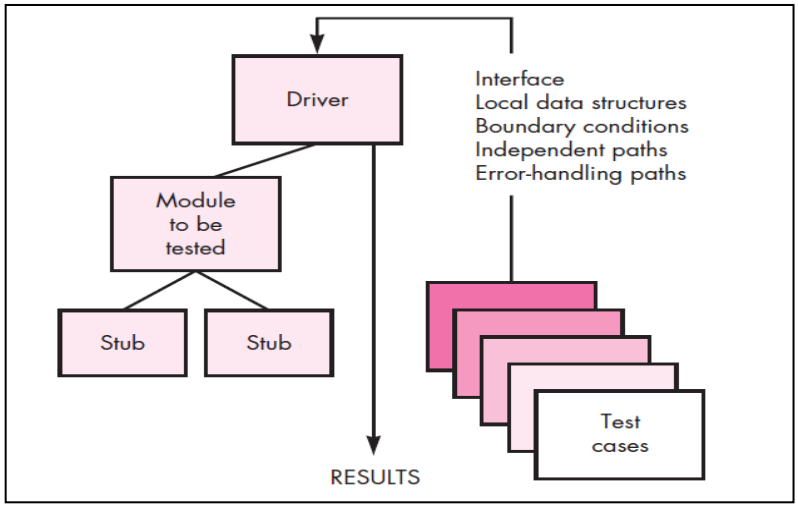
SUMMER– 18 EXAMINATION

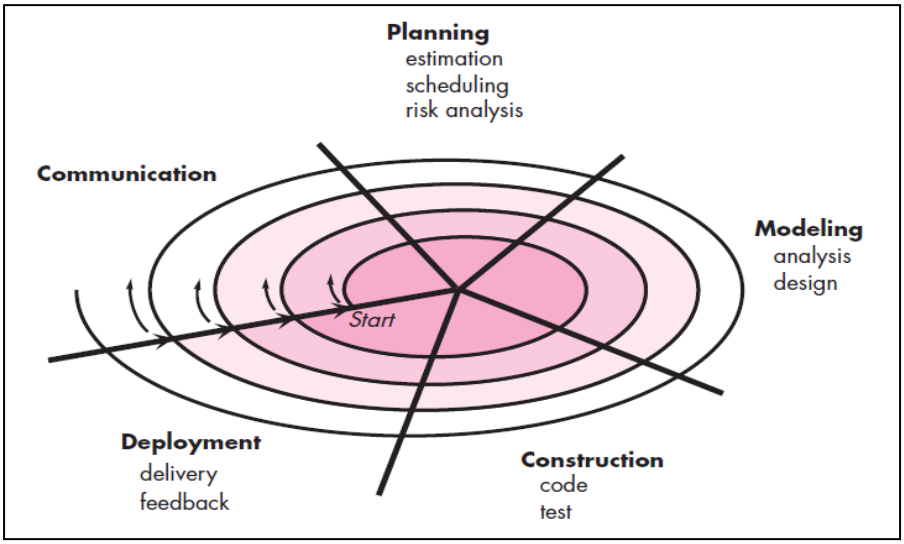
Subject Name: Software Engineering **Model Answer** **Subject Code:** 17513

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.	Answers	Marking Scheme
1.	**a)**	**Attempt any THREE of the following:**	**12 Marks**
	i)	**Define management spectrum and enlist characteristics of software.**	**4M**
	Ans:	**Management Spectrum:** The management spectrum describes the management/hierarchy of people associated with a software project or how to make a software project successful. It focuses on the four P's; People, Product, Process and Project. **Characteristics of software:** a) Software is developed or engineered; it is not manufactured in the classical sense. b) Software doesn't "wear out" like hardware and it is not degradable over a period. c) Although the industry is moving toward component-based construction, most software continues to be custom built. d) A software component should be designed and implemented so that it can be reused in many different programs. e) Software is not susceptible to the environmental maladies that cause hardware to wear out.	**(Definition: 1 mark, 3 Characteristics: 3 marks)**

	(ii)	**Draw stub and driver mechanism of unit testing and enlist various types of errors detected by unit testing.**	**4M**
	Ans:	 Types of errors detected by unit testing • Interface errors: - Connectivity between modules and functions along with function call. • Local data structure validation errors: - Testing local data structure during execution of program. • Boundary condition validations: - Validating boundary condition of loops • Verification of independent paths: - Errors that can be generated at independent flow path of module. • Error handling paths: - Validating exceptions and other run time errors.	**(Diagram: 1 mark, Errors: 3 marks)**
	(iii)	**Describe five steps for successfulness of project.**	**4M**
	Ans:	**{{Note:-Any other relevant steps shall be considered}}** **Step 1: Define the Scope of Project** While beginning with the project one shall define the scope of project. Once scope of project is define required analysis and design phase shall be completed. **Step 2: Set & Prioritize Goals; Establish measurable criteria for success**. After analysis is done one shall decide modules and also assign priority to each module. This gives preference to module and same shall be delivered first. This will also help to assess success of project.	**(Five steps: 4 marks)**

		Step 3: Create the Project Schedule and Test Deliverables The project manager must create a project schedule and verify deliverables during the process. **Step 4: Define Critical Project Milestones & Deliverables** In this step one shall define milestones in a project delivery. Milestones are small achievements that are completed during the completion of project. **Step 5: Test and Deliver project** This step ensures that the required modules are tested and delivered to the customers.	
	(iv)	**Draw the neat labelled diagram of spiral model and list two disadvantages of spiral model.**	**4M**
	Ans:	 **Disadvantages:** • Can be a costly model to use. • Risk analysis requires highly specific expertise. • Project's success is highly dependent on the risk analysis phase. • Doesn't work well for smaller projects. • It is not suitable for low risk projects. • May be hard to define objective, verifiable milestones. • Spiral may continue indefinitely.	**(Diagram: 2 marks, 2 Disadvantages: 1 mark each)**
	b)	**Attempt any ONE of the following:**	**6 Marks**
	(i)	**Elaborate any six types of software considering the changing nature.**	**6M**
	Ans:	**1. System Software**: System Software is a collection of programs written to serve other programs. Some system software (e.g.:- compliers, editors, and file management utilities) processes complex, but determinate information structures. Other system applications (e.g. operating system components, drivers, networking software, telecommunications processors) process largely indeterminate data. In either case, the systems software area is characterized by heavy interaction with	**(Any 6 Types: 1 mark each)**

computer hardware; heavy usage by multiple users; concurrent operation that requires scheduling, resource sharing, and sophisticated process management; complex data structures; and multiple external interfaces.

2. **Application Software:** Application Software consists of standalone programs that solve a specific business need. Applications in this area process business or technical data in a way that facilities business operations or management / technical decision making.

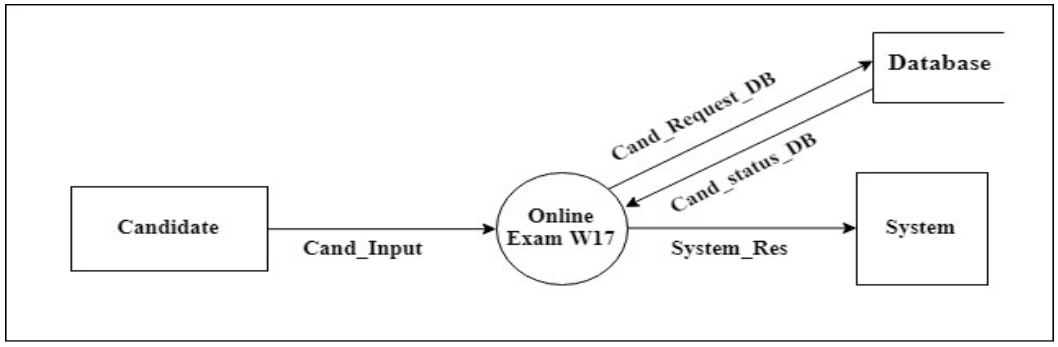
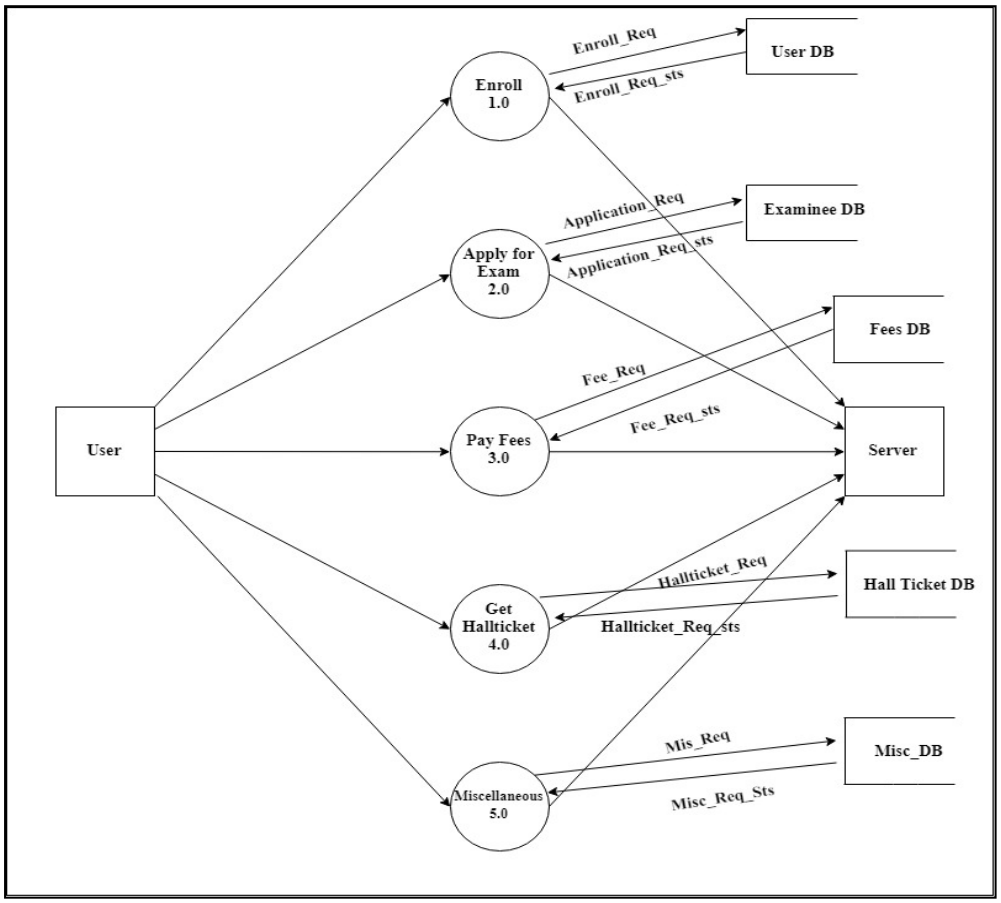
3. **Engineering / Scientific Software:** Formerly characterized by ―number crunching algorithms, engineering and scientific software applications range from astronomy to volcano logy, from automotive stress analysis to space shuttle orbital dynamics, and from molecular biology to automated manufacturing. Computer-aided design, system simulation, and other interactive applications have begun to take on real–time and even system software characteristics.

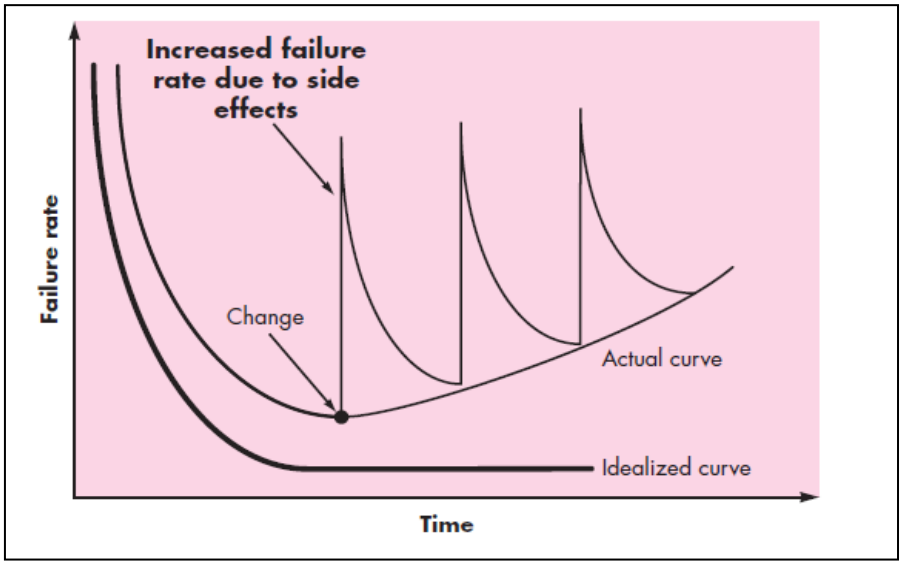
4. **Embedded Software:** Embedded Software resides within a product or system and is used to implement and control features and functions for the end-user and for the system itself. Embedded software can perform limited and esoteric functions (e.g. keypad control for a microwave oven) or provide significant function and control capability (e.g. digital functions in an automobile such as fuel control, dashboard displays, braking systems, etc.)

5. **Product–line Software:** Designed to provide a specific capability for use by many different customers, product–line software can focus on a limited & esoteric market place (e.g. – inventory control products) or address mass consumer markets (e.g. – word processing, spread-sheets, and computer graphics, and multimedia, entertainment, and database management, personal and business financial applications.)

6. **Web – applications:** Web Apps, span a wide array of applications. Web apps are evolving into sophisticated computing environments that not only provide standalone features, computing functions, and content to the end user, but also are integrated with corporate databases and business applications.

7. **Artificial Intelligence Software:** AI Software makes use of non–numerical algorithms to solve complex problems that are not amenable to computation or straightforward analysis. Applications within this area include robotics, expert systems, pattern recognition (image and voice), artificial neural networks, theorem proving, and game playing.

	(ii)	**Draw and explain level 0 and level 1 Dataflow diagram for "Online examination. Win17 of form filling on MSBTE website".**	**6M**
	Ans:	**{{Note:-Any other relevant diagram shall be considered}}**  DFD Level 0 for Online Examination win17 of form filling on MSBTE website In level 0, the candidate specifies his/her request to Online Exam W17 module. The module transfers request to Database. The Database returns the status and same will be transferred and reflected on System.  DFD Level 1 for Online Examination win17 of form filling on MSBTE website	**(Level 0: 2 marks, Level 1: 2 marks, Description: 1 mark each)**

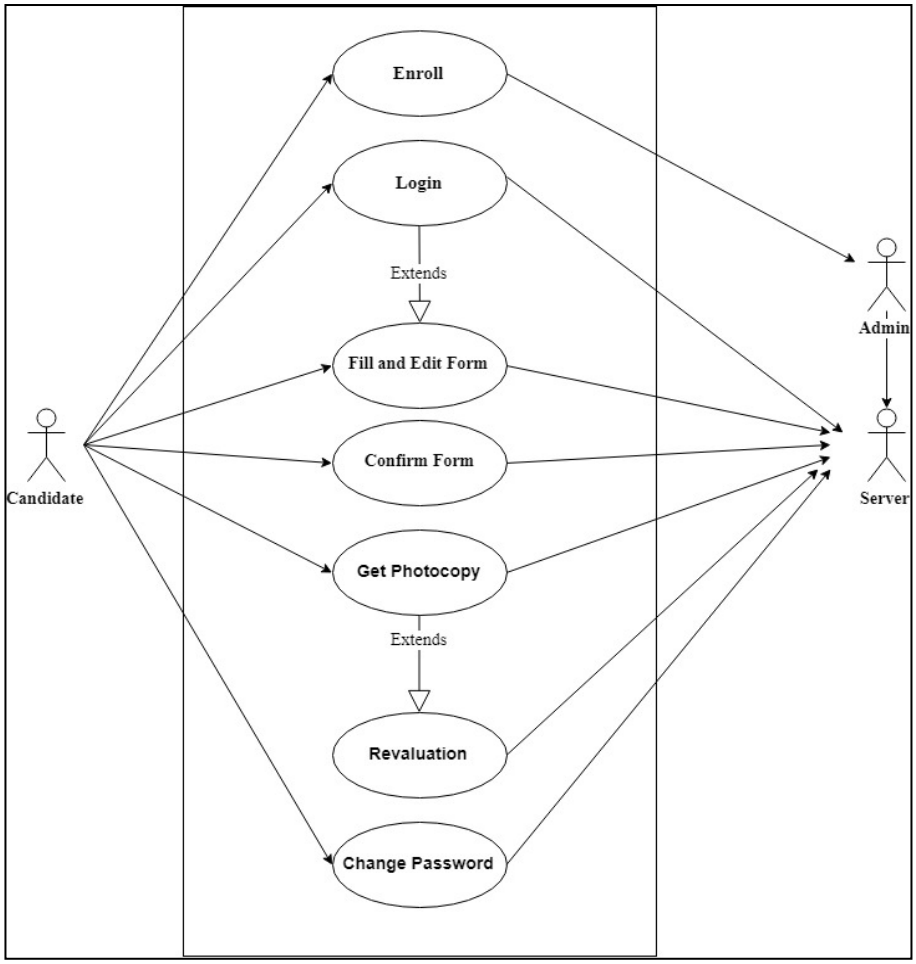
		In DFD Level 1, User/ Candidate can select any of the available menus, like enrol themselves, the module 2 allow them to apply for examination i.e. regular exam or backlog exam. In module 3 the candidate pays exam fees. In module 4 Student can get Hall-ticket as per their convenience. In module 5 the student can perform any other miscellaneous task such as report generation, change of password etc.	
2.		**Attempt any FOUR of the following:**	**16 Marks**
	a)	**Elaborate the software characteristic "Software does not wear out".**	**4M**
	Ans:	Software is not susceptible to the environmental maladies that cause hardware to wear out. Therefore, the failure rate for software should take the form of the "idealized" curve as shown in the diagram. Undiscovered defects will cause high failure rates early in the life of a program. However, these are corrected. However, the implication is clear—software doesn't wear out. But it does deteriorate! Another aspect of wear illustrates the difference between hardware and software. When a hardware component wears out, it is replaced by a spare part. There are no software spare parts. Every software failure indicates an error in design or in the process through which design was translated into machine executable code. Therefore, the software maintenance tasks that accommodate requests for change involve considerably more complexity than hardware maintenance.  **Failure Curve for Software**	**(Description: 4 marks , Diagram optional)**
	b)	**List and explain three principles of analysis modelling.**	**4M**
	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.	**(List :1 mark, Any 3 Description:1 mark each)**

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 effect 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 behaviour of the software (as a consequence of external events) must be represented. The behaviour 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 modelling 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 modelling.

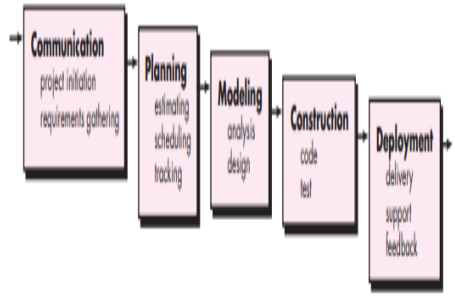
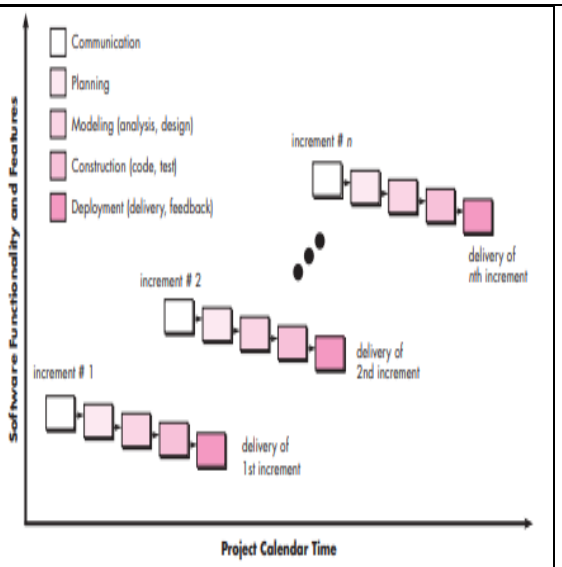
Principle 5: The analysis task should move from essential information toward implementation detail. Requirements modeling begin by describing the problem from the end-user's perspective. The essence of the problem is described without any consideration of how a solution will be implemented.

	c)	**Draw the usecase diagram for taking "photocopy of ansbooks from msbte" website.**	**4M**
	Ans:	**{{Note:-Any other relevant diagram shall be considered}}**  	**(Correct Use case: 4 marks)**
	d)	**Enlist advantages and disadvantages of smoke testing. (four points)**	**4M**
	Ans:	**Advantages:** **1.** Integration risk is minimized. Because smoke tests are conducted daily, incompatibilities and other show-stopper errors are uncovered early, thereby reducing the likelihood of serious schedule impact when errors are uncovered. **2.** The quality of the end product is improved. Because the approach is construction oriented, smoke testing is likely to uncover functional errors as well as architectural and component-level design errors. **3.** Error diagnosis and correction are simplified. Like all integration testing approaches, errors uncovered during smoke testing are likely to be associated with "new software increments". **4.** Progress is easier to assess. With each passing day, more of the software has been integrated and more has been demonstrated to work.	**(Advantages: 2 marks, Disadvantages: 2 Marks)**

		Disadvantages: **1.** The smoke test should exercise the entire system from end to end. **2.** The smoke test should be thorough enough that if the build passes, one can assume that it is stable enough to be tested more thoroughly. **3.** Needs to be done on regular basis hence require dedicated software team. **4.** The software is rebuilt and smoke tested every day. **5.** Smoke testing does not cover the detailed testing. **6.** It's a non-exhaustive testing with small number of test cases because of which we not are able to find the other critical issues. **7.** Smoke testing is not performed with negative scenarios and with invalid data.	
	e)	**Enlist and explain different types of Software Risks. (four points)**	**4M**
	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: 1 mark each)**

	f)	**Explain qualities of software considering.** **i) Quality of design** **ii) Quality of conformance**	**4M**
	Ans:	**i) Quality of design:** Quality of design refers to the characteristics that designers specify for a product. The grade of materials, tolerances, and performance specifications all contribute to the quality of design. As higher-grade materials are used, tighter tolerances and greater levels of performance are specified, the design quality of a product increases, if the product is manufactured according to specifications. In software development, quality of design encompasses the degree to which the design meets the functions and features specified in the requirements model. **ii) Quality of conformance:** Quality of conformance focuses on the degree to which the implementation follows the design and the resulting system meets its requirements and performance goals.	**(Quality of Design: 2 marks, Quality of Conformance : 2 marks)**
3.		**Attempt any FOUR of the following:**	**16 Marks**
	a)	**Give difference between waterfall model and incremental model. (four points)**	**4M**
	Ans:		**(Each Difference: 1 mark, any four difference)**

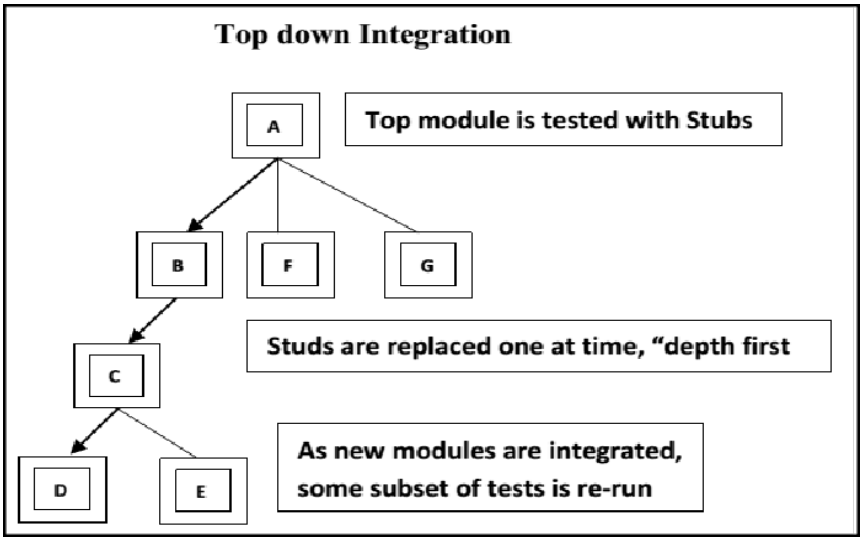
Sr. No	Waterfall Model	Incremental Model
1	When there is need to make well - defined adaptations or enhancements to an existing system and product definition is stable, waterfall model is used.	When there is need to provide limited set of functionality to users quickly and then refine and expand on that functionality in later Software releases, incremental approach is used.
2	It requires well understanding of requirement and familiar technology	Requirements of the complete system are clearly defined and understood
3	Sequential in nature	Incremental in nature
4	Difficult to accommodate changes after the process has started	Changes can be accommodated when planning for next increment
5	Risk can be identified at the end which may cause failure to the product	Risk can be identified in each Increment plan
6	The customer can see the working model of the project at the end. After review of the working model, if the customer get dissatisfied then it cause serious problems	The core product is used by the customer (or undergoes detailed review). As a result of use and/or evaluation, a plan is developed for next increment

		7	 	
	b)	**Explain following requirements of engineering tasks** **(i) Negotiation** **(ii) Validation**		**4M**
	Ans:	1. **Negotiation**: - 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 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 aspect of the system and then they can proceed towards the implementation of a system 2. **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.		**(Negotiation : 2 marks, Validation:2 marks)**
	c)	**Explain Architectural Design Elements.**		**4M**
	Ans:	• Requirements of the software should be transformed into an architecture that describes the software's top-level structure and identifies its components. • This is accomplished through architectural design (also called system design), which acts as a preliminary 'blueprint' from which software can be developed • The architecture design elements provides us overall view of the system.		**(Explanation: 4 marks)**

		• The architectural design element is generally represented as a set of interconnected subsystem that are derived from analysis packages in the requirement model. **The architecture model is derived from following sources:** • The information about the application domain to built the software. • Requirement model elements like data flow diagram or analysis classes, relationship and collaboration between them. • The architectural style and pattern as per availability. **Architectural Design Representation** Architectural design can be represented using the following models. 1. Structural model 2. Dynamic model 3. Process model 4. Functional model 5. Framework model **Architectural Design Output** The architectural design process results in an Architectural Design Document (ADD). This document consists of a number of graphical representations that comprises software models along with associated descriptive text. The software models include static model, interface model, relationship model, and dynamic process model. They show how the system is organized into a process at run-time.	
	d)	**State eight characteristics of software bugs.**	**4M**
	Ans:	**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.	**(For each point: ½ mark)**

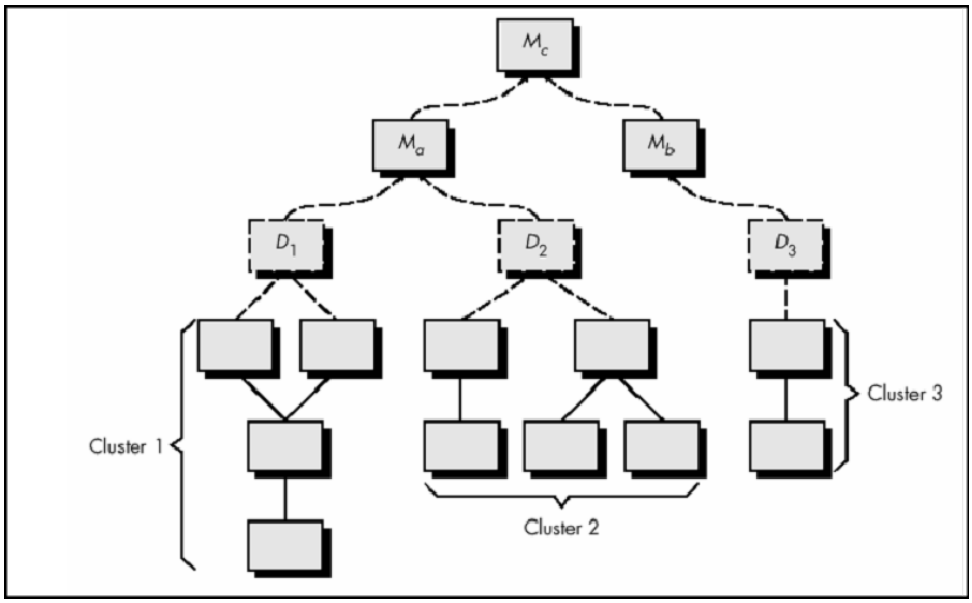
	e)	**Explain the functions of Software Configuration Management repository. (SCM)**	**4M**
	Ans:	The SCM repository is the set of mechanisms and data structures that allow a software team to manage change in an effective manner. It provides the obvious functions of a modern database management system by ensuring data integrity, sharing, and integration. In addition, the SCM repository provides a hub for the integration of software tools, is central to the flow of the software process, and can enforce uniform structure and format for software engineering work products. To achieve these capabilities, the repository is defined in terms of a meta-model. The meta-model determines how information is stored in the repository, how data can be accessed by tools and viewed by software engineers, how well data security and integrity can be maintained, and how easily the existing model can be extended to accommodate new needs. Some of the functions are described below: **1. Data Integrity:-** It includes functions to validate entries to the repository, ensure consistency among related objects and automatically perform "cascading" modifications when a change to one object demands some change to objects related to it. **2. Information sharing:-** provides a mechanism for sharing information among multiple developers and between multiple tools, manages and controls multiuser access to data, and locks or unlocks objects so that changes are not inadvertently overlaid on one another. **3. Tool Integration:-** Establishes a data model that can be accessed by many software engineering tools, controls access to the data, and performs appropriate configuration management functions. **4. Data Integration: -** provides database functions that allow various SCM tasks to be performed on one or more SCIs. **5. Methodology Enforcement:-** It defines an entity- relationship model stored in the repository that implies a specific process model for software engineering; at a minimum, the relationships and objects define a set of steps that must be conducted to build the contents of the repository. **6. Document Standardization:-** It is the definition of objects in the database that leads directly to a standard approach for the creation of software engineering documents	**(Any 4 functions: 1mark each)**

4.	a)	**Attempt any THREE of the following:**	**12 Marks**
	(i)	**Explain the testing concept with its Testing Principles. (any four principles)**	**4M**
	Ans:	Testing is an activity that is used to discover errors and correct them, so that we are able to create a defect free product for the customer. Software testing is a process of executing a program or software with the intent of finding an error. Testing begins at the component level and works "outward" toward the integration of the entire computer-based system. Different testing techniques are appropriate for different software engineering approaches and at different points in time. Testing is conducted by the developer of the software and (for large projects) an independent test group. **1. All tests should be traceable to customer requirements.** The objective of software testing is to uncover errors. It follows that the most severe defects (from the customer's point of view) are those that cause the program to fail to meet its requirements. **2. Tests should be planned long before testing begins.** Test planning can begin as soon as the requirements model is complete. Detailed definition of test cases can begin as soon as the design model has been solidified. Therefore, all tests can be planned and designed before any code has been generated. **3. The Pareto principle applies to software testing.** Stated simply, the Pareto principle implies that 80 percent of all errors uncovered during testing will likely be traceable to 20 percent of all program components. The problem, of course, is to isolate these suspect components and to thoroughly test them. **4. Testing should begin "in the small" and progress toward testing "in the large."** The first tests planned and executed generally focus on individual components. As testing progresses, focus shifts in an attempt to find errors in integrated clusters of components and ultimately in the entire system.	**(Description : 2 marks, Any four principles: 2 marks)**

		5. Exhaustive testing is not possible. The number of path permutations for even a moderately sized program is exceptionally large. For this reason, it is impossible to execute every combination of paths during testing. It is possible, however, to adequately cover program logic and to ensure that all conditions in the component-level design have been exercised. **6. To be most effective, testing should be conducted by an independent third party.** By most effective, we mean testing that has the highest probability of finding errors (the primary objective of testing). The software engineer who created the system is not the best person to conduct all tests for the software.	
	(ii)	**Compare Bottom-up integration testing and top-down integration testing. (four points)**	**4M**
	Ans:	**Top- down integration** • Main control module used as a test driver and stubs are substitutes for components directly subordinate to it. • Subordinate stubs are replaced one at a time with real components (following the depth-first or breadth-first approach). • Tests are conducted as each component is integrated. • On completion of each set of tests and other stub is replaced with a real component. Regression testing may be used to ensure that new errors not introduced.  	**(Each point of comparison: 1 mark, any 4 points shall be considered)**

Bottom- up integration

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



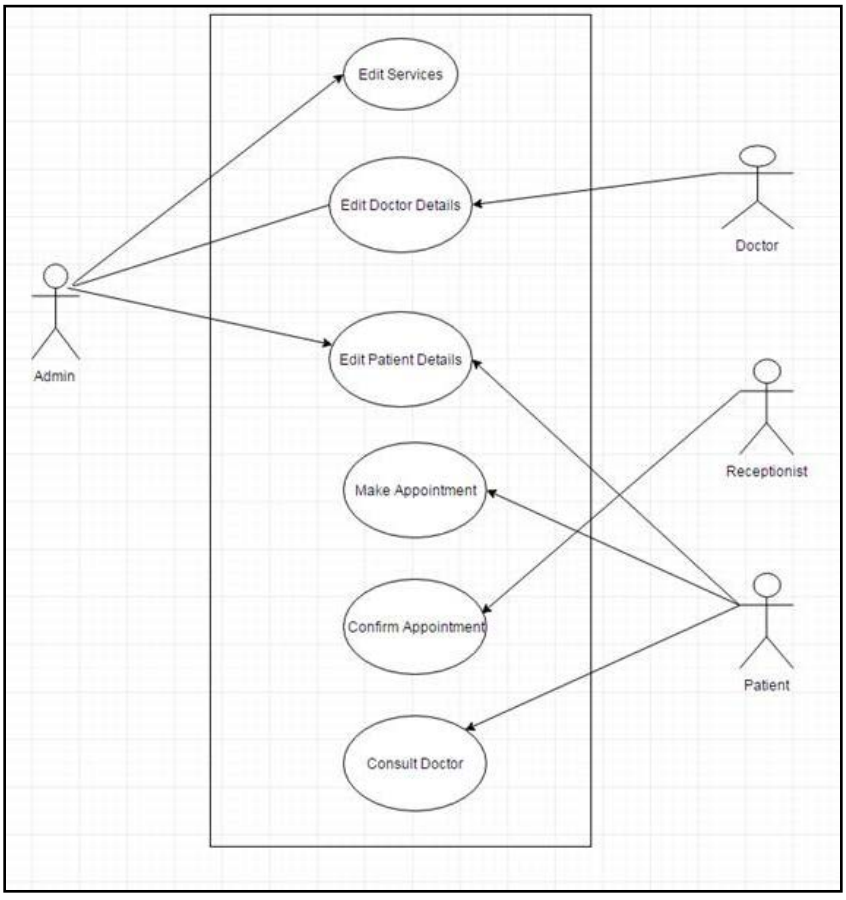
OR

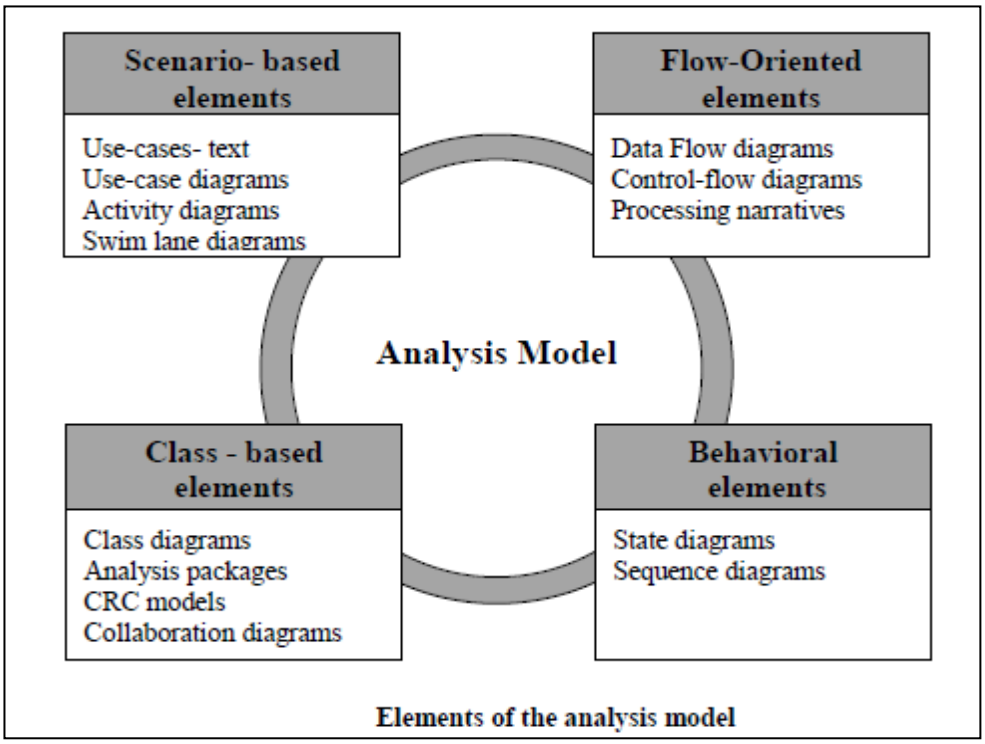
Top- Down integration	Bottom –Up Integration
This is incremental approach to construction of the software architecture.	Bottom up integration begins with sub modules and atomic checking
Modules are integrated by moving downward through the control hierarchy, beginning with the main control module (main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

			Depth-first integration integrates all components on a major control path of the program structure.	Different clusters are formed for the testing.
	(iii)	**Explain the factors that Delay Project Schedule.**		**4M**
	Ans:	Although there are many reasons why software is delivered late, most can be traced to one or more of the following root causes: **1.** An unrealistic deadline established by someone outside the software development group and forced on managers and practitioners within the group. **2.** Changing customer requirements that are not reflected in schedule changes. **3.** An honest underestimate of the amount of effort and/or the number of resources that will be required to do the job. **4.** Predictable and/or unpredictable risks that were not considered when the project commenced. **5.** Technical difficulties that could not have been foreseen in advance. **6.** Human difficulties that could not have been foreseen in advance. **7.** Miscommunication among project staff that results in delays. **8.** A failure by project management to recognize that the project is falling behind schedule and a lack of action to correct the problem.	**(Any four factors:4 marks)**
	(iv)	**Explain the six sigma for software engineering.**		**4M**
	Ans:	**Six Sigma** is the most widely developed by Motorola in 1980, used strategy for statistical quality assurance in industry today. Six Sigma strategy "is a rigorous and disciplined methodology that uses data and statistical analysis to measure and improve a company's operational performance by identifying and eliminating defects' in manufacturing and service-related processes" **DMAIC** The DMAIC project methodology has five phases: **Define** the system, the voice of the customer and their requirements, and the project goals, specifically. **Measure** key aspects of the current process and collect relevant data.	**(Explanation: 4 marks)**

		Analyze the data to investigate and verify cause-and-effect relationships. Determine what the relationships are, and attempt to ensure that all factors have been considered. Seek out root cause of the defect under investigation. **Improve** or optimize the current process based upon data analysis using techniques such as design of experiments or mistake proofing, and standard work to create a new, future state process. Set up pilot runs to establish process capability. **Control** the future state process to ensure that any deviations from target are corrected before they result in defects. Implement control systems such as statistical process control, production boards, visual workplaces, and continuously monitor the process. Some organizations add a Recognize step at the beginning, which is to recognize the right problem to work on. **DMADV** The DMADV project methodology, known as DFSS ("Design For Six Sigma"), features five phases it has first three phases same as DMAIC : **Design** an improved alternative, best suited per analysis in the previous step **Verify** the design, set up pilot runs, implement the production process and hand it over to the process owner(s).	
	b)	**Attempt any <u>ONE</u> of the following:**	**6 Marks**
	(i)	**List and explain five framework activities defined in PSP (Personal software process).**	**6M**
	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.	**(List:1 mark, Explanation of five activities: 5 marks)**

	(ii)	**Explain 4 Ps is software project spectrum.**	**6M**
	Ans:	**The People** • People factor is very much important in the process of software development. • There are following areas for software people like, recruiting, selection performance management, training, compensation, career development, organization and work design, and team/culture development. • Organizations achieve high levels of maturity in the people management area. **The Product** • 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. • Without this information, it is impossible to define reasonable estimates of the cost, an effective assessment of risk, a realistic breakdown of project tasks, or a manageable project schedule. • Objectives identify the overall goals for the product without considering how these goals will be achieved. Scope identifies the primary data, functions and behaviors that characterize the product. • Once the product objectives and scope are understood, alternative solutions are considered. From the available various alternatives, managers and practitioners select a "best" approach. **The Process** • A software process provides the framework from which a comprehensive plan for software development can be established. • A small number of frame-work activities are applicable to all software projects, regardless of their size or complexity. • A number of different tasks, milestones, work products and quality assurance points enable the framework activities to be adapted to the characteristics of the software project and the requirements of the project team. • Finally, umbrella activities such as software quality assurance, software configuration management, and measurement overlay the process model.	**(All 4 Ps: 6 Marks)**

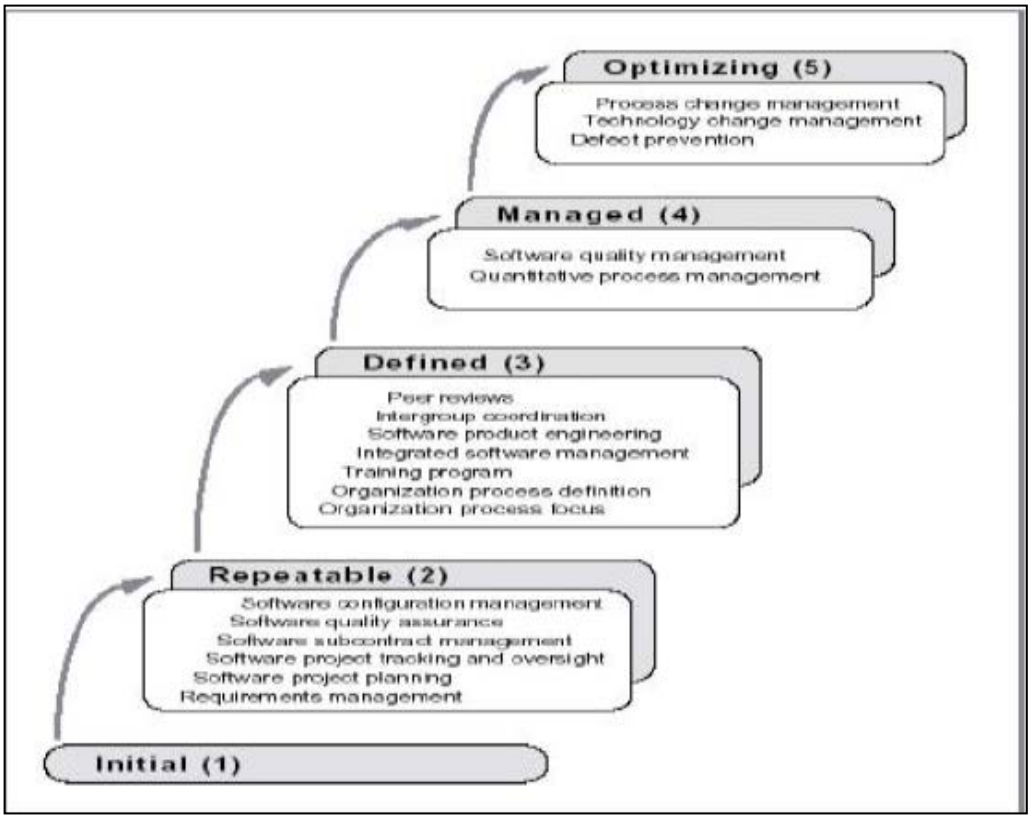
		The Project • We conduct planned and controlled software projects for one primary reason it is the only known way to manage complexity. • A software project manager and the software engineers who build the product must avoid a set of common warning signs, understand the critical success factors that lead to good project management, and develop a common sense approach for planning, monitoring and controlling the project.	
5.		**Attempt any TWO of the following:**	**16 Marks**
	a)	**Draw the behavioral analysis model for small hospital management system and illustrate the working of it.**	**8M**
	Ans:	**{{Note:-Any other relevant diagram shall be considered}}**  • The Admin actor can edit services provided by the hospital. Admin can also make necessary changes in Doctor details and Patient details as per requirement. • The Doctor actor specifies their details for latest updates.	**(Diagram:4 marks, Description: 4 marks)**

		• Patient request for appointment for assistance, once the details are matching the Receptionist confirms the appointments. • The patients consults with Doctor as per the Appointment given to them.	
	b)	**List and explain the elements of analysis model with neat labeled diagram.**	**8M**
	Ans:	**List of elements of analysis model:** **1. Flow-oriented modeling** **2. Scenario-based modeling** **3. Class-based modeling** **4. Behavioral modeling**  **Elements of the analysis model** • **Flow-oriented modeling** – provides an indication of how data objects are transformed by a set of processing functions • **Scenario-based modeling** – represents the system from the user's point of view • **Class-based modeling** – defines objects, attributes, and relationships • **Behavioral modeling** – depicts the states of the classes and the impact of events on these states	**(List:2marks, Diagram: 2marks, Explanation: 4marks)**

	c)	**Explain different levels of Capability Maturity Model Integration technique. (CMMI)**	**8M**
	Ans:	The Capability Maturity Model Integration (CMMI), a comprehensive process meta-model that is predicated on a set of system and software engineering capabilities that should be present as organizations reach different levels of process capability and maturity. The CMMI represents a process meta-model in two different ways :(1) Continuous model and (2) Staged model. The continuous CMMI meta-model describes a process in two dimensions. Each process area (e.g. project planning or requirements management) is formally assessed against specific goals and practices and is rated according to the following capability levels: **Level 0**: Incomplete—the process area (e.g., requirements management) is either not performed or does not achieve all goals and objectives defined by the CMMI for level 1 capability for the process area. **Level 1**: Performed—all of the specific goals of the process area (as defined by the CMMI) have been satisfied. Work tasks required to produce defined work products are being conducted. **Level 2:** Managed—all capability level 1 criteria have been satisfied. In addition, all work associated with the process area conforms to an organizationally defined policy; all people doing the work have access to adequate resources to get the job done; stakeholders are actively involved in the process area as required; all work tasks and work products are ―monitored, controlled, and reviewed; and are evaluated for adherence to the process description‖. **Level 3:** Defined—all capability level 2 criteria have been achieved. In addition, the process is ―tailored from the organization‘s set of standard processes according to the organization‘s tailoring guidelines, and contributes work products, measures, and other process-improvement information to the organizational process assets‖. **Level 4:** Quantitatively managed—all capability level 3 criteria have been achieved. In addition, the process area is controlled and improved using measurement and quantitative assessment. ―Quantitative objectives for quality and process performance are established and used as criteria in managing the process‖. **Level 5:** Optimized—all capability level 4 criteria have been achieved. In addition, the process area is adapted and optimized using quantitative (statistical) means to meet	**(Diagram: 3 marks, Description:5 marks)**

changing customer needs and to continually improve the efficacy of the process area under consideration.

OR



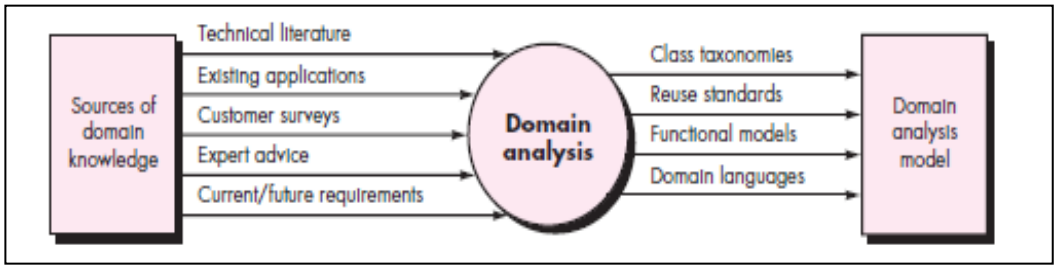
Level 1: Initial. The software process is characterized as ad hoc and occasionally even chaotic. Few processes are defined, and success depends on individual effort.

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.	
6.		**Attempt any FOUR of the following:**	**16 Marks**
	a)	**Explain principles of planning practices in software engineering (any four)**	**4M**
		Principle 1. Understand the scope of the project. It's impossible to use a road map if you don't know where you're going. Scope provides the software team with a destination. **Principle 2. Involve stakeholders in the planning activity.** Stakeholders define priorities and establish project constraints. To accommodate these realities, software engineers must often negotiate order of delivery, time lines, and other project-related issues. **Principle 3. Recognize that planning is iterative.** A project plan is never engraved in stone. As work begins, it is very likely that things will change. As a consequence, the plan must be adjusted to accommodate these changes. In addition, iterative, incremental process models dictate re-planning after the delivery of each software increment based on feedback received from users. **Principle 4. Estimate based on what you know.** The intent of estimation is to provide an indication of effort, cost, and task duration, based on the team's current understanding of the work to be done. If information is vague or unreliable, estimates will be equally unreliable. **Principle 5. Consider risk as you define 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 define 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. 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. **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,	**(Any four principles: 1mark each)**

		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.	
	b)	**Explain input and output of domain analysis.**	**4M**
	Ans:	**Input and Output for Domain Analysis:**  The role of domain analyst is to discover and define reusable analysis patterns, analysis classes and related information that may be used by many people working on similar but not necessarily the same applications. Input domain refers to all methodologies that are useful for gathering information of system to get acquainted with system. Good Input domain analysis leads to better understanding of system and ensure quality software development roadmap. Output domain refers to the result of methodologies that are used in Input Domain analysis. This gives a breakthrough for next step of SDLC in form of reusing modules that are already exists, finalizing platform, Basic models that will be part of system etc.	**(Diagram: 2marks, Description: 2marks)**
	c)	**Define white box testing and black box testing with its need and characteristics. (two points)**	**4M**
	Ans:	**White Box testing**: sometimes called glass-box testing, is a test-case design philosophy that uses the control structure described as part of component-level design to derive test cases. Using white-box testing methods, you can derive test cases that (1) Guarantee that all independent paths within a module have been exercised at least once, (2) Exercise all logical decisions on their true and false sides, (3) Execute all loops at their boundaries and within their operational bounds, and (4) Exercise internal data structures to ensure their validity.	**(White Box testing needs & characteristics:2 marks, Black Box testing needs &**

		Black Box testing: also called behavioral testing, focuses on the functional requirements of the software. That is, black-box testing techniques enable you to derive sets of input conditions that will fully exercise all functional requirements for a program. Black box testing is not an alternative to white-box techniques. Rather, it is a complementary approach that is likely to uncover a different class of errors than white box methods. Black-box testing attempts to find errors in the following categories: (1) Incorrect or missing functions, (2) Interface errors, (3) Errors in data structures or external database access, (4) Behavior or performance errors, and (5) Initialization and termination errors. **OR** **White Box testing**: Sometimes called glass-box testing, is a test-case design philosophy that uses the control structure described as part of component-level design to derive test cases. Need: - To assess and validate the code and internal structure of program/code. Characteristics: - Code is visible for Software Tester so they can verify correctness of the code. **Black Box testing**: Also called behavioral testing, focuses on the functional requirements of the software. That is, black-box testing techniques enable you to derive sets of input conditions that will fully exercise all functional requirements for a program. Need: To assess the correctness of the behavior of the Software. Characteristics: - To validate functional behavior and desired outcome/flow of the system.	**characteristics:2 marks)**
	d)	**Explain different activities done to track the software project.**	**4M**
	Ans:	**{{Note:-Any other relevant activities shall be considered}}** • Conducting periodic project status meetings in which each team member reports progress and problems. • Evaluating the results of all reviews conducted throughout the software engineering process. • Determining whether formal project milestones have been accomplished by the scheduled date. • Comparing the actual start date to the planned start date for each project task listed in the resource table. • Meeting informally with practitioners to obtain their subjective assessment of progress to date and problems on the horizon.	**(Any four activities: 1mark each)**

		• Using earned value analysis to assess progress quantitatively.	
	e)	**Prepare any four software quality assurance guidelines and describe them.**	**4M**
	Ans:	**{{Note:-Any other relevant guidelines shall be considered}}** **Product Quality: -** While preparing SQA one shall specify the quality of product i.e. least quality accepted for product. Normally it has to be 100% bug free system is desired but to meet deadline one shall define minimum working modules. Data Quality: - This specifies the quality of input and output data along with their data structure. While processing of input decided data structure is likely to change so SQA team needs to set guidelines for same and shall be note down in deviation made in software project. **Meet User Needs: -** Understand the compliance steps of software product and ensure all modules meets users' requirement. Security: - This step specifies security concerns of the software in view of Database Security, Internal data security. Also it shall consider access level to the unauthorized end users. **Functional Safety: -** There should be proper guidelines for the functions used in software specifying coupling and cohesion of the modules and functions. This allows the developers to make necessary changes in the system as per the changes specified by the customers. **Standards: -** While preparing for SQA guidelines the SQA team needs to specify the standards that software will follow i.e. IEEE or ISO along with its specification. This will be helpful to convey appropriate message about software in front of customer regarding quality standards. **Reviews and audits: -** All feedbacks and reviews received regarding projects shall be stored in centralized repository. These feedbacks may be used in later phases of project. Testing and Error/defect collection with analysis: - There needs to be definite procedure that needs to be follow as far as testing is concerns. All errors and defects needs to be collected and analyze to reduce testing time in subsequent modules/ projects. **Change management: -** Audit changes in the system as deviation from earlier road map with valid reason. Same shall be conveying to both stakeholders and higher authorities. **Risk management: -** Decide the strategy to be used in case of any risk. If proactive strategy is adopted one shall specify steps to be followed to encounter such risk. If reactive strategy is employed the team needs to record successful steps in order to tackle similar kind of risk that may arise in future.	**(Any four guidelines:1 mark each)**