called as transparent box / clear box testing.

c) **Describe a cocomo and cocomo-II models.** **6 M**

Ans **COCOMO :**

COCOCO model description : 3 M, COCOMO II model description : 3 M, Total - 6 M

COCOMO is a hierarchy of cost estimation models it includes basic, intermediate and detailed sub model.

1. Basic Model

The basic model aims at estimating, in a quick and rough fashion, most of the small to medium sized software projects.

Three modes of software development are considered in this model:

Organic: A small team of experienced developers develops software in a very familiar environment.

Embedded: The project has tight constraints, which might be related to the target processor.

Semidetached: It is an intermediate mode between the organic mode and embedded mode.

Depending on the problem at hand, the team might include a mixture of experienced and less experienced people with only a recent history of working together.

The Basic COCOMO equations take the form:

$E = a^{b} (KLOC)^{b}$

$D = c^{b} (E)^{d}$

SS = E/D persons

P = KLOC/E

Where E = effort

D = Deployment time

SS = staff size

P = productivity

ab ,bb ,cb ,db = Coefficients

2. Intermediate Model

In the Intermediate model Boehm introduced an additional set of 15 predictors called cost drivers in the intermediate model to take account of the software development environment.

Cost drivers are used to adjust the nominal cost of a project to the actual project environment, hence increasing the accuracy of the estimate.

The cost drivers are grouped into 4 categories:-

1. Product attributes

a. Required software reliability (RELY)

b. Database size (DATA)

c. Product complexity (CPLX)

2. Computer attributes

a. Execution time constraint (TIME)

b. Main store constraint (STOR)
c. Virtual machine volatility (VIRT)
d. Computer turnaround time (TURN)
3. Personnel attributes
a. Analyst capability (ACAP)
b. Application experience (AEXP)
c. Programmer capability (PCAP)
d. Virtual machine experience (VEXP)
e. Programming Language experience (LEXP)
4. Project attributes
a. Morden programming practices (MODP)
b. Use of software tool (TOOL)
c. Required development schedule (SCED)

3. Detailed COCOMO

A large amount of work is done by Boehm to capture all significant aspects of a software development.

It offers a means for processing all the project characteristics to construct a software estimate.

A software development is carried out in four successive phases:-

1. Plan/ requirements: This is the first phase of the development cycle. The requirement is analyzed, the product plan is set up and a full product specification is generated. This phase consumes from 6% to 8% of the effort and 10% to 40% of the development time.

2. Product Design: The second phase of the COCOMO development cycle is concerned with the determination of the product architecture and the specification of the subsystem. This phase requires from 16% to 18% of the nominal effort and can last from 19% to 38% of the development time.

3. Programming: The third phase of the COCOMO development cycle is divided into two sub phases: detailed design and code/unit test. This phase requires from 48% to 68% of the effort and lasts from 24% to 64% of the development time.

4. Integration/test: This phase of the COCOMO development cycle occurs before delivery.

This mainly consist of putting the tested parts together and then testing the final product this phase requires from 16% to 34% of the nominal effort and can last from 18% to 34% of the development time.

COCOMO II :

COCOMO-II is the revised version of the original Cocomo (Constructive Cost Model) and is developed at University of Southern California. It is the model that allows one to estimate the cost, effort and schedule when planning a new software development activity.

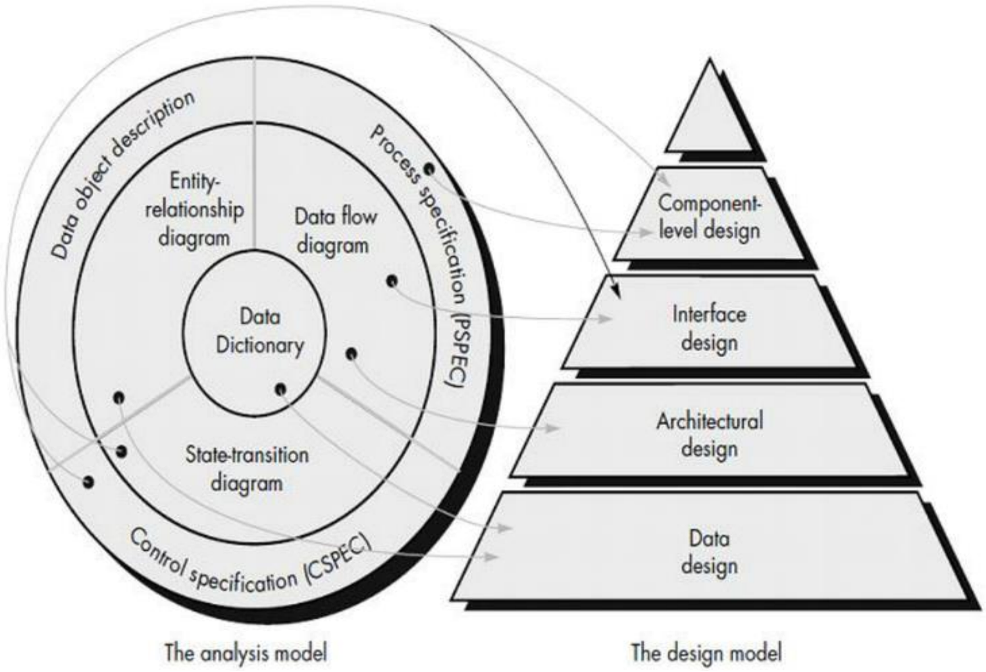
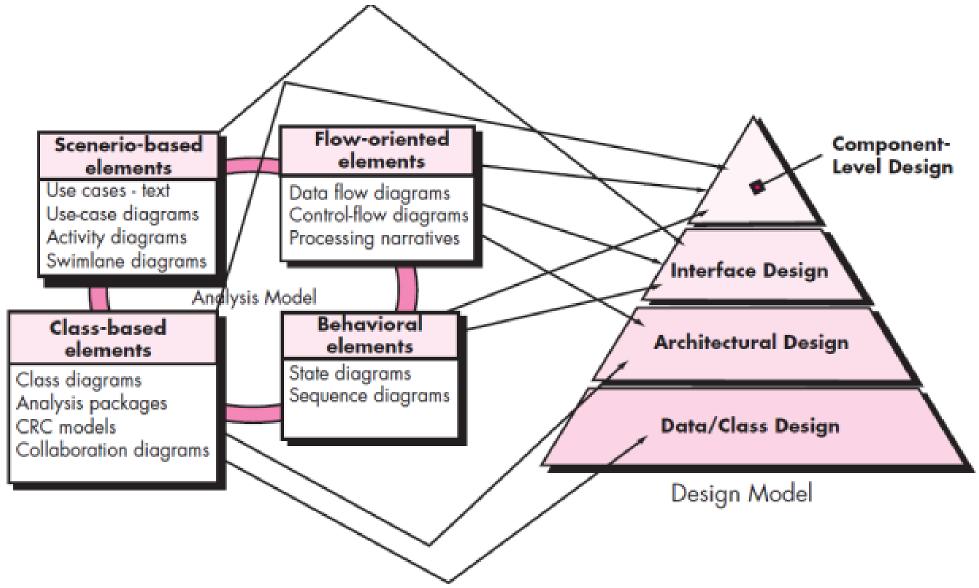
COCOMO II provides the following three-stage series of models for estimation of Application Generator, System Integration, and Infrastructure software projects:

COCOMO II has three different models:

1. **The Application Composition Model**

• Supports prototyping projects and projects where there is extensive reuse.

		• Based on standard estimates of developer productivity in Application (object) points /month. • Takes CASE tool use into account Formula is – PM = (NAP × (1 - %reuse/100)) / PROD – PM is the effort in person-months, NAP is the number of application points and PROD is the productivity **2. Early design model** • Estimates can be made after the requirements have been agreed. • Based on a standard formula for algorithmic models – PM = A × SizeB × M where – M = PERS × RCPX × RUSE × PDIF × PREX × FCIL × SCED; – A = 2.94 in initial calibration, Size in KLOC, B varies from 1.1 to 1.24 depending on novelty of the project, development flexibility, risk management approaches and the process maturity. **3. The reuse model** • Takes into account account black-box code that is reused without change and code that has to be adapted to integrate it with new code. • There are two versions: – Black-box reuse where code is not modified. An effort estimate (PM) is computed. – White-box reuse where code is modified. A size estimate equivalent to the number of lines of new source code is computed. This then adjusts the size estimate for new code. Reuse model estimates • For generated code: PM = (ASLOC * AT/100)/ATPROD – ASLOC is the number of lines of generated code – AT is the percentage of code automatically generated. – ATPROD is the productivity of engineers in integrating this code. Reuse model also estimates When code has to be understood and integrated: ESLOC = ASLOC * (1-AT/100) AT/100) * AAM. – ASLOC and AT as before. – AAM is the adaptation adjustment multiplier computed from the costs of changing the reused code, the costs of understanding how to integrate the code and the costs of reuse decision making.	
6.		**Attempt any TWO of the following:**	**12 M**
	a)	**Draw neat labelled diagram of translation of requirement model into design model and explain it with details.**	**6 M**
	Ans	**Diagram of translation of requirement model into design model**	labeled diagram of translation of

		 Data object description Entity-relationship diagram Data flow diagram Process specification (PSPEC) Data Dictionary State-transition diagram Control specification (CSPEC) Component-level design Interface design Architectural design Data design The analysis model The design model **OR**  Scenerio-based elements Use cases - text Use-case diagrams Activity diagrams Swimlane diagrams Flow-oriented elements Data flow diagrams Control-flow diagrams Processing narratives Analysis Model Class-based elements Class diagrams Analysis packages CRC models Collaboration diagrams Behavioral elements State diagrams Sequence diagrams Component-Level Design Interface Design Architectural Design Data/Class Design Design Model Each of the elements of the requirements model provides information that is necessary to create the four design models required for a complete specification of design. The flow of information during software design is illustrated in Figure above The requirements model, manifested by scenario-based, class-based, flow-oriented, and behavioral elements, feed the design task. Using design notation and design methods, design produces a data/class design, an architectural design, an interface design, and a component design. The data/class design	requirement model into design model : 3 M and explanation(any other valid description shall be given marks) : 3 M , total : 6 M

		transforms class models into design class realizations and the requisite data structures required to implement the software. The objects and relationships defined in the CRC diagram and the detailed data content depicted by class attributes and other notation provide the basis for the data design action. Part of class design may occur in conjunction with the design of software architecture. More detailed class design occurs as each software component is designed. The architectural design defines the relationship between major structural elements of the software, the architectural styles and design patterns that can be used to achieve the requirements defined for the system, and the constraints that affect the way in which architecture can be implemented. The architectural design representation—the framework of a computer-based system—is derived from the requirements model. The interface design describes how the software communicates with systems that interoperate with it, and with humans who use it. An interface implies a flow of information (e.g., data and/or control) and a specific type of behavior. Therefore, usage scenarios and behavioral models provide much of the information required for interface design. The component-level design transforms structural elements of the software architecture into a procedural description of software components. Information obtained from the class-based models, flow models, and behavioral models serve as the basis for component design. During design you make decisions that will ultimately affect the success of software construction and, as important, the ease with which software can be maintained.	
	b)	**Describe CMMI. Give significance of each level.**	**6 M**
	Ans	SEI CMMI is a process improvement approach that provides organizations with the essential elements of effective processes. • CMMI can help you make decisions about your process improvement plans. CMM stands for Capability Maturity Model. • Focuses on elements of essential practices and processes from various bodies of knowledge. • Describes common sense, efficient, proven ways of doing business (which you should already be doing) - not a radical new approach. • CMM is a method to evaluate and measure the maturity of the software development process of an organization. • CMM measures the maturity of the software development process on a scale of 1 to 5. **Capability Level 0: Incomplete** An "incomplete process" is a process that is either not performed or partially performed. One or more of the specific goals of the process area are not satisfied and no generic goals exist for this level since there is no reason to institutionalize a partially performed process. This is tantamount to Maturity Level 1 in the staged representation. **Capability Level 1: Performed** A Capability Level 1 process is a process that is expected to perform all of the Capability Level 1 specific and generic practices. Performance may not be stable and may not meet specific objectives such as quality, cost, and schedule, but useful work can be done. This is only a start, or baby-step, in process improvement. It means that you are doing something but you cannot prove that it is really working for you.	Description of CMMI : 1 M; significance of all levels : 5 M, total 6 M

		Capability Level 2: Managed A managed process is planned, performed, monitored, and controlled for individual projects, groups, or stand-alone processes to achieve a given purpose. Managing the process achieves both the model objectives for the process as well as other objectives, such as cost, schedule, and quality. As the title of this level indicates, you are actively managing the way things are done in your organization. You have some metrics that are consistently collected and applied to your management approach. **Capability Level 3: Defined** A capability level 3 process is characterized as a "defined process." A defined process is a managed (capability level 2) process that 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. **Capability Level 4: Quantitatively Managed** A capability level 4 process is characterized as a "quantitatively managed process." A quantitatively managed process is a defined (capability level 3) process that is controlled using statistical and other quantitative techniques. Quantitative objectives for quality and process performance are established and used as criteria in managing the process. Quality and process performance is understood in statistical terms and is managed throughout the life of the process. **Capability Level 5: Optimizing** An optimizing process is a quantitatively managed process that is improved, based on an understanding of the common causes of process variation inherent to the process. It focuses on continually improving process performance through both incremental and innovative improvements. Both the defined processes and the organization's set of standard processes are the targets of improvement activities.	
	c)	**Identify and enlist requirement for given modules of employee management software :** **i) Employee detail** **ii) Employee salary** **iii) Employee performance.**	**6 M**
	Ans	i. Employee detail ii. Employee salary iii. Employee performance This is with perspective of employee management software. Requirements for following modules will be as	Identification and enlisting requirements for 3 modules : 2 M each, total 6 M

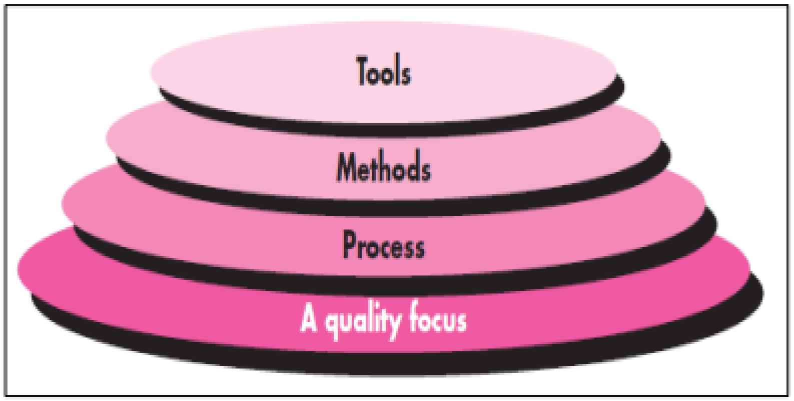
		i. Employee details a. Getting information about the customer b. Updation of employee details (department, change of address, emp_code etc.) c. Assignment of tasks, duties and responsibilities. d. Recording of employee attendance. **ii. Employee salary** a. Salary calculation b. Allowances, special bonus calculation and approval c. Tax statement/certificate d. Apply loan/approvals **iii. Performance** a. Recording annual performance b. Details about parameters for performance appraisal c. Analysis performance and determining hike in payment.	

SUMMER – 2023 EXAMINATION
Model Answer – Only for the Use of RAC Assessors

Subject Name: Software Engineering **Subject Code:** 22413

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 judgement 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.
8) As per the policy decision of Maharashtra State Government, teaching in English/Marathi and Bilingual (English + Marathi) medium is introduced at first year of AICTE diploma Programme from academic year 2021-2022. Hence if the students in first year (first and second semesters) write answers in Marathi or bilingual language (English +Marathi), the Examiner shall consider the same and assess the answer based on matching of concepts with model answer.

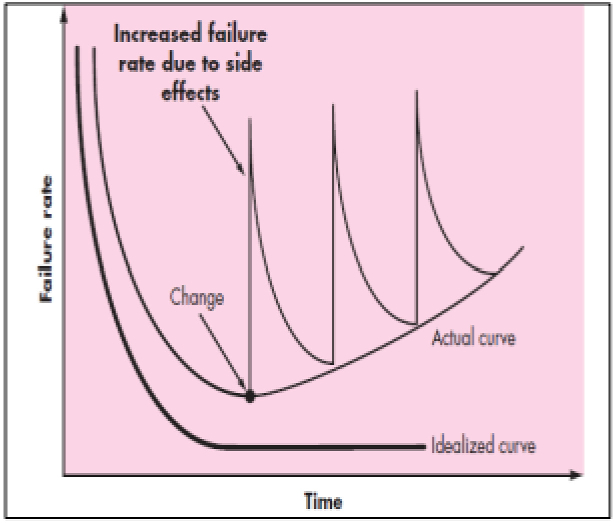
Q. No.	Sub Q. N.	Answer	Marking Scheme
1		**Attempt any FIVE of the following:**	**10 M**
	a)	**Draw layered approach of software engineering.**	**2 M**
	Ans		Correct Diagram -2 M
	b)	**Define software engineering.**	**2 M**
	Ans	Software engineering is the establishment and use of sound engineering principles in order to obtain economically software that is reliable and works efficiently on real machines.	Correct Definition-2 M

<table>
<tr><td></td><td><b>c)</b></td><td><b>List any four characteristics of good SRS.</b></td><td><b>2 M</b></td></tr>
<tr><td></td><td><b>Ans</b></td><td>Characteristics of SRS are:
• Correct
• Complete
• Unambiguous
• Verifiable
• Consistent
• Ranked for importance and/or stability
• Modifiable
• Traceable</td><td>Any 4 correct characteristics of SRS: 2 M</td></tr>
<tr><td></td><td><b>d)</b></td><td><b>Enlist four types of risks.</b></td><td><b>2 M</b></td></tr>
<tr><td></td><td><b>Ans</b></td><td>Types of risks are:
1. Generic risk
2. Product specific risk
3. Schedule / Time-Related / Delivery Related Planning Risks
4. Budget / Financial Risks
5. Operational / Procedural Risks
6. Technical / Functional / Performance Risks
7. Other Unavoidable Risks</td><td>Any 4 correct risk types: 2 M</td></tr>
<tr><td></td><td><b>e)</b></td><td><b>Define empirical estimation approach.</b></td><td><b>2 M</b></td></tr>
<tr><td></td><td><b>Ans</b></td><td>Empirical estimation techniques are based on making an educated guess of the project parameters. While using this technique, prior experience with development of similar products is helpful. Although empirical estimation techniques are based on common sense, different activities involved in estimation have been formalized over the years. Two popular empirical estimation techniques are: Expert judgment technique and Delphi cost estimation.</td><td>Correct Definition: 2 M</td></tr>
<tr><td></td><td><b>f)</b></td><td><b>Differentiate between quality assurance and quality control. (Any two points)</b></td><td><b>2 M</b></td></tr>
<tr><td></td><td><b>Ans</b></td><td>
<table>
<tr><th colspan="2">Quality Assurance</th><th colspan="2">Quality Control</th></tr>
<tr><td>1.</td><td>It is a procedure that focuses</td><td>1.</td><td>It is a procedure that</td></tr>
</table>
</td><td>Any two correct differentiation points: 2 M</td></tr>
</table>

	on providing assurance that quality requested will be achieved.		focuses on fulfilling the quality requested.
2.	QA aims to prevent defects.	2.	QC aims to identify and fix defects.
3.	It is a method to manage the quality- verification.	3.	It is a method to verify the quality-validation.
4.	It does not involve executing the program.	4.	It always involves executing a program.
5.	It's a preventive technique.	5.	It's a corrective technique.
6.	It's a proactive measure.	6.	It's a reactive measure
7.	It is the procedure to create the deliverables	7.	It is the procedure to verify that deliverables.
8.	QA involves in full software development life cycle.	8.	QC involves in full software testing life cycle.
9.	QA defines standards and methodologies in order to meet the customer requirements.	9.	QC confirms that the standards are followed while working on the product.
10.	It is performed before Quality Control.	10.	It is performed only after QA activity is done.
11.	It is a low-level activity, which can identify an error and mistakes which QC cannot.	11.	It is a High-Level Activity, which can identify an error that QA cannot.
12.	Its main motive is to prevent defects in the system. It is less time-consuming activity.	12.	Its main motive is to identify defects or bugs in the system. It is a more time-consuming activity.
13.	QA ensures that everything is executed in the right way, and that is why it falls under	13.	QC ensures that whatever we have done is as per the requirement, and that is why it falls under

	verification activity		validation activity.
14.	It requires the involvement of the whole team.	14.	It requires the involvement of the Testing team.
15.	The statistical technique applied on QA is known as SPC or Statistical Process Control (SPC).	15.	The statistical technique applied to QC is known as SQC or Statistical Quality Control.

	g)	**Define Software Quality Assurance Plan (SQAP).**	**2 M**
	Ans	• Software quality assurance plan consists of the auditing and reporting functions of management. • The goal of quality assurance is to provide management with the data necessary to be informed about product quality, thereby gaining insight and confidence that product quality is meeting its goals.	Correct Definition: 2 M
2.		**Attempt any <u>THREE</u> of the following:**	**12 M**
	a)	**Explain characteristics of software.**	**4 M**
	Ans	1. Software is developed or engineered; it is not manufactured in the classical sense. • Although some similarities exist between software development and hardware manufacture, the two activities are fundamentally different. • In both activities, high quality is achieved through good design, but the manufacturing phase for hardware can introduce quality problems that are non-existent (or easily corrected) for software. • Both activities are dependent on people, but the relationship between people applied and work accomplished is entirely different. • Software costs are concentrated in engineering. This means that software projects cannot be managed as if they were manufacturing projects. 1. Software doesn’t “wear out.”	Correct characteristics: 4 M



The idealized curve as shown in above figure is an oversimplification of actual failure models for software.

However, the implication is clear software doesn't wear out.

But it does deteriorate!

- This contradiction can best be explained by considering the “actual curve” shown in Figure.
- During its life, software will undergo change (maintenance). As changes are made, it is likely that some new defects will be introduced, causing the failure rate curve to spike as shown in figure.
- Before the curve can return to the original steady-state failure rate, another change is requested, causing the curve to spike again. Slowly, the minimum failure rate level begins to rise—the software is deteriorating due to change.

2. Although the industry is moving toward component-based construction, most software continues to be custom built.

The reusable components have been created so that the engineer can concentrate on the truly innovative elements of a design, that is, the parts of the design that represent something new.

- In the software world, it is something that has only begun to be achieved on a broad scale. A software component should be designed and implemented so that it can be reused in many different programs.
- A software component should be designed and implemented so that it can be reused in many different programs. Modern reusable components encapsulate both data and the processing that is applied to the data, enabling the software engineer to create new applications from reusable parts.
- For example, today’s interactive user interfaces are built with reusable

components that enable the creation of graphics windows, pull-down menus, and a wide variety of interaction mechanisms.

3. A software component should be designed and implemented so that it can be reused in many different programs.
    - It is the responsibility of software engineer to design and implement a software component in such a way that it should be reused easily in many different programs.
    - Latest reusable components summarize both data as well as the processing, which is applied to the data, which helps the software engineer to develop new applications from existing components.

b) Describe the notations used for preparing a structured chart. **4 M**

Ans Notations used in construction of structured chart are:

Any two correct notations with description: 4 M

1. Module

   It represents the process or task of the system. It is of three types.

   a. Control Module

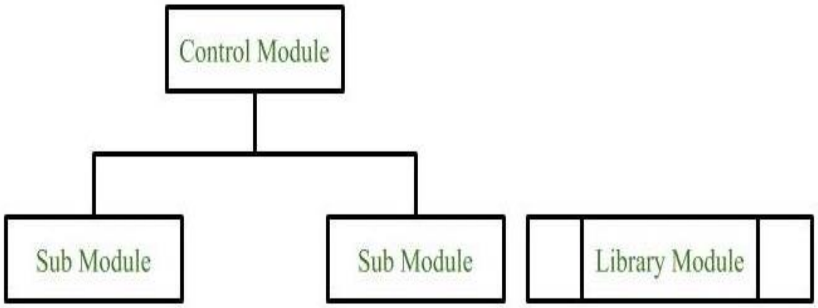
      A control module branches to more than one sub module.

   b. Sub Module

      Sub Module is a module which is the part (Child) of another module.

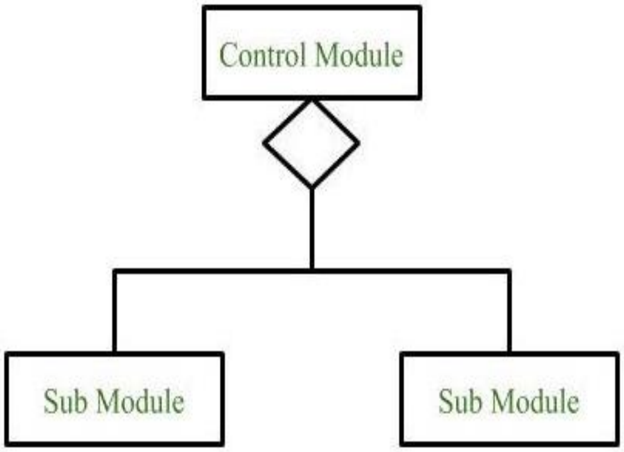
   c. Library Module

      Library Module are reusable and invokable from any module.



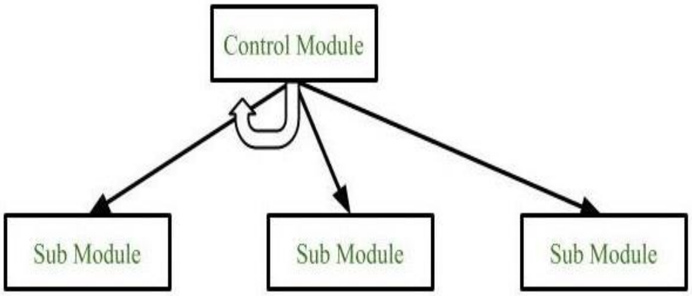
2. Conditional Call

   It represents that control module can select any of the sub module on the basis of some condition.



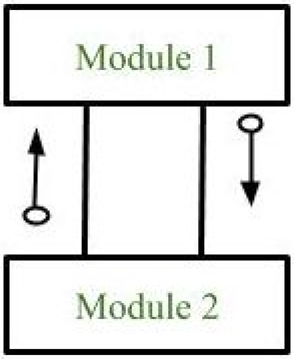
3. Loop (Repetitive call of module)

It represents the repetitive execution of module by the sub module.



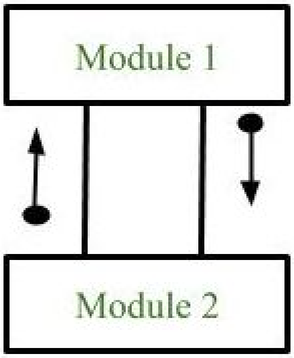
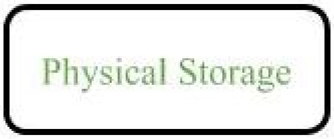
4. Data Flow

It represents the flow of data between the modules. It is represented by a directed arrow with empty circle at the end.



5. Control Flow

It represents the flow of control between the modules. It is represented by directed arrow with filled circle at the end.

		 6. Physical Storage Physical Storage is where all the information are to be stored. 	
	c)	**Explain the following 4P's management spectrum.**	**4 M**
	Ans	The management Spectrum: 4p’s Effective software project management focuses on the four P’s: People, Product, Process, and Project. The People: The “people factor” is so important that the Software Engineering Institute has developed a People Capability Maturity Model (People CMM) to continually improve its ability to attract, develop, motivate, organize, and retain the workforce needed to accomplish its strategic business objectives. 2. The people capability maturity model defines the following key practice areas for software people: a. Staffing b. communication and coordination c. work environment d. performance management e. Training, compensation, competency analysis and development, career development,	Explanation each element of management spectrum: 1 M

workgroup development, team/culture development and others.

3. Organizations that achieve high levels of People-CMM maturity have higher likelihood of implementing effective software project management practices.

The Product:

1. Before a project can be planned, product objectives and scope should be established, alternative solutions should be considered and technical and management constraints should be identified.

2. Without this information, it is impossible to define reasonable (and accurate) estimates of the cost, an effective assessment of risk, a realistic breakdown of project tasks, or a manageable project schedule that provides a meaningful indication of progress.

3. Objectives identify the overall goals for the product (from the stakeholders' points of view) without considering how these goals will be achieved.

4. Scope identifies the primary data, functions, and behaviors that characterize the product.

5. The alternatives enable managers and practitioners to select a "best" approach, given the constraints imposed by delivery deadlines, budgetary restrictions, personnel availability, technical interfaces, and other factors.

The Process:

1. A software process provides the framework from which a comprehensive plan for software development can be established.

2. A small number of framework activities are applicable to all software projects, regardless of their size or complexity.

3. Number of different task sets—tasks, milestones, work products, and quality assurance points enable the framework activities to be adapted to the characteristics of the software project and the requirements of the project team.

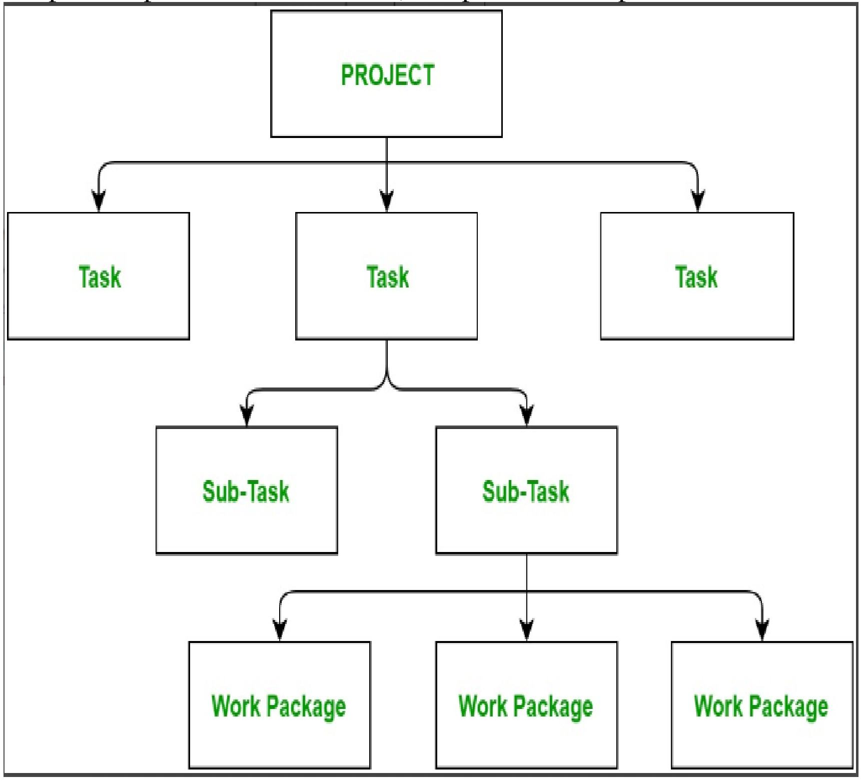
4. Finally, umbrella activities—such as software quality assurance, software configuration management, and measurement occur throughout the process.

The Project:

1. To manage complexity, we conduct planned and controlled software projects.

2. The success rate for present-day software projects may have improved but our project failure rate remains much higher than it should be.

3. To avoid project failure, a software project manager and the software engineers who

		build the product must avoid a set of common warning signs, understand the critical success factors that lead to good project management, and develop a common-sense approach for planning, monitoring, and controlling the project.	
	d)	**Describe work breakdown structure with diagram.**	**4 M**
	Ans	Dividing complex projects into simpler and manageable tasks is the process identified as Work Breakdown Structure (WBS). Usually, the project managers use this method for simplifying the project execution. In WBS, much larger tasks are broken down to manageable chunks of work. These chunks can be easily supervised and estimated. WBS is not restricted to a specific field when it comes to application. This methodology can be used for any type of project management. Steps: • Step-1: Identify the major activities of the project. • Step-2: Identify the sub-activities of the major activities. • Step-3: Repeat till undividable, simple and independent activities are created.  PROJECT Task Task Task Sub-Task Sub-Task Work Package Work Package Work Package <u>Construction of a WBS</u> Identifying the main deliverables of a project is the starting point for deriving a work breakdown structure. This important step is usually done by the project managers and the subject matter experts (SMEs) involved in the project. Once this step is completed, the subject matter experts start breaking down the high-level tasks into smaller chunks of work. In the process of breaking down the tasks, one can break them down into different levels of detail. One can detail a high-level task into ten sub-tasks while another can detail the same high-level task into 20 sub-tasks. Therefore, there is no hard and fast rule on how you should breakdown a task in WBS.	Correct explanation with diagram: 4 M

		Rather, the level of breakdown is a matter of the project type and the management style followed for the project. In general, there are a few "rules" used for determining the smallest task chunk. In "two weeks" rule, nothing is broken down smaller than two weeks' worth of work. This means, the smallest task of the WBS is at least two-week long. 8/80 is another rule used when creating a WBS. This rule implies that no task should be smaller than 8 hours of work and should not be larger than 80 hours of work. One can use many forms to display their WBS. Some use tree structure to illustrate the WBS, while others use lists and tables. Outlining is one of the easiest ways of representing a WBS.	
3.		**Attempt any <u>THREE</u> of the following:**	**12 M**
	a)	**State the requirements to apply RAD.**	**4 M**
	Ans	• Following are the requirements to apply RAD model: • RAD model can be applied successfully to the projects in which clear modularization is possible. • It should be used where the requirements change during the project and working prototypes are to be presented to customer in small iterations of 2-3 months. • Rapid Application Development is best when you've got a tight deadline to meet or are under pressure to deliver something that works. • RAD works well only if high skilled engineers (developers / designers) are available and the customer is also committed to achieve the targeted prototype in the given time frame. • RAD model should be chosen only if domain experts are available with relevant business knowledge. • RAD works well wherever there's a greater focus on user interface rather than non-GUI programs. • RAD should be used only when a system can be modularized to be delivered in an incremental manner. • It should be used only if the budget permits use of automated code generating tools.	Relevant requirement explanation give marks
	b)	**Describe any four software coding principles.**	**4 M**
	Ans	**The coding principles are that guide the coding task and are closely aligned with programming style , programming language , and programming methods. Coding principles can be stated as : -** **Preparation principles: Before you write one line of code, be sure you** • Understand of the problem you're trying to solve. • Understand basic design principles and concepts. • Pick a programming language that meets the needs of the software to be built and the environment in which it will operate. • Select a programming environment that provides tools that will make your work easier. • Create a set of unit tests that will be applied once the component you code is	For each principle give 1 M. Explain any 4 principles

completed.

Programming principles: As you begin writing code, be sure you.

- Constrain your algorithms by structured programming practice.
- Consider the use of pair programming.
- Select data structures that will meet the needs of the design.
- Understand the software architecture and create interfaces that are consistent with it.
- Keep conditional logic as simple as possible.
- Create nested loops in a way that makes them easily testable.
- Select meaningful variable names and follow other local coding standards.
- Write code that is self-documenting.
- Create a visual layout (e.g., indentation and blank lines) that aids understanding.

Validation Principles: After you've completed your first coding pass, be sure you

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

c) **Prepare decision table for accessing secured network.** **4 M**

Ans

- Decision table is a software testing technique used to test system behavior for different input combinations.
- This is a systematic approach where the different input combinations and their corresponding system behavior (Output) are captured in a tabular form. That is why it is also called as a **Cause-Effect table** where Cause and effects are captured for better test coverage.

For correct decision table with proper condition give 3 M and explanation 1 M

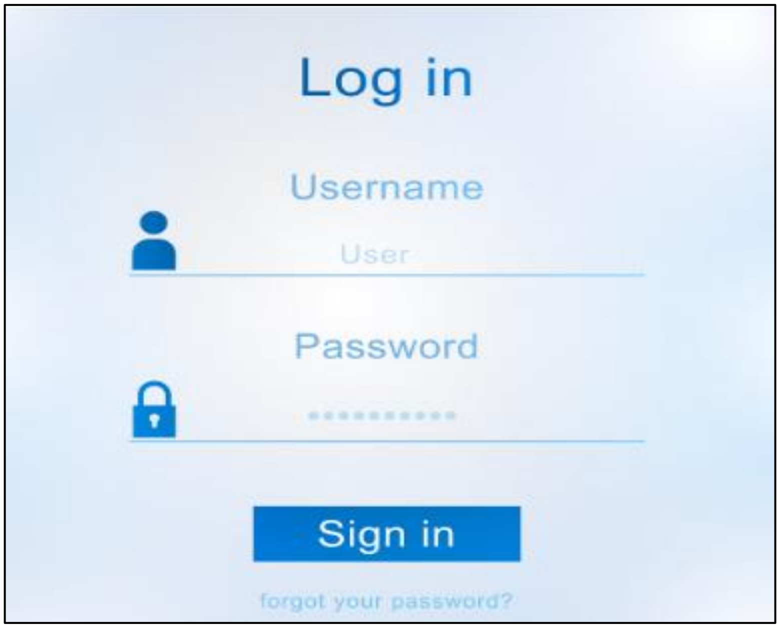
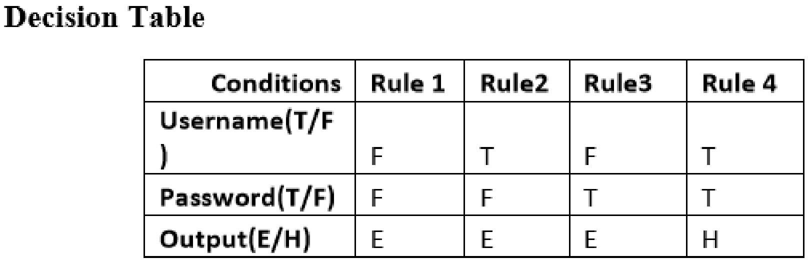


Fig: Login Screen of Network

- The condition is simple if the user provides correct username and password the user will be redirected to the homepage.

- If any of the input is wrong, an error message will be displayed.

Decision Base Table for Login Screen



Decision Table

Conditions	Rule 1	Rule2	Rule3	Rule 4
Username(T/F)	F	T	F	T
Password(T/F)	F	F	T	T
Output(E/H)	E	E	E	H

Legend:

T – Correct username/password

F – Wrong username/password

E – Error message is displayed

H – Home screen is displayed

Interpretation:

- Case 1 – Username and password both were wrong. The user is shown an error message.
- Case 2 – Username was correct, but the password was wrong. The user is shown an error message.
- Case 3 – Username was wrong, but the password was correct. The user is shown an error message.
- Case 4 – Username and password both were correct, and the user navigated to homepage.

d)	**State project size estimation techniques and explain any one.**	**4 M**

Ans

Currently two metrics are popularly being used widely to estimate size:

Lines of code (LOC) and function point (FP).

For listing project size estimation technique 1M, 3 M for explanation of any one technique

a) Lines of Code (LOC)

- LOC is the simplest among all metrics available to estimate project size.
- This metric is very popular because it is the simplest to use.
- Using this metric, the project size is estimated by counting the number of source instructions in the developed program.
- Obviously, while counting the number of source instructions, lines used for commenting the code and the header lines should be ignored.
- Accurate estimation of the LOC count at the beginning of a project is

very difficult.

- To estimate the LOC count at the beginning of a project, project managers usually divide the problem into modules, and each module into submodules and so on, until the sizes of the different leaf-level modules can be approximately predicted.
- To be able to do this, experience in developing similar products is helpful. By using the estimation of the lowest level modules, project managers arrive at the total size estimation.

Consider example of mechanical CAD software.

The mechanical CAD software will accept two dimensional and three-dimensional data. User will interact with system through user interface and peripheral devices like mouse, plotter, laser printer. All geometric data and supporting information are stored in database. Design analysis modules will be developed to produce the required output, which will be displayed on variety of graphic devices.

For size estimation CAD software is divided into major functions and size estimation of each function is done.

Function	Estimated LOC
User interface	2,300
Two-dimension geometric analysis	5,300
Three-dimension geometric analysis	6,800
Database management	3,500
Computer Graphics display facilities	4,950
Peripheral Control Function	2,100
Design analysis module	8,400
Estimated line of code	33,200

b) Function Point (FP):

- The conceptual idea behind the function point metric is that the size of a software product is directly dependent on the number of different Functions or features it supports.
- A software product supporting many features would certainly be of larger size than a product with a less number of features.
- Each function when invoked reads some input data and transforms it to the corresponding output data.
- For example, the issue book feature of a Library Automation Software

		takes the name of the book as input and displays its location and the number of copies available. • Thus, a computation of the number of input and the output data values to a system gives some indication of the number of functions supported by the system. • In addition to the number of basic functions that a software performs, the size is also dependent on the number of files and the number of interfaces. • Besides using the number of input and output data values, function point metric computes the size of a software product (in units of functions points or FPs) using three other characteristics of the product as shown in the following expression. • The size of a product in function points (FP) can be expressed as the weighted sum of these five problem characteristics. • The weights associated with the five characteristics were proposed empirically and validated by the observations over many projects. • Function point is computed in two steps. The first step is to compute the unadjusted function point (UFP). UFP = (Number of inputs)*4 + (Number of outputs)*5 + (Number of inquiries)*4 + (Number of files)*10 + (Number of interfaces)*10 • Once the unadjusted function point (UFP) is computed, the technical complexity factor (TCF) is computed next. • TCF refines the UFP measure by considering fourteen other factors such as high transaction rates, throughput, and response time requirements, etc. • Each of these 14 factors is assigned from 0 (not present or no influence) to 6 (strong influence). • The resulting numbers are summed, yielding the total degree of influence (DI). • Now, TCF is computed as TCF=(0.65+0.01*DI). As DI can vary from 0 to 70, TCF can vary from 0.65 to 1.35. Finally, FP=UFP*TCF.	
4.		**Attempt any <u>THREE</u> of the following:**	**12 M**
	a)	**Explain adaptive software development method w.r.to** **i) Speculation.** **ii) Collaboration.**	**4 M**
	Ans	**i) Speculation** • During *speculation*, the project is initiated, and *adaptive cycle planning* is	For Speculation

		conducted. • Adaptive cycle planning uses project initiation information—the customer's mission statement, project constraints (e.g., delivery dates or user descriptions), and basic requirements—to define the set of release cycles (software increments) that will be required for the project. • No matter how complete and farsighted the cycle plan, it will invariably change. • Based on information obtained at the completion of the first cycle, the plan is reviewed and adjusted so that planned work better fits the reality in which an ASD team is working. **ii)Collaboration** • Motivated people use *collaboration* in a way that multiplies their talent and creative output beyond their absolute numbers. • This approach is a recurring theme in all agile methods. But collaboration is not easy. • It encompasses communication and teamwork, but it also emphasizes individualism, because individual creativity plays an important role in collaborative thinking. • It is, above all, a matter of trust. • People working together must trust one another to (1) criticize without animosity, (2) assist without resentment, (3) work as hard as or harder than they do, (4) have the skill set to contribute to the work at hand, and (5) Communicate problems or concerns in a way that leads to effective action.	Explanation 2 M and for collaboration Explanation 2 M
	b)	**Describe any four core principles of software engineering practices.**	**4 M**
	Ans	Following are core principles of Software Engineering Practices: **1. Reason it all exists.** i. The software system exists to provide value for the user. ii. Before specifying the problem the requirement and the specifications have to be laid down. iii. The hardware and the software platform to be decided for implementation. **2. Keep it simple stupid** i. The terms and the design used for development of the project should be kept simple and easily understandable. ii. All the terms used should be easy to facilitate the basic concept of the project. **3. Maintain the vision** i. A clear vision is important for the development of software. ii. Compromising the architectural vision of the project weakens the development of the software.	For each principle give 1 M

iii. The developer should hold the vision and ensure the successful development and deployment of the software.

4. What you reproduce, someone else will have to consume. (Implement knowing someone else will have to understand what you are doing)

i. Always specify, design, and implement knowing that someone else is going to understand what is being developed.
ii. Customers for the product development are very large.
iii. Design the data structure and the implementation keeping implementation in mind and the end user.
iv. Code with the concern that the product must be implemented and maintained by the end user.

5. Be open to the future

i. The system designed today should be adaptable to the development and changes in the future at a low cost.
ii. There should not be many changes to the software to adapt to the new changes in the future development.

6. Plan ahead for reuse

i. The design and specifications should be developed in such a way that they can be reused for other implementations.
ii. The code and the design should be well documented for the use in future.

7. Think!

i. Before designing and implementation a proper thought should be to the result.
ii. Proper data structure and the design and implementation strategy should be developed if the software needs modification in the future.

	c)	**Describe RMMM strategy.**	**4 M**
	Ans	Risk mitigation, monitoring, and management (RMMM) plan. A risk management strategy can be included in the software project plan, or the risk management steps can be organized into a separate Risk Mitigation, Monitoring and Management Plan. The RMMM plan documents all work performed as part of risk analysis and is used by the project manager as part of the overall project plan. Once RMMM has been documented and the project has begun, risk mitigation and monitoring steps commence. Risk mitigation is a problem avoidance activity. Risk monitoring is a project tracking activity with three primary objectives: (1) To assess whether predicted risks do, in fact, occur. (2) To ensure that risk aversion steps defined for the risk are being properly applied;	Correct explanation give marks

and

(3) To collect information that can be used for future risk analysis.

In many cases, the problems that occur during a project can be traced to more than one risk.

Another job of risk monitoring is to attempt to allocate origin (what risk(s) caused which problems throughout the project).

An effective strategy must consider three issues: • Risk avoidance • Risk monitoring • Risk management and contingency planning.

If a software team adopts a proactive approach to risk, avoidance is always the best strategy.

This is achieved by developing a plan for risk mitigation.

Consider example of Staff Turnover risk:

To mitigate this risk, project management must develop a strategy for reducing turnover.

Among the possible steps to be taken are

• Meet with current staff to determine causes for turnover (e.g., poor working conditions, low pay, and competitive job market).

• Mitigate those causes that are under our control before the project starts.

• Once the project commences, assume turnover will occur and develop techniques to ensure continuity when people leave.

• Organize project teams so that information about each development activity is widely dispersed.

• Define documentation standards and establish mechanisms to be sure that documents are developed in a timely manner.

• Conduct peer reviews of all work (so that more than one person is "up to speed).

• Assign a backup staff member for every critical technologist. As the project proceeds, risk monitoring activities commence.

The project manager monitors factors that may provide an indication of whether the risk is becoming more or less likely.

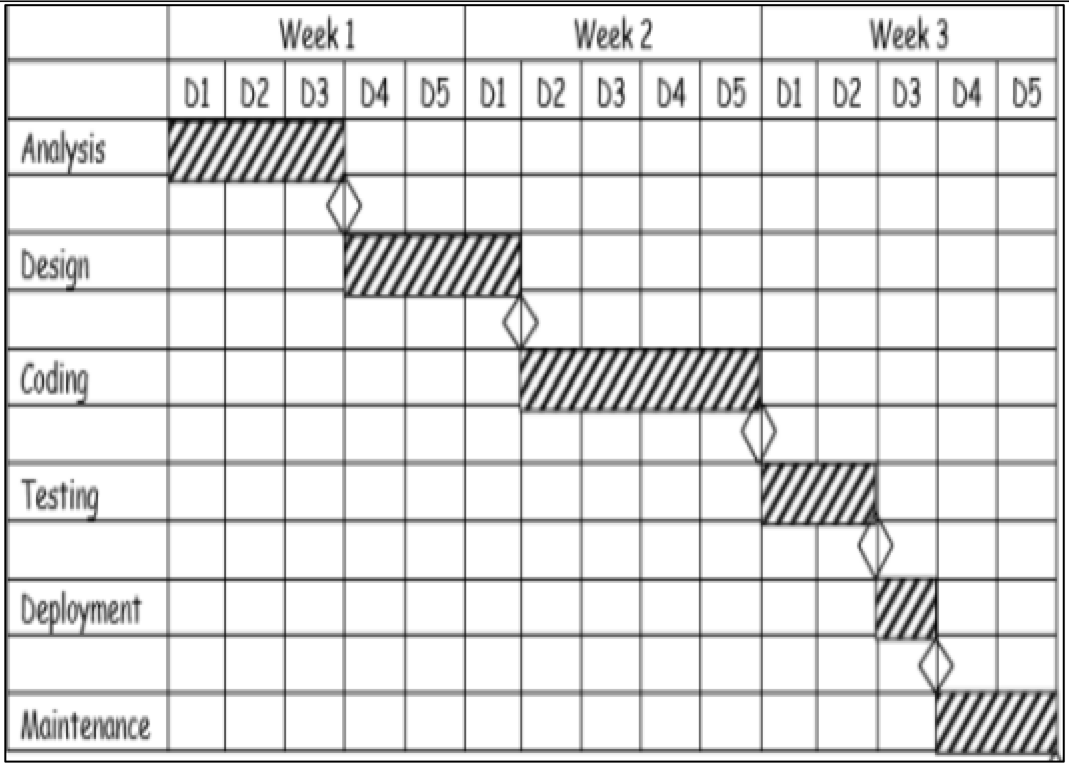
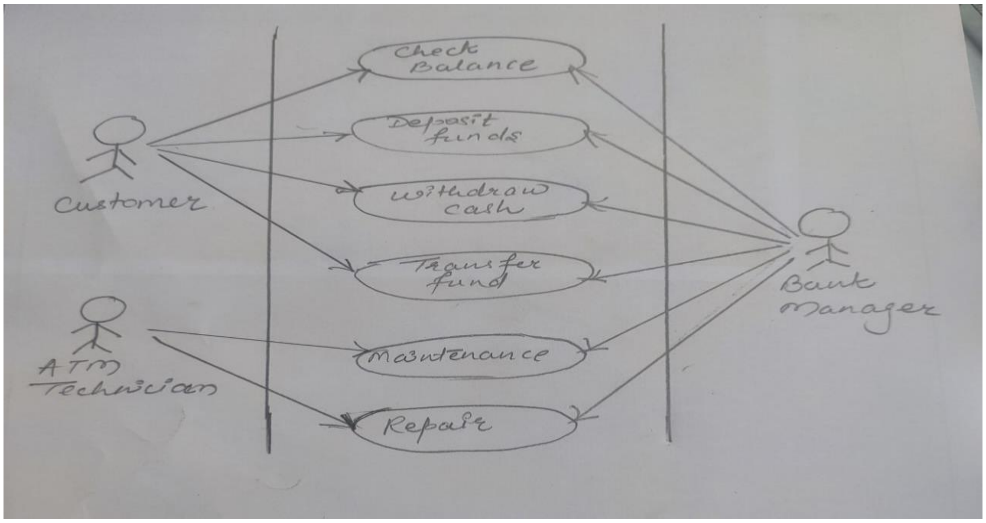
In the case of high staff turnover, the following factors can be monitored:

• General attitude of team members based on project pressures.

• The degree to which the team has jelled.

• Interpersonal relationships among team members.

		• Potential problems with compensation and benefits. • The availability of jobs within the company and outside it. In addition to monitoring these factors, the project manager should monitor the effectiveness of risk mitigation steps. RMMM steps incur additional project cost. Part of risk management, therefore, is to evaluate when the benefits accrued by the RMMM steps are outweighed by the costs associated with implementing them. Project planner performs a classic cost/benefit analysis.	
	d)	**Explain Heuristic method of cost estimation approach.**	**4 M**
	Ans	Heuristic techniques assume that the relationships among the different project parameters can be modeled using suitable mathematical expressions. Once the basic (independent) parameters are known, the other (dependent) parameters can be easily determined by substituting the value of the basic parameters in the mathematical expression. **Different heuristic estimation models can be divided into the following two classes: single variable model and the multi variable model.** **Single variable estimation models** provide a means to estimate the desired characteristics of a problem, using some previously estimated basic (independent) characteristic of the software product such as its size. A single variable estimation model takes the following form: Estimated Parameter = c1 * e1d1 In the above expression, e is the characteristic of the software which has already been estimated (independent variable). Estimated Parameter is the dependent parameter to be estimated. The dependent parameter to be estimated could be effort, project duration, staff size, etc. c1 and d1 are constants. The values of the constants c1 and d1 are usually determined using data collected from past projects (historical data). The basic COCOMO model is an example of single variable cost estimation model. **A multivariable cost estimation model takes the following form:** Estimated Resource = c1 * e1d1 + c 2 * e2 d2 + ... Where e1, e2, … are the basic (independent) characteristics of the software already estimated, and c1, c2, d1, d2, … are constants.	2 M for Single variable estimation model and 2 M for multi variable estimation model
	e)	**Prepare Gantt chart for hostel management system.** **(Five days a week). Consider phases of SDLC.**	**4 M**

	Ans		Correct timeline chart 4 M
5.		**Attempt any TWO of the following:**	**12 M**
	a)	**Draw use-case diagram for ATM system with minimum four use cases and two actors.**	**6 M**
	Ans	Use case diagram for ATM system with minimum four use cases and two actors  Check Balance Deposit funds Withdraw cash Transfer fund Maintenance Repair Customer ATM Technician Bank Manager	Use case diagram, 2 actors : 2 M : 1 M for each Four use cases: 4 M: 1 M each: (total 6 M) Any other relevant use cases and actors shall be given marks.
	b)	**Differentiate between black box testing and white box testing (any six points).**	**6 M**
	Ans	Differences between black box testing and white box testing	Any valid 6

S. No.	Black Box Testing	White Box Testing
1.	It is a way of software testing in which the internal structure or the program or the code is hidden and nothing is known about it.	It is a way of testing the software in which the tester has knowledge about the internal structure or the code or the program of the software.
2.	Implementation of code is not needed for black box testing.	Code implementation is necessary for white box testing.
3.	It is mostly done by software testers.	It is mostly done by software developers.
4.	It can be referred to as outer or external software testing.	It is the inner or the internal software testing.
5.	It is a functional test of the software.	It is a structural test of the software.
6.	This testing can be initiated based on the requirement specifications document.	This type of testing of software is started after a detail design document.
7.	It is applicable to the higher levels of testing of software.	It is generally applicable to the lower levels of software testing.
8.	It is also called closed box testing.	It is also called as clear box testing.
9.	It is least time consuming.	It is most time consuming.

differences between black box testing and white box testing: 6 M, 1 M each) Any more relevant points shall be given marks.

10.	**Types of Black Box Testing:** • Functional Testing • Non-functional testing • Regression Testing	**Types of White Box Testing:** • Path Testing • Loop Testing • Condition testing
11.	It is less exhaustive as compared to white box testing.	It is comparatively more exhaustive than black box testing.

c) **Use COCOMO model to calculate** **6 M**

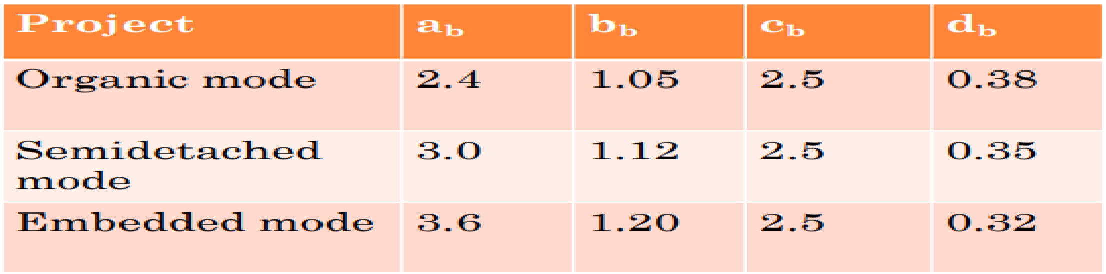
i) Effort

ii) Development time

if estimated size of project is 500 KLOC using organic,

Semi-detached and Embedded mode.

Ans



Project	a_b	b_b	c_b	d_b
Organic mode	2.4	1.05	2.5	0.38
Semidetached mode	3.0	1.12	2.5	0.35
Embedded mode	3.6	1.20	2.5	0.32

Effort ,E = a_b (KLOC)^{b_b} persons-months

Development time D = c_b (E)^{d_b} months

Effort : all 3 modes :3 M : 1 M each

Development time : all 3 modes :3 M : 1 M each = total 6 M

In organic mode :

i) E= 2.4 * (500) $^{1.05}$

= 2.4 * 682.21

=1637.30 person-months

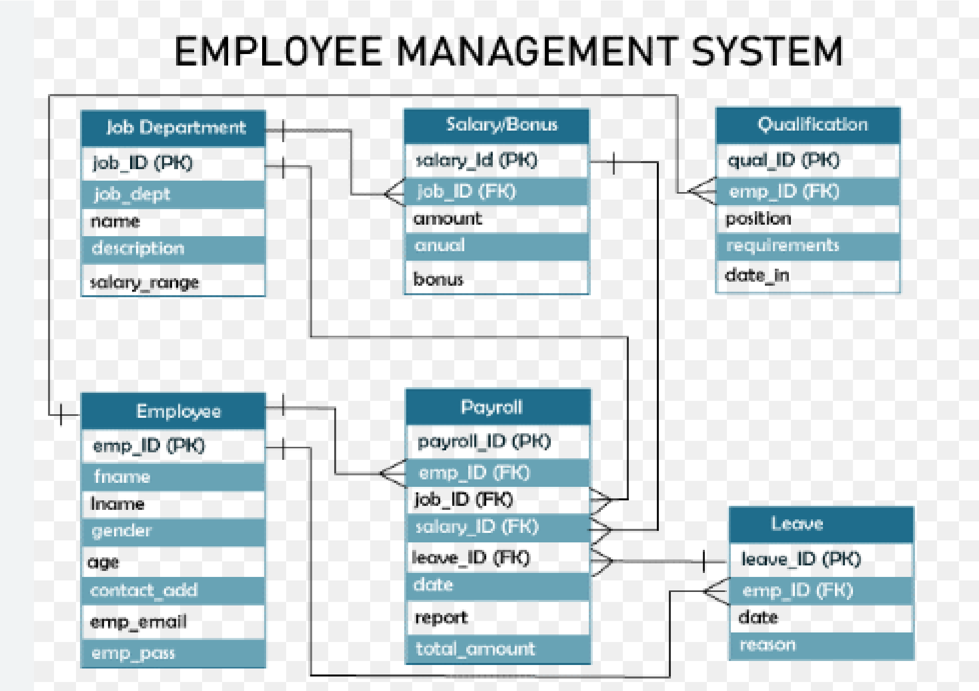
ii) D = 2.5 * (1637.30) $^{0.38}$

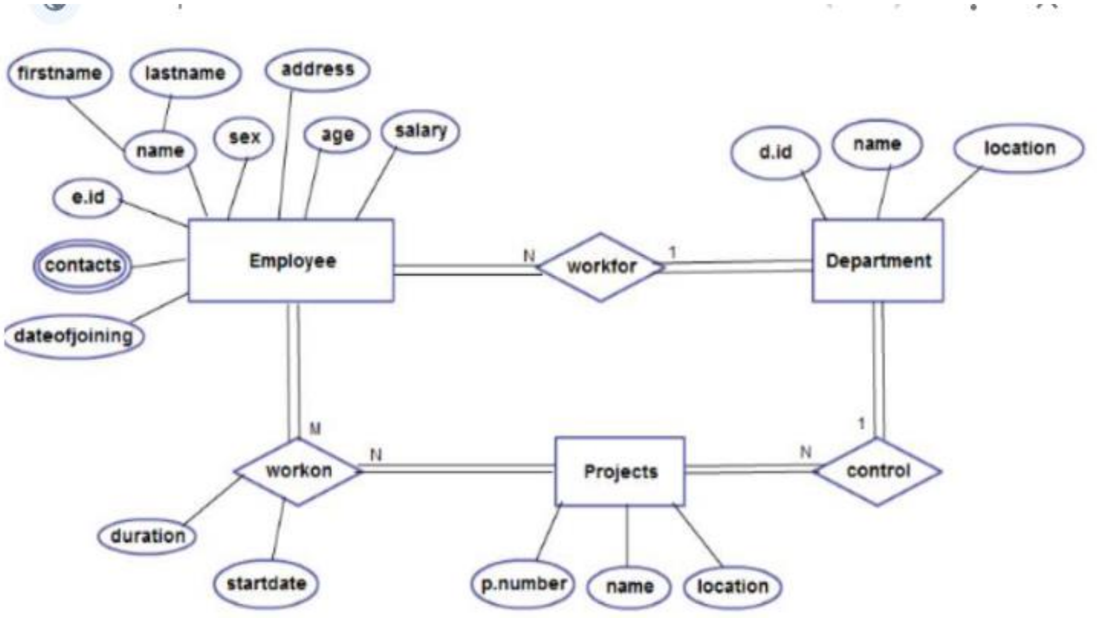
=2.5 *16.64

=41.6 months

In Embedded mode :

i) E= 3.6*(500) $^{1.20}$

		$= 3.6 * 1732.86$ $=6238.29$ Person-Months ii) $D= 2.5 * (6238.29\ )^{0.32}$ $=2.5 * 16.38$ $= 40.95$ Months **In Semidetached mode :** i) $E= 3.0 *(500)^{1.12}$ $= 3.0 * 1054.01$ $=3162.04$ Person-Months ii) $D= 2.5 * (3162.04)^{0.35}$ $=2.5 * 16.78$ $= 41.95$ Months	
6.		**Attempt any <u>TWO</u> of the following:**	**12 M**
	a)	**Draw and explain conceptual data model with E-R diagram for employee management system.**	**6 M**
	Ans	DATA MODEL 	Only Data model done – 3 marks Only ER dia drawn – 3 marks (Number of entities, attributes identified may differ. Consider any relevant terms)

		E-R DIAGRAM  firstname lastname address name sex age salary e.id contacts dateofjoining Employee N workfor 1 d.id name location Department M 1 workon N Projects N control duration startdate p.number name location	
	b)	**Describe six sigma and state the phases of DMAIC and DMADV.**	**6 M**
	Ans	Six sigma : 1. Six sigma is the process of producing high and improved quality output. 2. This can be done in two phases – identification and elimination. The cause of defects is identified and appropriate elimination is done which reduces variation in whole processes. 3. A six sigma method is one in which 99.99966 percentage of all the products to be produced have the same features and are of free from defects. 4. The Characteristics of Six Sigma are as follows: (a) Statistical Quality Control: Standard Deviation in statistics is used for measuring the quality of output. (b) Methodical Approach:-The Six Sigma is not a merely quality improvement strategy in theory, as it features a well-defined systematic approach of application in DMAIC and DMADV which can be used to improve the quality of production. DMAIC is an acronym for Design-Measure- Analyze-Improve-Control. The alternative method DMADV stands for Design-Measure- Analyze-Design-Verify. (c) Fact and Data-Based Approach:-The statistical and methodical as pect of Six Sigma shows the scientific basis of the technique. (d) Project and Objective-Based Focus:- The Six Sigma process is implemented for an organization’s project tailored to its specification and requirements. (e) Customer Focus:- The customer focus is fundamental to the Six Sigma	Description of six sigma : 3 M ; DMAIC phases -1.5 M and DMADV phases -1.5 M

approach. The quality improvement and control standards are based on specific customer requirements.

(f) Teamwork Approach to Quality Management: - The Six Sigma process requires organizations to get organized when it comes to controlling and improving quality.

DMADV PHASES

THE PHASES OF DMADV ARE:

1. **Define:** It covers the process mapping and flow-charting, project charter development, problem-solving tools, and so-called 7-M tools.
2. **Measure:** It includes the principles of measurement, continuous and discrete data, and scales of measurement, an overview of the principle of variations and repeatability and reproducibility (RR) studies for continuous and discrete data.
3. **Analyze:** It covers establishing a process baseline, how to determine process improvement goals, knowledge discovery, including descriptive and exploratory data analysis and data mining tools, the basic principle of Statistical Process Control (SPC), specialized control charts, process capability analysis, correlation and regression analysis, analysis of categorical data, and non-parametric statistical methods.
4. **Improve:** It covers project management, risk assessment, process simulation, and design of experiments (DOE), robust design concepts, and process optimization.
5. **Control:** It covers process control planning, using SPC for operational control and PRE-Control.

DMAIC PHASES ARE:

1. **Define:** It defines the problem or project goal that needs to be addressed.
2. **Measure:** It measures and determines the customer's needs and specifications.
3. **Analyze:** It analyzes the process to meet customer needs.
4. **Design:** It can design a process that will meet customer needs.
5. **Verify:** It can verify the design performance and ability to meet customer needs.

c) **State requirements for given modules of online shopping system.** **6 M**

i) Order module
ii) Accountant module
iii) Categories module

	Ans	This is with perspective of online shopping system. Requirements for following modules will be as **i) Order module :** a. Getting name of Item b. Getting the Item id c. Getting information of item price d. Getting information of quantity of item e. Information of availability of Item **ii) Accountant module** a. Getting the information of list of items purchased b. Bill generation c. Bill calculation d. Getting information of item price e. Generating the bill identification numbers **iii) Categories module** a. Getting information of number of categories b. Information of sub-categories of product c. Getting information on brands of various categories d. Information about the sizes in categories	Requirements of 3 modules : 2 M each; total 6 M (any other relevant requirements shall be given marks)

WINTER – 2022 EXAMINATION

Subject Name: Software Engineering **Model Answer** **Subject Code:** 22413

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 judgement 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.
8) As per the policy decision of Maharashtra State Government, teaching in English/Marathi and Bilingual (English + Marathi) medium is introduced at first year of AICTE diploma Programme from academic year 2021-2022. Hence if the students in first year (first and second semesters) write answers in Marathi or bilingual language (English +Marathi), the Examiner shall consider the same and assess the answer based on matching of concepts with model answer.

Q. No.	Sub Q. N.	Answer	Marking Scheme
1		**Attempt any FIVE of the following:**	**10 M**
	a)	**State the characteristics of Software Engineering.**	**2 M**
	Ans	**Characteristics of software engineering are :** 1. Software is developed or engineered; it is not manufactured in the classical sense. 2. Software doesn't "wear out." 3. Although the industry is moving toward component-based construction, most software continues to be custom built.	**Any 2 characteristics=2 M**
	b)	**Define : i) Software ii) Software Engineering**	**2 M**
	Ans	**Software:** Software is: 1. Instructions (computer programs) that when executed provide desired features, function, and performance; 2. Data structures that enable the programs to adequately manipulate information, and 3. Descriptive information (documents) in both hard copy and virtual forms that describes the operation and use of the programs. **Software Engineering:** Software engineering is the establishment and use of sound engineering principles in order to obtain economically software that is reliable and works efficiently on real machines.	**Software definition: 1 M; Software Engineering definition: 1 M or any other relevant definition shall be given marks**

	c)	**State the characteristics of SRS.**	**2 M**
	Ans	**<u>Characteristics of SRS are :</u>** • Correct • Complete • Unambiguous • Verifiable • Consistent • Ranked for importance and/or stability • Modifiable • Traceable	**any 4 characteristics of SRS : 2 M**
	d)	**List the project cost Estimation Approaches.**	**2 M**
	Ans	**<u>Project cost Estimation Approaches are :</u>** 1. Heuristic Estimation Approach 2. Analytical Estimation Approach 3. Empirical Estimation Approach	**Any two project cost Estimation Approaches : 2 M ; 1 M each**
	e)	**Define risk and list any two types of risk.**	**2 M**
	Ans	**<u>Risk:</u>** A risk is "an uncertain event or condition that, if it occurs, has a positive or negative effect on a project's objectives." **OR** Risk is the uncertainty which is associated with a future event which may or may not occur and a corresponding potential for loss. Types of risks are : 1. Generic risk 2. Product specific risk **OR** 1.Schedule / Time-Related / Delivery Related Planning Risks 2. Budget / Financial Risks 3. Operational / Procedural Risks 4. Technical / Functional / Performance Risks 5. Other Unavoidable Risks	**Risk : definition: 1 M; any two types of risks : 1 M**
	f)	**Define Software Quality Control and Quality Assurance.**	**2 M**
	Ans	**<u>Software Quality Control</u>**: It is a procedure that focuses on fulfilling the quality requested. **<u>Quality Assurance:</u>** It is a procedure that focuses on providing assurance that quality requested will be achieved. Quality assurance consists of the auditing and reporting functions of management.	**Definition of Software Quality Control : 1 M and Quality Assurance : 1 M** **(any other relevant definitions should**

			be given marks)
	g)	**List the phases of Software Quality Assurance.**	**2 M**
	An s	**<u>Phases of Software Quality Assurance are :</u>** • SQA Planning. • Activities. • Review and Audit.	**List of phases of Software Quality Assurance : 2 M**
2.		**Attempt any <u>THREE</u> of the following:**	**12 M**
	a)	**State and describe any four types of Software.**	**4 M**
	An s	**1. System software:**System software is a collection of programs written to service other programs. Some system software (e.g., compilers, editors, and file management utilities) process complex, but determinate, information structures. Other systems applications (e.g., operating system components, drivers, telecommunications processors) process largely indeterminate data. In either case, the system software area is characterized by heavy interaction with computer hardware; heavy usage by multiple users; concurrent operation that requires scheduling, resource sharing, and sophisticated process management; complex data structures; and multiple external interfaces. **2. Real-time software:** Software that monitors/analyzes/controls real-world events as they occur is called real time. Elements of real-time software include a data gathering component that collects and formats information from an external environment, an analysis component that transforms information as required by the application, a control/output component that responds to the external environment, and a monitoring component that coordinates all other components so that real-time response (typically ranging from 1 millisecond to 1 second) can be maintained. **3. 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. Applications in this area restructure existing data in a way that facilitates business operations or management decision making. In addition to conventional data processing application, Business software applications also encompass interactive computing (e.g., point-of-sale	**Stating and describing 4 types of software : 4 M; 1 M each**

transaction processing).

4. 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. However, modern applications within the engineering/scientific area are moving away from conventional numerical algorithms. Computer-aided design, system simulation, and other interactive applications have begun to take on real-time and even system software characteristics.

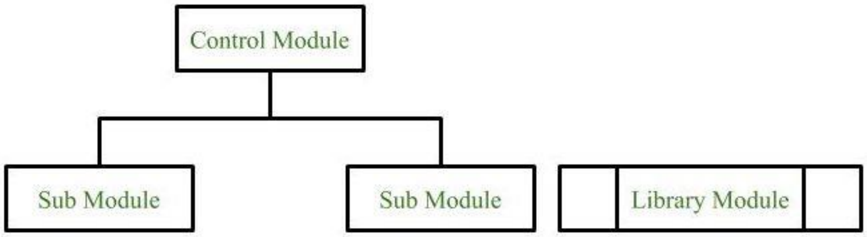
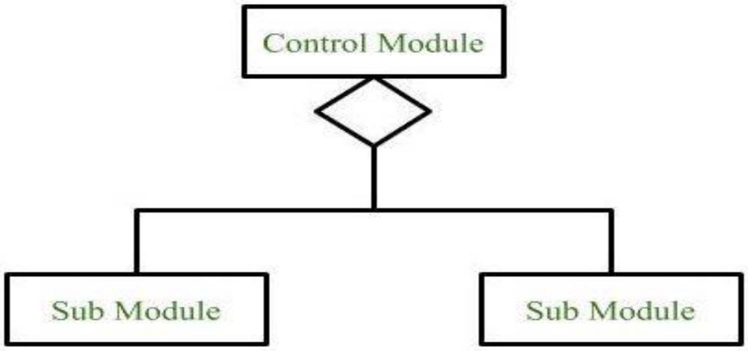
5. Embedded software: Intelligent products have become commonplace in nearly every consumer and industrial market. Embedded software resides in read-only memory and is used to control products and systems for the consumer and industrial markets.
Embedded software can perform very limited and esoteric functions (e.g., keypad control for a microwave oven) or provide significant function and control capability (e.g., digital functions in an automobile such as fuel control, dashboard displays, and braking systems).

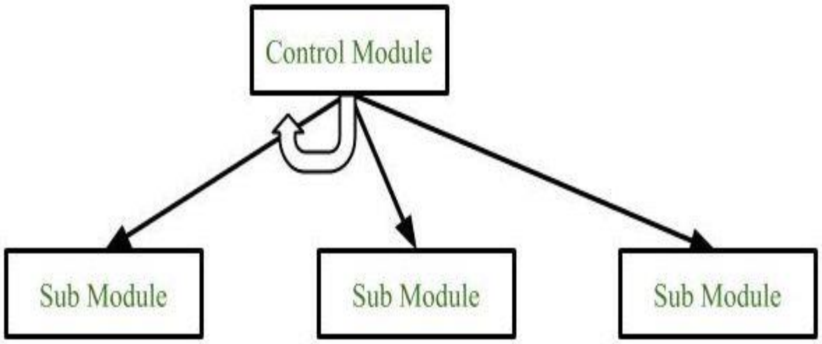
6. Product line or Personal computer software: The personal computer software market has burgeoned over the past two decades. Word processing, spreadsheets, computer graphics, multimedia, entertainment, database management, personal and business financial applications, external network, and database access are only a few of hundreds of applications.

7. 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 (e.g., hypertext and a variety of visual and audio formats). In essence, the network becomes a massive computer providing an almost unlimited software resource that can be accessed by anyone with a modem.

8. 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. Expert systems, also called knowledge-based systems, pattern recognition (image and voice), artificial neural networks, theorem proving, and game playing are representative of applications within this category.

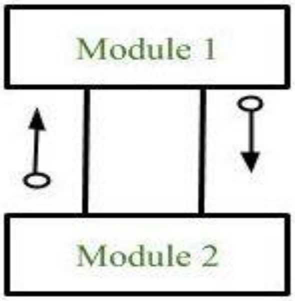
	b)	**Explain structured flowchart with suitable example.**	**4 M**
	An	**Structured flowchart** represents hierarchical structure of modules. It breaks down the entire system into lowest functional modules; describe functions	**Structured flowchart with**

	s	and sub-functions of each module of a system to a greater detail. Structured flowchart partitions the system into black boxes (functionality of the system is known to the users but inner details are unknown). Inputs are given to the black boxes and appropriate outputs are generated. Modules at top level called modules at low level. Components are read from top to bottom and left to right. When a module calls another, it views the called module as black box, passing required parameters and receiving results. **Symbols used in construction of structured chart** 1. **Module** It represents the process or task of the system. It is of three types. • **Control Module** A control module branches to more than one sub module. • **Sub Module** Sub Module is a module which is the part (Child) of another module. • **Library Module** Library Module are reusable and invokable from any module.  Control Module Sub Module Sub Module Library Module 2. **Conditional Call** It represents that control module can select any of the sub module on the basis of some condition.  3. **Loop (Repetitive call of module)** It represents the repetitive execution of module by the sub module. A curved arrow represents loop in the module.	**example : 4 M** **Or** **any other relevant structured flowchart with example give 4M**

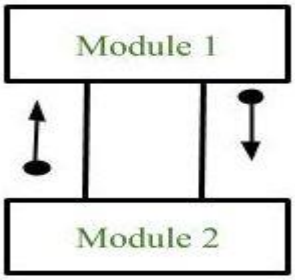


All the sub modules cover by the loop repeat execution of module.

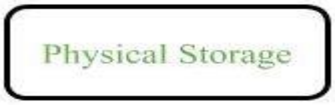
4. **Data Flow**
   It represents the flow of data between the modules. It is represented by directed arrow with empty circle at the end.



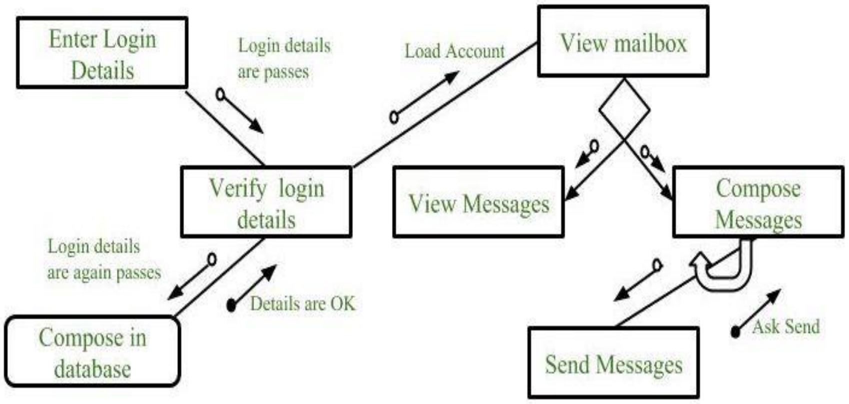
5. **Control Flow**
   It represents the flow of control between the modules. It is represented by directed arrow with filled circle at the end.



6. **Physical Storage**
   Physical Storage is that where all the information are to be stored.



Example : Structure chart for an Email server



c) **Describe 4P's of management spectrum.** **4 M**

Ans

The management spectrum focuses on the four P_s; people, product, process and project.

4 P’s of management spectrum : 4 M; 1 M each

1. **The People:**
   - People of a project includes from manager to developer, from customer to end user .But mainly people of a project highlight the developers.
   - It is so important to have highly skilled and motivated developers that the Software Engineering Institute has developed a People Management Capability Maturity Model (PM-CMM),
   - Organizations that achieve high levels of maturity in the people management area have a higher likelihood of implementing effective software engineering practices.
2. **The Product:**
   - The product is the ultimate goal of the project.
   - This is any types of software product that has to be developed.
   - To develop a software product successfully, all the product objectives and scopes should be established, alternative solutions should be considered, and technical and management constraints should be identified beforehand.
   - Lack of these information, it is impossible to define reasonable and accurate estimation of the cost, an effective assessment of risks, a realistic breakdown of project tasks or a manageable project schedule that provides a meaningful indication of progress.
3. **The 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 project and the requirements of the project team. • Finally, umbrella activities overlay the software process model. • Umbrella activities are independent of any one framework activity and occur throughout the process. 4. **The Project:** • The project is the complete software project that includes requirement analysis, development, delivery, maintenance and updates. • The project manager of a project or sub-project is responsible for managing the people, product and process. • The responsibilities or activities of software project manager would be a long list but that has to be followed to avoid project failure. • A software project could be extremely complex and as per the industry data the failure rate is high. • Its merely due to the development but mostly due to the steps before development and sometimes due to the lack of maintenance.	
	d)	**Describe critical path method with suitable example.**	**4 M**
	Ans	**CPM:-** (a) A critical path in project management is certain tasks that need to be performed in a clear order and for a certain period. (b) If part of one task can be slowed down or postponed for a term without leaving work on others, then such a task is not critical. (c) While tasks with a critical value cannot be delayed during the implementation of the project and are limited in time. (d) Critical Path Method (CPM) is an algorithm for planning, managing and analyzing the timing of a project. (e) The step-by-step CPM system helps to identify critical and non-critical tasks from projects' start to completion and prevents temporary risks. (f) Critical tasks have a zero run-time reserve. If the duration of these tasks changes, the terms of the entire project will be "shifted." That is why critical tasks in project management require special control and timely detection of risks. (g) The method was developed by one of the American companies in 1957. Its employees planned to close, repair and restart chemical plants. (h) The tasks in this project were numerous and complex; that's why they required such a method. (i) After that, Critical Path Method was quickly spread to agricultural and construction projects where people wanted to learn how to avoid routine tasks.	**Critical path method with suitable example : 4 M**

(j) Today, this method of identifying critical tasks is widely used in many industries, including software development.

Example :

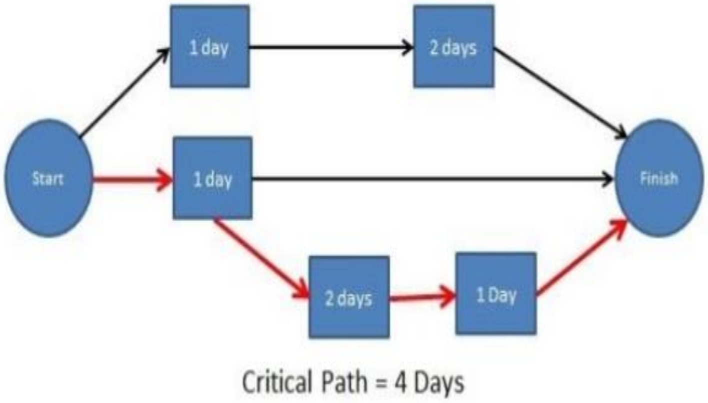


Figure 3: Critical Path Method.

3. **Attempt any <u>THREE</u> of the following:** **12 M**

a) **Distinguish between waterfall model and spiral model.** **4 M**

Ans **Difference Waterfall and Spiral Model:** **Any 4: 1 M each**

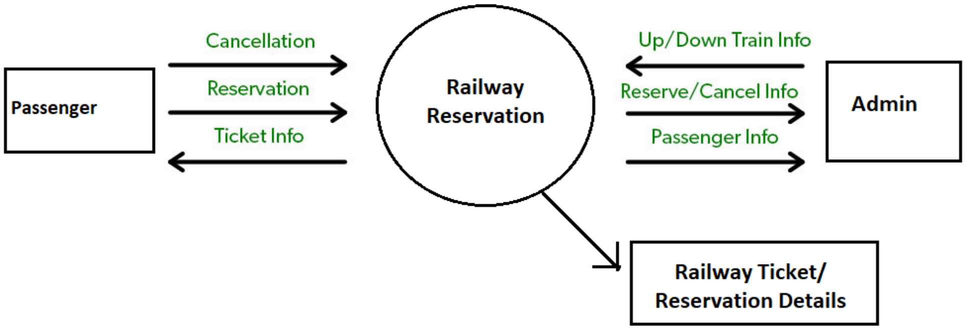
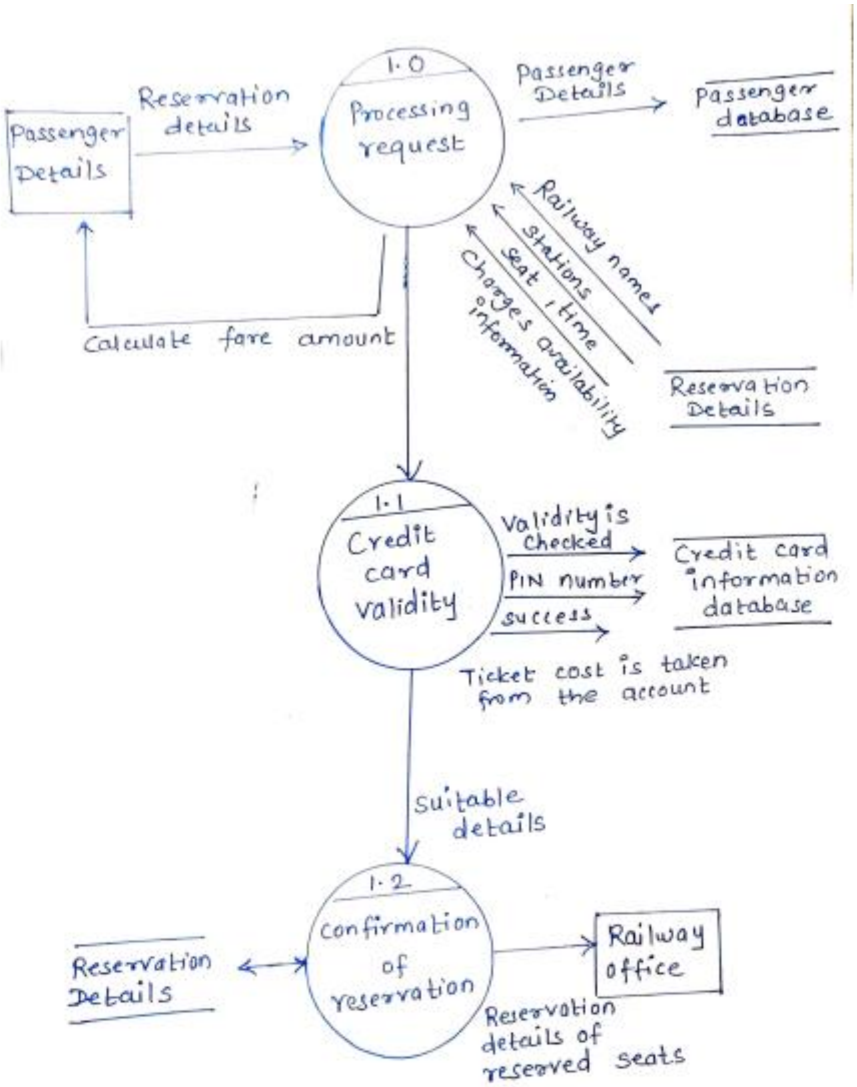
Parameters	Waterfall Model	Spiral Model
Working Nature	Sequential method	Evolutionary method
Involvement of Customer	Minimum, at end of project completion	Maximum, earlier in project development
Identification & rectification of errors	After completion of all stages	Earlier while developing the project
Simplicity	Easy Model	Complex Model
Flow of the Phases	One after another, difficult to go back	In iteration, easy to go back
Project size	Small	Large
Flexibility to change contents	Difficult	Easy
Cost	Less	More
Framework type	Linear	Iterative

b) **Describe any four Software analysis-modeling principles.** **4 M**

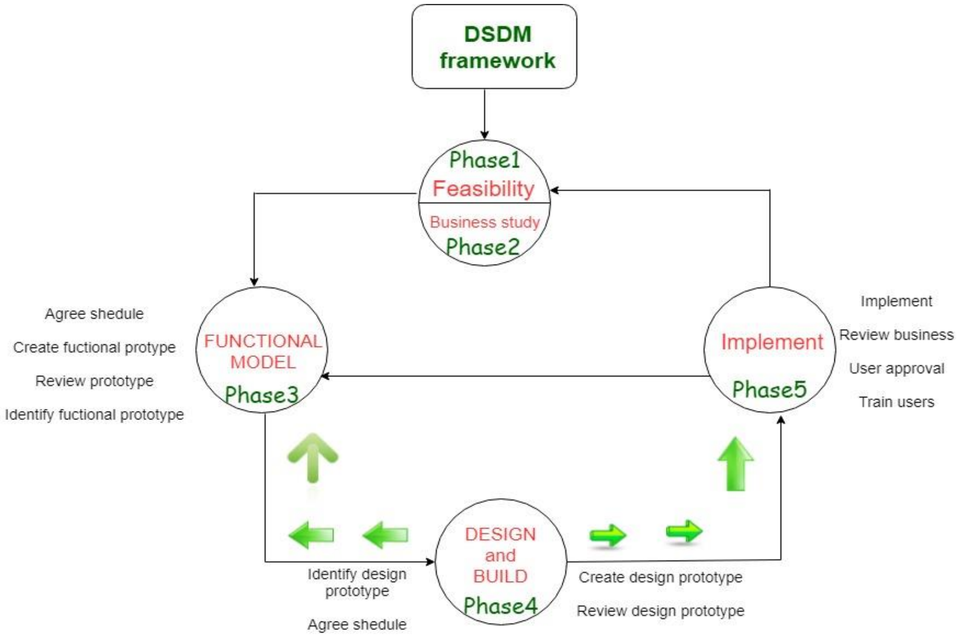
Ans **Four Software analysis modeling principles:** **Any 4: 1 M each**

1. **Principle 1:** The information domain of a problem must be represented and understood.

		The information domain encompasses the data that flow into the system, the data that flow out of the system, and the data stores that collect and organize persistent data objects. **2. Principle 2:** The functions that the software performs must be defined. Software functions provide direct benefit to end users and also provide internal support for those features that are user visible. Some functions transform data that flow into the system. In other cases, functions affect some level of control over internal software processing or external system elements. Functions can be described at many different levels of abstraction, ranging from a general statement of purpose to a detailed description of the processing elements that must be invoked. **3. Principle 3:** The behavior of the software must be represented. The behavior of computer software is driven by its interaction with the external environment. Input provided by end users, control data provided by an external system, or monitoring data collected over a network all cause the software to behave in a specific way. **4. Principle 4:** The models that depict information function and behavior must be partitioned in a manner that uncovers detail in a layered (or hierarchical) fashion. Requirement's modeling is the first step in software engineering problem solving. It allows you to better understand the problem and establishes a basis for the solution. Complex problems are difficult to solve in their entirety. For this reason, you should use a divide and-conquer strategy. A large, complex problem is divided into sub problems until each sub problem is relatively easy to understand. This concept is called partitioning or separation of concerns, and it is a key strategy in requirements modeling. **5. Principle 5:** The analysis task should move from essential information toward implementation detail. Requirements modeling begin by describing the problem from the end-user 's perspective. The essence of the problem is described without any consideration of how a solution will be implemented.	
	c)	**Draw DFD for Railway Reservation Management System for level 0 and level 1.**	**4 M**
	Ans	**Level 0 DFD: railway Reservation System:**	**2 M for Level 0 and 2M for Level 1** **OR** **any other relevant Level o and 1 shall**

		 Passenger Cancellation Reservation Ticket Info Railway Reservation Up/Down Train Info Reserve/Cancel Info Passenger Info Admin Railway Ticket/ Reservation Details **Fig: Level 0 DFD: Railway Reservation System**  Passenger Details Reservation details 1.0 Processing request Passenger Details Passenger database Calculate fare amount Railway names Stations Seat, time Charges availability information Reservation Details 1.1 Credit card validity Validity is checked PIN number success Credit card information database Ticket cost is taken from the account suitable details 1.2 confirmation of reservation Reservation Details Railway office Reservation details of reserved seats **Fig: Level 1 DFD: Railway Reservation System**	**be given marks**

	d)	**Explain line of code metrics for size estimation.**	**4M**
	Ans	Line of code metrics for size estimation: LOC count the total number of lines of source code in a project. **The units of LOC are:** KLOC- Thousand lines of code NLOC- Non-comment lines of code KDSI- Thousands of delivered source instruction 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. **Parameters to count LOC:** 1. count only executable lines. 2. count executable lines plus data definitions. 3. count executable lines, data definitions and comments. 4. count physical lines on input screen. Consider the following example for counting LOC: KCSI: thousands changed source instructions. KSSI: thousands shipped source instructions. First Release of Product Y KCSI = KSSI = 50 KLOC Defects/KCSI = 2.0 Total number of defects = 2.0 × 50 = 100 Second Release, KCSI = 20 KSSI = 50+ 20 (new and changed lines of code) -4 (assuming 20% are changed lines of code) = 66 Defect/KCSI = 1.8 (assuming 10% improvement over the first release). Total number of additional defects = 1.8 x 20 = 36. Third Release, KCSI=30 KSSI 66+30 (new and changed lines of code) -6 (assuming 20% of changed lines of code) = 90. Targeted number of additional defects (no more than previous release) = 36. Defect rate target for the new and changed lines of code: 36/30= 1.2 defects/KCSI or lower.	**Proper explanation=4M**

		Advantages: 1)Universally accepted and is used in many models like COCOMO. 2)Estimation is closer to the developer's perspective. **Disadvantages:** 1)Different programming languages contain a different number of lines. 2)No proper industry standard exists for this technique. 3)It is difficult to estimate the size using this technique in the early stages of the project.	
4.		**Attempt any <u>THREE</u> of the following:**	**12 M**
	a)	**Explain Dynamic Systems Development Method (DSDM).**	**4 M**
	Ans	**Dynamic Systems Development Method (DSDM):**  DSDM framework Phase1 Feasibility Business study Phase2 Agree shedule Create fuctional protype Review prototype Identify fuctional prototype FUNCTIONAL MODEL Phase3 Implement Phase5 Implement Review business User approval Train users DESIGN and BUILD Phase4 Identify design prototype Agree shedule Create design prototype Review design prototype Dynamic Systems Development Method life cycle 1. **Feasibility Study:** It establishes the essential business necessities and constraints related to the applying to be designed then assesses whether or not the application could be a viable candidate for the DSDM method. 2. **Business Study:** It establishes the use and knowledge necessities that may permit the applying to supply business value; additionally, it is the essential application design and identifies the maintainability necessities for the applying. 3. **Functional Model Iteration:** It produces a collection of progressive prototypes that demonstrate practicality	**1 M for diagram** **3M for explanation**

		for the client. (Note: All DSDM prototypes are supposed to evolve into the deliverable application.) The intent throughout this unvarying cycle is to collect further necessities by eliciting feedback from users as they exercise the paradigm. 4. **Design and Build Iteration:** It revisits prototypes designed throughout useful model iteration to make sure that everyone has been designed during a manner that may alter it to supply operational business price for finish users. In some cases, useful model iteration and style and build iteration occur at the same time. 5. **Implementation:** It places the newest code increment (an "operationalized" prototype) into the operational surroundings. It ought to be noted that: **(a)** the increment might not 100% complete or, **(b)** Changes are also requested because the increment is placed into place. In either case, DSDM development work continues by returning to the useful model iteration activity.	
	b)	**State software engineering practices and its importance.**	**4 M**
	Ans	**Software Engineering practices and its importance:** **<u>Software Engineering Practices:</u>** 1. Understand the problem (communication and analysis). 2. Plan a solution (modeling and software design). 3. Carry out the plan (code generation). 4. Examine the result for accuracy (testing and quality assurance). **Understand the problem:** • Who has a stake in the solution to the problem? That is, who are the stakeholders? • What are the unknowns? What data, functions, features, and behavior are required to properly solve the problem? • Can the problem be compartmentalized? Is it possible to represent smaller problems that may be easier to understand? Can the problem be represented graphically? Can an analysis model be created? **Plan the solution:** • Have you seen similar problems before? Are there patterns that are recognizable in a potential solution? Is there existing software that implements the data, functions, features, and behavior that are required? • Has a similar problem been solved? If so, are solutions readily apparent for the sub-problems? • Can you represent a solution in a manner that leads to effective implementation? Can a design model be created? **Carry out the plan:** • Does the solution confirm to the plan? IS source code traceable to the design	**Software Engineering practices=2M** **(any 2 Points)** **and** **Software Engineering importance=2M** **(any 2 Points)**

model?

- Is each component part of the solution probably correct? Have the design and code been received, or better, has correctness proof been applied to the algorithm?

Examine the result:

- Is it possible to test each component part of the solution? Has a reasonable testing strategy been implemented?
- Does the solution produce results that confirm to the data? Functions, features and behavior that are required? Has the software been validated against all stakeholder requirements?

Importance of Software Engineering:

The importance of software engineering lies in the fact that a specific piece of Software is required in almost every industry, every business, and purpose. As time goes on, it becomes more important for the following reasons.

1. Reduces Complexity

Dealing with big Software is very complicated and challenging. Thus to reduce the complications of projects, software engineering has great solutions. It simplifies complex problems and solves those issues one by one.

2. Handling Big Projects

Big projects need lots of patience, planning, and management, which you never get from any company. The company will invest its resources; therefore, it should be completed within the deadline. It is only possible if the company uses software engineering to deal with big projects without problems.

3. To Minimize Software Costs

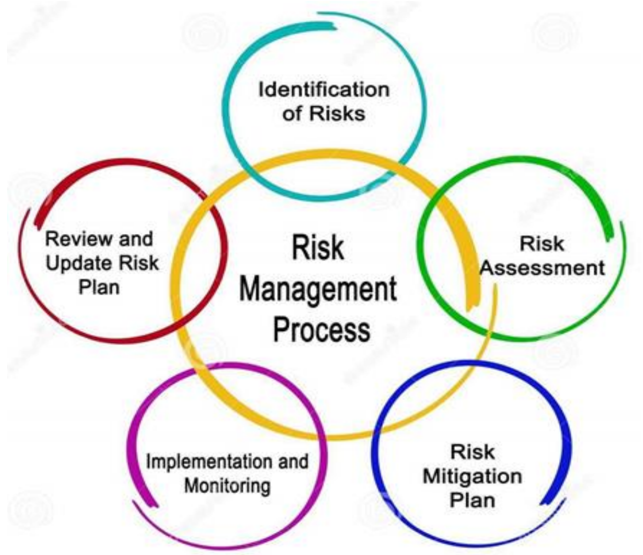
Software engineers are paid highly as Software needs a lot of hard work and workforce development. These are developed with the help of a large number of codes. But programmers in software engineering project all things and reduce the things which are not needed. As a result of the production of Software, costs become less and more affordable for Software that does not use this method.

4. To Decrease Time

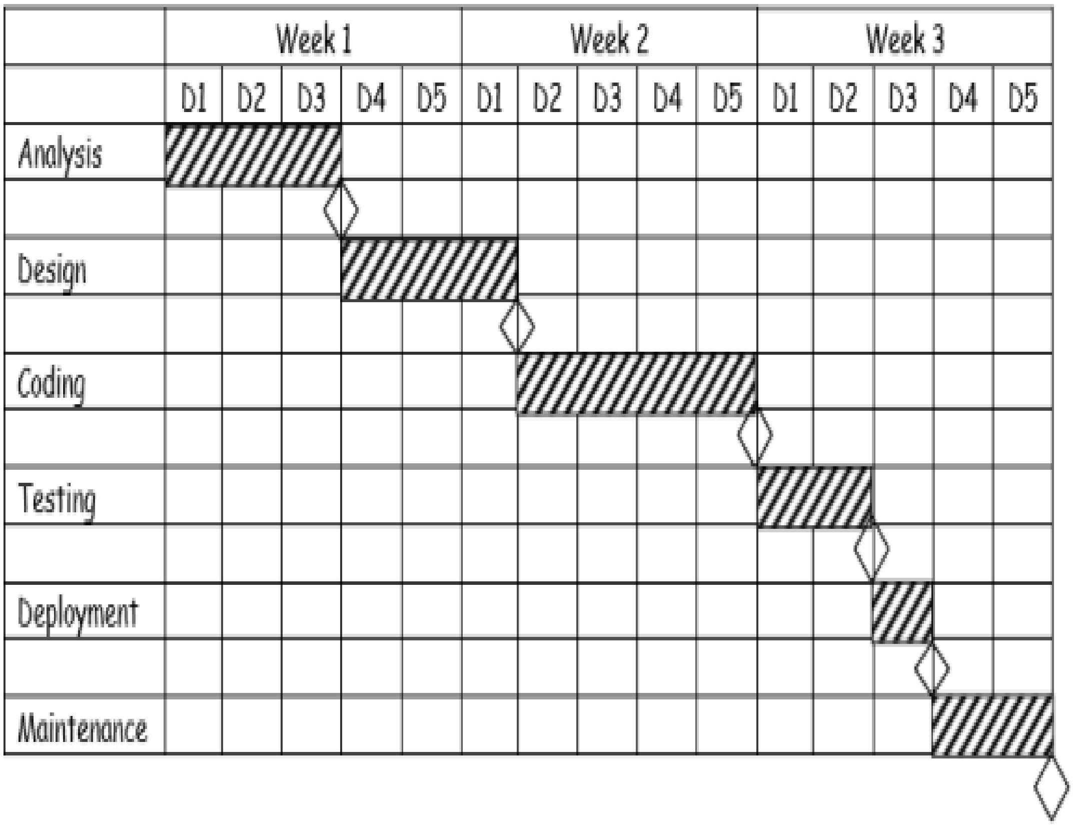
If things are not made according to the procedures, it becomes a huge loss of time. Accordingly, complex Software must run much code to get definitive running code. So it takes lots of time if not handled properly. And if you follow the prescribed software engineering methods, it will save your precious time by decreasing it.

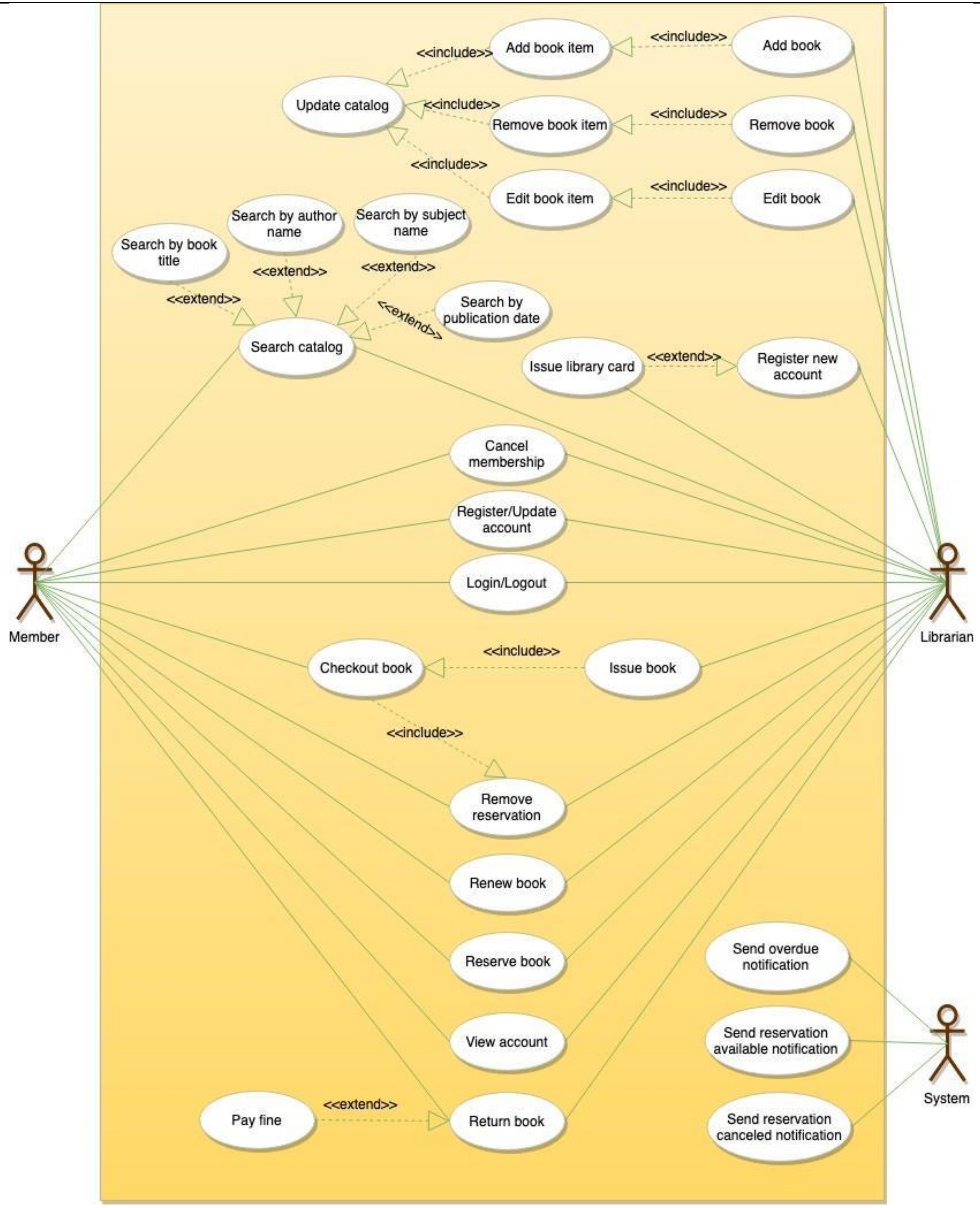
5. Effectiveness

Making standards decides the effectiveness of things. Therefore a company

		always targets the software standard to make it more effective. And Software becomes more effective only with the help of software engineering. **6. Reliable Software** The Software will be reliable if software engineering, testing, and maintenance are given. As a software developer, you must ensure that the Software is secure and will work for the period or subscription you have agreed upon.	
	c)	**State and explain the component of Risk Management.**	**4 M**
	Ans	**Component of Risk Management:** Risk Management is the system of identifying addressing and eliminating these problems before they can damage the project.  Identification of Risks Risk Assessment Risk Mitigation Plan Implementation and Monitoring Review and Update Risk Plan Risk Management Process **Fig: Components of Risk Management** **Components:** **1. Risk Identification** Risk identification is the process of documenting potential risks and then categorizing the actual risks the business faces. When identifying risk, it's also important to not just think about the risks that the business currently faces, but those that might emerge in the future, as well. **2. Risk Analysis** Once risks have been identified, the next step is to analyze their likelihood and potential impact. How exposed is the business to a particular risk? What is the potential cost of a risk becoming a reality? An organization might divide risks into "serious, moderate, or minor" or "high, medium, or low" depending on their potential for disruption.	**Any 4 components=** **4M**

		3. Response Planning Response planning answers the question: What are we going to do about it? For example, if during identification and analysis, you realized that the business is at risk of phishing attacks because its employees are unaware of email security best practices, your response plan might include security awareness training. **4. Risk Mitigation** Risk mitigation is the implementation of your response plan. It is the action your business and its employees take to reduce exposure. **5. Risk Monitoring** Risks are not static; they change over time. Risk monitoring is the process of “keeping an eye” on the situation through regular risk assessments.	
	d)	**Describe following project cost estimation approaches** **i) Heuristic** **ii) Empirical**	**4 M**
	Ans	**Project cost estimation approaches** **1. Empirical Estimation Technique:** Empirical estimation is a technique or model in which empirically derived formulas are used for predicting the data that are a required and essential part of the software project planning step. These techniques are usually based on the data that is collected previously from a project and also based on some guesses, prior experience with the development of similar types of projects, and assumptions. It uses the size of the software to estimate the effort. In this technique, an educated guess of project parameters is made. Hence, these models are based on common sense. However, as there are many activities involved in empirical estimation techniques, this technique is formalized. **2. Heuristic Technique:** Heuristic means “to discover”. The heuristic technique is a technique or model that is used for solving problems, learning, or discovery in the practical methods which are used for achieving immediate goals. These techniques are flexible and simple for taking quick decisions through shortcuts and good enough calculations, most probably when working with complex data. But the decisions that are made using this technique are necessary to be optimal. In this technique, the relationship among different project parameters is expressed using mathematical equations. The popular heuristic technique is given by Constructive Cost Model (COCOMO). This technique is also used to increase or speed up the analysis and investment decisions.	**2M for Heuristic and 2 M for Empirical**

	e)	**Prepare macro time line chart for 15 days of college management system (5 days a week) consider phases of SDLC.**	**4 M**
	Ans	**Time Chart:**  **Fig:Time line chart for 15 days of college management system**	**Correct Time line chart=4M**
5.		**Attempt any <u>TWO</u> of the following:**	**12 M**
	a)	**Sketch use-case diagram for Library management with minimum four use cases and two actors.**	**6 M**
	Ans	**Use-case diagram for Library management.**	**Correct Diagram for any four use cases and Actor =6M**

			
	b)	**Differentiate between white box testing and black box testing.**	**6 M**

Ans

6 M

1M = 1 Point

Sr.no	White box testing	Black Box Testing
1	The tester needs to have the knowledge of internal code or program.	This technique is used to test the software without the knowledge of internal code or program.
2	It aims at testing the structure of the item being tested.	It aims at testing the functionality of the software.
3	It is also called structural testing, clear box testing, code-based testing, or glass box testing.	It also knowns as data- driven, box testing, data and functional testing.
4	Testing is best suited for a lower level of testing like Unit Testing, Integration testing.	This type of testing is ideal for higher levels of testing like System Testing, Acceptance testing.
5	Statement Coverage, Branch coverage, and Path coverage are White Box testing technique.	Equivalence partitioning, Boundary value analysis are Black Box testing technique
6	Can be based on detailed design documents.	Can be based on Requirement specification document.
7	Example: By input to check and verify loops	Example: Search something on google by using keywords

c) **Describe a Cocomo and Cocomo-lI models.** **6 M**

Ans

<u>COCOMO Model :</u>

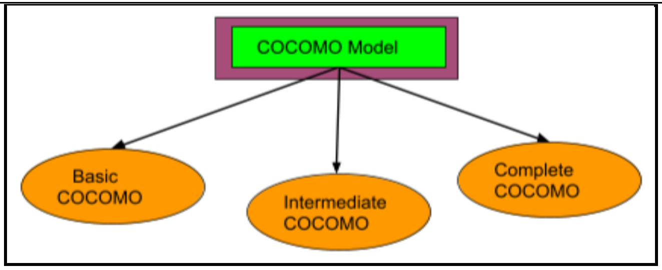
- COCOMO is one of the most widely used software estimation models in the world.
- This model is developed in 1981 by Barry Boehm to give estimation of number of man-months it will take to develop a software product.
- COCOMO predicts the efforts and schedule of software product based on size of software.
- COCOMO has three different models that reflect complexity

1. Basic Model
2. Intermediate Model
3. Complete Model

3 M for Explanation of COCOMO Model

and

3 M for Explanation of COCOMO Model-II



1) Basic COCOMO: The basic COCOMO is employed for rough calculations, limiting software estimation precision. This is because the model only considers lines of source code and constant values derived from software project types rather than other elements that significantly impact the software development process.

2) Intermediate COCOMO: The Intermediate COCOMO model expands the Basic COCOMO model that takes into account a collection of cost drivers to improve the cost estimating model's accuracy.

3) Complete/Detailed COCOMO: The model contains all qualities of both Basic COCOMO and Intermediate COCOMO techniques for each software engineering process. The model considers each project's development phase (analysis, design, and so on).

Estimation of Effort: Calculations –

Basic Model gives an approximate estimation of the project parameter.
The Basic COCOMO Estimation model given by following Expression ,

$$E= a(\mathrm{KLOC})^b$$

The Cocomo model divides software projects into 3 types-

1. Organic Project

It belongs to small & simple software projects which are handled by a small team with good domain knowledge and few rigid requirements.
Example: Small data processing or Inventory management system.

2. Semidetached Project

It is an intermediate (in terms of size and complexity) project, where the team having mixed experience (both experience & inexperience resources) to deals with rigid/nonrigid requirements.
Example: Database design or OS development.

3. Embedded Project

This project having a high level of complexity with a large team size by considering all sets of parameters (software, hardware and operational).
Example: Banking software or Traffic light control software.

<u>COCOMO II Model :</u>

COCOMO-II is the revised version of the original Cocomo (Constructive Cost Model) and is developed at University of Southern California. It is the model that allows one to estimate the cost, effort and schedule when planning a new software development activity.

End User Programming	Application Generators and composition aids	Infrastructure
	Application Composition	
	System Integration	

- The Application Composition Model

  This model involves prototyping efforts to resolve potential high- risk issues such as user interfaces, software/system interaction, performance, or technology maturity. The costs of this type of effort are best estimated by the Applications Composition model. It is suitable for projects built with modern GUI-builder tools. It is based on new Object Points.

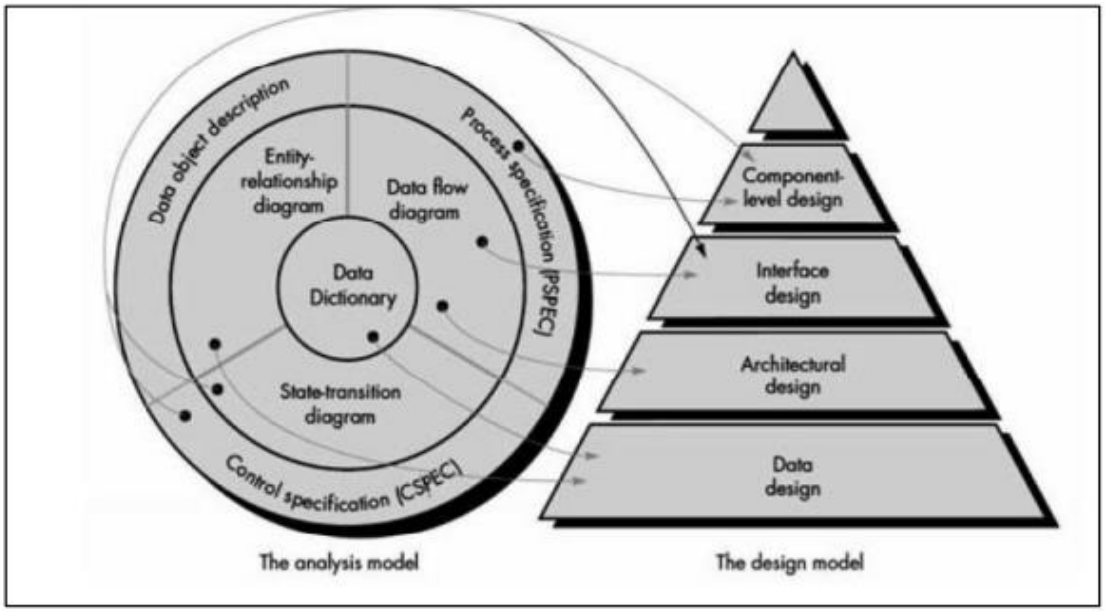
- The Early Design Model

  The Early Design model involves exploration of alternative software/system architectures and concepts of operation. It uses a small set of new Cost Drivers, and new estimating equations. Based on Unadjusted Function Points or KSLOC.

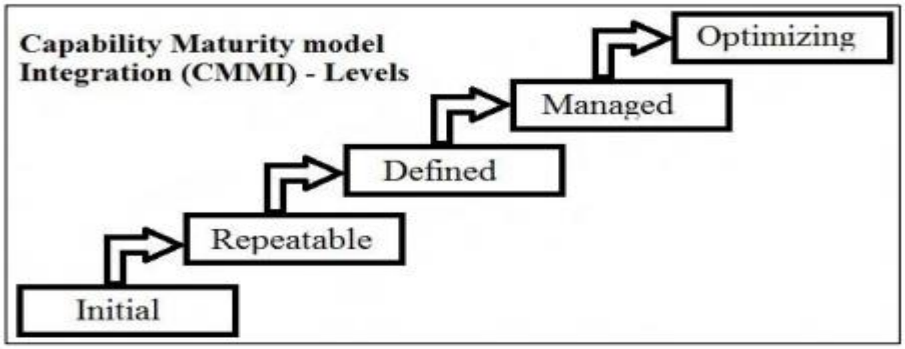
- The Post-Architecture Model

  The Post-Architecture model involves the actual development and maintenance of a software product Estimates. In COCOMO II effort is expressed as Person Months (PM). The inputs are the Size of software development, a constant, A, and a scale factor, B. The size is in units of thousands of source lines of code (KSLOC). The constant, A, is used to capture the multiplicative effects on effort with projects of increasing size.

6.		**Attempt any <u>TWO</u> of the following:**	**12 M**
	a)	**Recognize requirement for following modules of hospital management software** **i) Customer module** **ii) Administrator module** **iii) Account module**	**6 M**
	An	Requirement for following modules of hospital management software:	**6M=2M for each**

	s	**i) Customer module** a) Services provided in hospital b) Facility to register patients and view their report and history. c) Avaibility of beds and wards etc. d) Showing Dr qualification, avaibility for OPD. **ii) Administrator module** a)Administrator can view as well as alter ant information of the Hospital Management System b) Authority to all purchase and employee Management. c) Updating Availability beds and wards. d) Add patients details and assigning ID. **iii) Account module** a) Details about patients Health Insurance. b) Bill Generation. c) Develop maintains and analyses budgets, preparing periodic reports that compare budgeted costs to actual costs. d) Oversee the Hospitals Billing Department.	**Module**
	b)	**Draw neat labelled diagram of translation of requirement model in to design model and explain it with details.**	**6 M**
	Ans	**Translation of Requirement model into design model :**  Data object description Entity-relationship diagram Data flow diagram Process specification (PSPEC) Data Dictionary State-transition diagram Control specification (CSPEC) The analysis model Component-level design Interface design Architectural design Data design The design model Software requirements, manifested by the data, functional, and behavioral models, feed the design task. Using one of a number of design methods, the design task produces a data design, an architectural design, an interface design, and a component design. Each of the elements of the analysis model provides information that is necessary to create the four design models required for a complete specification of design.	**2M : Diagram** **4M : Explanation**

		Design is a meaningful engineering representation of something that is to be built. It can be traced to a customer's requirements and at the same time assessed for quality against a set of predefined criteria for —good‖ design. In the software engineering context, design focuses on four major areas of concern: data, architecture, interfaces, and components Design begins with the requirements model. The data design transforms the information domain model created during analysis into the data structures that will be required to implement the software. The data objects and relationships defined in the entity relationship diagram and the detailed data content depicted in the data dictionary provide the basis for the data design activity. Part of data design may occur in conjunction with the design of software architecture. More detailed data design occurs as each software component is designed. The architectural design defines the relationship between major structural elements of the software, the design pattern that can be used to achieve the requirements that have been defined for the system, and the constraints that affect the way in which architectural design patterns can be applied. The architectural design representation the framework of a computer based system can be derived from the system specification, the analysis model, and the interaction of subsystems defined within the analysis model. The interface design describes how the software communicates within itself, with systems that interoperate with it, and with humans who use it. An interface implies a flow of information (e.g., data and/or control) and a specific type of behavior. Therefore, data and control flow diagrams provide much of the information required for interface design. The component-level design transforms structural elements of the software architecture into a procedural description of software components. Information obtained from the PSPEC, CSPEC, and STD serve as the basis for component design.	
	c)	**Explain CMMI Techniques with its level.**	**6 M**
	Ans	**CMMI Techniques :** The Capability Maturity Model Integration (CMMI), a comprehensive process meta-model that is predicated on a set of system and software engineering capabilities that should be present as organizations reach different levels of process capability and maturity. The CMMI represents a process meta-model in two different ways: (1) Continuous model and (2) Staged model. The continuous CMMI meta-model describes a process in two dimensions. Each process area (e.g. project planning or requirements management) is formally assessed against specific goals and practices and is rated according to the following capability levels.	**1M : diagram , 5M : Any 5 Point**

		 Capability Maturity model Integration (CMMI) - Levels Optimizing Managed Defined Repeatable Initial **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 2 **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 3 **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.	

MAHARASHTRA STATE BOARD OF TECHNICAL EDUCATION
(Autonomous)
(ISO/IEC - 27001 - 2005 Certified)

SUMMER – 2022 EXAMINATION
MODEL ANSWER

Subject: Software Engineering **Subject Code: 22413**

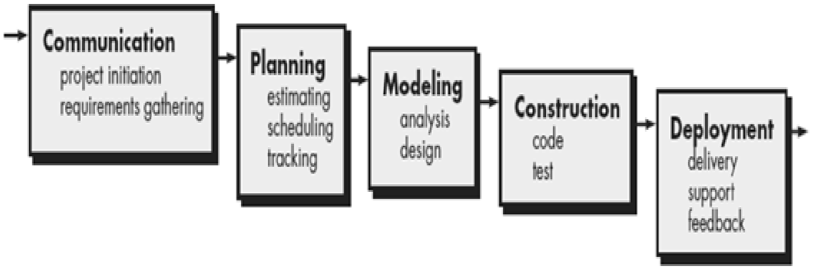
Important Instructions to examiners:

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 judgement 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.
8) As per the policy decision of Maharashtra State Government, teaching in English/Marathi and Bilingual (English + Marathi) medium is introduced at first year of AICTE diploma Programme from academic year 2021-2022. Hence if the students in first year (first and second semesters) write answers in Marathi or bilingual language (English +Marathi), the Examiner shall consider the same and assess the answer based on matching of concepts with model answer.

Q. No	Sub Q.N.	Answer	Marking Scheme
1.	 **a)** **Ans.**	**Attempt any FIVE of the following:** **List any four types of software** • System software • Application Software • Scientific software • Embedded software • Product line software • Web application • Artificial Intelligence	**10** **2M** ***1/2M each, any four types***
	b) **Ans.**	**List any four planning principles** 1.Understanding the scope of the project 2. Involve stakeholders in the planning activity 3.Planning is iterative 4.Planning should be based on the information available 5. Consider the risk as the plan is defined	**2M** ***1/2M each, any four principles***

		6. Being realistic 7. Adjust granularity as the plan is defined 8. Define how to ensure quality 9. Describe how to accommodate change 10. Track and monitor the plan frequently and make adjustments if required	
	c) Ans.	**Describe following design concepts** **i) Abstraction** **ii) Information hiding** **Abstraction** Abstraction is hiding the internal implementation and highlight the set of services. It is achieved by using the abstract class and interfaces and further implementing the same. **Information Hiding** It is the principle of segregation of the design decisions in a computer program that are most likely to change, thus protecting other parts of the program from extensive modification if the design decision is changed.	**2M** ***1M for each design concept***
	d) Ans.	**List 4P's of Management spectrum** People Product Process Project	**2M** ***1/2M each***
	e) Ans.	**Define Quality control and Quality Assurance** **Quality Control**: Software quality control is the set of procedures used by organizations to ensure that a software product meets its quality goals at the best value to the customer, and to continually improve the organization's ability to produce software products in the future **Quality Assurance**: Conformance to explicit stated functional and performance requirements, explicitly documented. It is also the development of standards and implicit characteristic that are expected of all professionally developed software.	**2M** ***1M for each definition***

	f) **Ans.**	**List any four selection criteria for Software Process Model** Following are the parameters which is used to select 1. Requirements Characteristics • Reliability of Requirements • How often the requirements can change • Types of requirements • Number of requirements • Can the requirements be defined at an early stage • Requirements indicate the complexity of the system 2. Development team : • Team size • Experience of developers on similar type of projects • Level of understanding of user requirements by the developers • Environment • Domain knowledge of developers • Experience on technologies to be used • Availability of training 3. User involvement in the project : • Expertise of user in project • Involvement of user in all phases of the project • Experience of user in similar project in the past 4. Project type and associated risk : • Stability of funds • Tightness of project schedule • Availability of resources • Type of project • Size of the project • Expected duration for the completion of project • Complexity of the project • Level and the type of associated risk	**2M** ***1/2M each, any four criterias***
	g) **Ans.**	**Define Project Cost Estimation.** Software cost estimation is the process of predicting the effort required to develop a software system. Project cost estimating is the process of predicting the total cost of the tasks, time, and resources required to deliver a project's scope of work. There are three approaches of project estimation, they are: i) Heuristic ii) Analytical iii) Empirical	**2M** ***2M for correct definition***

2.		**Attempt any <u>THREE</u> of the following:**	**12**
	a)	**Explain Waterfall Model with neat labeled diagram.**	**4M**
	Ans.	The Waterfall Model: The waterfall model is a traditional method, sometimes called the classic life cycle. 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.  Communication project initiation requirements gathering Planning estimating scheduling tracking Modeling analysis design Construction code test Deployment delivery support feedback 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. 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. 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.	***2M for diagram 2M for explanation***

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.

Among the problems that are sometimes encountered when the waterfall model is applied are

1. Real projects rarely follow the sequential flow that the model proposes. Although the linear model can accommodate iteration, it does so directly. As a result, changes can cause confusion as the project team proceeds.
2. It is often difficult for the customer to state all requirements explicitly. The Waterfall Model requires this and has difficulty accommodating the natural uncertainty that exists at the beginning of many projects.
3. The customer must have patience. A working version of the program will not be available until late in the project time-span. A major blunder, if undetected until the working program is received, can be disastrous.

The waterfall model is often inappropriate for such work. However, it can serve as a useful process model in situations where requirements are fixed and work is to proceed to completion in a linear manner.

	b) **Ans.**	**State and describe any four core principles.** The core principles are 1. The reason it all exists: The software system exists in the organization for providing value to its users with, the availability of hardware and software requirements. Hence all the decisions should be made by keeping this in mind. 2. Keep it Simple, Stupid (KISS) Software design is not a haphazard process. There are many factors considered in the design effort. The design should be straight forward and as simple as possible. This facilitates having a system which can be easily understood and easy to maintain. Simple doesn't mean quick and dirty. In fact, it requires lot of thought and effort to simplify multiple iterations of a complex task. This results in the advantage that the software is less error prone and easily maintainable. 3. Maintain the vision A clear vision is essential for the success of a software project. If the vision is missing, the project may end up of two or more minds. The team leader has a critical role to play for maintaining the vision and enforce compliance with the help of the team members. 4. What you produce, others will consume The design and implementation should be done by keeping in mind the user's requirements. The code should permit the system extension. Some other programmers debugging the code should not have any errors and satisfying all the user needs. 5. Be open to future The system with the long lifetime has more value. The industry standard software systems induce for longer. The system should be ready to accept and adapt to new changes. The systems which are designed by keeping in mind the future needs will be more successful and acceptable to the users. 6. Plan ahead for reuse Reuse saves time and efforts. The reuse of code and design is one of the advantages of object oriented technologies. The reuse of parts of the code helps in reducing the cost and time evolved, in the new software development. 7. Think Placing clear and complete thought before action almost always produces better results. With proper thinking, we are most likely to do	**4M** ***2M for stating 2M for description***

		it right. We also gain knowledge about how to do it right again. It becomes a valuable experience, even if something goes wrong, as there was adequate thought process. Hence when clear thought has gone into the system, value comes out, this provides potential rewards.	
	c) **Ans.**	**Explain Test Documentation with the help of following terms** **i) Test Case** **ii) Test Data** **iii) Test Plan** **Test Documentation** Test documentation is documentation of artifacts created before or during the testing of software. It helps the testing team to estimate testing effort needed, test coverage, resource tracking, execution progress, etc. It is a complete suite of documents that allows you to describe and document test planning, test design, test execution, test results that are drawn from the testing activity **Test Case** It is a detailed document that describes step by step procedure to test an application. It consists of the complete navigation steps and inputs and all the scenarios that need to be tested for the application. We will write the test case to maintain the consistency, or every tester will follow the same approach for organizing the test document. It is a document that is prepared by the managers or test lead. **Test Data** Data created or selected to satisfy the execution preconditions and inputs to execute one or more test cases **Test Plan** It consists of all information about the testing activities. The test plan consists of multiple components such as Objectives, Scope, Approach, Test Environments, Test methodology, Template, Role & Responsibility, Effort estimation, Entry and Exit criteria, Schedule, Tools, Defect tracking, Test Deliverable, Assumption, Risk, and Mitigation Plan or Contingency Plan.	**4M** ***1M for each***
	d) **Ans.**	**Explain CMMI in detail with neat diagram** The Capability Maturity Model Integration (CMMI), a comprehensive process meta-model that is predicated on a set of system and software engineering capabilities that should be present as organizations reach different levels of process capability and	**4M** ***3M for explanation***

maturity.

The CMMI represents a process meta-model in two different ways:

(1) As a "continuous" model and (2) as a "staged" model.

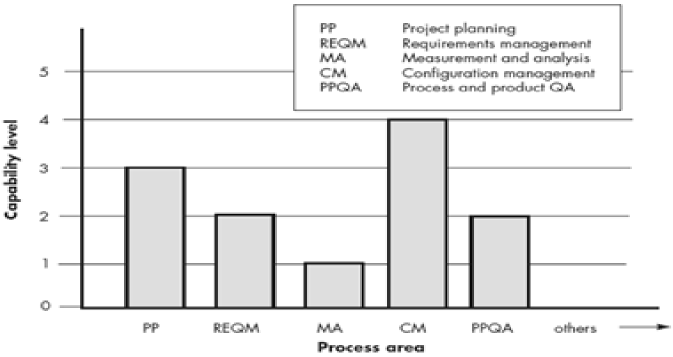
The continuous CMMI meta- model describes a process in two dimensions. Each process area (e.g., project planning or requirements management) is formally assessed against specific goals and practices and is rated according to the following capability levels:

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.

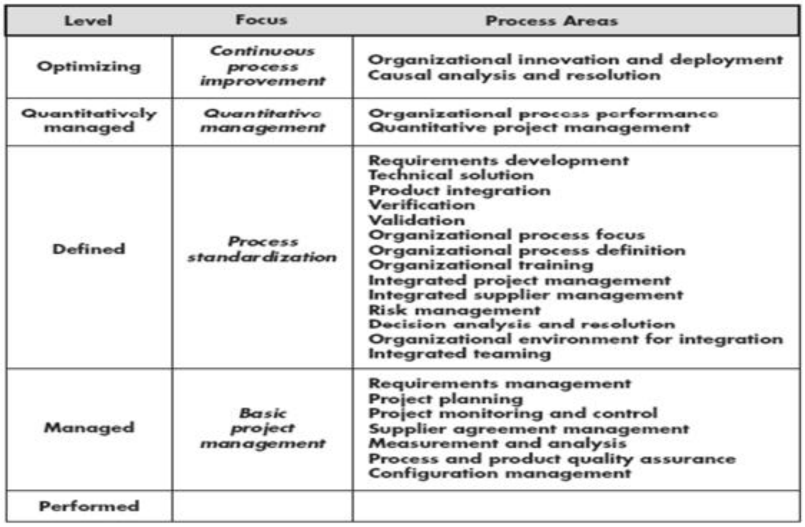
Level 1: Performed—all of the specific goals of the process area (as defined by the CMMI) have been satisfied. Work tasks required to produce defined work products are being conducted.

CMMI Process Area Capability Profile.

1M for diagram



OR



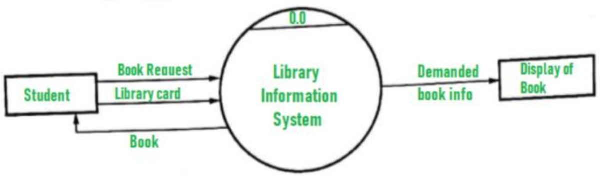
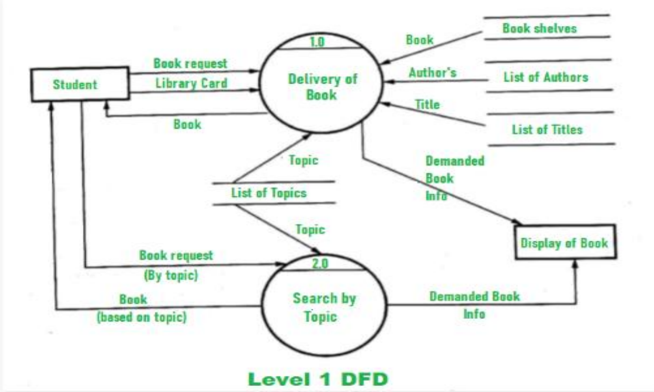
Level	Focus	Process Areas
Optimizing	*Continuous process improvement*	Organizational innovation and deployment Causal analysis and resolution
Quantitatively managed	*Quantitative management*	Organizational process performance Quantitative project management
Defined	*Process standardization*	Requirements development Technical solution Product integration Verification Validation Organizational process focus Organizational process definition Organizational training Integrated project management Integrated supplier management Risk management Decision analysis and resolution Organizational environment for integration Integrated teaming
Managed	*Basic project management*	Requirements management Project planning Project monitoring and control Supplier agreement management Measurement and analysis Process and product quality assurance Configuration management
Performed		

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.	
3.		**Attempt any THREE of the following:**	**12**
	a)	**State and describe any four deployment principles**	**4M**
	Ans.	**Principle 1: Manage customer's expectations.** It always happens that customer wants more than he has stated 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. **Principle 2: Assembly and test complete delivery package.** It is not the case that the deliverable package is ‗only software‘. The customer must get all supporting and essential help from developer‘s side. **Principle 3: Record-keeping mechanism must be established for customer support.** Customer support is important factor in deployment phase. If proper support is not provided, customer will not be satisfied. Hence support should be well planned and with record-keeping mechanism. **Principle 4: Provide essential instructions, documentations and manual**. Many times, developer thinks ―when project is successful deliverable part is only working program‖. But realty is that working program is just part of software product. Actual project delivery includes all documentations, help files and guidance for handling the software by user. **Principle 5: Don't deliver any defective or buggy software to the customer.** 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.	***1M for each principle***

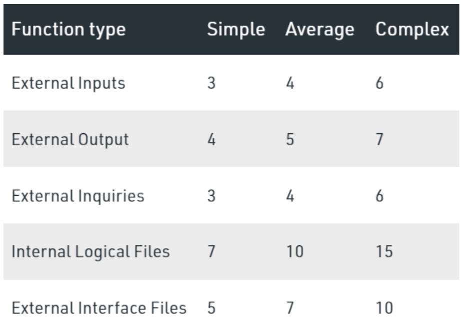
	b) **Ans.**	**Draw DFD 0 and DFD 1 diagram for Library Management System.**  0.0 Library Information System Student Book Request Library card Book Demanded book info Display of Book  1.0 Delivery of Book Student Book request Library Card Book Book Book shelves Author's List of Authors Title List of Titles Topic List of Topics Topic Demanded Book Info Display of Book Book request (By topic) Book (based on topic) 2.0 Search by Topic Demanded Book Info Level 1 DFD	**4M** ***2M for DFD 0*** ***2M for DFD 1***
	c) **Ans.**	**State and describe two metrics of project size estimation** **Metrics for project Size Estimation** 1.Line of Code 2. Function Point **Lines of Code (LOC)** LOC is the simplest among all metrics available to estimate project size. This metric is very popular because it is the simplest to use. Using this metric, the project size is estimated by counting the number of source instructions in the developed program while counting the number of source instructions, lines used for commenting the code and the header lines should be ignored. Estimation is dependent on programming language. For different programming language lines of code will vary. **Function Point metric** In this method, the number and type of function supported by the software are utilized to find FPC (Function point count).	**4M** ***2M for each metric***

The steps in function point analysis are:

- Count the number of functions of each proposed type
- Compute the unadjusted function point (UFP)
- Find total degree of influence (TDI)
- Compute value adjustment factor (VAF)
- Find the function point count (FPC)

Count the number of functions of each proposed type:
Functions belonging to the following types:
External Input: Functions related to data entering the system.
External Outputs: Functions related to data existing from the system.
External Enquires: They lead to data retrieval from the system.
Internal Files: Logical files maintained within the system.
External interface files: These are logical files of other application used by our application.

Compute the unadjusted function point (UFP)
Categories each of the function types like simple, average or complex based on their complexity. Multiply the count of each function type with its weighing factor and find the weighted sum.

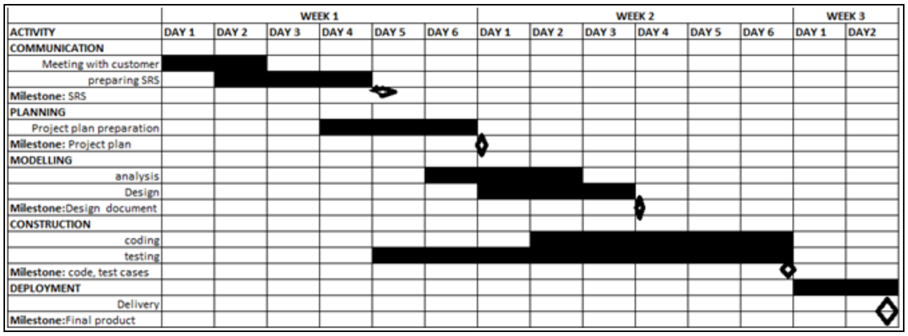
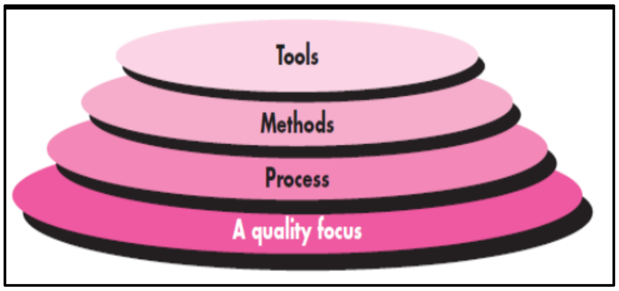


Function type	Simple	Average	Complex
External Inputs	3	4	6
External Output	4	5	7
External Inquiries	3	4	6
Internal Logical Files	7	10	15
External Interface Files	5	7	10

Find total degree of influence (TDI)
Use the 14 general characteristics of system to find the degree of influence of each of them. The sum of all 14 degree of influence will give TDI. The range of TDI is 0 to 70.

Compute value adjustment factor (VAF)
VAF=(TDI*0.01)+0.65

Find the function point count (FPC)
FPC=UFP*VAF

	d) **Ans.**	**Prepare Macro Timeline chart for 20 days of Hotel Management system (6 days a week) consider broad phase of SDLC.**  	4M *4M for correct timeline chart*
4.	 **a)** **Ans.**	**Attempt any <u>THREE</u> of the following:** **Draw and explain Software Engineering Layered technology approach.** Software engineering is a layered technology. The layers of software engineering as shown in the below 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.	12 4M *2M for diagram* *2M for explanation*

		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 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.	
	b) **Ans.**	**State the need of SRS and also enlist the characteristics**. The need of SRS document is to provide • A detailed overview of software product, its parameters and goals. • The description regarding the project's target audience and its user interface hardware and software requirements. • How client, team and audience see the product and its functionality. **Characteristics of SRS:** • Correctness • Completeness • Consistency	**4M** ***2M for enlisting*** ***2M for characteristics***

- Unambiguousness
- Modifiability
- Traceability
- Testability
- Understandable by stakeholder

c) **Distinguish between Black Box and White Box testing. (Write any four points)** **4M**

Ans.

1M for each valid point

White box testing	**Black Box Testing**
The tester needs to have the knowledge of internal code or program.	This technique is used to test the software without the knowledge of internal code or program
It aims at testing the structure of the item being tested.	It aims at testing the functionality of the software
It is also called structural testing, clear box testing, code-based testing, or glass box testing.	It also known as data driven, closed box testing, data-, and functional testing.
Testing is best suited for a lower level of testing like Unit Testing, Integration testing.	This type of testing is ideal for higher levels of testing like System Testing, Acceptance testing.
Statement Coverage, Branch coverage, and Path coverage are White Box testing technique.	Equivalence partitioning, Boundary value analysis are Black Box testing technique
Can be based on detailed design documents.	Can be based on Requirement specification document.

d) **Explain RMMM strategy.** **4M**

Ans. Risk mitigation, monitoring, and management (RMMM) plan. A risk management strategy can be included in the software project plan or the risk management steps can be organized into a separate **Risk Mitigation, Monitoring and Management Plan.**

4M for correct explanation

The RMMM plan documents all work performed as part of risk analysis and is used by the project manager as part of the overall project plan.

Once RMMM has been documented and the project has begun, risk mitigation and monitoring steps commence.

Risk mitigation is a problem avoidance activity.

Risk monitoring is a project tracking activity with three primary objectives:

1) To assess whether predicted risks do, in fact, occur;
2) To ensure that risk aversion steps defined for the risk are being properly applied; and
3) To collect information that can be used for future risk analysis.

In many cases, the problems that occur during a project can be traced to more than one risk. Another job of risk monitoring is to attempt to allocate origin (what risk(s) caused which problems throughout the project).

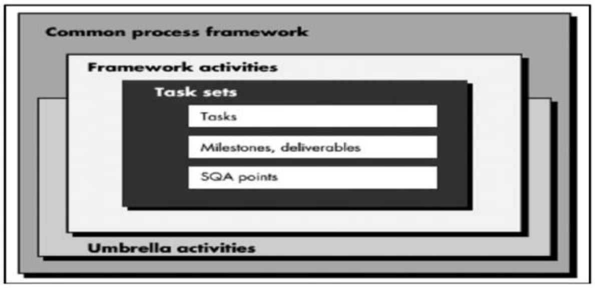
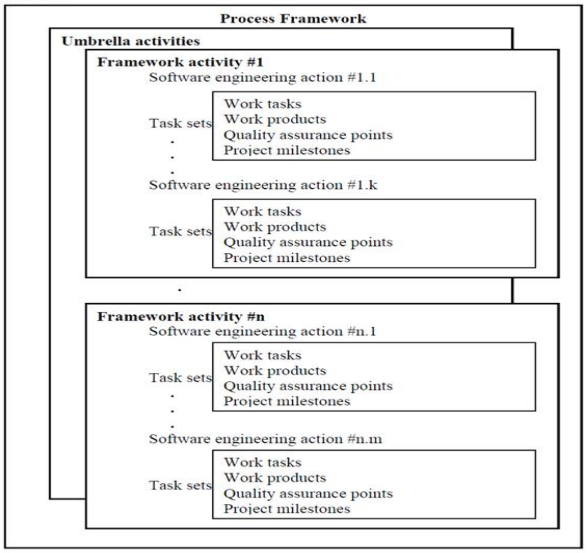
An effective strategy must consider three issues: • Risk avoidance • Risk monitoring • Risk management and contingency planning.

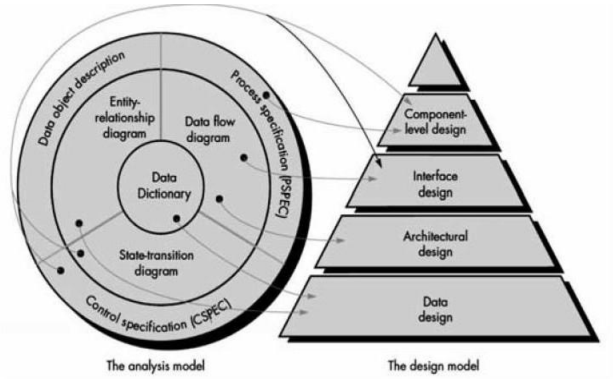
If a software team adopts a proactive approach to risk, avoidance is always the best strategy. This is achieved by developing a plan for risk mitigation.

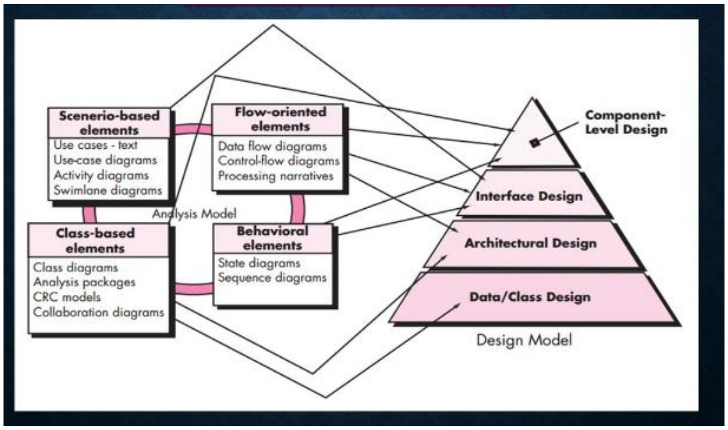
To **mitigate this risk**, project management must develop a strategy for **reducing turnover.** Among the possible steps to be taken are

- Meet with current staff to determine causes for turnover (e.g., poor working conditions, low pay, and competitive job market).
- Mitigate those causes that are under our control before the project starts.
- Once the project commences, assume turnover will occur and develop techniques to ensure continuity when people leave.
- Organize project teams so that information about each development activity is widely dispersed.
- Define documentation standards and establish mechanisms to be sure that documents are developed in a timely manner.
- Conduct peer reviews of all work (so that more than one person is "up to speed).

		• Assign a backup staff member for every critical technologist. As the project proceeds, **risk monitoring** activities commence. The project manager monitors factors that may provide an indication of whether the risk is becoming more or less likely. In the case of **high staff turnover, the following factors can be monitored**: • General attitude of team members based on project pressures. • The degree to which the team has jelled. • Interpersonal relationships among team members. • Potential problems with compensation and benefits. • The availability of jobs within the company and outside it. In addition to monitoring these factors, the **project manager should monitor the effectiveness of risk mitigation steps**. RMMM steps incur additional project cost. Part of risk management, therefore, is to evaluate when the benefits accrued by the RMMM steps are outweighed by the costs associated with implementing them. In essence, the project planner performs a classic cost/benefit analysis.	
	e) **Ans.**	**State and describe any four basic project scheduling principles**. **Basic principles software project scheduling are:** **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. **Interdependency:** 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 fulltime 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.	**4M** ***1M for each principle***

		Defined responsibilities: Every task that is scheduled should be assigned to a specific team member. **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. **Defined outcomes** – Every task that is scheduled should have a defined outcome for software projects such as a work product or part of a work product – Work products are often combined in deliverables.	
5.	**a)** **Ans.**	**Attempt any <u>TWO</u> of the following:** **Explain software process framework with neat labeled diagram and also describe software process framework activities.** Software process framework diagram :  Common process framework Framework activities Task sets Tasks Milestones, deliverables SQA points Umbrella activities OR  Process Framework Umbrella activities Framework activity #1 Software engineering action #1.1 Task sets Work tasks Work products Quality assurance points Project milestones Software engineering action #1.k Task sets Work tasks Work products Quality assurance points Project milestones Framework activity #n Software engineering action #n.1 Task sets Work tasks Work products Quality assurance points Project milestones Software engineering action #n.m Task sets Work tasks Work products Quality assurance points Project milestones	**12** **6M** ***3M for Diagram*** ***3M for description***

		A process framework establishes the foundation for a complete software process by identifying a small number of framework activities that are applicable to all software projects; In addition, the process framework encompasses a set of umbrella activities that are applicable across the entire software process. Basic framework activities: **1. Communication**: This framework activity involves heavy Communication & collaboration with the customer (and the stakeholders) and encompasses requirements gathering and other related activities. **2. Planning**: This activity establishes a plan for the software engineering work that follows. It describes the technical tasks to be conducted; the risks are analyzed. Project tracking should be done. Deadline is fixed. **3. Modeling**: This activity encompasses the creation of models that allow the developer & the customer to better understand software requirements & the design that will achieve those requirements. **4. Construction**: This activity combines code generation and the testing that is required uncovering errors in the code. **5. Deployment**: The software is delivered to the customer who evaluates the delivered product and provides feedback based on the evaluation.	
	b) **Ans.**	**Draw and explain translating requirement model into design model**. Translation of requirement model into design model diagram  Data object description Process specification (PSPEC) Entity-relationship diagram Data flow diagram Data Dictionary State-transition diagram Control specification (CSPEC) Component-level design Interface design Architectural design Data design The analysis model The design model OR	**6M** ***3M for diagram*** ***3M for description***



Software requirements, manifested by the data, functional, and behavioural models, feed the design task. Using one of a number of design methods, the design task produces a data design, an architectural design, an interface design, and a component design. Each of the elements of the analysis model provides information that is necessary to create the four design models required for a complete specification of design.

Design is a meaningful engineering representation of something that is to be built. It can be traced to a customer's requirements and at the same time assessed for quality against a set of predefined criteria for―good‖design. In the software engineering context, design focuses on four major areas of concern: data, architecture, interfaces, and components Design begins with the requirements model.

The data design transforms the information domain model created during analysis into the data structures that will be required to implement the software. The data objects and relationships defined in the entity relationship diagram and the detailed data content depicted in the data dictionary provide the basis for the data design activity. Part of data design may occur in conjunction with the design of software architecture. More detailed data design occurs as each software component is designed. The architectural design defines the relationship between major structural elements of the software, the design pattern that can be used to achieve the requirements that have been defined for the system, and the constraints that affect the way in which architectural design patterns can be applied.

	c)	**Describe following project cost estimation approaches** **i) Heuristic** **ii) Analytical** **iii) Empirical**	**6M**
	Ans.	i) Heuristic cost estimation approach: This technique basically use the concept of learning from the previous project and estimate the cost. The objective is to find a similar system produced earlier and through knowing how the properties of the new system vary from the existing one. Two classes of different heuristic Estimation Techniques: - Single variable model - Multi variable model 1. Single Variable Estimation Models: It provides a means to estimate the desired characteristics of a problem, using some previously estimated basic (independent) characteristic of the software product such as its size. A single variable estimator model takes the following form: Estimated Parameter = c1 * ed1 e= characteristic which already have been calculated. Estimated parameter is the dependent parameter to be estimated. The dependent parameters to be estimated could be effort, duration, staff size etc. c1 and d1 are constants- calculated from past projects. COCOMO is one of this type of models example. 2. Multi variable Cost Estimation Model: It has the following form Estimated Resources = c1 * e1d1 + c2 * e2d2 + - - - - e1 and e2 are the basic independent characteristics of the software already estimated. c1, c2, d1, d2, are constants. Multivariable Estimation Models are expected to give more accurate estimate compared to the Single Variable Models, since a project parameters is typically influenced by several independent parameters. The independent parameters influence the dependent parameter to different extents. This is modeled by the constants c1,c2,d1,d2..... ii) Analytical cost estimation approach Analytical estimation techniques derive the required results starting with basic assumptions regarding the project.	***2M for each approach***

- Thus, unlike empirical and heuristic techniques, analytical techniques do have scientific basis.
- Halstead's software science is an example of an analytical technique.
- Halstead_s software science can be used to derive some interesting results starting with a few simple assumptions. Halstead_s software science is especially useful for estimating software maintenance efforts.
- In fact, it outperforms both empirical and heuristic techniques when used for predicting software maintenance efforts.
- Halstead's Software Science – An Analytical Technique Halstead_s software science is an analytical technique to measure size, development effort, and development cost of software products.
- Halstead used a few primitive program parameters to develop the expressions for overall program length, potential minimum value, actual volume, effort, and development time.

iii) Empirical cost estimation approach

Empirical estimation techniques are based on making an educated guess of the project parameters. While using this technique, prior experience with development of similar products is helpful. Although empirical estimation techniques are based on common sense, different activities involved in estimation have been formalized over the years.
Two popular empirical estimation techniques are:

1. Expert judgment technique and
2. Delphi cost estimation.

1. Expert Judgment Technique

Expert judgment is one of the most widely used estimation techniques.

- In this approach, an expert makes an educated guess of the problem size after analyzing the problem thoroughly.
- Usually, the expert estimates the cost of the different components (i.e. modules or subsystems) of the system and then combines them to arrive at the overall estimate.
- However, this technique is subject to human errors and individual bias.

		• Also, it is possible that the expert may overlook some factors inadvertently. • Further, an expert making an estimate may not have experience and knowledge of all aspects of a project. 2. Delphi cost estimation • Delphi cost estimation approach tries to overcome some of the short comings of the expert judgment approach. • Delphi estimation is carried out by a team comprising of a group of experts and a coordinator. • In this approach, the coordinator provides each estimator with a copy of the software requirements specification (SRS) document and a form for recording his cost estimate. • Estimators complete their individual estimates anonymously and submit to the coordinator. In their estimates, the estimators mention any unusual characteristic of the product which has influenced his estimation. • The coordinator prepares and distributes the summary of the responses of all the estimators, and includes any unusual rationale noted by any of the estimators. • Based on this summary, the estimators re-estimate. This process is iterated for several rounds. However, no discussion among the estimators is allowed during the entire estimation process. The idea behind this is that if any discussion is allowed among the estimators, then many estimators may easily get influenced by the rationale of an estimator who may be more experienced or senior. After the completion of several iterations of estimations, the coordinator takes the responsibility of compiling the results and preparing the final estimate.	
6.	**a)** **Ans.**	**Attempt any TWO of the following:** **State and describe any six communication principles.** Communication principles are as given below: **1. Listen carefully** i. To collect lots of data from the client, the developer team has to listen carefully. ii. Maximum information with respect to requirement and the specifications should be collected before the implementation and the designing of the software.	**12** **6M** ***1M for each principles with description***

2. Prepare before you communicate

i. A proper agenda or the guidelines for the meetings should be prepared before the start of the meeting.

ii. Complete detail and the description about the clients and their work area should be gathered to deliver the software up to the best expectation.

3. Have a facilitator for any communication meeting

i. The requirement gathering and the specification are important for any software development, hence the communication should continue till the requirement gathering is over.

4. Face-to-face communication is best

i. It is always better to sit across the table and have discussion on the requirement on the software development by the client and the developer.

ii. Distant communication does not help gathering data properly.

5. Take notes and document decisions

i. The important points discussed should also be recorded.

ii. Proper notes and the documentation is important for the successful completion and deployment of the project.

6. Strive for collaboration

i. Collaboration in terms of teamwork is required for the successful completion of the software.

ii. The collective knowledge of the team members should be implemented in the development.

7. Stay focused and modularize your discussion

i. As the development is the working of many team members, so the possibility of the discussion going from one topic to the other topic is quite possible.

ii. As a good software developer it is required that the discussion remains focused on the specified area.

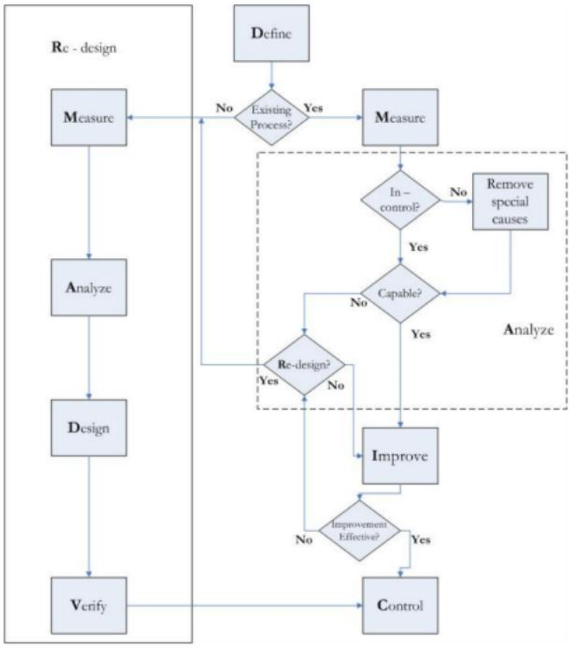
8. Draw a picture if something is unclear

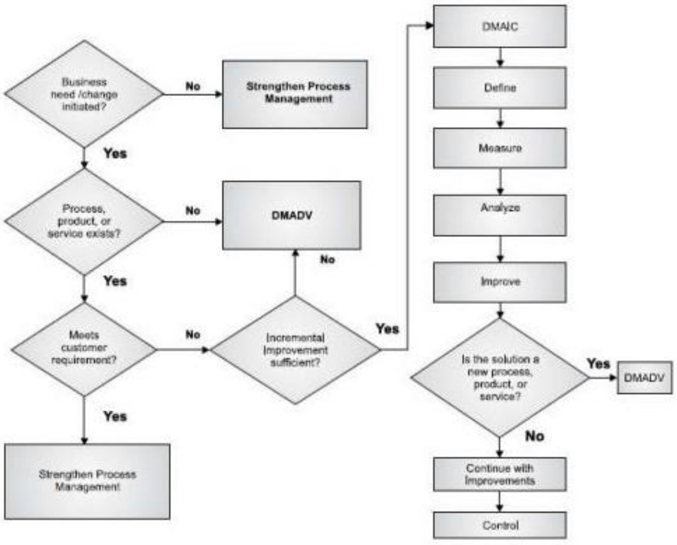
i. Drawing flowcharts, E-R diagrams and other supporting graphical representations give clarity to the discussion and the documentation.

9. Move on once you agree, move on when you can't agree, move on if something unclear can't be clarified at the moment

i. Healthy discussion leads to the final conclusion of successful implementation of the software.

ii. Once reached to final statement recorded should move to the next step.

		iii. If no conclusion is reached than that point should be left and move ahead with new implementation which is cost effective. **10. Negotiation is not a contest or game** i. Negotiation should be mutual not to put someone down or make them feel to be the loser.	
	b) Ans.	**Describe six sigma strategy in details.** 1. Six Sigma is the process of producing high and improved quality output. 2. This can be done in two phases – identification and elimination. The cause of defects is identified and appropriate elimination is done which reduces variation in whole processes. 3. Six Sigma projects follow two project methodologies:  Re - design Measure Analyze Design Verify Define Existing Process? No Yes Measure In – control? No Remove special causes Yes Capable? No Yes Analyze Re-design? Yes No Improve Improvement Effective? No Yes Control	6M ***3M (1.5 each) DMAIC & DMADV Description*** ***3M for diagram***


Business need /change initiated?
No
Strengthen Process Management
Yes
Process, product, or service exists?
No
DMADV
No
Yes
Meets customer requirement?
No
Incremental Improvement sufficient?
Yes
Yes
Strengthen Process Management
DMAIC
Define
Measure
Analyze
Improve
Is the solution a new process, product, or service?
Yes
DMADV
No
Continue with Improvements
Control

(Note :Any other correct diagram should be given marks)

a) DMAIC

It specifies a data-driven quality strategy for improving processes. This methodology is used to enhance an existing business process. The DMAIC project methodology has five phases:

i) Define:-It covers the process mapping and flow-charting, project charter development, problem- solving tools, and so-called 7-M tools.

ii) Measure:-It includes the principles of measurement, continuous and discrete data, and scales of measurement, an overview of the principle of variations and repeatability and reproducibility (RR) studies for continuous and discrete data.

iii) Analyze:-It covers establishing a process baseline, how to determine process improvement goals, knowledge discovery, including descriptive and exploratory data analysis and data mining tools, the basic principle of Statistical Process Control (SPC), specialized control charts, process capability analysis, correlation and regression analysis, analysis of categorical data, and non-parametric statistical methods.

iv) Improve:-It covers project management, risk assessment, process simulation, and design of experiments (DOE), robust design concepts, and process optimization.

		v) Control:-It covers process control planning, using SPC for operational control and PRE-Control. a) DMADV It specifies a data-driven quality strategy for designing products and processes. This method is used to create new product designs or process designs in such a way that it results in a more predictable, mature, and detect free performance. The DMADV project methodology has five phases: **a. Define:-**It defines the problem or project goal that needs to be addressed. **b. Measure:-**It measures and determines the customer's needs and specifications. **c. Analyze:-**It analyzes the process to meet customer needs. **d. Design:-**It can design a process that will meet customer needs. **e. Verify:-**It can verify the design performance and ability to meet customer needs.	
	c) **Ans.**	**Use COCOMO model to calculate** **i) Effort** **ii) Development Time** **iii) Average staff size** **iv) Productivity** **If estimated size of project is 400 KLOC using embedded mode.** Given size if project= 400 KLOC; mode = embedded In embedded mode : a= 3.6 b=1.20 c=2.5 d=0.32 i) Effort E = a*(KLOC)^b E=3.6* (400)^ 1.20 =3.6 * 1325.78 = 4772.80 per month ii) Development time D= c *(E)^ d = 2.5 *(4772.80)^0.32 =37.57 months	**6M** ***2M for each correct answer and formula of effort, development time and productivity***

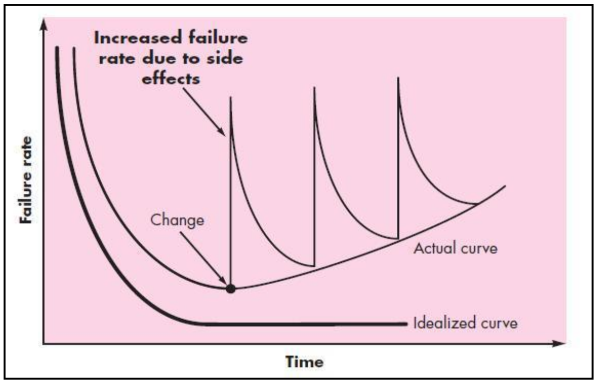
		iii) Average staff size SS= E/ D =4772.80 / 37.57 =127.03 persons iv) Productivity P=KLOC /E =400/ 4772.80 =0.083	

Winter – 19 EXAMINATION

Subject Name: Software Engineering **Model Answer** **Subject Code: 22413**

Important Instructions to examiners:

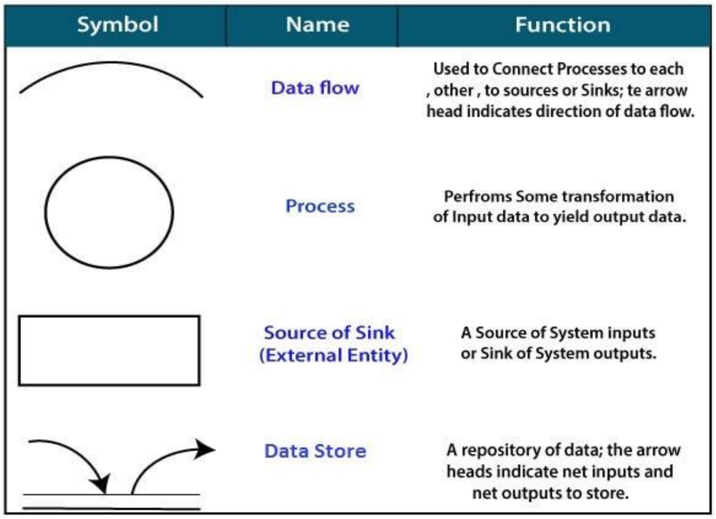
1) The answers should be examined by key words and not as word-to-word as given in the model answer scheme.
2) The model answer and the answer written by candidate may vary but the examiner may try to assess the understanding level of the candidate.
3) The language errors such as grammatical, spelling 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 judgement 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.

Q. No.	Sub Q. N.	Answer	Marking Scheme
1.		**Attempt any Five of the following:**	**10M**
	a	**Define software. Draw the failure curve for software.**	**2M**
	Ans	Definition of Software Software is: 1. Instructions (computer programs) that when executed provide desired features, function, and performance; 2. Data structures that enable the programs to adequately manipulate information, and 3. Descriptive information (documents) in both hard copy and virtual forms that describes the operation and use of the programs. 	Correct definition 1M and diagram 1M

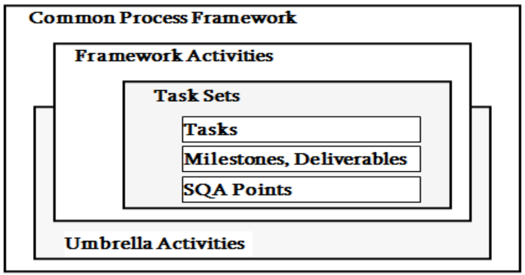
	b	**State two characteristics of Software.**	**2M**
	Ans	**Characteristics of software :** • Software is developed or engineered; it is not manufactured in the classical sense. • Software doesn't "wear out." But it does deteriorate! • Although the industry is moving toward component-based construction, most software continues to be custom built.	Any two correct Characteristics - 1M each
	c	**Define software requirement specification**	**2M**
	Ans	**Concept:** A software requirements specification (SRS) is a document that is created when a detailed description of all aspects of the software to be built that must be specified before the project is to commence. It is a primary document for development of software. It is written by Business Analysts who interact with client and gather the requirements to build the software.	Correct definition -2M
	d	**Define proactive and reactive risk strategy.**	**2M**
	Ans	**Reactive risk strategies** • Reactive risk strategy follows that the risks have to be tackled at the time of their occurrence. • No precautions are to be taken as per this strategy. • They are meant for risks with relatively smaller impact. • More commonly, the software team does nothing about risks until something goes wrong. • Then, the team flies into action in an attempt to correct the problem rapidly. This is often called a fire-fighting mode. • **Proactive risk strategies** • It follows that the risks have to be identified before start of the project. • They have to be analysed by assessing their probability of occurrence, their impact after occurrence, and steps to be followed for its precaution.	Correct definition -1M each
	e	**Name two cost estimation approaches.**	**2M**
	Ans	• Heuristic Estimation Approach • Analytical Estimation Approach • Empirical Estimation Approach	Any two techniques-1M each
	f	**Define software quality.**	**2M**
	Ans	1.Quality means that a product satisfies the demands of its specifications 2. It also means achieving a high level of customer satisfaction with the product 3. In software systems this is difficult • Customer quality requirements(e.g. efficiency or reliability) often conflict with developer quality requirements (e.g. maintainability or reusability)	Correct Definition-2M

		• Software specifications are often incomplete, inconsistent, or ambiguous	
	g	**Name four software quality assurance activities.**	**2M**
	Ans	These activities are performed (or facilitated) by an independent SQA group that: i. Prepares an SQA plan for a project. ii. Participates in the development of the project's software process description. iii. Reviews software engineering activities to verify compliance with the defined software process. iv. Audits designated software work products to verify compliance with those defined as part of the software process. v. Ensures that deviations in software work and work products are documented and handled according to a documented procedure. vi. Records any noncompliance and reports to senior management.	Any 4 activity name-1/2M each
2.		**Attempt any Three of the following:**	**12M**
	a	**State and explain with examples four categories of software.**	**4M**
	Ans	**Types / Categories of Software** **1. System Software** 1. System software is a collection of programs written to service other programs. 2. Few examples of system software are compilers, editors, and file management utilities, process complex, but determinate, information structures. 3. Other systems applications are operating system components, drivers, and telecommunications. Example : DOS, WINDOWS **2. Real-time Software** ***(*Question: Explain the features of real world software. – 3 Marks*)*** 1. Software that monitors or analyses or controls real-world events as they occur is called real time. 2. Elements of real-time software include a data gathering component that collects and formats information from an external environment, an analysis component that transforms information as required by the application. 3. A control/output component that responds to the external environment and a monitoring component that coordinates all other components so that real-time response can be maintained. Example : airline reservation system, railway reservation system **3. Business Software** 1. Business information processing is the largest single software application area. Discrete "systems".	Any 4 types explanation with example-4M

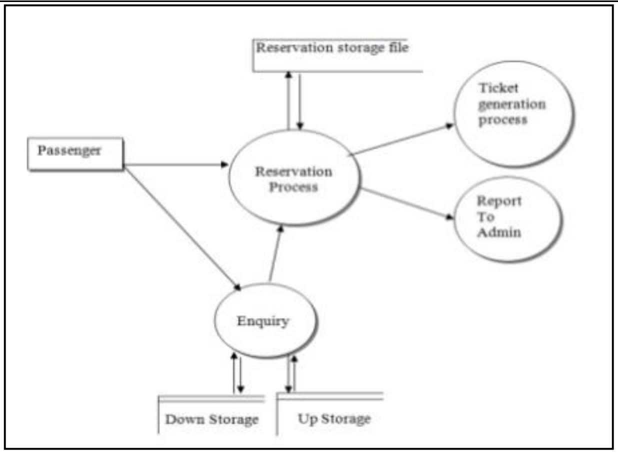
		2. For example: payroll, accounts receivable/payable, inventory have evolved into management information system (MIS) software that accesses one or more large databases containing business information. 3. Applications in this area restructure existing data in a way that facilitates business operations or management decision making. 4. In addition to conventional data processing application, business software applications also encompass interactive computing. Example : Tally **4. Engineering and Scientific Software** 1. Engineering and scientific software have been characterized by "number crunching" algorithms. 2. Applications range from astronomy to volcanology, from automotive stress analysis to space shuttle orbital dynamics, and from molecular biology to automated manufacturing. 3. However, modern applications within the engineering/scientific area are moving away from conventional numerical algorithms. 4. Computer-aided design, system simulation, and other interactive applications have begun to take on real-time and even system software characteristics. Example : CAD / CAM software **5. Embedded Software** 1. Intelligent products have become commonplace in nearly every consumer and industrial market. 2. Embedded software resides in read-only memory and is used to control products and systems for the consumer and industrial markets. 3. Embedded software can perform very limited and esoteric functions, for example: keypad control for a microwave oven. 4. To provide significant function and control capability, for example: digital functions in an automobile such as fuel control, dashboard displays, and braking systems. Example : Microwave, Washing machine software **6. Personal Computer Software** 1. The personal computer software market has burgeoned over the past two decades. 2.Word processing, spread sheets, computer graphics, multimedia, entertainment, database management, personal and business fi applications, external network, and database access are only a few of hundreds of applications. Example: Microsoft word, Excel.	
	b	**Explain the notations used for preparing a Data Flow diagram.**	**4M**
	Ans	**Circle:** A circle (bubble) shows a process that transforms data inputs into data outputs. **Data Flow:** A curved line shows the flow of data into or out of a process or data store.	Correct symbols with explanation -1M each

<table>
<tr><td></td><td></td><td><b>Data Store:</b> A set of parallel lines shows a place for the collection of data items. A data store indicates that the data is stored which can be used at a later stage or by the other processes in a different order. The data store can have an element or group of elements.
<b>Source or Sink:</b> Source or Sink is an external entity and acts as a source of system inputs or sink of system outputs.
<table><tr><th>Symbol</th><th>Name</th><th>Function</th></tr><tr><td></td><td>Data flow</td><td>Used to Connect Processes to each , other , to sources or Sinks; te arrow head indicates direction of data flow.</td></tr><tr><td></td><td>Process</td><td>Perfroms Some transformation of Input data to yield output data.</td></tr><tr><td></td><td>Source of Sink (External Entity)</td><td>A Source of System inputs or Sink of System outputs.</td></tr><tr><td></td><td>Data Store</td><td>A repository of data; the arrow heads indicate net inputs and net outputs to store.</td></tr></table><b>Symbols for Data Flow Diagrams</b></td><td></td></tr>
<tr><td></td><td><b>c</b></td><td><b>Describe 4 P’s of management spectrum giving their significance.</b></td><td><b>4M</b></td></tr>
<tr><td></td><td><b>Ans</b></td><td><b>The Management Spectrum – 4 Ps and their Significance</b>
Effective software project management focuses on these items (in this order) Deals with the cultivation of motivated, highly skilled people
<b>1. The people</b>
i. Consists of the stakeholders, the team leaders, and the software team
<b>2. The product</b>
i. Product objectives and scope should be established before a project can be planned.
<b>3. The process</b>
i. The software process provides the framework from which a comprehensive plan for software development can be established.
<b>4. The project</b>
i. Planning and controlling a software project is done for one primary reason…it is the only known way to manage complexity
ii. In a 1998 survey, 26% of software projects failed outright, 46% experienced cost and schedule overruns.</td><td>Description of each P’s-1M each</td></tr>
<tr><td></td><td><b>d</b></td><td><b>Explain four basic principles of software project scheduling.</b></td><td></td></tr>
</table>

	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. **Interdependency:** 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 fulltime 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 responsibilitie**s: 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. **Defined outcomes** – Every task that is scheduled should have a defined outcome for software projects such as a work product or part of a work product – Work products are often combined in deliverables	Any four correct principles -1M each
3.		**Attempt any Three of the following:**	**12M**
	a	**Explain Process framework with a suitable diagram.**	**4M**
	Ans	A process framework establishes the foundation for a complete software process by identifying a small number of framework activities that are applicable to all software projects; In addition, the process framework encompasses a set of umbrella activities that are applicable across the entire software process.	Description 2M Diagram 2 M

		 Common Process Framework Framework Activities Task Sets Tasks Milestones, Deliverables SQA Points Umbrella Activities Figure: Chart of Process Framework Basic framework activities: **1. Communication**: This framework activity involves heavy communication & collaboration with the customer (and the stakeholders) and encompasses requirements gathering and other related activities. **2. Planning**: This activity establishes a plan for the software engineering work that follows. It describes the technical tasks to be conducted; the risks are analyzed. Project tracking should be done. Deadline is fixed. **3. Modeling**: This activity encompasses the creation of models that allow the developer & the customer to better understand software requirements & the design that will achieve those requirements. **4. Construction**: This activity combines code generation and the testing that is required uncovering errors in the code. **5. Deployment**: The software is delivered to the customer who evaluates the delivered product and provides feedback based on the evaluation.	
	b	**Describe four principles of good planning.**	**4M**
	Ans	**Principle 1. Understand the scope of the project.** It's impossible to use a road map if you don't know where you‘re going. Scope provides the software team with a destination. **Principle 2. Involve stakeholders in the planning activity.** Stakeholders define priorities and establish project constraints. To accommodate these realities, software engineers must often negotiate order of delivery, time lines, and other project-related issues. **Principle 3. Recognize that planning is iterative.** A project plan is never engraved in stone. As work begins, it is very likely that things will change. As a consequence, the plan must be adjusted to accommodate these changes. In addition, iterative, incremental process models dictate re-planning after the delivery of each software increment based on feedback received from users. **Principle 4. Estimate based on what you know.** The intent of estimation is to provide an indication of effort, cost, and task duration, based on the team‘s current understanding of the work to be done. If information is vague or unreliable, estimates will be equally unreliable. **Principle 5.Consider risk as you defines the plan.** If you have identified risks that have high impact and high probability, contingency planning is necessary. In addition, the project plan (including the schedule) should be	Any 4 Principles; 1 M each

		adjusted to accommodate the likelihood that one or more of these risks will occur. **Principle 6. Be realistic.** People don't work 100 percent of every day. Noise always enters into any human communication. Omissions and ambiguity are facts of life. Change will occur. Even the best software engineers make mistakes. These and other realities should be considered as a project plan is established. **Principle 7.Adjust granularity as you defines the plan.** Granularity refers to the level of detail that is introduced as a project plan is developed. A high-granularity plan provides significant work task detail that is planned over relatively short time increments (so that tracking and control occur frequently). A low-granularity plan provides broader work tasks that are planned over longer time periods. In general, granularity moves from high to low as the project time line moves away from the current date. Over the next few weeks or months, the project can be planned in significant detail. Activities that won't occur for many months do not require high granularity (too much can change). **Principle 8. Define how you intend to ensure quality.** The plan should identify how the software team intends to ensure quality. If technical reviews are to be conducted, they should be scheduled. If pair programming is to be used during construction, it should be explicitly defined within the plan. **Principle 9. Describe how you intend to accommodate change.** Even the best planning can be obviated by uncontrolled change. You should identify how changes are to be accommodated as software engineering work proceeds. For example, can the customer request a change at any time? If a change is requested, is the team obliged to implement it immediately? How is the impact and cost of the change assessed? **Principle 10.Track the plan frequently and make adjustments as required.** Software projects fall behind schedule one day at a time. Therefore, it makes sense to track progress on a daily basis, looking for problem areas and situations in which scheduled work does not conform to actual work conducted. When slippage is encountered, the plan is adjusted accordingly.	
	c	**Draw and explain Level 1 DFD for railway reservation system.**	**4M**
	Ans		Diagram 2 M Description 2



The passenger can initiate either Reservation process or Enquiry process; If a user opts for Reservation process then the system shall proceed with ticket generation process and same needs to be notified to the Admin. If user opts for enquiry module then appropriate request shall be entertain and result to be displayed to the user.

d **With an example, explain Line of Code (LOC) based estimation.** **4M**

Ans LOC-Based Estimation: As an example of LOC and FP problem-based estimation techniques, let us consider a software package to be developed for a computer-aided design application for mechanical components.

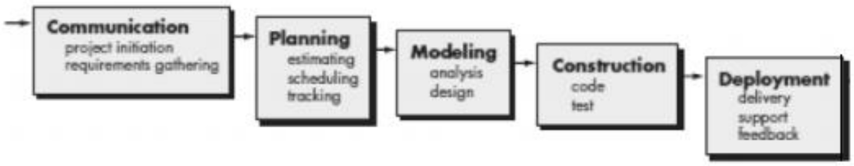
A review of the System Specification indicates that the software is to execute on an engineering workstation and must interface with various computer graphics peripherals including a mouse, digitizer, high resolution color display and laser printer.

Using the System Specification as a guide, a preliminary statement of software scope can be developed:

Description 2M
Example 2M

Example:

Function	**Estimated LOC**
User interface and control facilities (UICF)	2,300
Two-dimensional geometric analysis (2DGA)	5,300
Three-dimensional geometric analysis (3DGA)	6,800
Database management (DBM)	3,350
Computer graphics display facilities (CGDF)	4,950
Peripheral control function (PCF)	2,100
Design analysis modules (DAM)	8,400

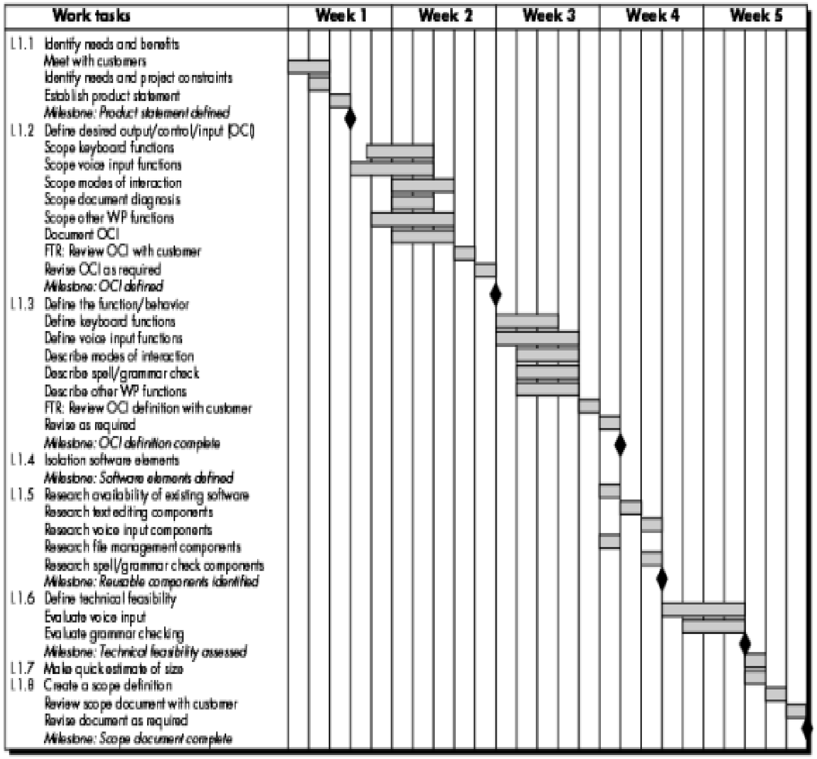
		Estimated lines of code | 33,200	
4.		**Attempt any Three of the following:**	**12M**
	a	**Explain waterfall process model. State its advantages and disadvantages.**	**4M**
	Ans	 Communication project initiation requirements gathering Planning estimating scheduling tracking Modeling analysis design Construction code test Deployment delivery support feedback The waterfall model is a traditional method, sometimes called the classic life cycle. This is one of the initial models. As the figure implies stages are cascaded and shall be developed one after the other. It suggests a systematic, sequential approach to software development that begins with customer specification of requirements and progresses through, communication, planning, modeling construction and deployment. 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. 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 limitations 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. **Advantages of waterfall model:** 1. This model is simple and easy to understand and use. 2. It is easy to manage due to the rigidity of the model – each phase has specific deliverables and a review process.	Description 2M Any 2 advantage 1M Any 2 Disadvantages 2M

		3. In this model phases are processed and completed one at a time. Phases do not overlap. 4. Waterfall model works well for smaller projects where requirements are very well understood. **Disadvantages of waterfall model:** 1. Once an application is in the testing stage, it is very difficult to go back and change something that was not well-thought out in the concept stage. 2. No working software is produced until late during the life cycle. 3. High amounts of risk and uncertainty. 4. Not a good model for complex and object-oriented projects. 5. Poor model for long and ongoing projects. 6. Not suitable for the projects where requirements are at a moderate to high risk of changing.	
	b	**Enlist core principles of software engineering practice.**	**4M**
	Ans	1. Reason it all exists. Provide value to the user 2.Keep it simple stupid 3.Maintain the vision 4. What you reproduce, someone else will have to consume. (implement knowing someone else will have to understand what you are doing) 5.Be open to the future 6. Plan ahead for reuse Plan ahead for reuse Think!	List of all 7 core principles 4M
	c	**Describe RMMM Strategy.**	**4M**
	Ans	Risk mitigation, monitoring, and management (RMMM) plan. A risk management strategy can be included in the software project plan or the risk management steps can be organized into a separate Risk Mitigation, Monitoring and Management Plan. The RMMM plan documents all work performed as part of risk analysis and is used by the project manager as part of the overall project plan. Once RMMM has been documented and the project has begun, risk mitigation and monitoring steps commence. Risk mitigation is a problem avoidance activity. Risk monitoring is a project tracking activity with three primary objectives: (1) To assess whether predicted risks do, in fact, occur; (2) To ensure that risk aversion steps defined for the risk are being properly applied; and (3) To collect information that can be used for future risk analysis. In many cases, the problems that occur during a project can be traced to more than one risk. Another job of risk monitoring is to attempt to allocate origin (what risk(s) caused which problems throughout the project). An effective strategy must consider three issues: • Risk avoidance • Risk monitoring • Risk management and contingency planning.	Description 4M any relevant description shall be considered

		If a software team adopts a proactive approach to risk, avoidance is always the best strategy. This is achieved by developing a plan for risk mitigation. To mitigate this risk, project management must develop a strategy for reducing turnover. Among the possible steps to be taken are • Meet with current staff to determine causes for turnover (e.g., poor working conditions, low pay, and competitive job market). • Mitigate those causes that are under our control before the project starts. • Once the project commences, assume turnover will occur and develop techniques to ensure continuity when people leave. • Organize project teams so that information about each development activity is widely dispersed. • Define documentation standards and establish mechanisms to be sure that documents are developed in a timely manner. • Conduct peer reviews of all work (so that more than one person is "up to speed). • Assign a backup staff member for every critical technologist. As the project proceeds, risk monitoring activities commence. The project manager monitors factors that may provide an indication of whether the risk is becoming more or less likely. In the case of high staff turnover, the following factors can be monitored: • General attitude of team members based on project pressures. • The degree to which the team has jelled. • Interpersonal relationships among team members. • Potential problems with compensation and benefits. • The availability of jobs within the company and outside it. In addition to monitoring these factors, the project manager should monitor the effectiveness of risk mitigation steps. RMMM steps incur additional project cost. Part of risk management, therefore, is to evaluate when the benefits accrued by the RMMM steps are outweighed by the costs associated with implementing them. In essence, the project planner performs a classic cost/benefit analysis.	
	d	**Describe the Analytical method of project cost estimation with example.**	**4M**
	Ans	Analytical estimation techniques derive the required results starting with basic assumptions regarding the project. Thus, unlike empirical and heuristic techniques, analytical techniques do have scientific basis. **Halstead's software science is an example of an analytical technique.** Halstead's software science can be used to derive some interesting results starting with a few simple assumptions. Halstead's software science is especially useful for estimating software maintenance efforts. In fact, it outperforms both empirical and heuristic techniques when used for predicting software maintenance efforts. Halstead's Software Science – An Analytical Technique Halstead's software science is an analytical technique to measure size, development	Description 2M Example 2M

		effort, and development cost of software products. Halstead used a few primitive program parameters to develop the expressions for overall program length, potential minimum value, actual volume, effort, and development time. For a given program, let: • η1 be the number of unique operators used in the program, • η2 be the number of unique operands used in the program, • N1 be the total number of operators used in the program, • N2 be the total number of operands used in the program. Example: Let us consider the following C program: main( ) { int a, b, c, avg; scanf(“%d %d %d”, &a, &b, &c); avg = (a+b+c)/3; printf(“avg = %d”, avg); } The unique operators are: main, (), {}, int, scanf, &, “, ”, =, +, /, printf The unique operands are: a, b, c, &a, &b, &c, a+b+c, avg, 3, “%d %d %d”, “avg = %d” Therefore, η1 = 12, η2 = 11 Estimated Length = (12*log12 + 11*log11) = (12*3.58 + 11*3.45) = (43+38) = 81 Volume = Length*log(23) = 81*4.52 = 366	
	e	**Explain GANTT chart and its application for project tracking with an example.**	**4M**
	Ans	When creating software project schedule, we begin with a set of tasks. If automated tools are used, the work breakdown is input as a task network or task outline. Effort, duration and start date are then input for each task, In addition, tasks may be assigned to specific individuals. As a consequence of this input, a time-line chart, also called a Gantt chart is generated. A time-line chart can be developed for the entire project. The figure below depicts a part of a software project schedule that emphasizes scoping task for a word-processing (WP) software product. All project tasks 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 diamond indicates milestones. Once the information necessary for the generation of a time-line chart has been input, the majority of software project scheduling tools produce project tables – a tabular listing of all project tasks, their planned and actual start and end dates, and a variety of related information. Used in conjunction with the time-line chart, project tables enable to track progress.	Description and Example 3M Application1M

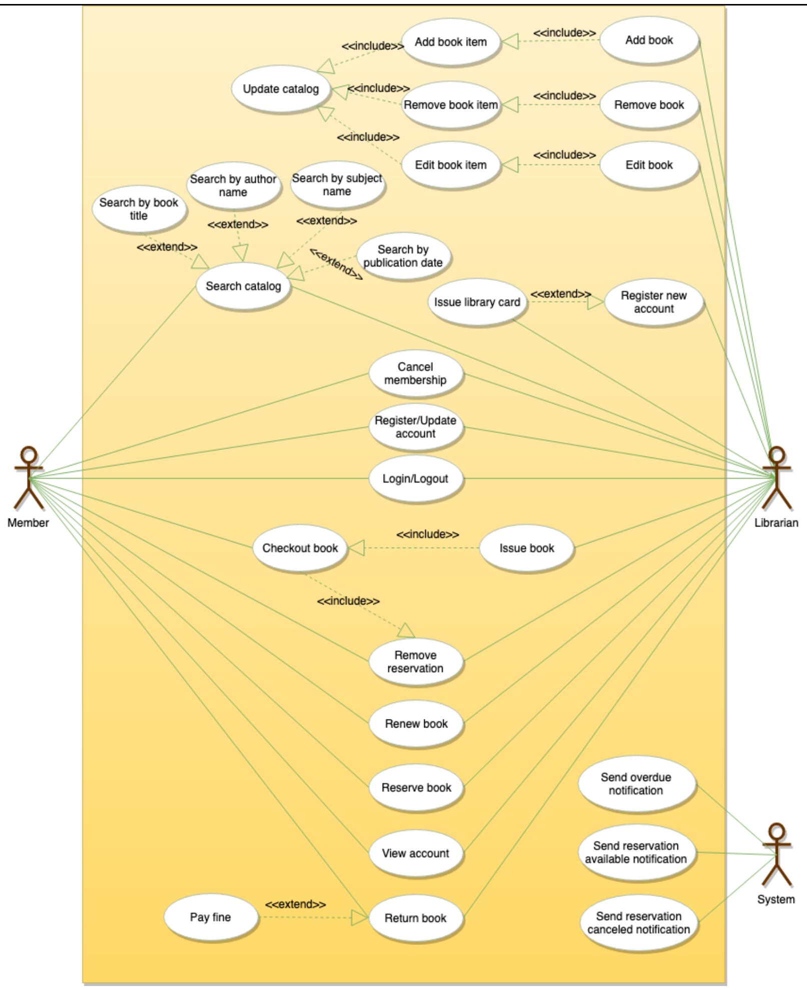
Time-Line chart - Micro-level Scheduling



Application of Gantt Chart

- The sheer simplicity and ease-of-access of all relevant information make Gantt charts an ideal choice for teams to use them for organizing their schedules. Due to this, Gantt charts are widely used in project management, IT and development teams.
- Apart from them, marketing, engineering, product launch, manufacturing teams can also use Gantt charts to get an overview of how things are rolling on the work front.

5.		**Attempt any Two of the following:**	**12M**
	a	**Sketch a use case diagram for library management system with minimum four use cases and two actors**.	**6M**
	Ans		Correct/relevant any four use cases -6M

		 Add book item <<include>> Add book Update catalog Remove book item Remove book Edit book item Edit book Search by author name Search by subject name Search by book title <<extend>> Search by publication date Search catalog Issue library card Register new account Cancel membership Register/Update account Login/Logout Member Librarian Checkout book Issue book Remove reservation Renew book Reserve book Send overdue notification Send reservation available notification System View account Pay fine Return book Send reservation canceled notification	
	b	**Explain the concept of black box testing and white box testing.**	**6M**
	Ans	**Black Box Testing:** • It is a way of software testing in which the internal structure or the program or the code is hidden and nothing is known about it. • It also known as data-driven, box testing, data-, and functional testing. • This type of testing is ideal for higher levels of testing like System Testing, Acceptance testing. • It is mostly done by software testers. • No knowledge of implementation is needed. • It is functional test of the software. • Testing can start after preparing requirement specification document.	Black box testing explanation -3M and white box testing explanation- 3M

- Techniques used:
    - **Equivalence partitioning**: Equivalence partitioning divides input values into valid and invalid partitions and selecting corresponding values from each partition of the test data.
    - **Boundary value analysis:** checks boundaries for input values.
- **Advantages of Black Box Testing**
    - Efficient when used on large systems.
    - Since the tester and developer are independent of each other, testing is balanced and unprejudiced.
    - Tester can be non-technical.
    - There is no need for the tester to have detailed functional knowledge of system.
    - Tests will be done from an end user's point of view, because the end user should accept the system. (This testing technique is sometimes also called Acceptance testing.)
    - Testing helps to identify vagueness and contradictions in functional specifications.
    - Test cases can be designed as soon as the functional specifications are complete.
- **Disadvantages of Black Box Testing**
    - Test cases are challenging to design without having clear functional specifications.
    - It is difficult to identify tricky inputs if the test cases are not developed based on specifications.
    - It is difficult to identify all possible inputs in limited testing time. As a result, writing test cases may be slow and difficult.
    - There are chances of having unidentified paths during the testing process.
    - There is a high probability of repeating tests already performed by the programmer.

White Box Testing:

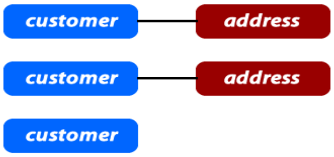
- It is a way of testing the software in which the tester has knowledge about the internal structure r the code or the program of the software.
- It is also called structural testing, clear box testing, code-based testing, or glass box testing.

		• Testing is best suited for a lower level of testing like Unit Testing, Integration testing. • It is mostly done by software developers. • Knowledge of implementation is required. • It is structural test of the software. • Testing can start after preparing for Detail design document. • Techniques Used: o Statement Coverage, Branch coverage, and Path coverage are White Box testing technique. o **Statement Coverage** validates whether every line of the code is executed at least once. o **Branch coverage** validates whether each branch is executed at least once. o **Path coverage** method tests all the paths of the program. • **Advantages of White Box Testing** • Code optimization by finding hidden errors. • White box tests cases can be easily automated. • Testing is more thorough as all code paths are usually covered. • Testing can start early in SDLC even if GUI is not available. • **Disadvantages of White Box Testing** White box testing can be quite complex and expensive. • Developers who usually execute white box test cases detest it. The white box testing by developers is not detailed can lead to production errors. • White box testing requires professional resources, with a detailed understanding of programming and implementation. • White-box testing is time-consuming, bigger programming applications take the time to test fully.	
	c	**Calculate using COCOMO model** **i)Effort** **ii)Project duration** **iii)Average staff size** **If estimated size of project is 200 KLOC using organic mode.**	**6M**

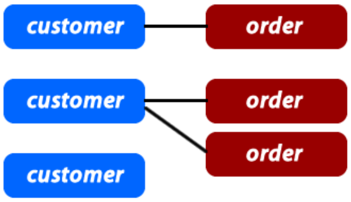
	Ans	Given data: size=200 KLOC mode= organic 1. Effort: $E = a (KLOC)^{b}$ For organic a=2.4 and b= 1.05 $E= 2.4 (200)^{1.05}$ = 626 staff members 2. Project duration: $TDEV= c (E)^{d}$ Where TDEV= time for development c and d are constant to be determined E = effort For organic mode, c= 2.5 and d= 0.38 $TDEV= 2.5 (626)^{0.38}$ = 29 months 3. Average staff size: SS = E/TDEV SS = 626/29 = 22 staffs	Correct Answer for each point asked -6M
6.		**Attempt any Two of the following:**	**12M**
	a	**Define data objects, attributes, relationship, and cardinality, with example of each.**	**6M**
	Ans	**Data Object:** A data object is an entity/object in the real world with an independent existence that can be differentiated from other objects. Example: An entity might be o An object with physical existence (e.g., a lecturer, a student, a car) o An object with conceptual existence (e.g., a course, a job, a position)	Definition of each one-4M and example of each-2M

Attributes: Each data object/ entity is described by a set of attributes (e.g., Employee = (Name, Address, Birthdate (Age), Salary).

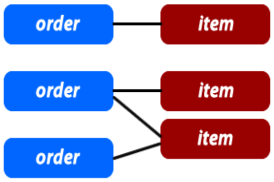
Each attribute has a name, and is associated with an entity and a domain of legal values.
Example: Employee = (Name, Address, Birthdate (Age), Salary).
Relationship: A relationship identifies names and defines an association between two entity types **One**-to-one relationship: Example: We have a relationship between the Customers table and the Addresses table. If each address can belong to only one customer, this relationship is "One to One".



One –to – many relationship:
Example:
Each customer may have zero, one or multiple orders. But an order can belong to only one customer.



Many- to – many Relationship:
Example: In some cases, you may need multiple instances on both sides of the relationship. For example, each order can contain multiple items. And each item can also be in multiple orders.

		 **Cardinality:** In the case of Data Modeling, **Cardinality** defines the number of attributes in one entity set, which can be associated with the number of attributes of other set via relationship set. Example: **One-to-one, One-to-many, Many-to-one, Many-to-many.**	
	b	**Compare CMMI and ISO for software w.r.t to** **i)scope** **ii)Approach** **Iii) Implementation.**	**6M**
	Ans	**Difference between CMMI and ISO based on** **SCOPE:** CMMI is rigid and extends only to businesses developing software intensive systems. ISO is flexible and applicable to all manufacturing industries. CMMI focuses on engineering and project management processes whereas ISO's focus is generic in nature. CMMI mandates generic and specific practices and businesses have a choice of selecting the model relevant to their business needs from 22 developed process areas. ISO requirements are same for all companies, industries, and disciplines. **APPROACH:**CMMI requires ingraining processes into business needs so that such processes become part of corporate culture and do not break down under the pressure of deadlines. ISO specifies to conformance and remains oblivious as to whether such conformance is of strategic business value or not.CMMI approaches risk management as an organized and technical discipline by identifying risk factors, quantifying such risk factors, and tracking them throughout the project life cycle. ISO was until recently neutral on risk management. ISO 31000:2009 now provides generic guidelines for the design, implementation, and maintenance of risk management processes throughout an organization.	Difference based on Scope-2M Approach-2M and Implementation 2M

		Although CMMI focuses on linkage of processes to business goals, customer satisfaction is not a factor in the ranking whereas customer satisfaction is an important part of ISO requirements. **IMPLEMENTATION:** Neither CMMI nor ISO requires the establishment of new processes. CMMI compares the existing processes to industry best practices whereas ISO requires adjustment of existing processes to confirm to the specific ISO requirements. In practice, some organizations tend to rely on extensive documentation while implementing both CMMI and ISO. Most organizations tend to constitute in-house teams, or rely on external auditors to see through the implementation process.	
	c	**Explain six function of requirement engineering process.**	**6M**
	Ans	**Requirement Engineering:** The broad spectrum of tasks and techniques that lead to an understanding of requirements is called requirements engineering. It starts during the communication activity and continues into the modeling activity. Requirements engineering provides the appropriate mechanism for understanding what the customer wants by analyzing need, assessing feasibility negotiating a reasonable solution, specifying the solution ambiguously, validating the specification, and managing the requirements as they are transformed into an operational system. It encompasses seven distinct tasks: inception, elicitation, elaboration, negotiation, specification, validation, and management. **Inception:** The question why you want to do this will be answered and analyses to identify business need, a potential new market with breadth and depth and services to be provided. The above points establish a basic understanding of the problem, the people who want a solution, the nature of the solution that is desired to understand the scope of the project. **Elicitation:** This answers for what are things need to do by asking the customer, the users, and others what the objectives for the system or product are, what is to be accomplished, how the system or product fits into the needs of the business, and finally, how the system or product is to be used on a day-to-day basis **Elaboration:** The information obtained from the customer during inception and elicitation is expanded and refined during elaboration. This	Correct/relevant explanation for each function-1M

task focuses on developing a refined requirements model that identifies requirements for three domains, information, functional and behavioral domain. It

- Describe how the end user (and other actors) will interact with the system.
- Business domain entities that is visible to the end user.
- The attributes of each analysis class are defined, and the services that are required by each class are identified.
- The relationships and collaboration between classes are identified, and a variety of supplementary diagrams are produced.

Negotiation: It answers for is it actually required? Through which Customers, users, and other stakeholders are asked to rank requirements and prioritized the same. Using an iterative approach that prioritizes requirements, assesses their cost and risk, and addresses internal conflicts, requirements are eliminated, combined, and/or modified so that each party achieves some measure of satisfaction.

Specification: A specification can be a written document, a set of graphical models, a formal mathematical model, a collection of usage scenarios, a prototype, or any combination of these to present gathered requirements. The formality and format of a specification varies with the size and the complexity of the software to be built.

Validation: As a part of this task documented software requirement specification will be examining by conducting technical reviews in order to examine errors in content or interpretation, areas where clarification may be required, missing information, inconsistencies (a major problem when large products or systems are engineered), conflicting requirements, or unrealistic (unachievable) requirements.

Requirements management: Requirements management is a set of activities that help the project team identify, control, and track requirements and changes to requirements at any time as the project proceeds.

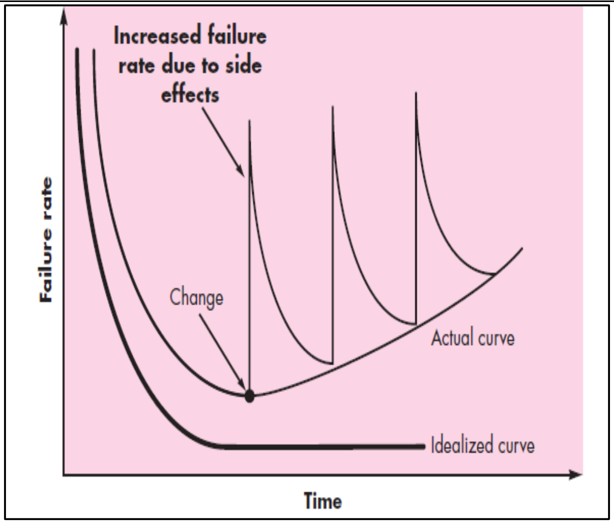
SUMMER – 19 EXAMINATION

Subject Name: Software Engineering **Model Answer** **Subject Code: 22413**

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 judgement 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.

Q. No.	Sub Q. N.	Answer	Marking Scheme
1		**Attempt any Five of the following:**	**10 M**
	a	**Enlist and explain software characteristics (any two).**	**2 M**
	Ans	1. Software is developed or engineered; it is not manufactured in the classical sense. ▪ Although some similarities exist between software development and hardware manufacture, the two activities are fundamentally different. ▪ In both activities, high quality is achieved through good design, but the manufacturing phase for hardware can introduce quality problems that are non-existent (or easily corrected) for software. ▪ Both activities are dependent on people, but the relationship between people applied and work accomplished is entirely different. ▪ Software costs are concentrated in engineering. This means that software projects cannot be managed as if they were manufacturing projects 2. Software doesn't "wear out."	Each Characteristics with explanation – 1M



- The idealized curve as shown in above figure is a gross oversimplification of actual failure models for software. However, the implication is clear—software doesn't wear out. But it does deteriorate!
- This contradiction can best be explained by considering the “actual curve” shown in Figure.
- During its life, software will undergo change (maintenance). As changes are made, it is likely that some new defects will be introduced, causing the failure rate curve to spike as shown in Figure.
- Before the curve can return to the original steady-state failure rate, another change is requested, causing the curve to spike again. Slowly, the minimum failure rate level begins to rise—the software is deteriorating due to change.

3. Although the industry is moving toward component-based construction, most software continues to be custom built.

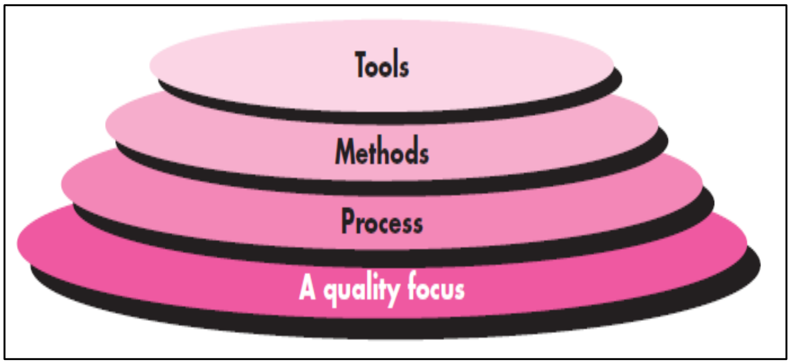
- The reusable components have been created so that the engineer can concentrate on the truly innovative elements of a design, that is, the parts of the design that represent something new.
- In the software world, it is something that has only begun to be achieved on a broad scale. A software component should be designed and implemented so that it can be reused in many different programs
- A software component should be designed and implemented so that it can be reused in many different programs. Modern

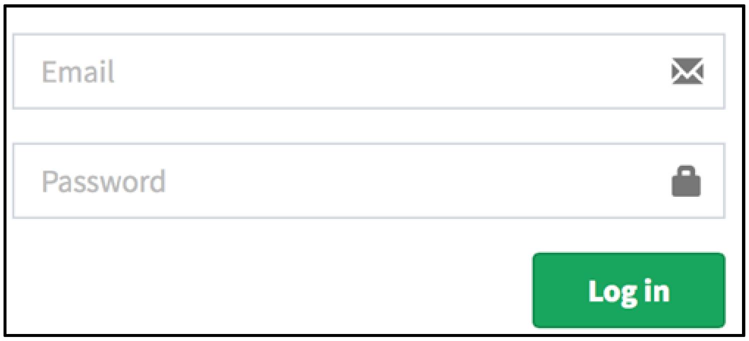
		reusable components encapsulate both data and the processing that is applied to the data, enabling the software engineer to create new applications from reusable parts. ▪ For example, today's interactive user interfaces are built with reusable components that enable the creation of graphics windows, pull-down menus, and a wide variety of interaction mechanisms.	
	b	**Define software on engineering.**	**2 M**
	Ans	Software engineering is the establishment and use of sound engineering principles in order to obtain economically software that is reliable and works efficiently on real machines.	Correct Definition-2M
	c	**State need of software requirement specification (SRS).**	**2 M**
	Ans	The need of SRS document is to provide • A detailed overview of software product, its parameters and goals. • The description regarding the project's target audience and its user interface hardware and software requirements. • How client, team and audience see the product and its functionality.	Any two points stating need of SRS-2M
	d	**Define Reactive Risk strategies.**	**2 M**
	Ans	A reactive risk strategy monitors the project for likely risks. Resources are set aside to deal with them, should they become actual problems. More commonly, the software team does nothing about risks until something goes wrong. Then, the team flies into action in an attempt to correct the problem rapidly. This is often called a fire-fighting mode. When this fails, "crisis management" takes over and the project is in real jeopardy.	Correct Definition-2M
	e	**Specify following cost directives of cocomo:** • **Product attributes (any two)** • **Hardware attributes (any two).**	**2 M**
	Ans	**Product attributes –** • Required software reliability extent • Size of the application database • The complexity of the product **Hardware attributes –**	Product attributes (any two)-1M, Hardware

<table>
<tr><td></td><td></td><td>• Run-time performance constraints
• Memory constraints
• The volatility of the virtual machine environment
• Required turnabout time</td><td>attributes (any two)-1M</td></tr>
<tr><td></td><td><b>f</b></td><td><b>Differentiate between Software Quality Management and Software Quality Assurance (any two points).</b></td><td><b>2 M</b></td></tr>
<tr><td></td><td><b>Ans</b></td><td>
<table>
<tr><th>Software Quality Assurance (QA)</th><th>Software Quality Control (QC)</th></tr>
<tr><td>• It is a procedure that focuses on providing assurance that quality requested will be achieved</td><td>• It is a procedure that focuses on fulfilling the quality requested.</td></tr>
<tr><td>• QA aims to prevent the defect</td><td>• QC aims to identify and fix defects</td></tr>
<tr><td>• It is a method to manage the quality- Verification</td><td>• It is a method to verify the quality-Validation</td></tr>
<tr><td>• It does not involve executing the program</td><td>• It always involves executing a program</td></tr>
<tr><td>• It's a Preventive technique</td><td>• It's a Corrective technique</td></tr>
<tr><td>• It's a Proactive measure</td><td>• It's a Reactive measure</td></tr>
<tr><td>• It is the procedure to create the deliverables</td><td>• It is the procedure to verify that deliverables</td></tr>
<tr><td>• QA involves in full software development life cycle</td><td>• QC involves in full software testing life cycle</td></tr>
<tr><td>• In order to meet the customer requirements,</td><td>• QC confirms that the standards are followed</td></tr>
</table>
</td><td>Each correct differentiation points- 1M</td></tr>
</table>

QA defines standards and methodologies	while working on the product
- It is performed before Quality Control	- It is performed only after QA activity is done
- It is a Low-Level Activity, it can identify an error and mistakes which QC cannot	- It is a High-Level Activity, it can identify an error that QA cannot
- Its main motive is to prevent defects in the system. It is a less time-consuming activity	- Its main motive is to identify defects or bugs in the system. It is a more time-consuming activity
- QA ensures that everything is executed in the right way, and that is why it falls under verification activity	- QC ensures that whatever we have done is as per the requirement, and that is why it falls under validation activity
- It requires the involvement of the whole team	- It requires the involvement of the Testing team
- The statistical technique applied on QA is known as SPC or Statistical Process Control (SPC)	- The statistical technique applied to QC is known as SQC or Statistical Quality Control

	g	**Define Software Quality Assurance.**	**2 M**
	Ans	• Quality assurance consists of the auditing and reporting functions of management. • The goal of quality assurance is to provide management with the data necessary to be informed about product quality, thereby gaining insight and confidence that product quality is meeting its goals.	Correct Definition-2M

2.		**Attempt any THREE of the following:**	**12M**
	a	**Explain Software Engineering as layered technology approach.**	**4 M**
	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:**	Correct Diagram -1M, explanation - 3M

		Software Engineering methods provide the technical —how to 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.	
	b	**Explain with example Decision table**	**4 M**
	Ans	• Decision table is a software testing technique used to test system behaviour for different input combinations. • This is a systematic approach where the different input combinations and their corresponding system behaviour (Output) are captured in a tabular form. That is why it is also called as a **Cause-Effect** table where Cause and effects are captured for better test coverage. • **Example 1: Decision Base Table for Login Screen**  • The condition is simple if the user provides correct username and password the user will be redirected to the homepage. If any of the input is wrong, an error message will be displayed.	Explanation-2 M, Example of Decision table- 2 M

Decision Table

Conditions	Rule 1	Rule2	Rule3	Rule 4
Username(T/F)	F	T	F	T
Password(T/F)	F	F	T	T
Output(E/H)	E	E	E	H

- **Legend:**

  T – Correct username/password

  F – Wrong username/password

  E – Error message is displayed

  H – Home screen is displayed

- **Interpretation:**
  - Case 1 – Username and password both were wrong. The user is shown an error message.
  - Case 2 – Username was correct, but the password was wrong. The user is shown an error message.
  - Case 3 – Username was wrong, but the password was correct. The user is shown an error message.
  - Case 4 – Username and password both were correct, and the user navigated to homepage.

c **Explain following elements of management spectrum:** **4 M**

i. **People**
ii. **Process**
iii. **Product**
iv. **Project**

Ans

The management Spectrum: 4p's

Effective software project management focuses on the four P's: people, product, process, and project.

The People:

Explanation each element of management spectrum – 1M

1. The “people factor” is so important that the Software Engineering Institute has developed a People Capability Maturity Model (People-CMM) to continually improve its ability to attract, develop, motivate, organize, and retain the workforce needed to accomplish its strategic business objectives.
2. The people capability maturity model defines the following key practice areas for software people:

a. Staffing
b. communication and coordination
c. work environment
d. performance management
e. Training, compensation, competency analysis and development, career development, workgroup development, team/culture development and others.

3. Organizations that achieve high levels of People-CMM maturity have higher likelihood of implementing effective software project management practices.

The Product:

1. Before a project can be planned, product objectives and scope should be established, alternative solutions should be considered and technical and management constraints should be identified.
2. Without this information, it is impossible to define reasonable (and accurate) estimates of the cost, an effective assessment of risk, a realistic breakdown of project tasks, or a manageable project schedule that provides a meaningful indication of progress.
3. Objectives identify the overall goals for the product (from the stakeholders’ points of view) without considering how these goals will be achieved.
4. Scope identifies the primary data, functions, and behaviors that characterize the product
5. The alternatives enable managers and practitioners to select a “best” approach, given the constraints imposed by delivery deadlines, budgetary restrictions, personnel availability, technical interfaces, and other factors.

The Process:

		1. A software process provides the framework from which a comprehensive plan for software development can be established. 2. A small number of framework activities are applicable to all software projects, regardless of their size or complexity. 3. A number of different task sets—tasks, milestones, work products, and quality assurance points enable the framework activities to be adapted to the characteristics of the software project and the requirements of the project team. 4. Finally, umbrella activities—such as software quality assurance, software configuration management, and measurement occur throughout the process. **The Project:** 1. To manage complexity, we conduct planned and controlled software projects. 2. The success rate for present-day software projects may have improved but our project failure rate remains much higher than it should be. 3. To avoid project failure, a software project manager and the software engineers who build the product must avoid a set of common warning signs, understand the critical success factors that lead to good project management, and develop a common-sense approach for planning, monitoring, and controlling the project.	
	d	**List and explain basic principles of project scheduling.**	**4 M**
	Ans	**Basic Principles** • **Compartmentalization:** The project must be compartmentalized into a number of manageable activities and tasks. • **Interdependency:** The interdependency of each compartmentalized activity or task must be determined. • **Time allocation:** Each task to be scheduled must be allocated some number of work units. • **Effort validation:** Every project has a defined number of staff members. • **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.	Correct listing – 2M, explanation – 2M

		• **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.		**Attempt any THREE of the following:**	**12 M**
	a	**Prescriptive process model and agile process model.**	**4 M**
	Ans	(see table below)	1 M for each Difference ,Any Four Difference

Prescriptive process model	**agile process mode**
Prescriptive process models stress detailed definition, identification, and application of process activates and tasks.	Agile process models emphasize project “agility” and follow a set of principles that lead to a more informal approach to software process.
A prescriptive model also describes how each of these elements are related to one another.	Agile methods note that not only do the software requirements change, but so do team members, the technology being used.
It is Process oriented.	It is people oriented.
It follows Life cycle model (waterfall, spiral) development model.	It follows Iterative and Incremental development model.
Documentation required is to be comprehensive and constant.	Documentation required is to be minimal and evolving.
Predictive planning is required	Adaptive planning is required.
Customers role is important.	Customers role is critical.
Formal communication is required.	Informal communication is required.
To maintain quality heavy planning and strict control with late heavy testing is required.	To maintain quality continuous control of requirements and

		development with continuous testing is required.	
	b	**Describe any four principles of communication for software engineering :**	**4 M**
	Ans	**Principle 1 Listen**: • Try to focus on the speaker‘s words, rather than formulating your response to those words. • Ask for clarification if something is unclear, but avoid constant interruptions. • Never become contentious in your words or actions (e.g., rolling your eyes or shaking your head) as a person is talking. **Principle 2 Prepare before you communicate:** • Spend the time to understand the problem before you meet with others. If necessary, perform some research to understand business domain. • If you have responsibility for conducting a meeting, prepare an agenda in advance of the meeting. **Principle 3 someone should facilitate the activity:** • Every communication meeting should have a leader (a facilitator) • To keep the conversation moving in a productive direction, • To mediate any conflict that does occur, and • To ensure that other principles are followed. **Principle 4 Face-to-face communication is best:** • It usually works better when some other representation of the relevant information is present. • For example, a participant may create a drawing /document that serve as a focus for discussion. **Principle 5 Take notes and document decisions:**	1M for one principle, Any four princple

- Someone participating in the communication should serve as a recorder and write down all important points and decisions.

Principle 6 Strive for collaboration:

- Collaboration occurs when the collective knowledge of members of the team is used to describe product or system functions or features.
- Each small collaboration builds trust among team members and creates a common goal for the team.

Principle 7 Stay focused; modularize your discussion:

- The more people involved in any communication, the more likely that discussion will bounce from one topic to the next.
- The facilitator should keep the conversation modular; leaving one topic only after it has been resolved.

Principle 8 If something is unclear, draw a picture:

- Verbal communication goes only so far.
- A sketch or drawing can often provide clarity when words fail to do the job.

Principle 9

(a) Once you agree to something, move on.

(b) If you can’t agree to something, move on.

(c) If a feature or function is unclear and cannot be clarified at the moment,

move on.

<table>
<tr>
<td></td>
<td></td>
<td>

- The people who participate in communication should recognize that many topics require discussion and that moving on is sometimes the best way to achieve communication agility.

Principle 10 Negotiation is not a contest or a game: It works best when both parties win.

- There are many instances in which you and other stakeholders must negotiate functions and features, priorities, and delivery dates.
- If the team has collaborated well, all parties have a common goal. Still, negotiation will demand compromise from all parties.

</td>
<td></td>
</tr>
<tr>
<td></td>
<td>c</td>
<td>

Draw proper labelled "LEVEL 1 Data Flow Diagram" (DFD) for student attendance system

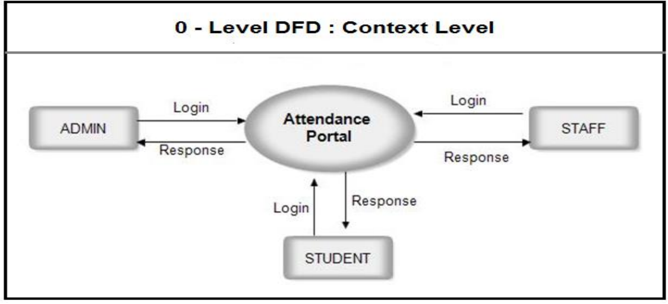
</td>
<td>

4 M

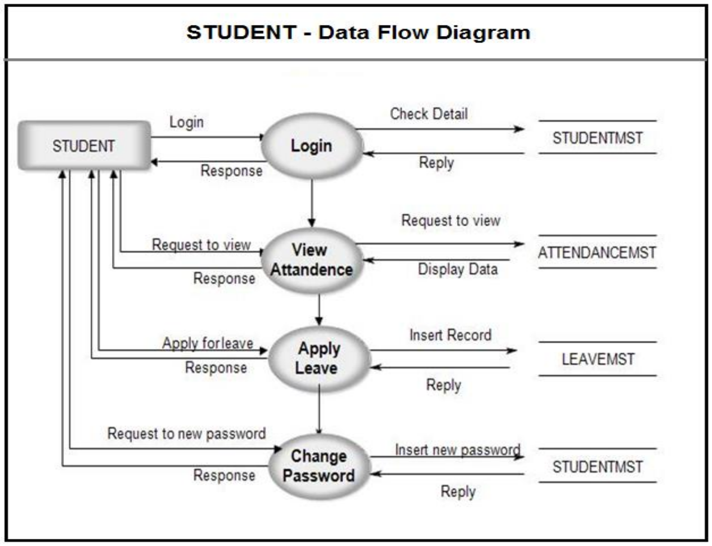
</td>
</tr>
<tr>
<td></td>
<td>

Ans

</td>
<td>



Level 0 Context Level

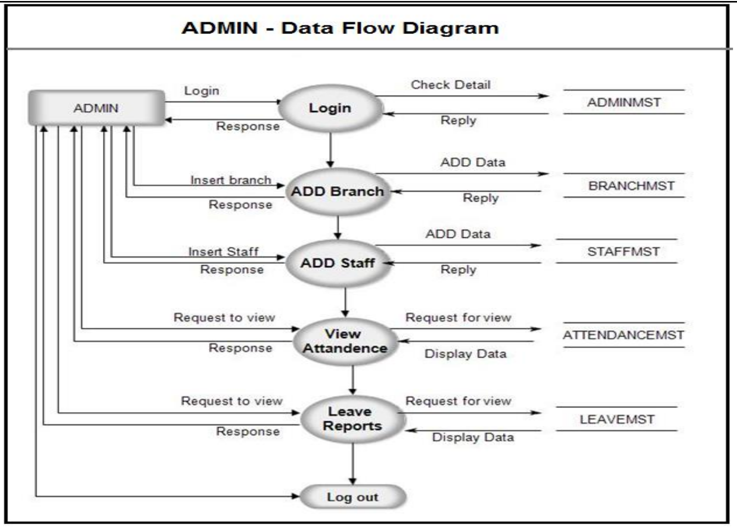


Level 1 DFD student

</td>
<td>

1 M for level 0 and 3 M for level 1 DFD

</td>
</tr>
</table>

		 ADMIN - Data Flow Diagram ADMIN Login Response Login Check Detail Reply ADMINMST Insert branch Response ADD Branch ADD Data Reply BRANCHMST Insert Staff Response ADD Staff ADD Data Reply STAFFMST Request to view Response View Attandence Request for view Display Data ATTENDANCEMST Request to view Response Leave Reports Request for view Display Data LEAVEMST Log out **Level 1 for admin**	
	d	**State importance of “Function point “ and “lines of code” in concerned with project estimation**	**4 M**
	Ans	Currently two metrics are popularly being used widely to estimate size: lines of code (LOC) and function point (FP). **Lines of Code (LOC)** LOC is the simplest among all metrics available to estimate project size. This metric is very popular because it is the simplest to use. Using this metric, the project size is estimated by counting the number of source instructions in the developed program. Obviously, while counting the number of source instructions, lines used for commenting the code and the header lines should be ignored. **Function Point (FP):** The conceptual idea behind the function point metric is that the size of a software product is directly dependent on the number of different Functions or features it supports. A software product supporting many features would certainly be of larger size than a product with less number	2 M for function point and 2 M for lines of code

		of features. Each function when invoked reads some input data and transforms it to the corresponding output data. For example, the issue book feature (as shown in figure) of a Library Automation Software takes the name of the book as input and displays its location and the number of copies available. Thus, a computation of the number of input and the output data values to a system gives some indication of the number of functions supported by the system. Albrecht postulated that in addition to the number of basic functions that a software performs, the size is also dependent on the number of files and the number of interfaces.	
4.		**Attempt any THREE of the following:**	**12 M**
	a	**Describe Extreme programming with proper diagram**	**4 M**
	Ans	Extreme programming is a lightweight, efficient, low-risk, flexible, predictable, scientific, and fun way to develop a software. eXtreme Programming (XP) was conceived and developed to address the specific needs of software development by small teams in the face of vague and changing requirements. Extreme Programming is one of the Agile software development methodologies. It provides values and principles to guide the team behavior. The team is expected to self-organize. Extreme Programming provides specific core practices where- • Each practice is simple and self-complete. • Combination of practices produces more complex and emergent behavior. Extreme Programming is based on the following values- • Communication • Simplicity • Feedback • Courage • Respect	1 M for Diagram and 3 M for explanation

Extreme Programming involves-

- Writing unit tests before programming and keeping all of the tests running at all times. The unit tests are automated and eliminates defects early, thus reducing the costs.

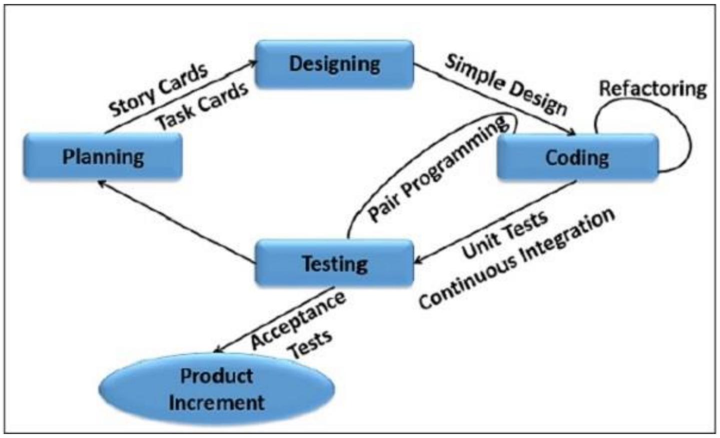
- Starting with a simple design just enough to code the features at hand and redesigning when required.

- Programming in pairs (called pair programming), with two programmers at one screen, taking turns to use the keyboard. While one of them is at the keyboard, the other constantly reviews and provides inputs.

- Integrating and testing the whole system several times a day.

- Putting a minimal working system into the production quickly and upgrading it whenever required.

- Keeping the customer involved all the time and obtaining constant feedback. Iterating facilitates the accommodating changes as the software evolves with the changing requirements.



Extreme Programming solves the following problems often faced in the software development projects-

- Slipped schedules: Short and achievable development cycles ensure timely deliveries.

		• Cancelled projects: Focus on continuous customer involvement ensures transparency with the customer and immediate resolution of any issues. • Costs incurred in changes: Extensive and ongoing testing makes sure the changes do not break the existing functionality. A running working system always ensures sufficient time for accommodating changes such that the current operations are not affected. • Production and post-delivery defects: Emphasis is on the unit tests to detect and fix the defects early. • Misunderstanding the business and/or domain: Making the customer a part of the team ensures constant communication and clarifications. • Business changes: Changes are considered to be inevitable and are accommodated at any point of time. • Staff turnover: Intensive team collaboration ensures enthusiasm and good will. Cohesion of multi-disciplines fosters the team spirit Extreme Programming takes the effective principles and practices to extreme levels. Extreme Programming • Code reviews are effective as the code is reviewed all the time. • Testing is effective as there is continuous regression and testing. • Design is effective as everybody needs to do refactoring daily. • Integration testing is important as integrate and test several times a day. • Short iterations are effective as the planning game for release planning and iteration planning.	
	b	**List and explain any four principles of “Core Principles” of Software Engineering.**	**4 M**
	Ans	**The First Principle: The Reason It All Exists**	1 M for one principle and explanation

- A software system exists for one reason: to provide value to its users. All decisions should be made with this in mind.
- Before specifying a system requirement, system functionality, before determining the hardware platforms, first determine, whether it adds value to the system.

The Second Principle: KISS (Keep It Simple, Stupid!)

- All design should be as simple as possible, but no simpler. This facilitates having a more easily understood and easily maintained system.
- It doesn't mean that features should be discarded in the name of simplicity.
- Simple also does not mean "quick and dirty." In fact, it often takes a lot of thought and work over multiple iterations to simplify.

The Third Principle: Maintain the Vision

- A clear vision is essential to the success of a software project.
- If you make compromise in the architectural vision of a software system, it will weaken and will eventually break even the well-designed systems.
- Having a powerful architect who can hold the vision helps to ensure a very successful software project.

The Fourth Principle: What You Produce, Others Will Consume

- Always specify, design, and implement by keeping in mind that someone else will have to understand what you are doing.
- The audience for any product of software development is potentially large.
- Design (make design), keeping the implementers (programmers) in mind. Code (program) with concern for those who will maintain and extend the system.
- Someone may have to debug the code you write, and that makes them a user of your code.

The Fifth Principle: Be Open to the Future

- A system with a long lifetime has more value.
- True "industrial-strength" software systems must last for longer.

		• To do this successfully, these systems must be ready to adapt changes. • Always ask "what if," and prepare for all possible answers by creating systems that solve the general problem. **The Sixth Principle: Plan Ahead for Reuse** • Reuse saves time and effort. • The reuse of code and designs has a major benefit of using object-oriented technologies. • Planning ahead for reuse reduces the cost and increases the value of both the reusable components and the systems into which they are incorporated. **The Seventh principle: Think!** • Placing clear, complete thought before action almost always produces better results. • When you think about something, you are more likely to do it right. You also gain knowledge about how to do it right again. • If you do think about something and still do it wrong, it becomes a valuable experience. • Applying the first six principles requires intense thought, for which the potential rewards are enormous.	
	c	**Explain RMMM plan with example .**	**4 M**
	Ans	A risk management plan or plan risk management is a document that a prepares to foresee risks, estimate impacts, and define responses to risks. It also contains a risk matrix. A risk is "an uncertain event or condition that, if it occurs, has a positive or negative effect on a project's objectives." Risk is inherent with any and project manager should assess risks continually and develop plans to address them. The risk management plan contains an analysis of likely risks with both high and low impact, as well as mitigation strategies to help the project avoid being derailed should common problems arise. Risk management plans should be periodically reviewed by the project team to avoid having the analysis become stale and not reflective of actual potential project risks. Most critically, risk management plans include a risk strategy. There are two characteristics of risk i.e. uncertainty and loss. Risk Mitigation, Monitoring and Management (RMMM)	1 M for introduction to risk and 3 M for RMMM plan example

Risk analysis support the project team in constructing a strategy to deal with risks.

There are three important issues considered in developing an effective strategy:

Risk avoidance or mitigation - It is the primary strategy which is fulfilled through a plan.

Risk monitoring - The project manager monitors the factors and gives an indication whether the risk is becoming more or less.

Risk management and planning - It assumes that the mitigation effort failed and the risk is a reality.

RMMM PlanIt is a part of the software development plan or a separate document.

The RMMM plan documents all work executed as a part of risk analysis and used by the project manager as a part of the overall project plan.

The risk mitigation and monitoring starts after the project is started and the documentation of RMMM is completed.

Risk :Computer Crash

Mitigation :

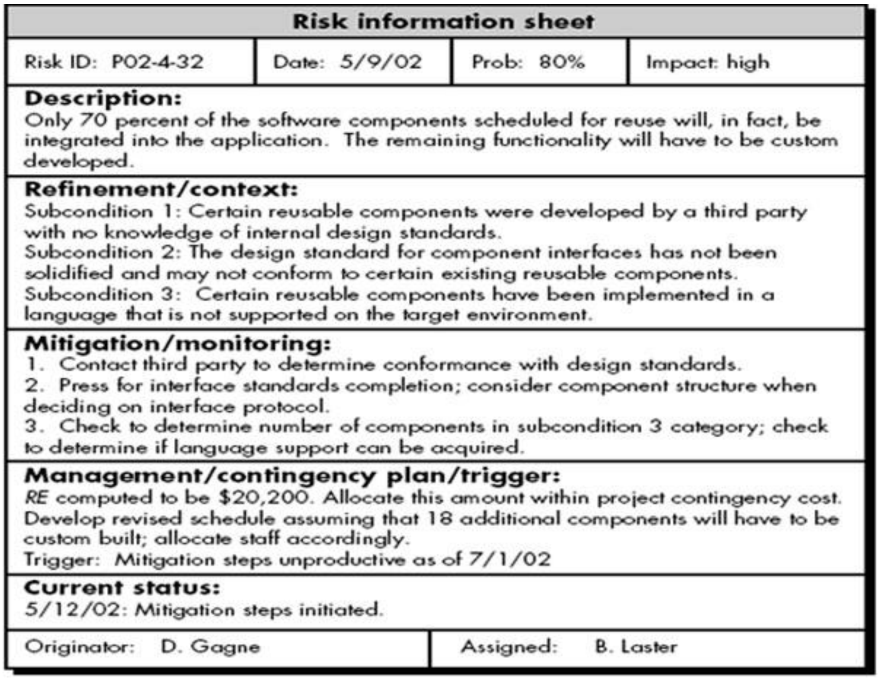
The cost associated with a computer crash resulting in a loss of data is crucial. A computer crash itself is not crucial, but rather the loss of data. A loss of data will result in not being able to deliver the product to the customer. This will result in a not receiving a letter of acceptance from the customer. Without the letter of acceptance, the group will receive a failing grade for the course. As a result the organization is taking steps to make multiple backup copies of the software in development and all documentation associated with it, in multiple locations. ·

Monitoring :

When working on the product or documentation, the staff member should always be aware of the stability of the computing environment they're working in. Any changes in the stability of the environment should be recognized and taken seriously. ·

Management :

The lack of a stable-computing environment is extremely hazardous to a software development team. In the event that the computing environment is found unstable, the development team should cease work on that system until the environment is made stable again, or should move to a system that is stable and continue working there.



Risk information sheet

Risk ID: P02-4-32	Date: 5/9/02	Prob: 80%	Impact: high

Description:
Only 70 percent of the software components scheduled for reuse will, in fact, be integrated into the application. The remaining functionality will have to be custom developed.

Refinement/context:
Subcondition 1: Certain reusable components were developed by a third party with no knowledge of internal design standards.
Subcondition 2: The design standard for component interfaces has not been solidified and may not conform to certain existing reusable components.
Subcondition 3: Certain reusable components have been implemented in a language that is not supported on the target environment.

Mitigation/monitoring:
1. Contact third party to determine conformance with design standards.
2. Press for interface standards completion; consider component structure when deciding on interface protocol.
3. Check to determine number of components in subcondition 3 category; check to determine if language support can be acquired.

Management/contingency plan/trigger:
RE computed to be $20,200. Allocate this amount within project contingency cost. Develop revised schedule assuming that 18 additional components will have to be custom built; allocate staff accordingly.
Trigger: Mitigation steps unproductive as of 7/1/02

Current status:
5/12/02: Mitigation steps initiated.

Originator: D. Gagne	Assigned: B. Laster

	d	**Explain any one project cost estimation approach.**	**4 M**
	Ans	**(i) Heuristic** Heuristic techniques assume that the relationships among the different project parameters can be modeled using suitable mathematical expressions. Once the basic (independent) parameters are known, the other (dependent) parameters can be easily determined by substituting the value of the basic parameters in the mathematical expression. Different heuristic estimation models can be divided into the following	Any one approach - Explanation 4 M

two classes: single variable model and the multi variable model.

Single variable estimation models provide a means to estimate the desired characteristics of a problem, using some previously estimated basic (independent) characteristic of the software product such as its size. A single variable estimation model takes the following form:

Estimated Parameter = $c_1 * e_1^{d_1}$

In the above expression, e is the characteristic of the software which has already been estimated (independent variable). Estimated Parameter is the dependent parameter to be estimated. The dependent parameter to be estimated could be effort, project duration, staff size, etc. c1 and d1 are constants. The values of the constants c1 and d1 are usually determined using data collected from past projects (historical data). The basic COCOMO model is an example of single variable cost estimation model.

A multivariable cost estimation model takes the following form:

Estimated Resource = $c_1 * e_1^{d_1} + c_2 * e_2^{d_2} + ...$

Where e1, e2, … are the basic (independent) characteristics of the software already estimated, and c1, c2, d1, d2, … are constants.

(ii) Analytical

Halstead’s Software Science – An Analytical Technique
Halstead’s software science is an analytical technique to measure size, development effort, and development cost of software products. Halstead used a few primitive program parameters to develop the expressions for over all program length, potential minimum value, actual volume, effort, and development time.

Example:

Let us consider the following C program:

```
main( )
{
      int a, b, c, avg;

      scanf("%d %d %d", &a, &b, &c);
      avg = (a+b+c)/3;
      printf("avg = %d", avg);
}
```

The unique operators are:
main,(),{},int,scanf,&,",",";",=,+,/, printf

The unique operands are:
a, b, c, &a, &b, &c, a+b+c, avg, 3, "%d %d %d", "avg = %d"

Therefore,
n1 = 12, n2 = 11

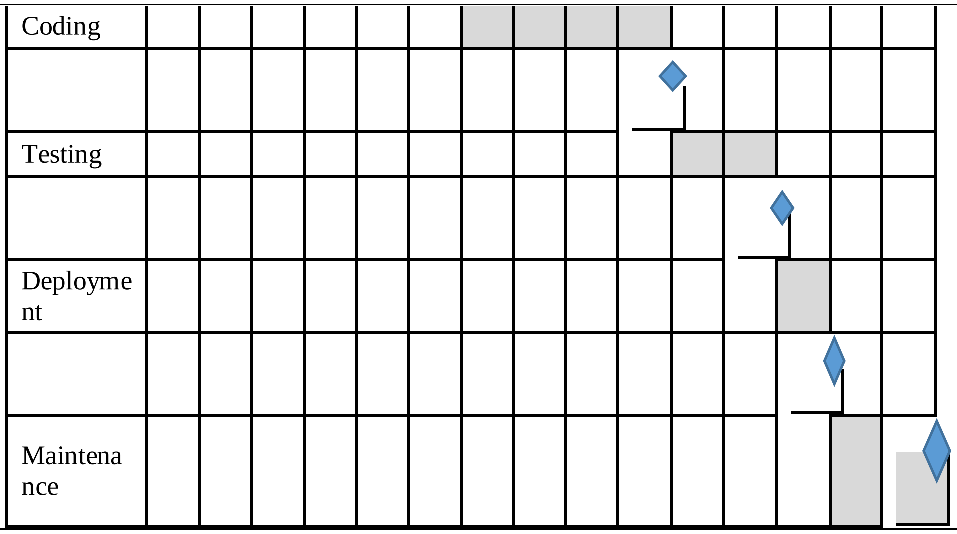
Estimated Length = (12*log12 + 11*log11)
= (12*3.58 + 11*3.45)
= (43+38) = 81

Volume = Length*log(23)
= 81*4.52
= 366

e **Draw time chart for Libraray management system System (5 days a week). Consider broad phases of SDLC.** 4 M

Ans

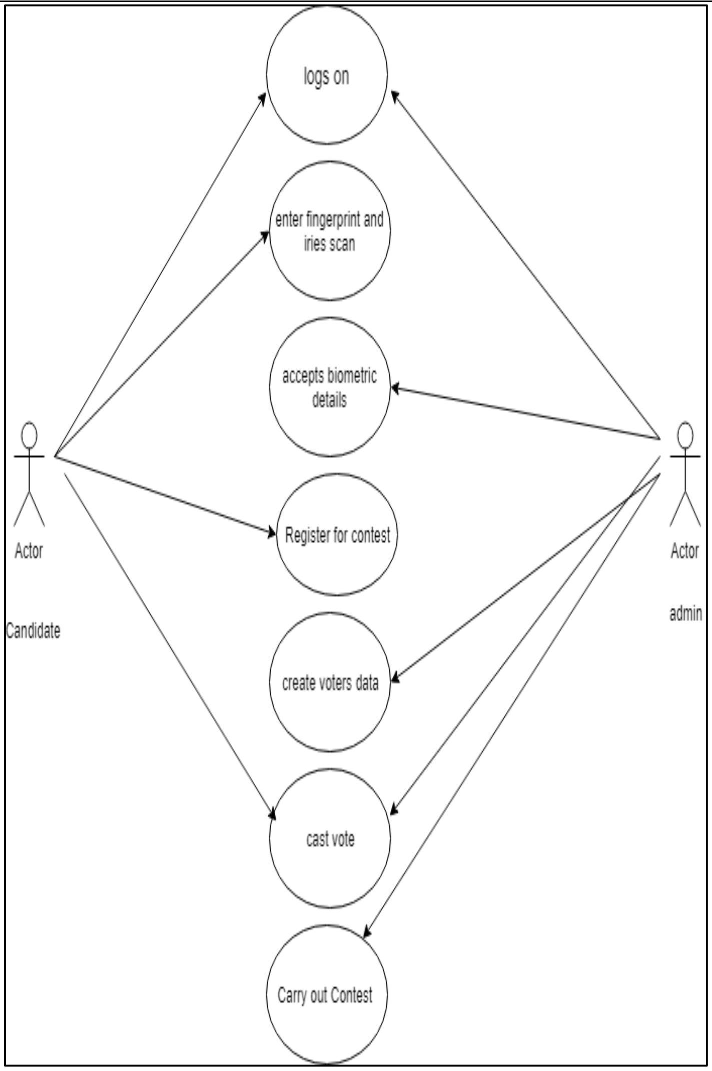
	Week 1					Week 2					Week 3				
	D1	D2	D3	D4	D5	D1	D2	D3	D4	D5	D1	D2	D3	D4	D5
Ananlysis	▨	▨	▨												
			◆												
Design				▨	▨	▨									
						◆									



5		**Attempt any TWO of the following:**	**12 M**
	a	**Enlist requirement Gathering and Analysis for web based project for registering candidates for contest**	**6 M**
	Ans	Requirement gathering includes suggestions and ideas for ways to best capture the different types of requirement (functional, system, technical, etc.) during the gathering process. **1. Functional requirements** The functional requirements are the requirements that will enable solving the real world problem. The web based project must be able to register the candidates for contest. **2. Non-functional requirements** These requirements aim at providing support, security and facilitate user interaction segment of the website. • The project must enable the candidates to safely enter their passwords and other biometric information. • There must be no repetition in registration of candidates i.e the candidates must be registered only once. 3. **Business requirements**: They are high-level requirements that are taken from the business case from the projects. For eg:-	6M – 1M for 1 point

Qualifying criteria	Allowed/Disallowed
Indian Nationality Registration	Allowed
Age>18	Allowed
No criminal record	Allowed

4. **Architectural and Design requirements**: These requirements are more detailed than business requirements. It determines the overall design required to implement the business requirement.
   - The web based project must be supported by different operating systems , PC and mobile compatibility etc.
   - The hardware must be integrated so as to accept the fingerprint details of a candidate and register him in the system.
   - The database of the project must be updated.
5. **System and Integration requirements**: At the lowest level, we have system and integration requirements. It is detailed description of each and every requirement. It can be in form of user stories which is really describing everyday business language. The requirements are in abundant details so that developers can begin coding.



6. **Documenting the requirement using traceability matrix**
   A Traceability Matrix is a document that co-relates any two-baseline documents that require a many-to-many relationship to check the completeness of the relationship.It is used to track the requirements and to check the current project requirements are met.

Req no	**Description**	**Test case ID**	**Status**	
1	Login	TC1	TC1 Pass	
2	Feed in biometric details	TC2	TC2 Pass	

	b	**Differentiate between White box and Black Box Testing.**	**6 M**

<table>
<tr><td></td><td><b>Ans</b></td><td>
<table>
<tr><th>Sr .no</th><th>White box testing</th><th>Black Box Testing</th></tr>
<tr><td>1</td><td>The tester needs to have the knowledge of internal code or program.</td><td>This technique is used to test the software without the knowledge of internal code or program.</td></tr>
<tr><td>2</td><td>It aims at testing the structure of the item being tested.</td><td>It aims at testing the functionality of the software.</td></tr>
<tr><td>3</td><td>It is also called structural testing, clear box testing, code-based testing, or glass box testing.</td><td>It also knowns as data-driven, box testing, data-, and functional testing.</td></tr>
<tr><td>4</td><td>Testing is best suited for a lower level of testing like Unit Testing, Integration testing.</td><td>This type of testing is ideal for higher levels of testing like System Testing, Acceptance testing.</td></tr>
<tr><td>5</td><td>Statement Coverage, Branch coverage, and Path coverage are White Box testing technique.</td><td>Equivalence partitioning, Boundary value analysis are Black Box testing technique</td></tr>
<tr><td>6</td><td>Can be based on detailed design documents.</td><td>Can be based on Requirement specification document.</td></tr>
</table>
</td><td>6M- 1M for 1point</td></tr>
<tr><td></td><td><b>c</b></td><td><b>Describe COCOMO II model for evaluating size of software project with any three parameters in detail</b></td><td><b>6 M</b></td></tr>
<tr><td></td><td><b>Ans</b></td><td>COCOMO-II is the revised version of the original Cocomo (Constructive Cost Model) and is developed at University of Southern California. It is the model that allows one to estimate the cost, effort and schedule when planning a new software development activity.</td><td>3M for Description, 3M for parameters</td></tr>
</table>

COCOMO II provides the following three-stage series of models for estimation of Application Generator, System Integration, and Infrastructure software projects:

End User Programming	Application Generators and composition aids	Infrastructure
	Application Composition	
	System Integration	

- The Application Composition Model

  This model involves prototyping efforts to resolve potential high-risk issues such as user interfaces, software/system interaction, performance, or technology maturity. The costs of this type of effort are best estimated by the Applications Composition model. It is suitable for projects built with modern GUI-builder tools. It is based on new Object Points.

- The Early Design Model

  The Early Design model involves exploration of alternative software/system architectures and concepts of operation. It uses a small set of new Cost Drivers, and new estimating equations. Based on Unadjusted Function Points or KSLOC.

- The Post-Architecture Model

The Post-Architecture model involves the actual development and maintenance of a software product

Estimates

In COCOMO II effort is expressed as Person Months (PM). The inputs are the Size of software development, a constant, A, and a scale factor, B. The size is in units of thousands of source lines of code (KSLOC). The constant, A, is used to capture the multiplicative effects on effort with projects of increasing size.

The parameters used in COCOMO II are described below:-

a. **Person month**- A person month is the amount of time one person spends working on the software development project for one month. The nominal effort for a given size project and expressed as person months (PM) is given by Equation 1.

$PM_{nominal} = A * (Size)^{B}$

Where

A- constant

$B = 0.91 + 0.01 \sum(\text{exponent driver ratings})$

- B ranges from 0.91 to 1.23

- 5 drivers; 6 rating levels each

b. **Maintenance size** is the amount of project code that is change. It is calculated as below:-

Size=[(BaseCodeSize) *MCF] *MAF

COCOMO II uses the reuse model for maintenance when the amount of added or changed base source code is less than or equal to 20% or the new code being developed. Base code is source code that already exists and is being changed for use in the current project. For maintenance projects that involve more than 20% change in the existing base code (relative to new code being developed) COCOMO II uses maintenance size.

c. Maintenance Change Factor MCF

The percentage of change to the base code is called the Maintenance Change Factor (MCF).

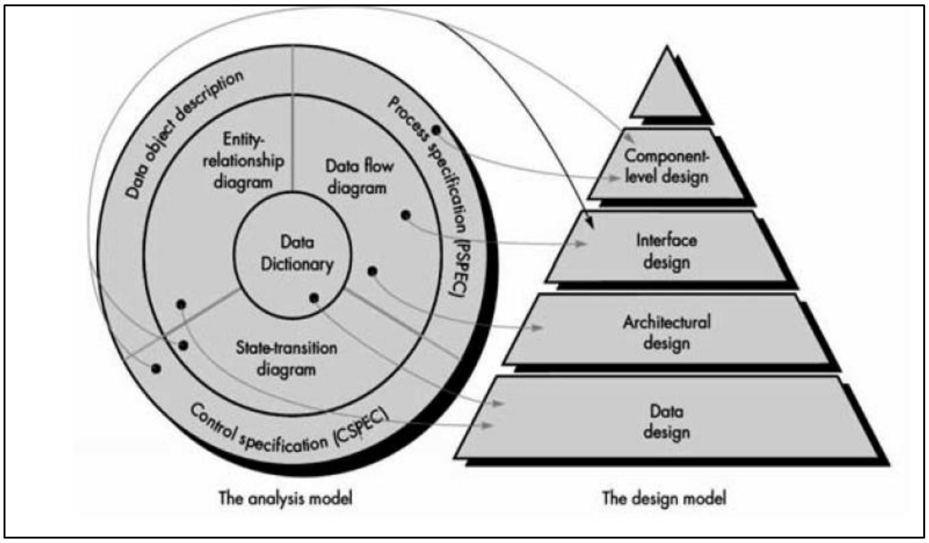
MCF= (SizeAdded +SizeModified)/BaseCodeSize

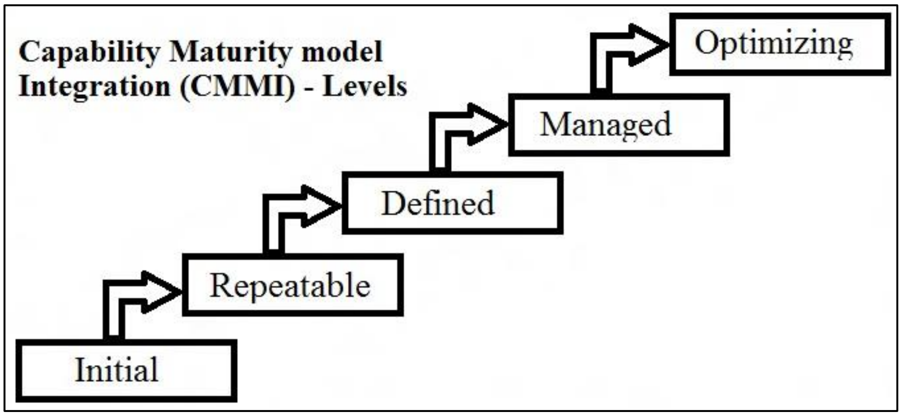
d. Maintenance effort (MAF)

COCOMO II instead used the Software Understanding (SU) and Programmer Unfamiliarity (UNFM) factors from its reuse model to model the effects of well or poorly structured/understandable software on maintenance effort.

MAF=1+ (SU.01*UNFM)

6		**Attempt any TWO of the following:**	**12 M**

	a	**Draw and explain Transition diagram from requirement model to design model**	**6 M**
	Ans	**Transition diagram from requirement model to design model**  Data object description Entity-relationship diagram Data flow diagram Process specification (PSPEC) Data Dictionary State-transition diagram Control specification (CSPEC) Component-level design Interface design Architectural design Data design The analysis model The design model Software requirements, manifested by the data, functional, and behavioural models, feed the design task. Using one of a number of design methods, the design task produces a data design, an architectural design, an interface design, and a component design. Each of the elements of the analysis model provides information that is necessary to create the four design models required for a complete specification of design. Design is a meaningful engineering representation of something that is to be built. It can be traced to a customer's requirements and at the same time assessed for quality against a set of predefined criteria for ―good‖ design. In the software engineering context, design focuses on four major areas of concern: data, architecture, interfaces, and components Design begins with the requirements model. The data design transforms the information domain model created during analysis into the data structures that will be required to implement the software. The data objects and relationships defined in the entity relationship diagram and the detailed data content depicted in the data dictionary provide the basis for the data design activity. Part of data design may occur in conjunction with the design of software architecture. More detailed data design occurs as each software component is designed. The architectural design defines the relationship between major structural elements of the software, the design pattern that can be used to achieve the requirements that have been defined for the system, and the constraints that affect the way in which architectural design patterns can be applied.	2M –diiagram, 4M – explanation

		The architectural design representation the framework of a computer-based system can be derived from the system specification, the analysis model, and the interaction of subsystems defined within the analysis model. The interface design describes how the software communicates within itself, with systems that interoperate with it, and with humans who use it. An interface implies a flow of information (e.g., data and/or control) and a specific type of behavior. Therefore, data and control flow diagrams provide much of the information required for interface design. The component-level design transforms structural elements of the software architecture into a procedural description of software components. Information obtained from the PSPEC, CSPEC, and STD serve as the basis for component design.	
	b	**Describe CMMI. Give significance of each level.**	**6 M**
	Ans	**The Capability Maturity Model Integration (CMMI)**, a comprehensive process meta-model that is predicated on a set of system and software engineering capabilities that should be present as organizations reach different levels of process capability and maturity. The CMMI represents a process meta-model in two different ways: (1) Continuous model and (2) Staged model. The continuous CMMI meta-model describes a process in two dimensions. Each process area (e.g. project planning or requirements management) is formally assessed against specific goals and practices and is rated according to the following capability levels:  Capability Maturity model Integration (CMMI) - Levels Optimizing Managed Defined Repeatable Initial **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.	1M- diagram , 5M- 5 points

		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 2 **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 3 **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.	
	c	**Identify and enlist requirement for given modules of employee management software**	**6 M**
	Ans	i. Employee detail ii. Employee salary iii.Employee performance This is with perspective of employee management software. Requirements for following modules will be as i. Employee details a. Getting information about the customer b. Updation of employee details (department, change of address, emp_code etc) c. Assignment of tasks , duties and responsibilities. d. Recording of employee attendance. ii. Employee salary a. Salary calculation	2 M for employee detail, salary, performance each

		b. Allowances, special bonus calculation and approval c. Tax statement/certificate d. Apply loan/approvals iii. Performance a. Recording annual performance b. Details about parameters for performance appraisal c. Analysis performance and determining hike in payment.	