MAHARASHTRA STATE BOARD OF TECHNICAL EDUCATION
(Autonomous)
(ISO/IEC - 27001 - 2013 Certified)

Winter – 19 EXAMINATION

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

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:**	**20M**
	a	**Describe changing nature of software.**	**4 M**
	Ans	Changing Nature of Software: Whenever one starts with the software implementation changes can occur any time. The software can be change due to any reason. But while implementing software one should be ready for such changes as if changes occur there shall not be drastic change in the system. The development team should manage to implement/mould the implemented system so that the changes can be reflected and the user requirements meet. When change occur, the team look for the current status of the system and from there onwards they start implementing a system with new requirements of a user or changes which is to be implemented in a system. **OR** Today's Software takes on a dual role. It is a product, and at the same time, the vehicle for delivering a product. As a product, it delivers the computing potential embodied by computer hardware or more broadly, by a network of computers that are accessible by local hardware. Whether it resides within a mobile phone or operates inside a mainframe computer,	Any relevant 4 points each carry 1M **Note: Any Relevant answer shall be considered.**

		software is information transformer— producing, managing, acquiring, modifying, displaying, or transmitting information that can be as simple as a single bit or as complex as a multimedia presentation derived from data acquired from dozens of independent sources. As the vehicle used to deliver the product, software acts as the basis for the control of the computer (operating systems), the communication of information (networks), and the creation and control of other programs (software tools and environments). Software delivers the most important product of our time—information. It transforms personal data (e.g., an individual's financial transactions) so that the data can be more useful in a local context; it manages business information to enhance competitiveness. It provides a gateway to worldwide information networks (e.g., the Internet), and provides the means for acquiring information in all of its forms. The role of computer software has undergone significant change over the last half century. Dramatic improvements in hardware performance, profound changes in computing architectures, vast increases in memory and storage capacity, and a wide variety of exotic input and output options, have all precipitated more sophisticated and complex computer-based systems. Sophistication and complexity can produce dazzling results when a system succeeds, but they can also pose huge problems for those who build complex systems.	
	b	**What is extreme programming?**	**4 M**
	Ans	**Definition** Extreme Programming (XP) is an agile software development framework that aims to produce higher quality software, and higher quality of life for the development team. XP is the most specific of the agile frameworks regarding appropriate engineering practices for software development. **When Applicable** The general characteristics where XP is appropriate were described by Don Wells: • Dynamically changing software requirements • Risks caused by fixed time projects using new technology • Small, co-located extended development team • The technology you are using allows for automated unit and functional tests. Due to XP's specificity when it comes to it's full set of software engineering practices, there are several situations where you may not want to fully practice XP.	Definition-1M When applicable-1M Values-2M

Values

The five values of XP are communication, simplicity, feedback, courage, and respect and are described in more detail below.

Communication

Software development is inherently a team sport that relies on communication to transfer knowledge from one team member to everyone else on the team. XP stresses the importance of the appropriate kind of communication – face to face discussion with the aid of a white board or other drawing mechanism.

Simplicity

Simplicity means "what is the simplest thing that will work?" The purpose of this is to avoid waste and do only absolutely necessary things such as keep the design of the system as simple as possible so that it is easier to maintain, support, and revise. Simplicity also means address only the requirements that you know about; don't try to predict the future.

Feedback

Through constant feedback about their previous efforts, teams can identify areas for improvement and revise their practices. Feedback also supports simple design. Your team builds something, gathers feedback on your design and implementation, and then adjust your product going forward.

Courage

Kent Beck defined courage as "effective action in the face of fear" (Extreme Programming Explained P. 20). This definition shows a preference for action based on other principles so that the results aren't harmful to the team. You need courage to raise organizational issues that reduce your team's effectiveness. You need courage to stop doing something that doesn't work and try something else. You need courage to accept and act on feedback, even when it's difficult to accept.

Respect

The members of your team need to respect each other in order to communicate with each other, provide and accept feedback that honours your relationship, and to work together to identify simple designs and solutions.

	c	**Describe any two core principles of software engineering**	**4 M**
	Ans	**The First Principle: The Reason It All Exists** A software system exists for one reason: To provide value to its users. All decisions should be made with this in mind. Before specifying a system	

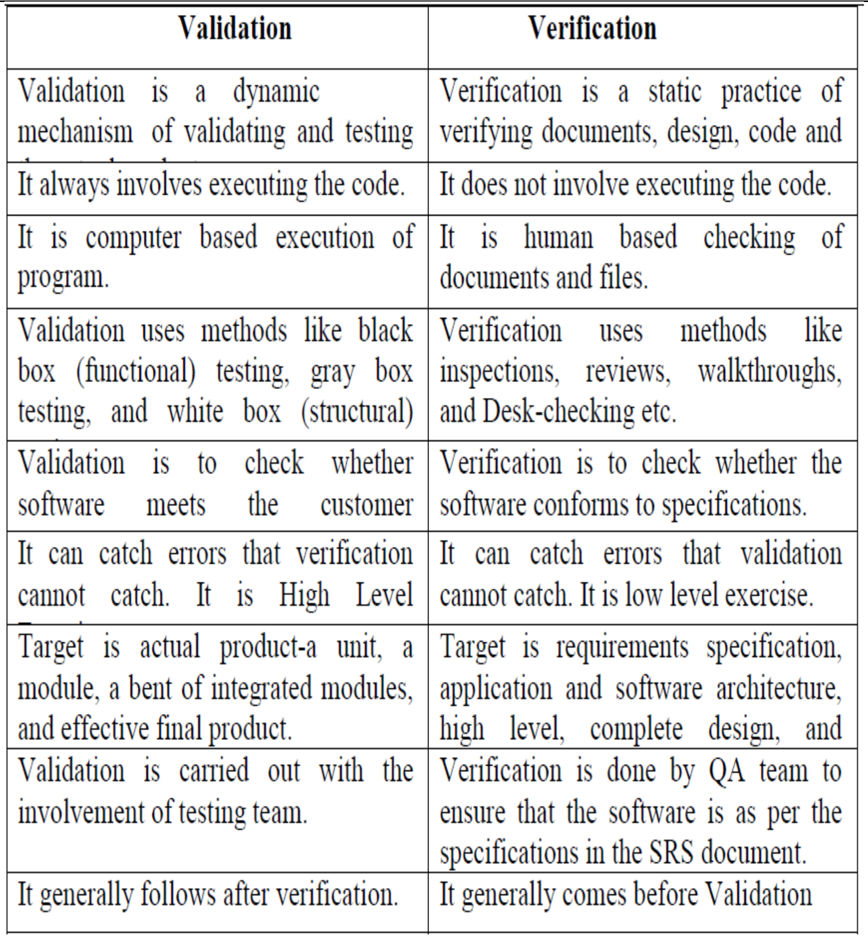
		requirement, before noting a piece of system functionality, before determining the hardware platforms or development processes, ask yourself questions such as: "Does this add real VALUE to the system?" If the answer is "no", don't do it. All other principles support this one. **The Second Principle: KISS (Keep It Simple, Stupid!)** There are many factors to consider in any design effort. All design should be as simple as possible, but no simpler. This facilitates having a more easily understood, and easily maintained system. **The Third Principle: Maintain the Vision** A clear vision is essential to the success of a software project. Without one, a project almost unfailingly ends up being "of two [or more] minds" about itself. Compromising the architectural vision of a software system weakens and will eventually break even the most well-designed systems. Having an empowered Architect who can hold the vision and enforce compliance helps ensure a very successful software project. **The Fourth Principle: What You Produce, Others Will Consume.** Seldom is an industrial-strength software system constructed and used in a vacuum. In some way or other, someone else will use, maintain, document, or otherwise depend on being able to understand your system. So, always specify, design, and implement knowing someone else will have to understand what you are doing. The audience for any product of software development is potentially large. Specify with an eye to the users. Design, keeping the implementers in mind. Code with concern for those that must maintain and extend the system. Someone may have to debug the code you write, and that makes them a user of your code. Making their job easier adds value to the system. **The Fifth Principle: Be Open to the Future** A system with a long lifetime has more value. In today's computing environments, where specifications change on a moment's notice and hardware platforms are obsolete when just a few months old, software lifetimes are typically measured in months instead of years. However, true "industrial-strength" software systems must endure far longer. To do this successfully, these systems must be ready to adapt to these and other changes. Systems that do this successfully are those that have been designed this way from the start. Never design yourself into a corner. Always ask "what if ", and prepare for all possible answers by creating systems that solve the general problem, not just the specific one. This could very possibly lead to the reuse of an entire system. **The Sixth Principle: Plan Ahead for Reuse** Reuse saves time and effort. Achieving a high level of reuse is arguably the hardest goal to accomplish in developing a software system. The reuse of code and designs has been proclaimed as a major benefit of using object-oriented technologies. However, the return on this investment is not automatic. To leverage the reuse possibilities that OO programming	Explain any two principle 2M each

		provides requires forethought and planning. There are many techniques to realize reuse at every level of the system development process. Those at the detailed design and code level are well known and documented. New literature is addressing the reuse of design in the form of software patterns. However, this is just part of the battle. Communicating opportunities for reuse to others in the organization is paramount. How can you reuse something that you don't know exists? Planning ahead for reuse reduces the cost and increases the value of both the reusable components and the systems into which they are incorporated. **Seventh Principle: Think!** This last Principle is probably the most overlooked. 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 valuable experience. A side effect of thinking is learning to recognize when you don t know something, at which point you can research the answer. When clear thought has gone into a system, value comes out. Applying the first six Principles requires intense thought, for which the potential rewards are enormous.	
	d	**What are different data design elements?**	**4 M**
	Ans	**Data design** is the first design activity, which results in fewer complexes, modular and efficient program structure. The information domain model developed during analysis phase is transformed into data structures needed for implementing the software. The data objects, attributes, and relationships depicted in entity relationship diagrams and the information stored in data dictionary provide a base for data design activity. During the data design process, data types are specified along with the integrity rules required for the data. For specifying and designing efficient data structures, some principles should be followed. These principles are listed below. 1. The data structures needed for implementing the software as well-as the operations that can be applied on them should be identified. 2. A data dictionary should be developed to depict how different data objects interact with each other and what constraints are to be imposed on the elements of data structure. 3. Stepwise refinement should be used in data design process and detailed design decisions should be made later in the process. 4. Only those modules that need to access data stored in a data structure directly should be aware of the representation of the data structure.	Correct explanation related to topic-4M

5. A library containing the set of useful data structures along with the operations that can be performed on them should be maintained. Language used for developing the system should support abstract data types. The structure of data can be viewed at three levels, namely, program component level, application level, and business level. At the program component level, the design of data structures and the algorithms required to manipulate them is necessary, if high-quality software is desired. At the application level, it is crucial to convert the data model into a database so that the specific business objectives of a system could be achieved. At the business level, the collection of information stored in different databases should be reorganized into data warehouse, which enables data mining that has an influential impact on the business itself.

e **Differentiate between Validation and Verification.** **4 M**

Ans



Validation	Verification
Validation is a dynamic mechanism of validating and testing	Verification is a static practice of verifying documents, design, code and
It always involves executing the code.	It does not involve executing the code.
It is computer based execution of program.	It is human based checking of documents and files.
Validation uses methods like black box (functional) testing, gray box testing, and white box (structural)	Verification uses methods like inspections, reviews, walkthroughs, and Desk-checking etc.
Validation is to check whether software meets the customer	Verification is to check whether the software conforms to specifications.
It can catch errors that verification cannot catch. It is High Level	It can catch errors that validation cannot catch. It is low level exercise.
Target is actual product-a unit, a module, a bent of integrated modules, and effective final product.	Target is requirements specification, application and software architecture, high level, complete design, and
Validation is carried out with the involvement of testing team.	Verification is done by QA team to ensure that the software is as per the specifications in the SRS document.
It generally follows after verification.	It generally comes before Validation

Any correct minimum 4 points – 4M

f **Give possible reasons of why software is delivered late.** **4 M**

Ans **1.Expansion of functionality**

The expansion of functionality is a phenomenon in which new functionalities continue to be conceived and requested as the project

Any correct minimum 4 reasons – 4M

proceeds. The software can never be completed in this way.

2. Gold plating
Gold plating is a phenomenon in which programmers and designers try to make many details of the software or design too elaborate. Much time is spent improving details, even though the improvements were not requested by the customer or client. The details often add little to the desired result.

3. Neglecting quality control
Time pressure can sometimes cause programmers or project teams to be tempted to skip testing. This frequently causes more delays than it prevents. The time that elapses before an error is discovered in the software is associated with an exponential increase in the time that is needed to repair it.

4.Overly optimistic schedules
Overly optimistic schedules place considerable pressure on the project team. The team will initially attempt to reach the (unrealistic) deadlines. These attempts lead to sloppy work and more errors, which cause further delays.
In this regard, be particularly wary of schedules that are imposed from above. The desire to complete a project (more) quickly sometimes arises for primarily strategic reasons; if it is not feasible, however, it should not be attempted. The project will not proceed more quickly and the product will ultimately suffer.

5. Working on too many projects at the same time
Dividing work across many different projects (or other tasks) causes waiting times that lead to many delays in projects.

6. Poor design:The absence (or poor realisation) of designs leads to delays, as it requires many revisions at later stages.

7. The 'one-solution-fits-all' syndrome
Using the right software for a project is important. Some software platforms are more suited to particular applications than others are. Thinking that the use of particular software will greatly improve productivity, however, is also a trap.

8. Research-oriented projects
Projects in which software must be made and research must be conducted are difficult to manage. Research is accompanied by high levels of uncertainty. When or if progress will be achieved in research is unclear. When software development is dependent upon the results of research, the former frequently comes to a standstill.

		9. Mediocre personnel insufficiently qualified personnel can cause project delays. Technically substantive knowledge of the subject of the project plays a role, as do knowledge and skills in working together to play the game of the project. **10. Customers fail to fulfil agreements** Customers are not always aware that they are expected to make a considerable contribution to the realisation of a project. When customers do not react in a timely manner to areas in which they must be involved, projects can come to a standstill. Worse yet, the team may proceed further without consulting the customer, which can lead to later conflicts. **11. Tension between customers and developers** The tension that can arise between customers and developers (e.g.because the project is not proceeding quickly enough) can cause additional delays, as it disturbs the necessary base of trust and the working atmosphere.	
	g	**What is software reliability?**	**4 M**
	Ans	There is no doubt that the reliability of a computer program is an important element of its overall quality. If a program repeatedly and frequently fails to perform, it matters little whether other software quality factors are acceptable. Software reliability, unlike many other quality factors, can be measured directly and estimated using historical and developmental data. Software reliability is defined in statistical terms as “the probability of failure-free operation of a computer program in a specified environment for a specified time”. **Measures of Reliability and Availability** Early work in software reliability attempted to extrapolate the mathematics of hardware reliability theory to the prediction of software reliability. Most hardware-related reliability models are predicated on failure due to wear rather than failure due to design defects. In hardware, failures due to physical wear (e.g., the effects of temperature, corrosion, shock) are more likely than a design-related failure. Unfortunately, the opposite is true for software. In fact, all software failures can be traced to design or implementation problems; wear does not enter into the picture. There has been an on-going debate over the relationship between key concepts in hardware reliability and their applicability to software. Although an irrefutable link has yet to be established, it is worthwhile to consider a few simple concepts that apply to both system elements. If we consider a computer-based system, a simple measure of reliability is meantime-between-failure (MTBF): MTBF =MTTF + MTTR where the acronyms MTTF and MTTR are mean-time-to-failure and mean-time-to repair, in addition to a reliability measure, you should also develop a measure of availability.	Appropriate explanation-4 M

Software availability is the probability that a program is operating according to requirements at a given point in time and is defined as

$$\text{Availability} = \frac{\text{MTTF}}{\text{MTTF} + \text{MTTR}} \times 100\%$$

The MTBF reliability measure is equally sensitive to MTTF and MTTR. The availability measure is somewhat more sensitive to MTTR, an indirect measure of the maintainability of software.

Software Safety

Software safety is a software quality assurance activity that focuses on the identification and assessment of potential hazards that may affect software negatively and cause an entire system to fail. If hazards can be identified early in the software process, software design features can be specified that will either eliminate or control potential hazards.

2. **Attempt any Three of the following:** **16M**

a **List characteristics of software.** **4 M**

Ans **Characteristics of software:**

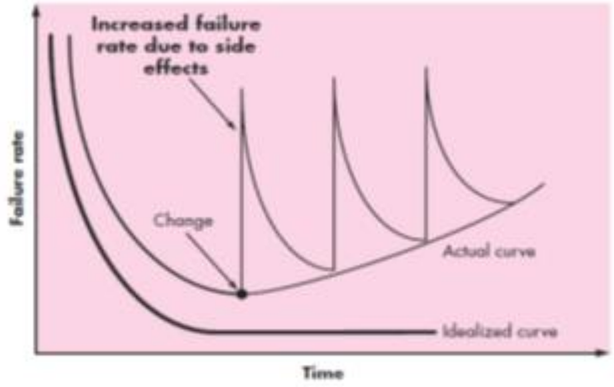
Correct 3 characteristic – 1M each
Diagram-1M

i) Software is developed or engineered; it is not manufactured in the classical sense.

Software is virtual. That is, software can be used using proper hardware. And we can only use it. But we can use, touch, and see hardware. Thus, software never gets manufactured, they are developed.

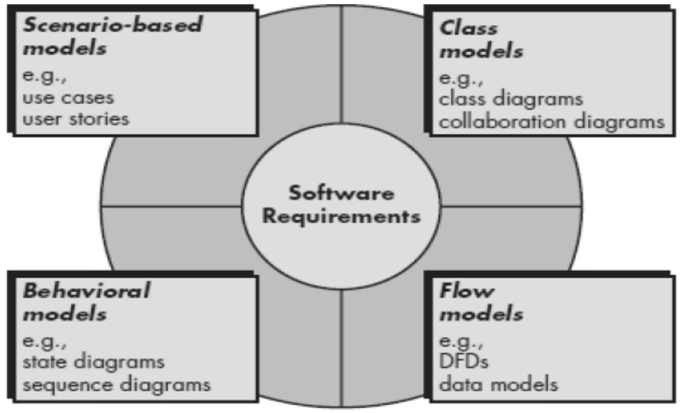
ii) Software doesn't "wear out" like hardware and it is not degradable over a period.

Software is not susceptible to the environmental maladies that cause hardware to wear out. In theory, therefore, the failure rate curve for software should take the form of the "idealized curve" shown in Figure



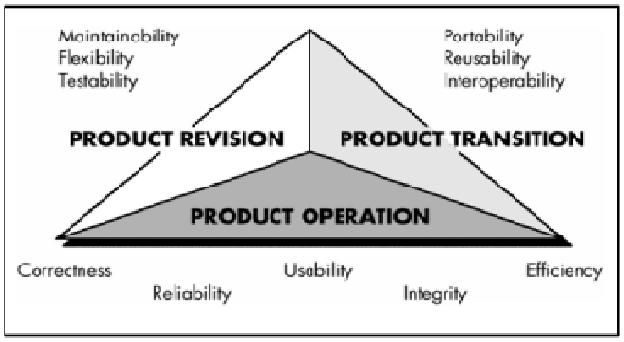
During its life, software will undergo change. As changes are made, it is likely that errors will be introduced, causing the failure rate curve to spike

		as shown in the “actual curve”. 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. **iii) Although the industry is moving toward component-based construction, most Software continues to be custom built.** Custom software or application is a kind of software, which is specifically design and develops for an organization or a group of users with unique needs and requirements. Most of the organizations are opting for custom built applications for its unique benefits.	
	b	**Explain behavioral model.**	**4 M**
	Ans	Behavioural models are used to describe the overall behaviour of a system. The behavioural model indicates how software will respond to external events or stimuli. To create the model, you should perform the following steps: 1. Evaluate all use cases to fully understand the sequence of interaction within the system. 2. Identify events that drive the interaction sequence and understand how these events relate to specific objects. 3. Create a sequence for each use case. 4. Build a state diagram for the system. 5. Review the behavioural model to verify accuracy and consistency **Two types of behavioural model are:** **Data processing models** that show how data is processed as it moves through the system and **State machine models** that show the systems response to events These models show different perspectives, so both of them are required to describe the system’s behaviour. **Data Processing Model** Data flow diagrams (DFDs) may be used to model the system’s data processing. These show the processing steps as data flows through a system. DFDs are an intrinsic part of many analysis methods. It is Simple and intuitive notation that customers can understand. It shows end-to-end processing of data. DFDs model the system from a functional perspective. It is helpful to develop an overall understanding of the system. **State machine Model** These model the behaviour of the system in response to external and internal events. They show the system’s responses to stimuli so are often used for modelling real modelling real-time systems. State machine models show system states as nodes and events as arcs between these	Description of each step -2M Types of behavioral model-2M **Note: Any Relevant answer shall be considered.**

		nodes. When an event occurs, the system moves from one state to another. State charts, an integral part of the UML is used to represent state machine models.	
	c	**What is an object-oriented analysis?**	**4 M**
	Ans	Object-oriented Analysis focuses on the definition of classes and the manner in which they collaborate with one another to effect customer requirements.  Scenario-based models e.g., use cases user stories Class models e.g., class diagrams collaboration diagrams Software Requirements Behavioral models e.g., state diagrams sequence diagrams Flow models e.g., DFDs data models Elements of the Analysis Model The intent is to define all classes, relationship, behaviour associated with them, that are to the problem to be solved. To achieve this following task should occur. Task 1. Basic user requirements must be communicated between user and developer. Task 2. Classes must be identified (i.e. attributes, methods defined) Task 3. A class hierarchy is defined Task 4. Object- object relationships (object connection) should be represented. Task 5. Object behaviour must be modelled. Task 6. Task-1 to Task-5 is reapplied iteratively till model is complete.	Definition-1M Diagram-1M List of task-2M
	d	**Enlist guidelines those leads to a successful software testing strategy.**	**4 M**
	Ans	1. Testing is a process of executing a program with the intent of finding an error. 2. A good test case is one that has a high probability of finding an as-yet undiscovered error. 3. A successful test is one that uncovers an as-yet-undiscovered error 4. All tests should be traceable to customer requirements. 5. A good test has a high probability of finding an error. 6. A good test is not redundant. 7. A good test should be ―best of breed. 8. A good test should be neither too simple nor too complex.	Any four relevant points -1M each

		OR 1. Finding programming defects. 2. Gaining confidence in and providing information about the level of quality. 3. To make sure that the end result meets the business and user requirements. 4. To ensure that it satisfies SRS that is System Requirement Specifications	
	e	**Describe SCM repository.**	**4 M**
	Ans	Software configuration management (SCM), also called change management, is a set of activities designed to manage change by identifying the work products that are likely to change, establishing relationships among them, defining mechanisms for managing different versions of these work products, controlling the changes imposed, and auditing and reporting on the changes made. SCM is an umbrella activity that is applied throughout the software process. SCM is a set of tracking and control activities that are initiated when SE project begin and terminate only when the software is taken out of operation. SCM helps to improve software quality and on time delivery. SCM defines the project strategy for change management. When formal SCM is invoked, the change control process produces software change requests, reports and engineering change orders. SCM helps to track, analyse and control every work product. **Need of SCM** • To Identify all items that define the software configuration • To Manage changes to one or more configuration items • To Facilitate construction of different versions of a software application • To ensure that software quality is maintained as configuration evolves. **Functions of SCM repository** 1.**Data integrity: Validates** entries, ensures consistency, cascades modifications 2.**Information sharing: Shares** information among developers and tools, manages and controls multi-user access 3.**Tool integration:** Establishes a data model that can be accessed by many software engineering tools, controls access to the data. 4.**Data integration: Allows** various SCM tasks to be performed on one or more CSCIs 5.**Methodology enforcement: Defines** an entity-relationship model for the repository that implies a specific process model for software engineering. 6. **Document standardization: Defines** objects in the repository to guarantee a Standard approach for creation of software engineering documents.	Explanation-1M Need of SCM-1M Functions of SCM-2M

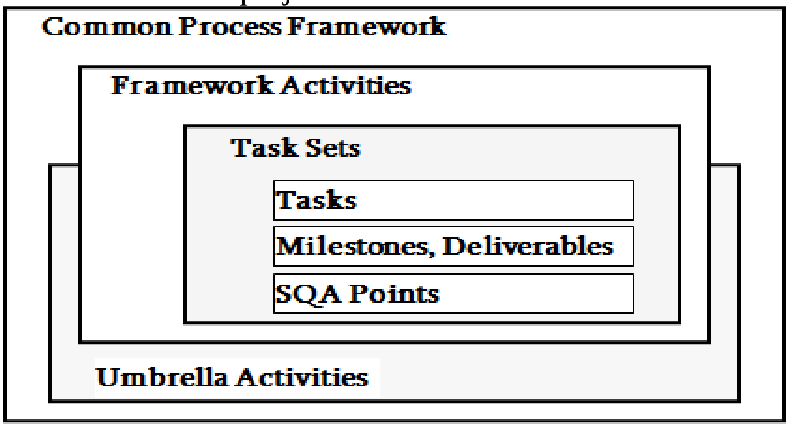
	f	**State and explain SQA activities.**	**4 M**
	Ans	Software quality assurance is composed of a variety of tasks associated with two different constituencies - the software engineers who do technical work and an SQA group that has responsibility for quality assurance planning, oversight, record keeping, analysis, and reporting. Software engineers address quality (and perform quality assurance and quality control activities) by applying solid technical methods and measures, conducting formal technical reviews, and performing well-planned software testing. **Activities of SQA:** 1) Prepare an SQA plan for a project. The plan is developed during project planning and is reviewed by all interested parties. Quality assurance activities performed by the software engineering team and the SQA group are governed by the plan. The plan identifies • evaluations to be performed • audits and reviews to be performed • standards that are applicable to the project • procedures for error reporting and tracking • documents to be produced by the SQA group • amount of feedback provided to the software project team 2) Participate in the development of the project’s software process description. The software team selects a process for the work to be performed. The SQA group reviews the process description for compliance with organizational policy, internal software standards, externally imposed standards (e.g., ISO-9001), and other parts of the software project plan. 3) Review software engineering activities to verify compliance with the defined software process. The SQA group identifies, documents, and tracks deviations from the process and verifies that corrections have been made. 4) Audits designated software work products to verify compliance with those defined as part of the software process. The SQA group reviews selected work products; identifies, documents, and tracks deviations; verifies that corrections have been made; and periodically reports the results of its work to the project manager. 5) Ensure that deviations in software work and work products are documented and handled according to a documented procedure. Deviations may be encountered in the project plan, process description,	State-1M Activities-3M

		applicable standards, or technical work products. 6) Records any noncompliance and reports to senior management. Noncompliance items are tracked until they are resolved. **Or** Software quality assurance is composed of a variety of tasks associated with two different constituencies - the software engineers who do technical work and an SQA group that has responsibility for quality assurance planning, oversight, record keeping, analysis, and reporting. Software engineers address quality (and perform quality assurance and quality control activities) by applying solid technical methods and measures, conducting formal technical reviews, and performing well-planned software testing. 1. Evaluations to be performed 2. Prepares an SQA plan for a project. 3. Participates in the development of the project's software process description. 4. Reviews software engineering activities to verify compliance with the Defined software process. 5. Audits designated software work products to verify compliance with those Defined as part of the software process. 6. Ensures that deviations in software work and work products are documented and handled according to a documented procedure. 7. Records any noncompliance and reports to senior management. 8. Audits and reviews to be performed 9. Standards those are applicable to the project 10. Procedures for error reporting and tracking 11. Documents to be produced by the SQA group 12. Amount of feedback provided to the software project team.	
3.		**Attempt any Three of the following:**	**16M**
	a	**Explain McCall's quality factor?**	**4 M**
	Ans	 Maintainability Flexibility Testability Portability Reusability Interoperability PRODUCT REVISION PRODUCT TRANSITION PRODUCT OPERATION Correctness Reliability Usability Integrity Efficiency **Figure: McCall's Quality Factor**	8points – ½ M each

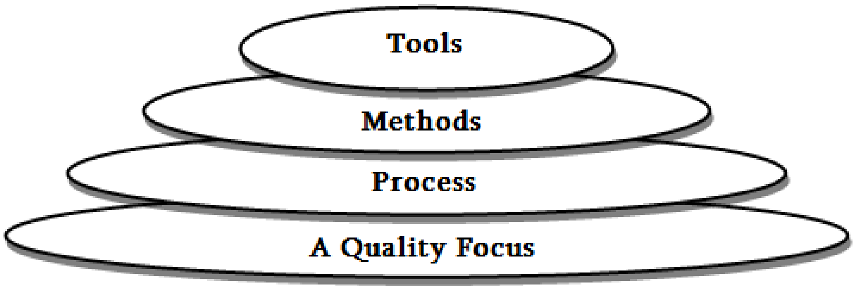
		The McCall's quality factor are as follows: **(a) Correctness**: The extent to which a program satisfies its specs and fulfills the customer 's mission objectives. **(b) Reliability**: The extent to which a program can be expected to perform its intended function with required precision. **(c) Efficiency**: The amount of computing resources and code required to perform is function. **(d) Integrity**: The extent to which access to S/W or data by unauthorized persons can be controlled. **(e) Usability**: The effort required to learn, operate, prepare input for, and interpret output of a program. **(f) Maintainability**: The effort required to locate and fix errors in a program. **(g) Flexibility**: The effort required to modify an operational program. **(h) Testability**: The effort required to test a program to ensure that it performs its intended function. **(i) Portability**: The effort required to transfer the program from one hardware and/or software system environment to another. **(j) Reusability**: The extent to which a program can be reused in other applications related to the packaging and scope of the functions that the program performs. **(k) Interoperability**: The effort required to couple one system to another.	
	b	**What are reactive and proactive risk strategies?**	**4 M**
	Ans	**i) Reactive risk strategy** •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. The reactive risk management is an essential element of:	(2M for each strategy)

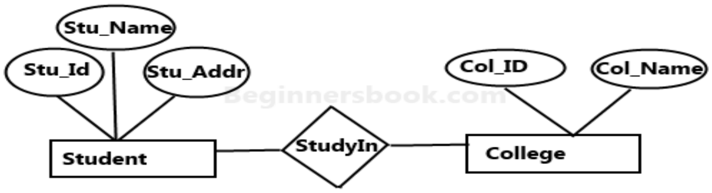
		•Mitigating safety events after hazard has occurred; •Minimizing damage from critical safety situations; •Acting quickly and efficiently in response to undesirable incidents; and •High quality decision making in reaction to safety data (threats, risk, etc.). **ii)Proactive risk strategy** It follows that the risks have to be identified before start of the project. They have to be analyzed by assessing their probability of occurrence, their impact after occurrence, and steps to be followed for its precaution. •They are meant for risks with relatively higher impact. The primary goals of proactive risk management are: •Identify behaviors that lead to hazard occurrence, and stop it before it happens; •Identify root causes before they lead to hazard occurrence; and •Understand safety "inputs" of your program – i.e., underlying causes that lead to safety performance.	
	c	**Why stress and performance testing is necessary?**	**4 M**
	Ans	**Stress testing** is a form of deliberately intense or thorough testing used to determine the stability of a given system or entity. It involves testing beyond normal operational capacity, often to a breaking point, in order to observe the results. Reasons can include: • to determine breaking points or safe usage limits • to confirm mathematical model is accurate enough in predicting breaking points or safe usage limits • to confirm intended specifications are being met • to determine modes of failure (how exactly a system fails) • to test stable operation of a part or system outside standard usage **Performance testing** is a testing practice performed to determine how a system performs in terms of responsiveness and stability under a particular workload. It can also serve to investigate, measure, validate or verify other quality attributes of the system, such as scalability, reliability and resource usage. • Performance testing strives to build performance standards into the implementation, design and architecture of a system. Performance testing can serve different purposes: • It can demonstrate that the system meets performance criteria. • It can compare two systems to find which performs better. • It can measure which parts of the system or workload cause the system to perform badly.	(4M for relevant description)

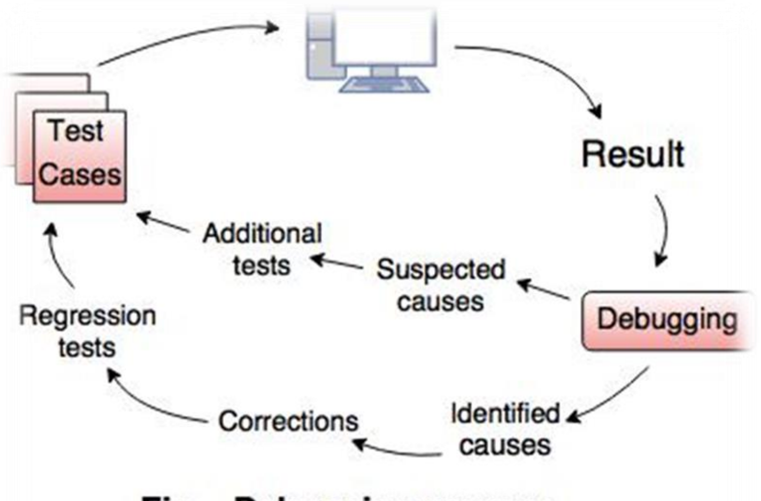
<table>
<tr><td></td><td>d</td><td><b>Compare cardinality and modality.</b></td><td><b>4 M</b></td></tr>
<tr><td></td><td>Ans</td><td>
<table>
<tr><th>Cardinality</th><th>Modality</th></tr>
<tr><td>1) It represents the number of occurrences of one object related to number of occurrences of another object.</td><td>1) A modality of relationship is zero if occurrence of relationship is optional and modality of relationship is 1 if occurrence of relationship is mandatory (i.e. compulsory).</td></tr>
<tr><td>2) It gives maximum number occurrences in relationship</td><td>2) The modality specifies the minimum number of relationships</td></tr>
<tr><td>3)Expected values are 1:1,1:M, M:N</td><td>3)Expected values are 0 or 1</td></tr>
<tr><td>4)E.g. many employees occupy one room.</td><td>4)E.g. exactly one (maximum 1 and minimum 1) room is occupied by zero or many (maximum many and minimum 0) employees.</td></tr>
</table>
</td><td>4M points each</td></tr>
<tr><td></td><td>e</td><td><b>Explain seven major tasks of Requirement Engineering.</b></td><td><b>4 M</b></td></tr>
<tr><td></td><td>Ans</td><td>
<p><b>Requirements engineering tasks:</b></p>
<p><b>1.Inception</b>: - Inception means beginning. It is usually said that requirement engineering is a—communication intensive activity. The customer and developer meet and they the overall scope and nature of the problem statements. By having proper inception phase the developer will have clear idea about the system and as a result of that better understanding of a system can be achieved. Once the system is clear to the developer, they can implement a system with better efficiency.</p>
<p><b>2. Elicitation</b>: - Elicitation task will help the customer to define the actual requirement of a system. To know the objectives of the system or the project to be developed is a critical job. This phase will help people to determine the goal of a system and clear idea about the system can be achieved.</p>
<p><b>3.Elaboration</b>: - The information obtained from the customer during inception and elicitation is expanded and refined during elaboration. This requirement engineering activity focuses on developing a refined technical model of software functions, features and constraints.</p>
<p><b>4. Negotiation</b>: - This phase will involve the negotiation between what user actual expects from the system and what is actual feasible for the developer to build. Often it is seen that user always expect lot of things</p>
</td><td>7points for 4M</td></tr>
</table>

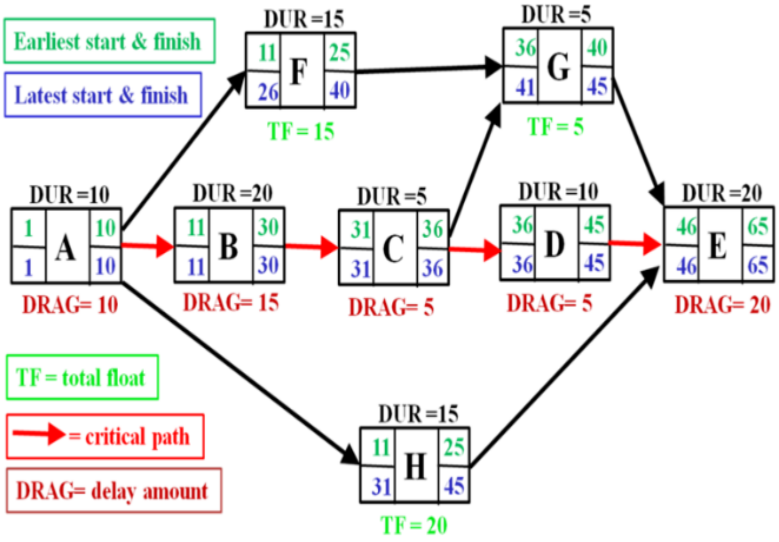
		from the system for lesser cost. But based on the other aspect and feasibility of a system the customer and developer can negotiate on the few key aspects of the system and then they can proceed towards the implementation of a system . **5. Specification**: - A specification can be a re-written document, a set of graphical models, a formal mathematical model, a collection of usage scenario, a prototype, or any combinations of these. The specification is the final work product produced by the requirement engineers. It serves as the foundation for subsequent software engineering activities. It describes the function and performance of a computer-based system and the constraints that will govern its development. **6.Validation**: - The work products produced as a consequence of requirements engineering are assessed for quality during a validation step. Requirements validation examines the specification to ensure that all software requirements have been stated unambiguously; that inconsistencies, omissions and errors have been detected and corrected, and that the work products conform to the standards established for the process, the project, and the product. **7. Requirements management**: - Requirement management begins with identification. Each requirement is assigned a unique identifier. Once requirement have been identified, traceability tables are developed.	
	f	**Explain software development process umbrella activities.**	**4 M**
	Ans	The phases and related steps of the generic view of software engineering are complemented by a number of umbrella activities. Typical activities in this category include: **01. Software project tracking and control:** When plan, tasks, models all have been done then a network of software engineering tasks that will enable to get the job done on time will have to be created. **02. Formal technical reviews:** This includes reviewing the techniques that has been used in the project.  Common Process Framework Framework Activities Task Sets Tasks Milestones, Deliverables SQA Points Umbrella Activities	4M for explanation

		03. Software quality assurance: This is very important to ensure the quality measurement of each part to ensure them. **04. Software configuration management:** Software configuration management (SCM) is a set of activities designed to control change by identifying the work products that are likely to change, establishing relationships among them, defining mechanisms for managing different versions of these work products. **05. Document preparation and production:** All the project planning and other activities should be hardly copied and the production gets started here. **06. Reusability management:** This includes the backing up of each part of the software project they can be corrected or any kind of support can be given to them later to update or upgrade the software at user/time demand. **07. Measurement:** This will include all the measurement of every aspects of the software project. **08. Risk management:** Risk management is a series of steps that help a software team to understand and manage uncertainty. It’s a really good idea to identify it, assess its probability of occurrence, estimate its impact, and establish a contingency plan that– ‘should the problem actually occur’.	
4.		**Attempt any Three of the following:**	**16M**
	a	**Describe software engineering as a Layered Technology.**	**4 M**
	Ans	**Software engineering as a Layered Technology can be explained as follows: -** **1. A quality Process: -**Any engineering approach must rest on an quality. The "Bed Rock" that supports software Engineering is Quality. **2. Process: -**Foundation for SE is the Process Layer SE process is the glue that holds all the technology layers together and enables the timely development of computer software. It forms the base for management control of software project. Process defines a framework that must be established for effective delivery of software engineering technology. IT includes works products (models, documents, data, reports, forms etc.) are produced, milestones are established, quantity is ensured and change is properly managed. **3. Methods: -**SE methods provide the "Technical Questions" for building Software. Methods contain a broad array of tasks that include communication requirement analysis, design modeling, program, construction, testing and support.	1M for diagram and 3M Explanation

		4. Tools: -SE tools provide automated or semi-automated support for the "Process" and the "Methods". Tools are combined and interrelated so that information created by one tool can be used by another.  **Figure: Flowchart of the Layers of Software Development**	
	b	**Explain Development Principles.**	**4 M**
	Ans	**The development principles are as follows: -** **Preparation principles**: Before you write one line of code, be sure you • Understand of the problem you're trying to solve. • Understand basic design principles and concepts. • Pick a programming language that meets the needs of the software to be built and the environment in which it will operate. • Select a programming environment that provides tools that will make your work easier. • Create a set of unit tests that will be applied once the component you code is completed. **Programming principles**: As you begin writing code, be sure you •Constrain your algorithms by following structured programming [Boh00] practice. • Consider the use of pair programming. • Select data structures that will meet the needs of the design. • Understand the software architecture and create interfaces that are consistent with it.	4M for explanation

		• Keep conditional logic as simple as possible. • Create nested loops in a way that makes them easily testable. • Select meaningful variable names and follow other local coding standards. · Write code that is self-documenting. • Create a visual layout (e.g., indentation and blank lines) that aids understanding. **Validation Principles**: After you've completed your first coding pass, be sure you • Conduct a code walkthrough when appropriate. • Perform unit tests and correct errors you've uncovered. • Refactor the code.	
	c	**Describe data objects and data attributes**	**4 M**
	Ans	**A data object** can be an external entity (e.g., anything that produces or consumes information), a thing (e.g., a report or a display), an occurrence (e.g., a telephone call) or event (e.g., an alarm), a role (e.g., salesperson), an organizational unit (e.g., accounting department), a place (e.g., a warehouse), or a structure (e.g., a file). For example, a person or a car (Figure 6.2) can be viewed as a data object in the sense that either can be defined in terms of a set of attributes. **Attributes:** Attributes define the properties of a data object and take on one of three different characteristics. They can be used to: (1) name an instance of the data object (2) describe the instance (3) make reference to another instance in another table.  Stu_Name Stu_Id Stu_Addr Col_ID Col_Name Beginnersbook.com Student StudyIn College In the above example, student and college are entities and Stu_ID, Col_ID are attributes.	2M for each description

	d	**Explain Brute Force approaches used as debugging strategies.**	**4 M**
	Ans	**Brute Force:** It is the most common and least efficient method for isolating the cause of a software error. Brute force debugging methods are applied when all else fails. Using a "let the computer find the error" philosophy, memory dumps are taken, run-time traces are invoked, and the program is loaded with WRITE statements. In the morass of information that is produced a clue is found that can lead us to the cause of an error. Although the mass of information produced may ultimately lead to success, it more frequently leads to wasted effort and time. Thought must be expended first. The brute force debugging process starts with the test cases. It gives two results, i.e. the cause is found and corrected second is the cause is not found.  Test Cases Result Debugging Suspected causes Additional tests Regression tests Identified causes Corrections **Fig. - Debugging process**	4M for explanation
	e	**Explain CPM**	**4 M**
	Ans	**Critical Path Method (CPM)** **i.** CPM is a project planning technique that is used in projects that have predictable activities and tasks such as in construction projects. **ii.** It allows project planners to decide which aspect of the project to reduce or increase when a trade-off is needed. **iii.** It is a deterministic tool and provides an estimate on the cost and the amount of time to spend in order to complete the project. **iv**. It allows planners to control both the time and cost of the project. **v.** CPM uses a single estimate for the time that a project can be completed. **vi.**CPM is best suited for routine and those projects where the project is performed for projects where time and cost estimates can the first time and the estimate of duration be accurately calculated are uncertain.	4M for explanation

		 Earliest start & finish Latest start & finish DUR =15 11 F 25 26 40 TF = 15 DUR =5 36 G 40 41 45 TF = 5 DUR =10 1 A 10 1 10 DRAG= 10 DUR =20 11 B 30 11 30 DRAG= 15 DUR =5 31 C 36 31 36 DRAG= 5 DUR =10 36 D 45 36 45 DRAG= 5 DUR =20 46 E 65 46 65 DRAG= 20 TF = total float = critical path DRAG= delay amount DUR =15 11 H 25 31 45 TF = 20	
	f	**What is Quality Control?**	**4 M**
	Ans	**Software quality control** is the set of procedures used by organizations to ensure that a software product will meet its quality goals at the best value to the customer, and to continually improve the organization's ability to produce software products in the future. Software quality control refers to specified functional requirements as well as non-functional requirements such as supportability, performance and usability.QC aims to identify defects - It's a Corrective technique.It is a method to verify the quality-Validation.Its main motive is to identify defects or bugs in the system. **SQC Activities** It includes the following activities: • **Reviews** ○ Requirement Review ○ Design Review ○ Code Review ○ Deployment Plan Review ○ Test Plan Review ○ Test Cases Review	4M for relevant explanation

		• **Testing** o Unit Testing o Integration Testing o System Testing o Acceptance Testing	
5.		**Attempt any Three of the following:**	**16M**
	a	**How effort distribution is done?**	**4 M**
	ns.	• Each of the software project estimation techniques leads to estimates of work units (e.g., person-months) required to complete software development. • A recommended distribution of effort across the software process is often referred to as the 40–20–40 rule. Forty percent of all effort is allocated to frontend analysis and design. A similar percentage is applied to back-end testing and coding (20 percent of effort) is deemphasized. • This effort distribution should be used as a guideline only. • The characteristics of each project dictate the distribution of effort. Work expended on project planning rarely accounts for more than 2 to 3 percent of effort, unless the plan commits an organization to large expenditures with high risk. • Customer communication and requirements analysis may comprise 10 to 25 percent of project effort. Effort expended on analysis or prototyping should increase in direct proportion with project size and complexity. • A range of 20 to 25 percent of effort is normally applied to software design. Time expended for design review and subsequent iteration must also be considered. • Because of the effort applied to software design, code should follow with relatively little difficulty. • A range of 15 to 20 percent of overall effort can be achieved. Testing and subsequent debugging can account for 30 to 40 percent of software development effort. • The criticality of the software often dictates the amount of testing that is required.	Description: 4 M; any relevant description shall be considered.
	b	**Describe RMMM Plan.**	**4 M**
	Ans	RMMM Plan 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.	Description – 4 Any relevant description shall be considered

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.

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

An effective strategy must consider three issues:

- Risk avoidance
- Risk monitoring
- Risk management and contingency planning.

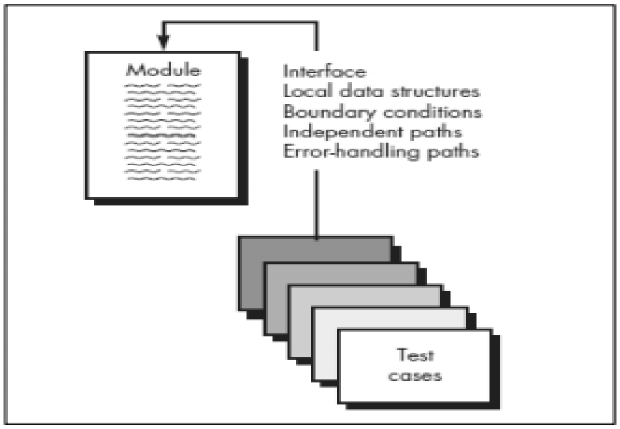
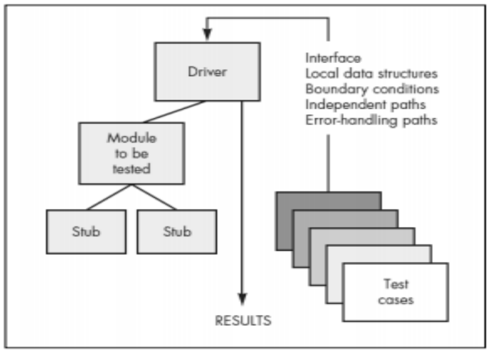
If a software team adopts a proactive approach to risk; avoidance is always the best strategy. This is achieved by developing a plan for risk mitigation. To mitigate this risk, project management must develop a strategy for reducing turnover Among the possible steps to be taken are

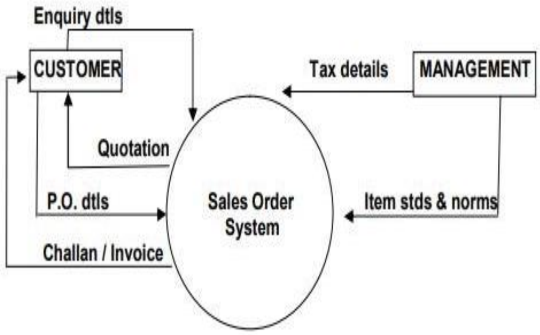
- Meet with current staff to determine causes for turnover (e.g., poor working conditions, low pay, and competitive job market).
- Mitigate those causes that are under our control before the project starts.
- Once the project commences, assume turnover will occur and develop techniques to ensure continuity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

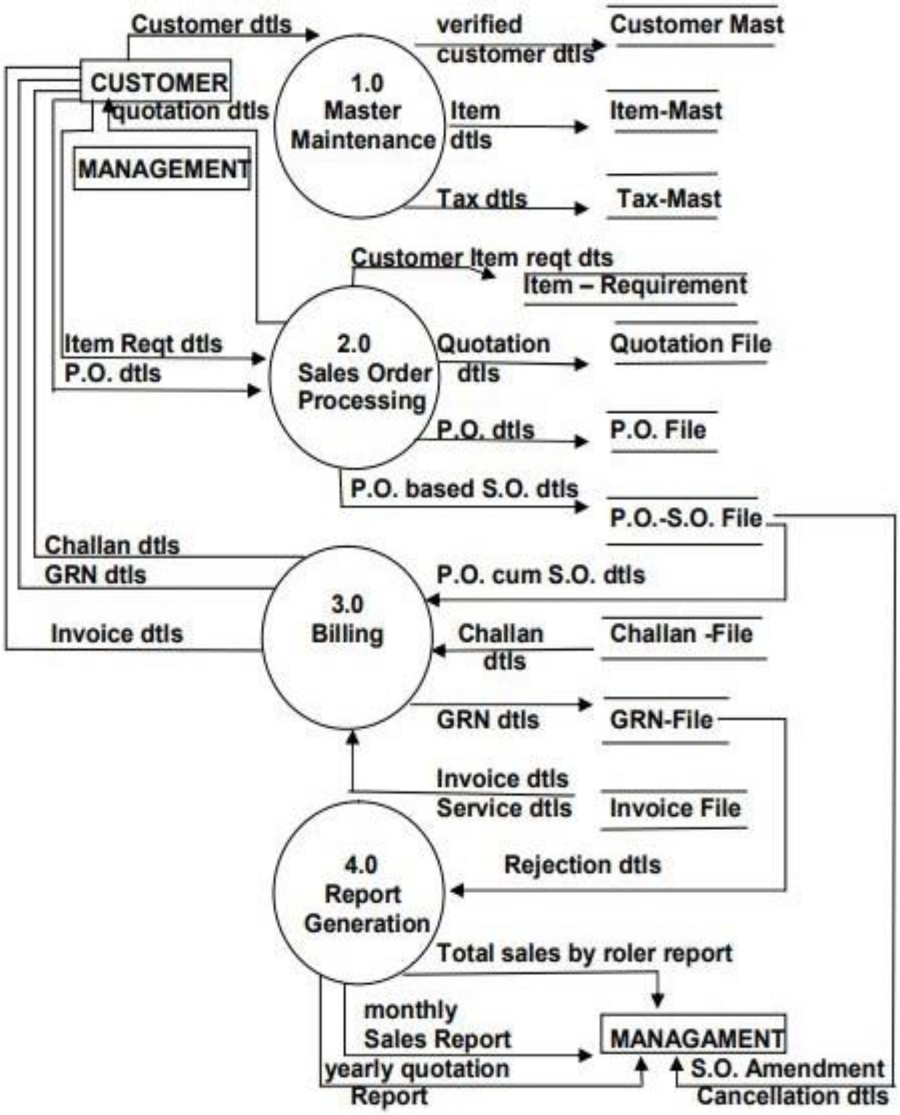
As the project proceeds, risk monitoring activities commence. The project manager monitors factors that may provide an indication of whether the risk is becoming more or less likely.

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

- General attitude of team members based on project pressures.

		• The degree to which the team has jelled. • Interpersonal relationships among team members. • Potential problems with compensation and benefits. • The availability of jobs within the company and outside it	
	c	**Explain Unit testing in detail.**	**4 M**
	Ans	 Module Interface Local data structures Boundary conditions Independent paths Error-handling paths Test cases **OR**  Driver Interface Local data structures Boundary conditions Independent paths Error-handling paths Module to be tested Stub Stub RESULTS Test cases • Unit testing focuses verification effort on the smallest unit of software design—the software component or module. • Using the component-level design description as a guide, important control paths are tested to uncover errors within the boundary of the module. • The relative complexity of tests and the errors those tests uncover is limited by the constrained scope established for unit testing. • The unit test focuses on the internal processing logic and data structures within the boundaries of a component. This type of testing can be conducted in parallel for multiple components. Unit tests are illustrated schematically in above Figure	Description – 3 Many one Diagram 1 M Any relevant description shall be considered

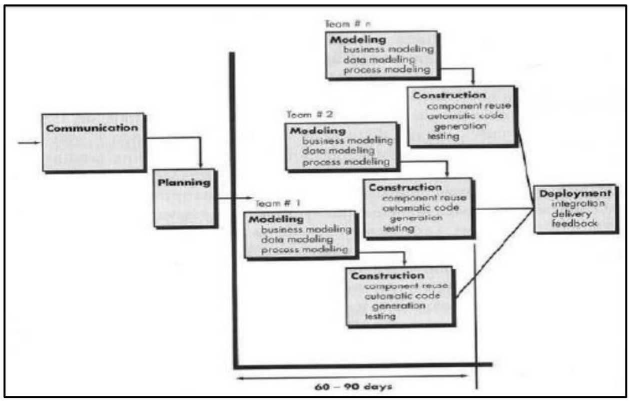
		•The module interface is tested to ensure that information properly flows into and out of the program unit under test. • Local data structures are examined to ensure that data stored temporarily maintains its integrity during all steps in an algorithm's execution. • All independent paths through the control structure are exercised to ensure that all statements in a module have been executed at least once. • Boundary conditions are tested to ensure that the module operates properly at boundaries established to limit or restrict processing. • And finally, all error-handling paths are tested. The unit test environment is illustrated in above Figure. •A driver is nothing more than a "main program" that accepts test case data, passes such data to the component (to be tested), and prints relevant results. • Stubs serve to replace modules that are subordinate (invoked by) the component to be tested. A stub or "dummy subprogram" uses the subordinate module's interface, may do minimal data manipulation, prints verification of entry, and returns control to the module undergoing testing.	
	d	**Development DFD for sales order processing of a department store.**	**4 M**
	Ans	 Enquiry dtls CUSTOMER Tax details MANAGEMENT Quotation P.O. dtls Sales Order System Item stds & norms Challan / Invoice **Level 0 DFD of Sales order processing**	Correct DFD diagram 4M diagrams; Any relevant diagram shall be considered



Level 1DFD of Sales order processing

	e	**Explain four principles of analysis modeling**	**4 M**
	Ans	**Principle 1:** The information domain of a problem must be represented and understood**.** The information domain encompasses the data that flow into the system, the data that flow out of the system, and the data stores that collect and organize persistent data objects. **Principle 2:** The functions that the software performs must be defined**.** Software functions provide direct benefit to end users and also provide internal support for those features that are user visible. Some functions transform data that flow into the system. In other cases, functions affect some level of control over internal software processing or external system elements. Functions can be described at many different levels of abstraction, ranging from a general statement of purpose to a detailed description of the processing elements that must be invoked. **Principle 3:** The behavior of the software (as a consequence of external events) must be represented**.** The behavior of computer software is driven by its interaction with the external environment. Input provided by end users, control data provided by an external system, or monitoring data collected over a network all cause the software to behave in a specific way. **Principle 4:** The models that depict information function and behavior must be partitioned in a manner that uncovers detail in a layered (or hierarchical) fashion. Requirement's modeling is the first step in software engineering problem solving. It allows you to better understand the problem and establishes a basis for the solution. Complex problems are difficult to solve in their entirety. For this reason, you should use a divide-and-conquer strategy. A large, complex problem is divided into sub problems until each sub problem is relatively easy to understand. This concept is called partitioning or separation of concerns, and it is a key strategy in requirements modeling. **Principle 5:** The analysis task should move from essential information toward implementation detail. Requirements modeling begin by describing the problem from the end-user 's perspective. The essence of the problem is described without any consideration of how a solution will be implemented.	Any 4 Principle; 1 M each principle
	f	**Explain RAD model with neat diagram.**	**4 M**
	Ans	• Rapid application on Development (RAD) is a modern software process model that emphasizes a short development cycle. • The RAD Model is a ―high-speed‖ adaptation of the waterfall model, in which rapid development is achieved by using a	Description – 2 M; Diagram 2 M Any relevant description

component-based construction approach.

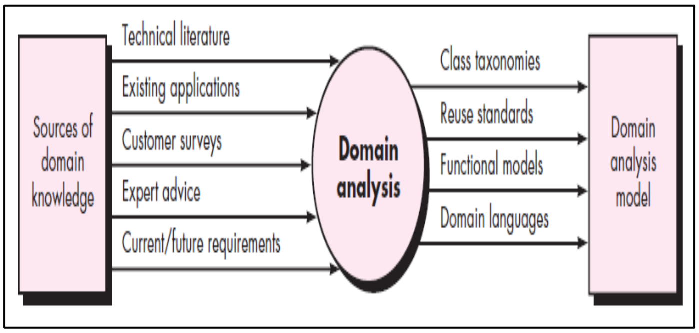


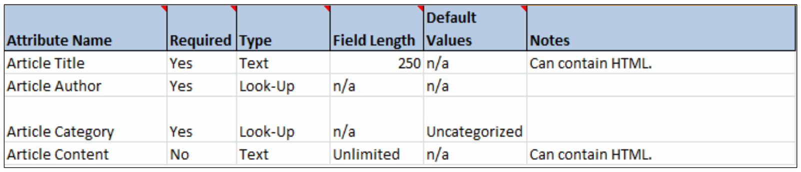
- If requirements are well understood and project scope is considered, the RAD process enables a development team to create a ―Fully Functional System within a very short period of time (e.g. 60 to 90 days).
- One of the distinct features of RAD model is the possibility of cross life cycle activities which will be assigned to teams, teams #1 to team #n leading to each module getting developed almost simultaneously.
- This approach is very useful if the business application requirements are modularized as function to be completed by individual teams and finally to integrate into a complete system.
- As such compared to waterfall model the team will be of larger size to function with proper coordination.
- RAD model distributes the analysis and construction phases into a series of short iterative development cycles.
- The activities of each phase per team are Business modeling, Data modeling and process modeling.
- This model is useful for projects with possibility of modularization.
- RAD may fail if modularization is difficult.
- This model should be used if domain experts are available with relevant business knowledge.
- Communication works to understand the business problem and the information characteristics that the software must accommodate.
- Planning is essential because multiple software teams ‘work is parallel on different system functions.
- Modeling encompasses three major phases – business modeling, data modeling and process modeling- and establishes design representations that serve as the basis for RAD‘s construction activity.

shall be considered

		• Construction emphasizes the use of pre-existing software components and the application of automatic code generation. • Finally, deployment establishes a basis for subsequent iterations, if required. **Advantages:** 1. Changing requirements can be accommodated and progress can be measured. 2. Powerful RAD tools can reduce development time. 3. Productivity with small team in short development time and quick reviews, risk control increases reusability of components, better quality. 4. Risk of new approach only modularized systems is recommended through RAD. 5. Suitable for scalable component-based systems. **Limitations/Disadvantages:** 1. RAD model success depends on strong technical team expertise and skills. 2. Highly skilled developers needed with modeling skills. 3. User involvement throughout life cycle. If developers & customers are not committed to the rapid-fire activities necessary to complete the System in a much-abbreviated time frame, RAD projects will fail. 4. May not be appropriate for very large-scale systems where the technical risks are high.	
6.		**Attempt any Four of the following:**	**16M**
	a	**What are PSP and TSP framework activities? Explain their meaning.**	**4 M**
	Ans	**PSP model defines following five frame work activities:** **Planning-** isolates requirements, develops size and resource estimates. Tests are identified and project schedule is created. **High level design:** External specification for each component to be constructed is developed and a component design is created. **High level design review:** formal verification methods are applied to uncover errors in the design. **Development:** component level design is refined and reviewed. Code is generated, reviewed, compiled and tested. **Post-mortem:** Using measures and matrix collected, the effectiveness of the process is determined. They provide guidance for improvement. **TSP defines the following framework activities:** **Project Launch:** It reviews core objective and describes the TSP	PSP activities Explanation 2M, TSP activities explanation 2M

		structure and content. It assigns terms and roles to developers and describes the customer needs statement. It also establishes team and individual goals. **High Level Design:** It creates high-level design, specifies the design, and inspects the design develop an integration test plan. **Implementation:** Implementation uses the TSP to implement modules/unit, creates a detailed design of modules/units, reviews the design, translates the design to code, review the code, compile and test the modules/units and analyze the quality of the modules/units. **Integration and Test:** Testing builds and integrates these builds into a system. It conducts a system test and produce user documentation. **Post-mortem**: It conducts a post-mortem analysis, writes a cycle report and produce peer and team evaluations.	
	b	**Write importance of SRS.**	**4 M**
	Ans	**The importance SRS:** 1. Establish the basis for agreement between the customers and the suppliers on what the software product is to do. 2. The complete description of the functions to be performed by the software specified in the SRS will assist the potential users to determine if the software specified meets their needs or how the software must be modified to meet their needs. 3. Reduce the development effort. The preparation of the SRS forces the various concerned groups in the customer's organization to consider rigorously all of the requirements before design begins and reduces later redesign, recoding, and retesting. 4. Careful review of the requirements in the SRS can reveal omissions, misunderstandings, and inconsistencies early in the development cycle when these problems are easier to correct. 5. Provide a basis for estimating costs and schedules. The description of the product to be developed as given in the SRS is a realistic basis for estimating project costs and can be used to obtain approval for bids or price estimates. 6. Provide a baseline for validation and verification. Organizations can develop their validation and Verification plans much more productively from a good SRS. As a part of the development contract, the SRS provides a baseline against which compliance can be measured. 7. Facilitate transfer. The SRS makes it easier to transfer the software product to new users or new machines. Customers thus find it easier to transfer the software to other	Importance of SRS – 4 M; Any relevant description shall be considered

		8. parts of their organization, and suppliers find it easier to transfer it to new customers. Serve as a basis for enhancement. Because the-SRS discusses the product but not the project that developed it, the SRS serves as a basis for later enhancement of the finished.	
	c	**What is meant by Domain Analysis in modeling?**	**4 M**
	Ans	• Analysis patterns often reoccur across many applications within a specific business domain. If these patterns are defined and categorized in a manner that allows you to recognize and apply them to solve common problems, the creation of the analysis model is expedited. • More important, the likelihood of applying design patterns and executable software components grows dramatically. This improves time-to-market and reduces development costs.  Sources of domain knowledge Technical literature Existing applications Customer surveys Expert advice Current/future requirements Domain analysis Class taxonomies Reuse standards Functional models Domain languages Domain analysis model • Domain Analysis gives answer of following questions: • How are analysis patterns and classes recognized in the first place? • Who defines them, categorizes them, and readies them for use on subsequent projects? • Software domain analysis is the identification, analysis, and specification of common requirements from a specific application domain, typically for reuse on multiple projects within that application domain,[Object-oriented domain analysis is] the identification, analysis, and specification of common, reusable capabilities within a specific application domain, in terms of common objects, classes, subassemblies, and frameworks. • The "specific application domain" can range from avionics to banking, from multimedia video games to software embedded within medical devices. • The goal of domain analysis is straightforward: to find or create those analysis classes and/or analysis patterns that are broadly applicable so that they may be reused.	Description – 4 M; Diagram optional Any relevant description shall be considered

<table>
<tr><td></td><td></td><td>• Domain analysis may be viewed as an umbrella activity for the software process. Domain analysis is an ongoing software engineering activity that is not connected to any one software project.
• The role of the domain analyst is to discover and define analysis patterns, analysis classes, and related information that may be used by many people working on similar but not necessarily the same applications.</td><td></td></tr>
<tr><td></td><td>d</td><td><b>Explain Data dictionary with diagram.</b></td><td><b>4 M</b></td></tr>
<tr><td></td><td>Ans</td><td>Data dictionaries are lists of all of the names used in the system models. Descriptions of the entities, relationships and attributes are also included. Data dictionary as the central database for the description of all data objects. Once entries in the dictionary are defined, entity-relationship diagrams can be created and object hierarchies can be developed. Many CASE workbenches support data dictionaries.

<table>
<tr><th>Attribute Name</th><th>Required</th><th>Type</th><th>Field Length</th><th>Default Values</th><th>Notes</th></tr>
<tr><td>Article Title</td><td>Yes</td><td>Text</td><td>250</td><td>n/a</td><td>Can contain HTML.</td></tr>
<tr><td>Article Author</td><td>Yes</td><td>Look-Up</td><td>n/a</td><td>n/a</td><td></td></tr>
<tr><td>Article Category</td><td>Yes</td><td>Look-Up</td><td>n/a</td><td>Uncategorized</td><td></td></tr>
<tr><td>Article Content</td><td>No</td><td>Text</td><td>Unlimited</td><td>n/a</td><td>Can contain HTML.</td></tr>
</table></td><td>Description – 2 M; Diagram 2 M Any relevant description shall be considered</td></tr>
<tr><td></td><td>e</td><td><b>State eight characteristics of software bug.</b></td><td><b>4 M</b></td></tr>
<tr><td></td><td>Ans</td><td>1. The symptom and the cause may be geographically remote. That is, the symptom may appear in one part of a program, while the cause may actually be located at a site that is far removed. Highly coupled program structures exacerbate this situation.
2. The symptom may disappear (temporarily) when another error is corrected.
3. The symptom may actually be caused by non-errors (e.g., round-off inaccuracies).
4. The symptom may be caused by human error that is not easily traced.
5. The symptom may be a result of timing problems, rather than processing problems.
6. It may be difficult to accurately reproduce input conditions (e.g., a real-time application in which input ordering is indeterminate).
7. The symptom may be intermittent. This is particularly common in embedded systems that couple hardware and software inextricably.
8. The symptom may be due to causes that are distributed across a number of tasks running on different processors.</td><td>Each Characteristics – ½ M</td></tr>
</table>

	f	**Explain the types of risk.**	**4 M**
	Ans	1. **Generic risks** are a potential threat to every software project. 2. **Product-specific risks** can be identified only by those with a clear understanding of the technology, the people, and the environment. 3. **Product size**—risks associated with the overall size of the software to be built or modified that is specific to the software that is to be built. 4. **Business impact**—risks associated with constraints imposed by management or the marketplace. 5. **Project risks** threaten the project plan. That is, if project risks become real, it is likely that the project schedule will slip and that costs will increase. 6. **Technical risks** threaten the quality and timeliness of the software to be produced. 7. **Business risks** threaten the viability of the software to be built and often jeopardize the project or the product. 8. **Known risks** are those that can be uncovered after careful evaluation of the project plan, the business and technical environment in which the project is being developed, and other reliable information sources. 9. Predictable risks are extrapolated from past project experience of user. 10. **Unpredictable risks** are one that they can and do occur, but they are extremely difficult to identify in advance.	Any 4 types of risk; Each type of risk– 1 M