22413

21819

3 Hours / 70 Marks

Seat No. □□□□□□□□

Instructions – (1) All Questions are *Compulsory.*

(2) Answer each next main Question on a new page.

(3) Illustrate your answers with neat sketches wherever necessary.

(4) Assume suitable data, if necessary.

(5) Use of Non-programmable Electronic Pocket Calculator is permissible.

(6) Mobile Phone, Pager and any other Electronic Communication devices are not permissible in Examination Hall.

(7) Preferably, write the answers in sequential order.

Marks

1. Attempt any FIVE of the following: 10

a) Enlist and explain software characteristics (any two).

b) Define software on engineering.

c) State need of software requirement specification (SRS).

d) Define Reactive Risk strategies.

e) Specify following cost directives of cocomo:

(i) Product attributes (any two)

(ii) Hardware attributes (any two).

f) Differentiate between Software Quality Management and Software Quality Assurance (any two points).

g) Define Software Quality Assurance.

Marks

2. Attempt any THREE of the following: 12

a) Explain Software Engineering as layered technology approach.

b) Explain with example Decision table.

c) Explain following elements of management spectrum:

(i) People

(ii) Process

(iii) Product

(iv) Project

d) List and explain basic principles of project scheduling.

3. Attempt any THREE of the following: 12

a) Distinguish between perspective process model and agile process model.

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

c) Draw proper labeled "LEVEL I Data Flow Diagram" (DFD) for student attendance system.

d) State importance of "Function Point (FP)" and "Lines of codes (LOC)" in concerned with project estimation.

4. Attempt any THREE of the following: 12

a) Describe extreme programming with proper diagram

b) List and explain any "four core principles" of software engineering.

c) Explain RMMM plan with example.

d) Explain any one project cost estimation approach.

e) Prepare time line chart for Library Managements System (five days a week) Consider phases of SDLC.

Marks

5. Attempt any TWO of the following: 12

a) Enlist Requirement Gathering and Analysis for web based project for registering candidates for contest (any six points).

b) Differentiate between White box and Black box testing (any six points).

c) Describe Co-Como II model for evaluating size of software project with any three parameters in detail.

6. Attempt any TWO of the following: 12

a) Draw and explain Transition diagram from requirement model to design model.

b) Describe CMMI. Give significance of each level.

c) Identify and enlist requirement for given modules of employee management software:

(i) Employee detail

(ii) Employee salary

(iii) Employee performance.

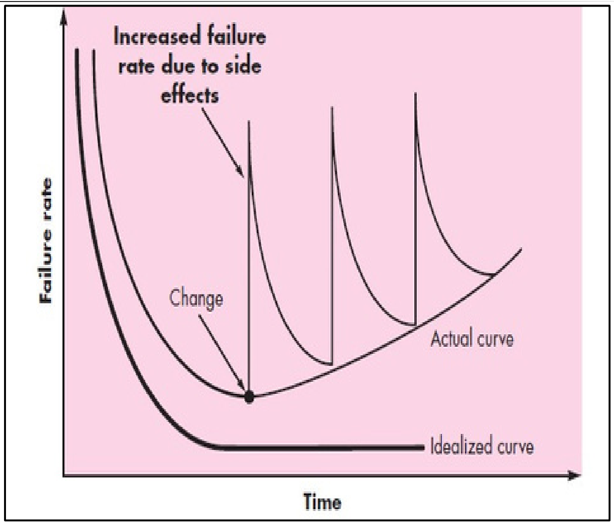
SUMMER – 19 EXAMINATION

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

		□ Run-time performance constraints □ Memory constraints □ The volatility of the virtual machine environment □ Required turnabout time	attributes (any two)-1M
	f	**Differentiate between Software Quality Management and Software Quality Assurance (any two points).**	**2 M**

Ans

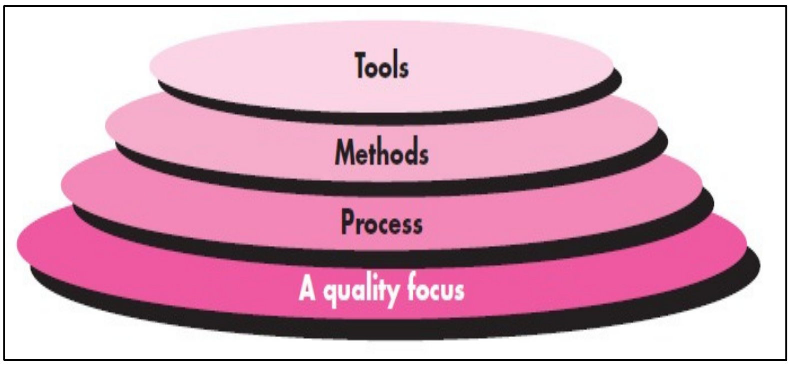
Software Quality Assurance (QA)	**Software Quality Control (QC)**
□ It is a procedure that focuses on providing assurance that quality requested will be achieved	□ It is a procedure that focuses on fulfilling the quality requested.
□ QA aims to prevent the defect	□ QC aims to identify and fix defects
□ It is a method to manage the quality- Verification	□ It is a method to verify the quality-Validation
□ It does not involve executing the program	□ It always involves executing a program
□ It's a Preventive technique	□ It's a Corrective technique
□ It's a Proactive measure	□ It's a Reactive measure
□ It is the procedure to create the deliverables	□ It is the procedure to verify that deliverables
□ QA involves in full software development life cycle	□ QC involves in full software testing life cycle
□ In order to meet the customer requirements,	□ QC confirms that the standards are followed

Each correct differentiation points- 1M

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:-	Correct Diagram -1M, explanation - 3M



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 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.

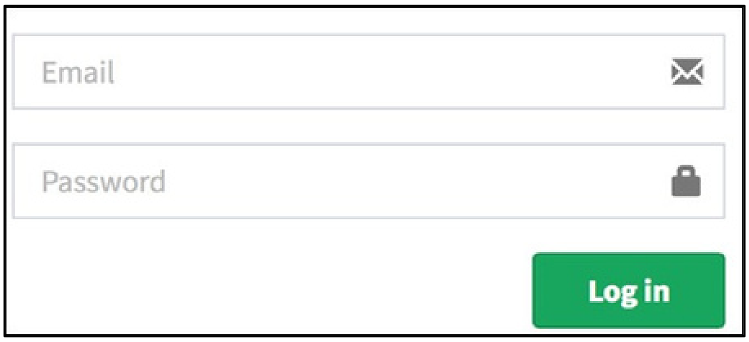
	b	**Explain with example Decision table**	**4 M**

Ans

Explanation-2 M, Example of Decision table- 2 M

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.

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		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** 1M for one principle, Any four princple
	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:**	

- 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.

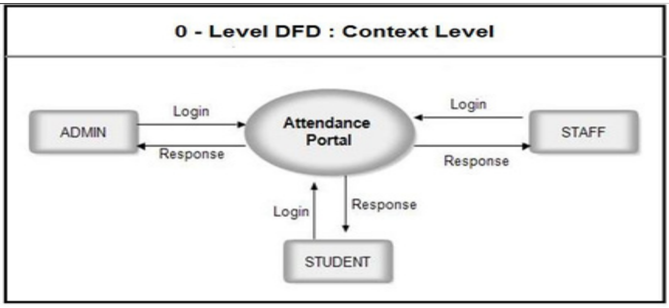
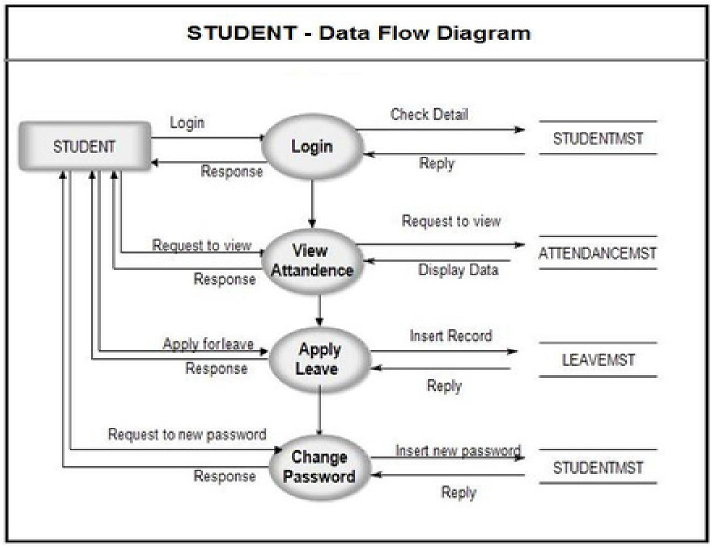
Principle 9

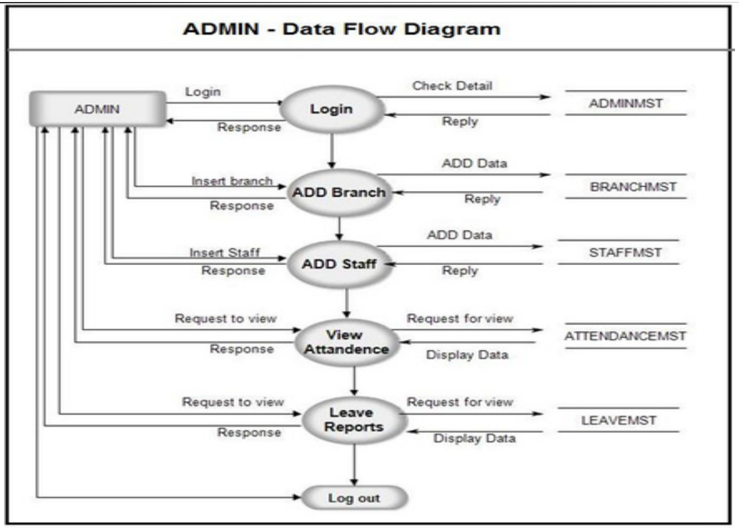
(a) Once you agree to something, move on.

(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 proper labelled “LEVEL 1 Data Flow Diagram” (DFD) for student attendance system**	**4 M** 1 M for level 0 and 3 M for level 1 DFD
	Ans	 0 - Level DFD : Context Level ADMIN Login Response Attendance Portal Login Response STAFF Login Response STUDENT **Level 0 Context Level**  STUDENT - Data Flow Diagram STUDENT Login Response Login Check Detail Reply STUDENTMST Request to view Response View Attandence Request to view Display Data ATTENDANCEMST Apply forleave Response Apply Leave Insert Record Reply LEAVEMST Request to new password Response Change Password Insert new password Reply STUDENTMST **Level 1 DFD student**	



Level 1 for admin

d **State importance of "Function point " and "lines of code" in concerned with project estimation**

4 M
2 M for function point and 2 M for lines of code

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.

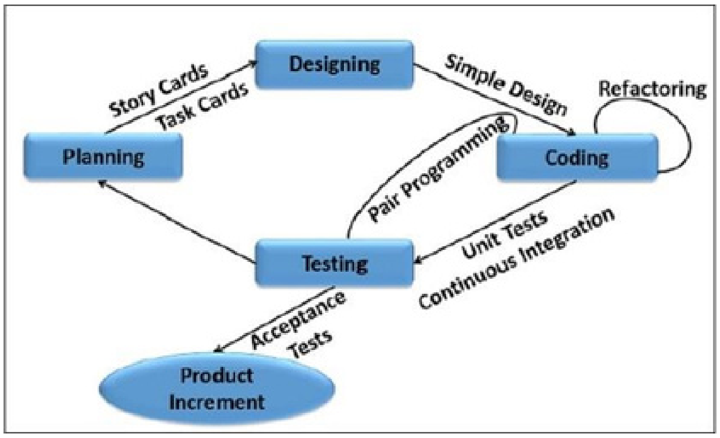
Function Point (FP):

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

		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		1 M for introduction to risk and 3 M for RMMM plan example

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)

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 :

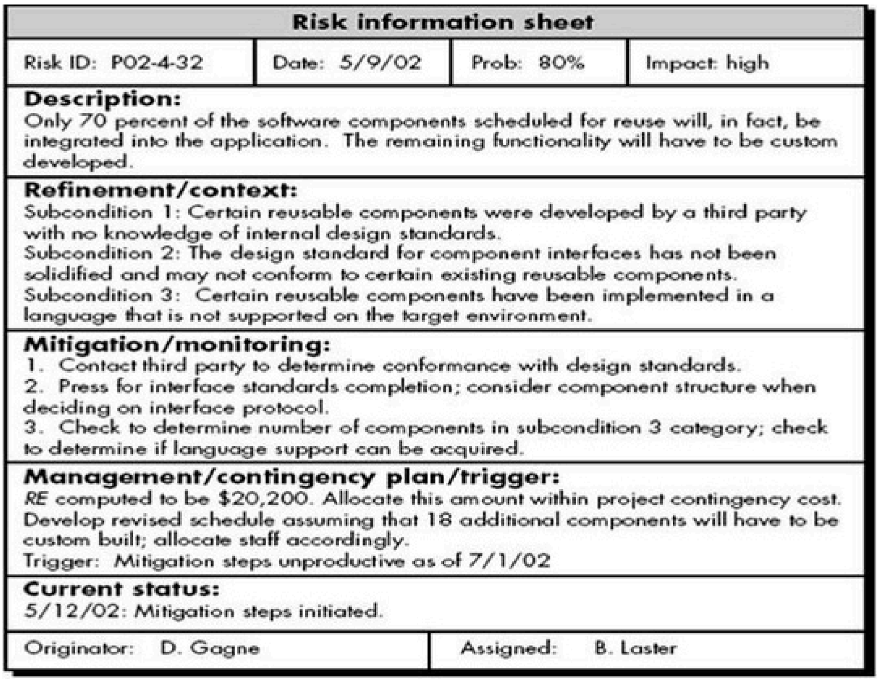
The cost associated with a computer crash resulting in a loss of data is crucial. A computer crash itself is not crucial, but rather the loss of data. A loss of data will result in not being able to deliver the product to the customer. This will result in a not receiving a letter of acceptance from the 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:

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

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 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:

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

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

(ii) 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				▨	▨	▨									
						◆									

Coding							▨	▨	▨	▨					
										◆					
Testing											▨	▨			
												◆			
Deployment													▨		
													◆		
Maintenance														▨	▨ ◆

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** 6M – 1M for 1 point

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.

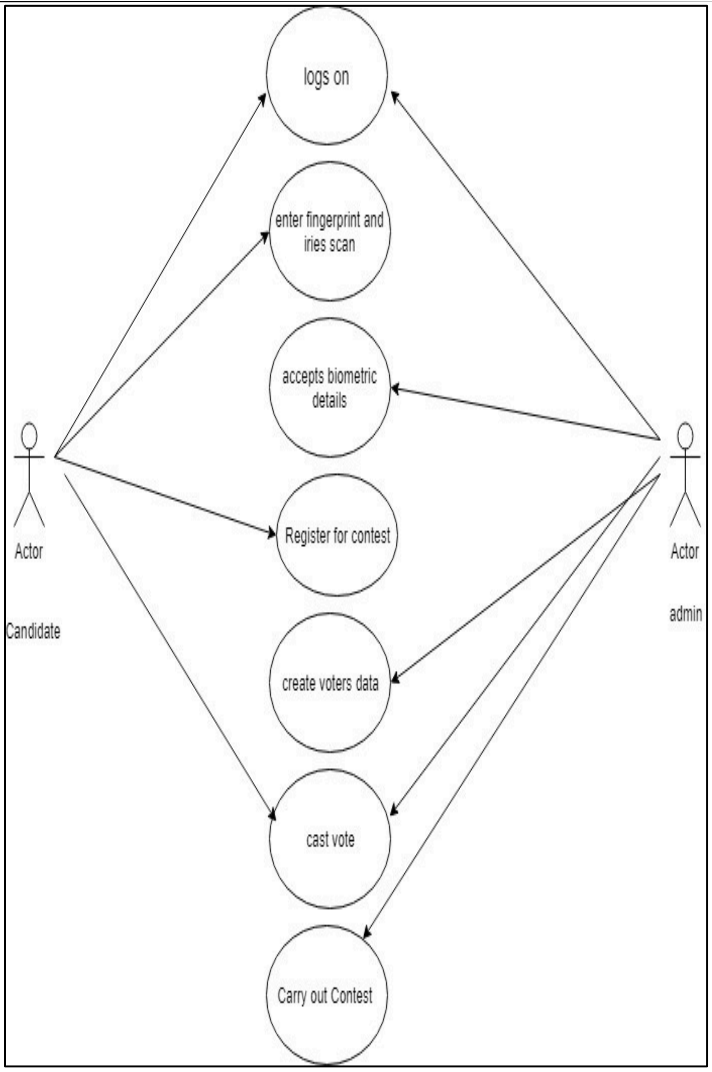
3. **Business requirements**: They are high-level requirements that are taken from the business case from the projects.
For eg:-

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**

Ans

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.

6M- 1M for 1point

	c	**Describe COCOMO II model for evaluating size of software project with any three parameters in detail**	**6 M**
	Ans	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.	3M for Description, 3M for parameters

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 Application	Infrastructure
	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 ∑(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

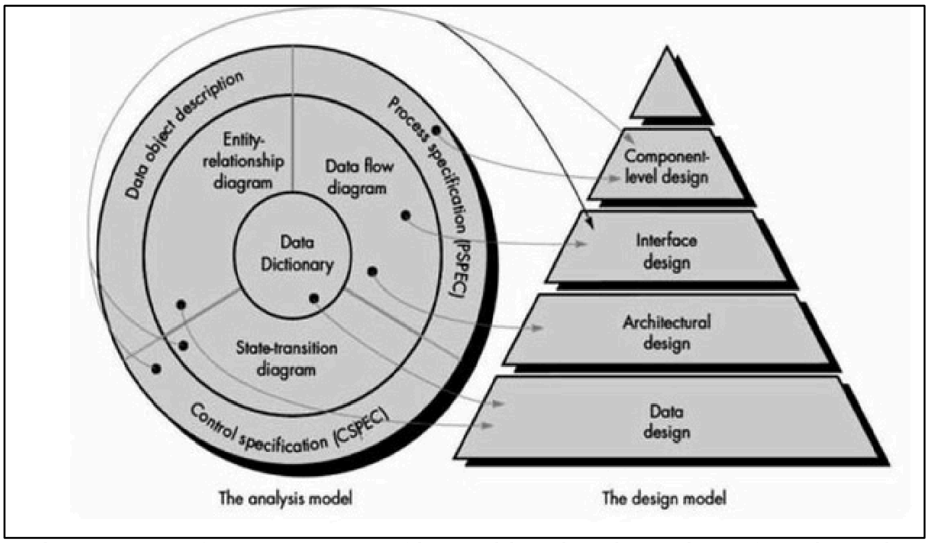
d. Maintenance effort (MAF)

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**	2M –diiagram, 4M – explanation



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.

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

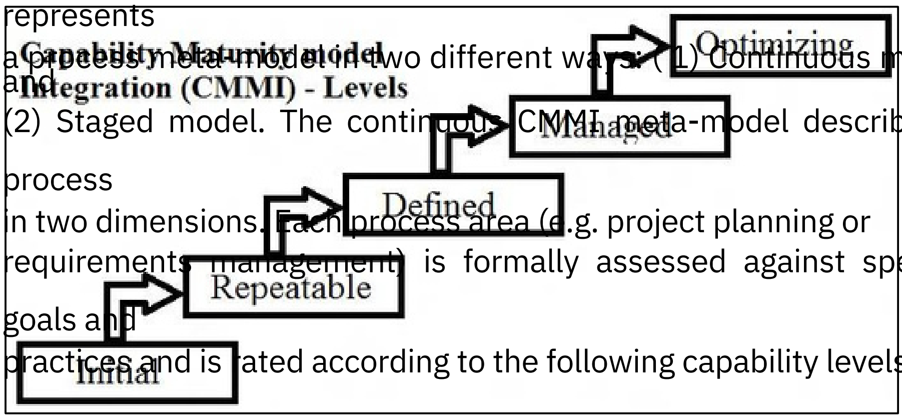
Describe CMMI. Give significance of each level.

6 M

Ans

1M- diagram , 5M- 5 points

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:



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.

c	**Identify and enlist requirement for given modules of employee management software** i. Employee detail ii. Employee salary iii.Employee performance	**6 M** 2 M for employee detail, salary, performance each

Ans

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

		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.	

22413

11920

3 Hours / 70 Marks Seat No. □□□□□□□□

Instructions – (1) All Questions are *Compulsory.*

(2) Answer each next main Question on a new page.

(3) Illustrate your answers with neat sketches wherever necessary.

(4) Figures to the right indicate full marks.

(5) Mobile Phone, Pager and any other Electronic Communication devices are not permissible in Examination Hall.

Marks

1. Attempt any FIVE of the following: 10

a) Define software. Draw the failure curve for Software.

b) State two characteristics of Software.

c) Define software requirements specifications.

d) Define Proactive and Reactive risk strategy.

e) Name two cost estimation approaches.

f) Define software quality.

g) Name four software quality assurance activities.

2. Attempt any THREE of the following: 12

a) State and explain with examples four broad categories of software.

b) Explain the notations used for preparing a Data Flow diagram.

c) Describe 4 ‘P’s of management spectrum giving their significance.

d) Explain four basic principles of software Project Scheduling.

P.T.O.

Marks

3. Attempt any THREE of the following: 12

a) Explain process framework with a suitable diagram.

b) Describe four principles of good planning.

c) Draw and explain Level 1 DFD for railway reservation system.

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

4. Attempt any THREE of the following: 12

a) Explain waterfall process model. State its advantages and disadvantages.

b) Enlist core principles of Software engineering practice.

c) Describe RMMM Strategy.

d) Describe the Analytical method of project cost estimation with example.

e) Explain GANTT chart and its application for project tracking with an example.

5. Attempt any TWO of the following: 12

a) Sketch a use case diagram for library management system with minimum four use cases and two actors.

b) Explain the concept of Black box testing and white box testing.

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.

Marks

6. Attempt any TWO of the following: 12

a) Define data objects, attributes, relationship, cardinality with example of each.

b) Compare CMMI and ISO for software w.r.to

i) Scope

ii) Approach

iii) Implementation.

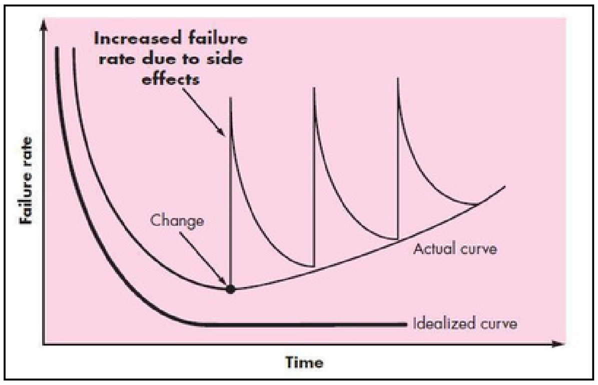
c) Explain six functions of requirement engineering process.

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
		Attempt any Three of the following:	
2.		**State and explain with examples four categories of software.**	**12M**
	a	**Types / Categories of Software**	**4M**
	Ans	**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.

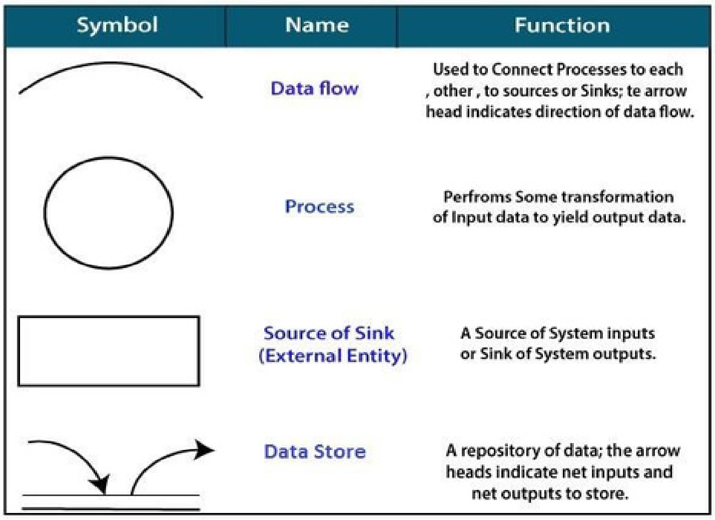
Example : CAD / CAM software

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

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.



Symbol	Name	Function
	Data flow	Used to Connect Processes to each , other , to sources or Sinks; te arrow head indicates direction of data flow.
	Process	Perfroms Some transformation of Input data to yield output data.
	Source of Sink (External Entity)	A Source of System inputs or Sink of System outputs.
	Data Store	A repository of data; the arrow heads indicate net inputs and net outputs to store.

Symbols for Data Flow Diagrams

c **Describe 4 P's of management spectrum giving their significance.** **4M**

Ans

The Management Spectrum – 4 Ps and their Significance
Effective software project management focuses on these items (in this order) Deals with the cultivation of motivated, highly skilled people

Description of each P's-1M each

1. The people
i. Consists of the stakeholders, the team leaders, and the software team
2. The product
i. Product objectives and scope should be established before a project can be planned.
3. The process
i. The software process provides the framework from which a comprehensive plan for software development can be established.
4. The project
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.

d **Explain four basic principles of software project scheduling.**

	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

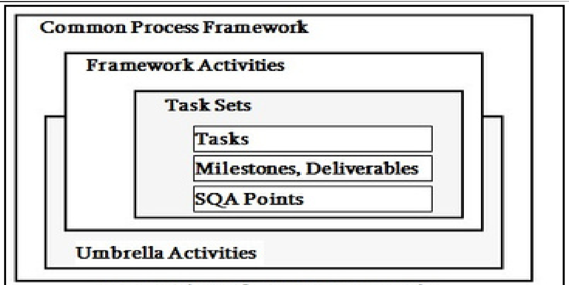


Figure: Chart of Process Framework

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.

Any 4 Principles; 1 M each

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.

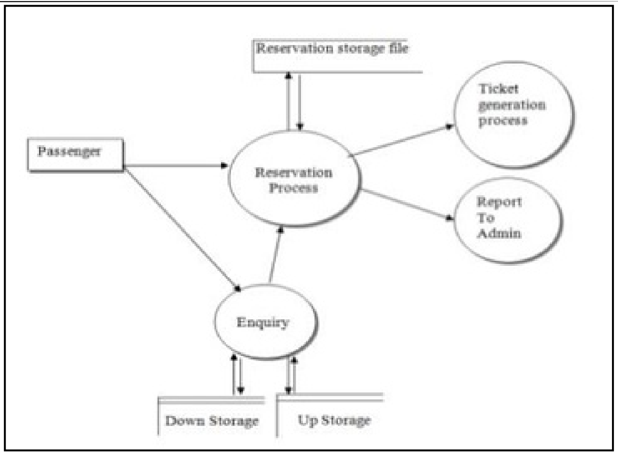
Principle 5.Consider risk as you defines the plan. If you have identified risks that have high impact and high probability, contingency planning is necessary. In addition, the project plan (including the schedule) should be

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. **Draw and explain Level 1 DFD for railway reservation system.**

	c		**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. Description 2M Example 2M

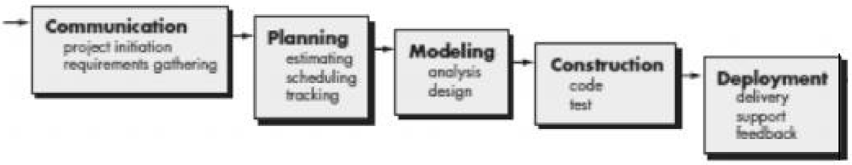
A review of the System Specification indicates that the software is to execute on an engineering workstation and must interface with various computer graphics peripherals including a mouse, digitizer, high resolution color display and laser printer.

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

Example:

Function User interface and control	**Estimated LOC**
facilities (UICF) Two-dimensional	2,30 0
analysis (2DGA) geometric Three-dimensional	5,30
geometric analysis (3DGA)	0
	6,80
Database management (DBM)	0
Peripheral control function (PCF) Computer graphics display module s facilities (CGDF)	3,35 0
Design analysis (DAM)	4,95 0 2,10 0 8,40 0

		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		Description 2M Any 2 advantage 1M Any 2 Disadvantages 2M



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.

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.

Advantages of waterfall model:

		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

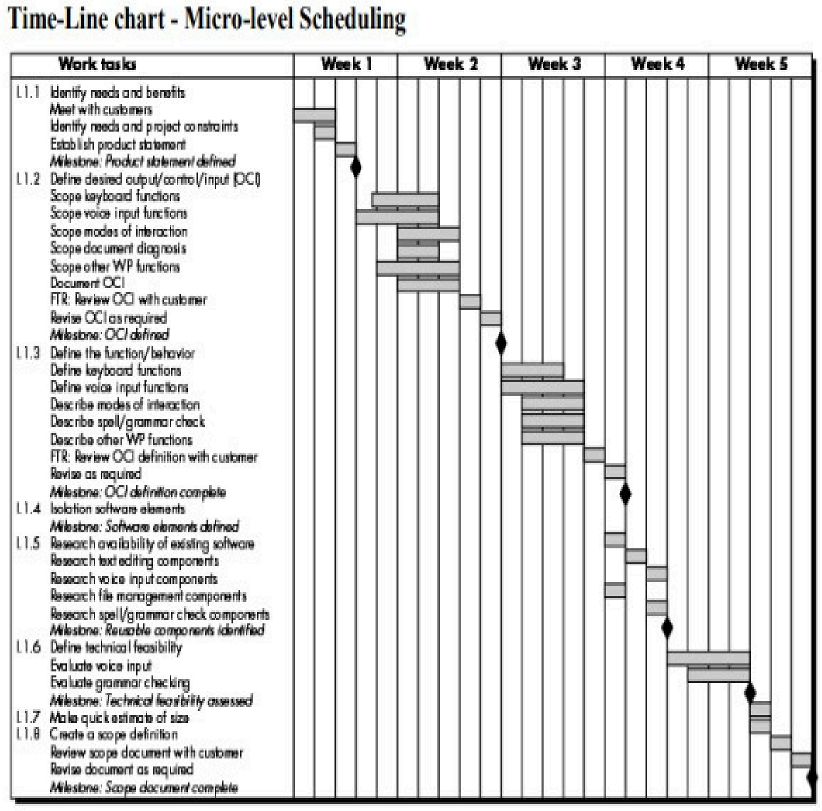
The unique operands are: a, b, c, &a, &b, &c, a+b+c, avg, 3, "%d %d %d", "avg = %d"

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

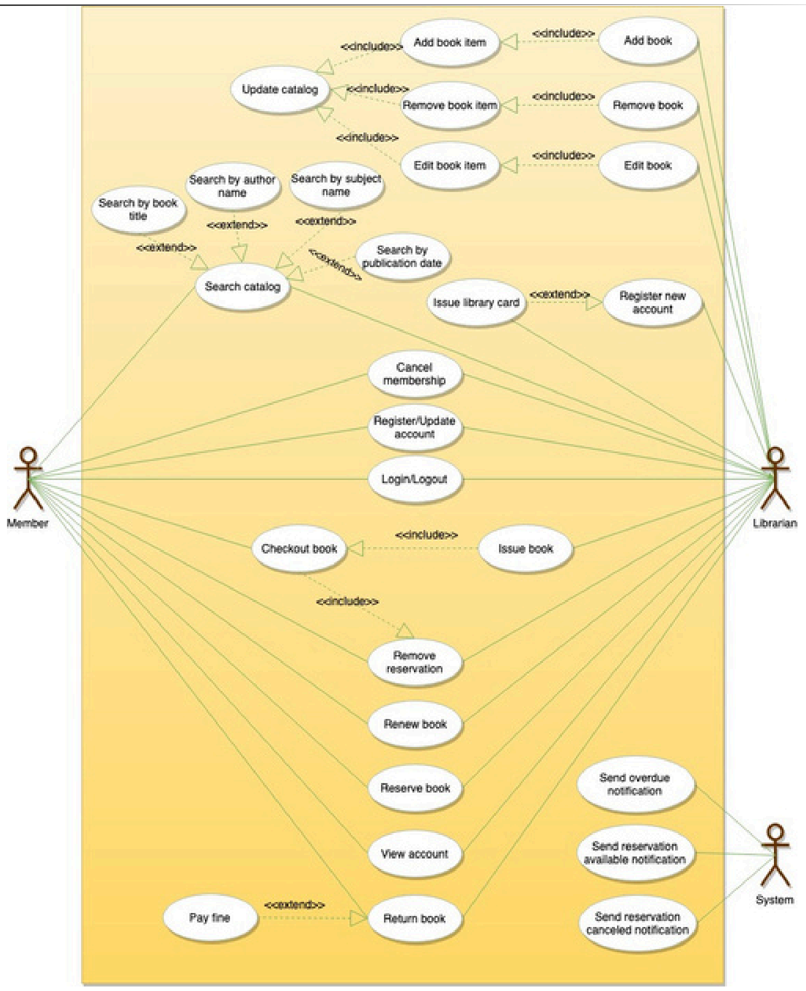


Application of Gantt Chart

 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 **Ans**	**Sketch a use case diagram for library management system with minimum four use cases and two actors.**	**6M**
			Correct/relevant any four use cases -6M

			
	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:
  - o **Equivalence partitioning**: Equivalence partitioning divides input values into valid and invalid partitions and selecting corresponding values from each partition of the test data.
  - o **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-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).

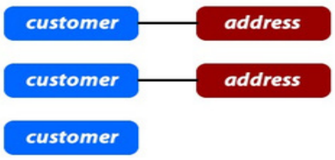
Each attribute has a name, and is associated with an entity and a

domain

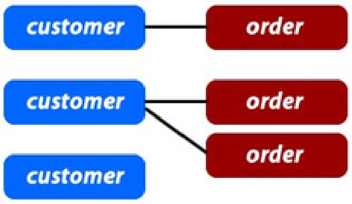
of legal values.
Example: Employee = (Name, Address, Birthdate (Age), Salary).
Relationship: A relationship identifies names and defines an
association
between two entity types **One**-to-one relationship: Example: We

have a
relationship between the Customers table and the Addresses table.
If each
address can belong to only one customer, this relationship is "One

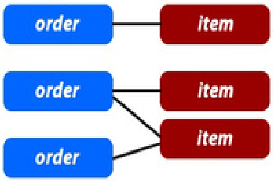
to
One".



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.	Correct/relevant explanation for each function- 1M

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 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.