USN □□□□□□□□□□

CELEBRATING 25 YEARS CMRIT CMR INSTITUTE OF TECHNOLOGY, BENGALURU. ACCREDITED WITH A+ GRADE BY NAAC

Internal Assessment Test 4– March 2022

Sub:	Software Engineering					Sub Code:	18CS35	Branch	ISE
Date:	15/03/2022	Duration:	3 Hrs	Max Marks:	100	Sem/Sec:	III / A, B and C		OBE

		Answer any FIVE FULL Questions	MARKS	CO	RBT
		Module 1			
1	a	What is software engineering? Bring out the differences between generic and bespoke products. List software engineering attributes	10	CO1	L2
	b	Illustrate requirement engineering process with a neat block diagram.	10	CO1	L3
		or			
2	a	With a neat block diagram, explain the spiral process model.	10	CO1	L2
	b	What is ethnography? How ethnography is effective in discovering the types of requirements.	10	CO1	L2
		Module 1,2			
3	a	Explain briefly class model, state model and interaction model.	10	CO 3	L2
	b	Briefly explain links, associations, ordering, bags, and sequences with an example.	10	CO3	L2
		or			
4	a	Explain the following important terms with example? i) Identity ii) Classification iii) Inheritance iv) Polymorphism	10	CO3	L2
	b	With a neat diagram, explain the waterfall model of software development process?	10	CO1	L2
		Module 2,5			
5	a	Write and explain UML notation for objects and classes, values and attributes.	10	CO4	L2
	b	Define the purpose of the following terms with suitable example and UML notation with respect to class model a) Multiplicity b) Association class.	10	CO4	L1
		or			
6	a	Illustrate Algorithmic cost modeling	10	CO5	L3
	b	Explain COCOMO II model.	10	CO5	L2
		Module 1,5			
7	a	Explain Incremental Development process model with a neat block diagram. List its benefits and problems.	10	CO2	L2
	b	Explain the IEEE standard requirement document with its structure.	10	CO2	L2
		or			
8	a	With a neat diagram explain project scheduling process.	10	CO5	L2
	b	Explain briefly the key stages in the process of product measurement	10	CO5	L2
		Module 1,5			
9	a	Discuss software quality and its attributes. Explain process based quality.	10	CO4	L2
	b	Explain software review and inspections of quality assurance.	10	CO4	L2
		or			
10	a	Explain the various section and supplements of the project plan.	10	CO5	L2
	b	Explain requirement elicitation and analysis process	10	CO1	L2

Faculty Signature CCI Signature HOD Signature

USN	□□□□□□□□□□							CELEBRATING 25 YEARS CMRIT CMR INSTITUTE OF TECHNOLOGY, BENGALURU. ACCREDITED WITH A+ GRADE BY NAAC
Internal Assessment Test 4 – March 2022								
Sub:	Software Engineering				Sub Code:	18CS35	Branch:	ISE
Date:	15/3/2022	Duration:	180 min's	Max Marks: 100	Sem/Sec:	III / A, B and C		OBE

	Answer any FIVE FULL Questions	MARKS	CO	RBT
1	a. What is software engineering? Bring out the differences between generic and bespoke products. List software engineering attributes **Solution:** Software engineering is an engineering discipline that is concerned with all aspects of software production. 1. Generic products These are stand-alone systems that are produced by a development organization and sold on to buy them. Examples of this type of product include software for PCs such as databases, word processors, drawing packages, and project-management tools. It also includes so-called vertical applications designed for some specific purpose such as library information systems, accounting systems, or systems for maintaining dental records. 2. Customized (or bespoke) products These are systems that are commissioned by a particular customer. A software contractor develops the software especially for that customer. Examples of this type of software include control systems for electronic devices, systems written to support a particular business process, and air traffic control systems. The attributes of software engineering are Maintainability Dependability and security Efficiency Acceptability	10	CO1	L2
	b. Illustrate requirement engineering process with a neat block diagram. **Solution:** There are four main activities in the requirements engineering process: 1. Feasibility study: An estimate is made of whether the identified user needs may be satisfied using current software and hardware technologies. The study considers whether the proposed system will be cost-effective from a business point of view and if it can be developed within existing budgetary constraints. 2. Requirements elicitation and analysis This is the process of deriving the system requirements through observation of existing systems, discussions with potential users and procurers, task analysis, and so on. This may involve the development of one or more system models and prototypes. These help you understand the system to be specified. 3. Requirements specification Requirements specification is the activity of translating the information gathered during the analysis activity into a document. 4. Requirements validation This activity checks the requirements for realism, consistency, and completeness. During this process, errors in the requirements document are inevitably discovered. It must then be modified to correct these problems.	10	CO1	L3

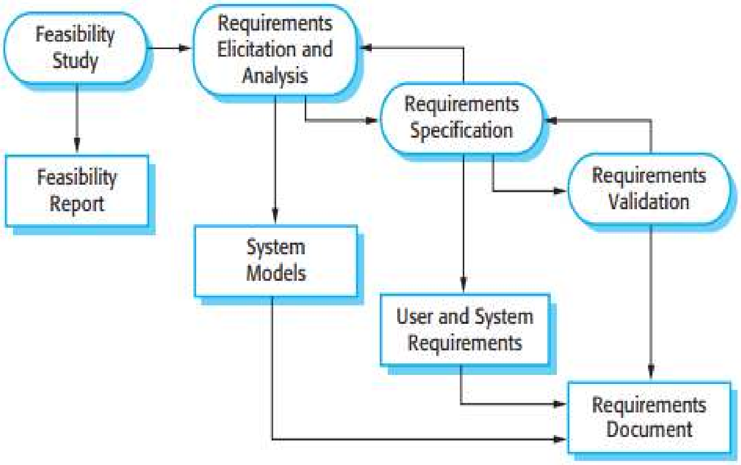


Fig: Requirement Engineering Process

2 a. With a neat block diagram, explain the spiral process model. 10 CO1 L2

Solution:

Each loop in the spiral is split into four sectors:

1. Objective setting Specific objectives for that phase of the project are defined. Constraints on the process and the product are identified and a detailed management plan is drawn up. Project risks are identified. Alternative strategies, depending on these risks, may be planned.

2. Risk assessment and reduction for each of the identified project risks, a detailed analysis is carried out. Steps are taken to reduce the risk. For example, if there is a risk that the requirements are inappropriate, a prototype system may be developed.

3. Development and validation After risk evaluation, a development model for the system is chosen. For example, throwaway prototyping may be the best development approach if user interface risks are dominant. If safety risks are the main consideration, development based on formal transformations may be the most appropriate process, and so on. If the main identified risk is sub-system integration, the waterfall model may be the best development model to use.

4. Planning The project is reviewed and a decision made whether to continue with a further loop of the spiral. If it is decided to continue, plans are drawn up for the next phase of the project.

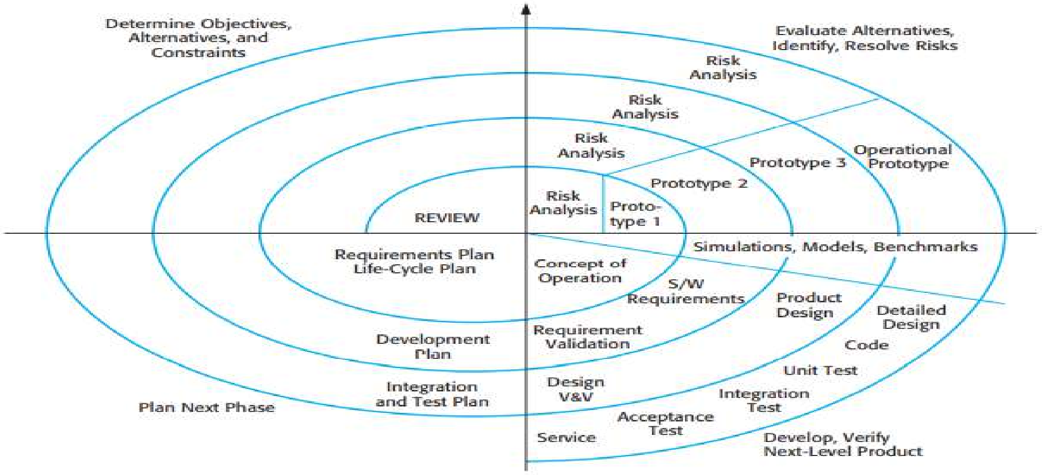
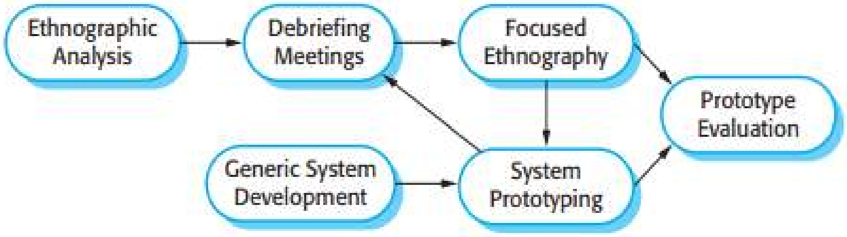
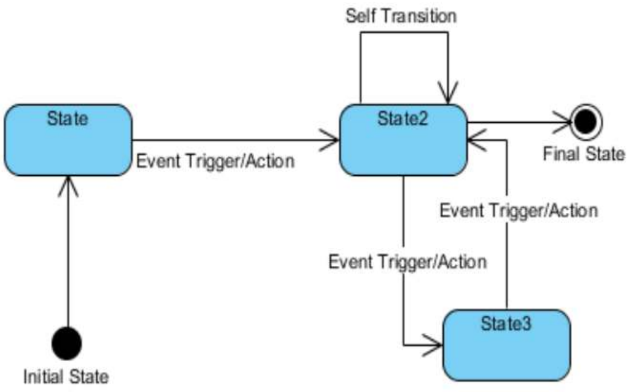
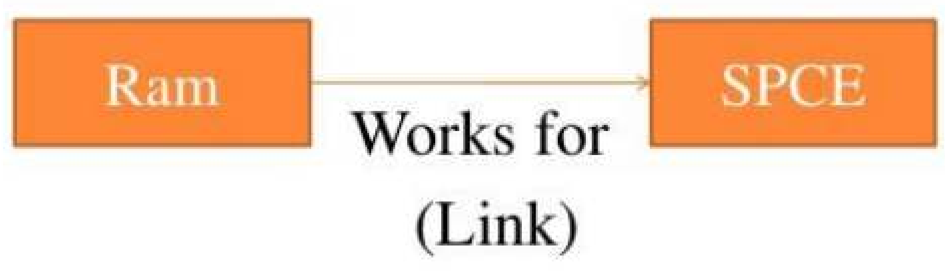
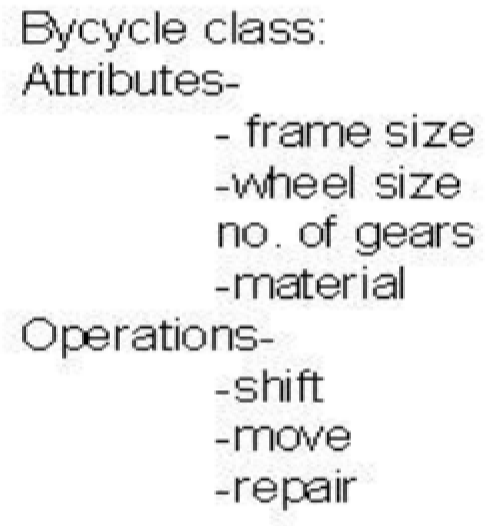
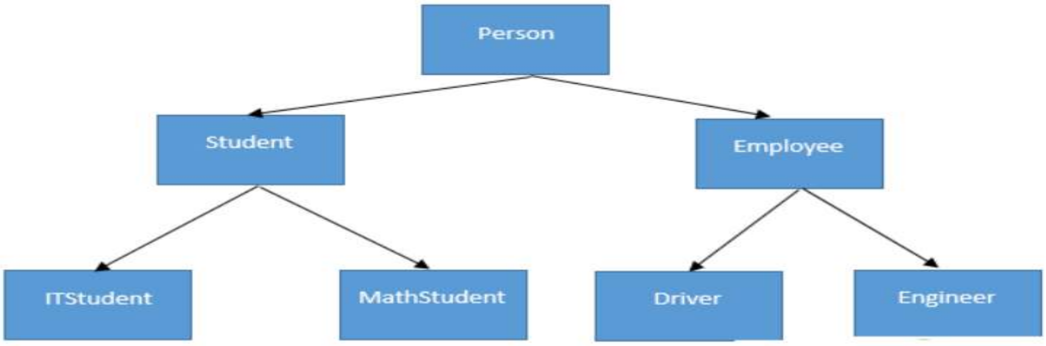


Fig: Spiral Process

2	b. What is ethnography? How ethnography is effective in discovering the types of requirements. **Solution:** Ethnography is an observational technique that can be used to understand operational processes and help derive support requirement for these processes. Ethnography is particularly effective for discovering two types of requirements: 1. Requirements that are derived from the way in which people actually work, rather than the way in which process definitions say they ought to work. For example, air traffic controllers may switch off a conflict alert system that detects aircraft with intersecting flight paths, even though normal control procedures specify that it should be used. They deliberately put the aircraft on conflicting paths for a short time to help manage the airspace. 2. Requirements that are derived from cooperation and awareness of other people's activities. For example, air traffic controllers may use an awareness of other controllers' work to predict the number of aircrafts that will be entering their control sector. They then modify their control strategies depending on that predicted workload. Therefore, an automated ATC system should allow controllers in a sector to have some visibility of the work in adjacent sectors.  Ethnographic Analysis Debriefing Meetings Focused Ethnography Prototype Evaluation Generic System Development System Prototyping Fig: Ethnography and prototyping for requirements analysis	10	CO1	L2
3	a. Explain briefly class model, state model and interaction model. **Solution:** Class Model—for the objects in the system & their relationships. It describes the static structure of the objects in the system and their relationships. • Class model contains class diagrams- a graph whose nodes are classes and arcs are relationships among the classes. 2. State model—for the life history of objects. It describes the aspects of an object that change over time. • It specifies and implements control with state diagrams-a graph whose nodes are states and whose arcs are transition between states caused by events. 3. Interaction Model—for the interaction among objects. • It describes how the objects in the system co-operate to achieve broader results. • This model starts with use cases that are then elaborated with sequence and activity diagrams.	10	CO3	L2

	 Self Transition State Event Trigger/Action State2 Final State Event Trigger/Action Event Trigger/Action State3 Initial State Fig: State Model			
	b. Briefly explain links, associations, ordering, bags, and sequences with an example. **Solution:** Links and associations are the means for establishing relationships among objects and classes. • A link is a physical connection between the objects. • E.g. JoeSmith WorksFor Simplex Company. • Mathematically, we define a link as a tuple – that is, a list of objects. • A link is an instance of an association. • An association is a description of a group of links with common structure and common semantics. • E.g. a person WorksFor a company. Ordering is an inherent part of association. You can indicate an ordered set of objects by writing "{ordered}" next to the appropriate association end. Bag is a collection of elements with duplicates allowed. • A sequence is an ordered collection of elements with duplicates allowed. • Note: {ordered} and {sequence} annotations are same, except that the first • disallows duplicates and the other allows them. 	10	CO3	L2

4	a. Explain the following important terms with example? i) Identity ii) Classification iii) Inheritance iv) Polymorphism **Solution:** Identity means that data is quantized into discrete, distinguishable entities called objects. • E.g. for objects: personal computer, bicycle, queen in chess etc. • Objects can be concrete (such as a file in a file system) or conceptual (such as scheduling policy in a multiprocessing OS). Classification means that objects with the same data structure (attribute) and behavior (operations) are grouped into a class. • Each object is said to be an instance of its class.  Bycycle class: Attributes- - frame size -wheel size no. of gears -material Operations- -shift -move -repair Inheritance: • It is the sharing of attributes and operations (features) among classes based on a hierarchical relationship. A super class has general information that sub classes refine and elaborate.  Person Student Employee ITStudent MathStudent Driver Engineer Polymorphism • means that the same operation may behave differently for different classes.	10	CO3	L2
	b. With a neat diagram, explain the waterfall model of software development process? **Solution:** There are separate identified phases in the waterfall model: – Requirements analysis and definition – System and software design – Implementation and unit testing – Integration and system testing	10	CO1	L2

– Operation and maintenance

Requirements analysis and definition The system’s services, constraints, and goals are established by consultation with system users. They are then defined in detail and serve as a system specification.

- System and software design The systems design process allocates the requirements to either hardware or software systems by establishing an overall system architecture.

- Implementation and unit testing During this stage, the software design is realized as a set of programs or program units. Unit testing involves verifying that each unit meets its specification.

Integration and system testing The individual program units or programs are integrated and tested as a complete system to ensure that the software requirements have been met. After testing, the software system is delivered to the customer.

- Operation and maintenance involves correcting errors which were not discovered in earlier stages of the life cycle, improving the implementation of system units and enhancing the system’s services as new requirements are discovered.

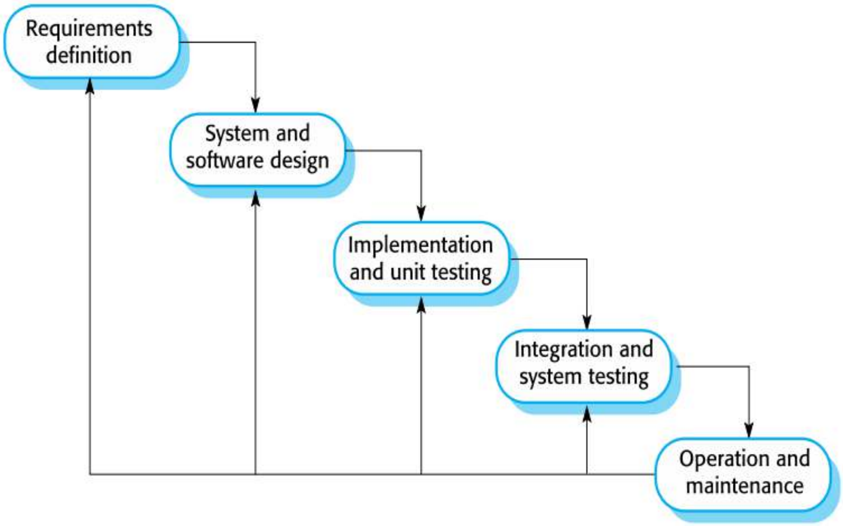

Requirements definition
System and software design
Implementation and unit testing
Integration and system testing
Operation and maintenance

Fig: Waterfall Model

5 a. Write and explain UML notation for objects and classes, values and attributes 10 CO4 L2

Solution:

Objects

Conventions used (UML):

- UML symbol for both classes and objects is box.
- Objects are modeled using box with object name followed by colon followed by class name.
- Use boldface to list class name, center the name in the box and capitalize the first letter.
- To run together multiword names (such as JoeSmith), separate the words with intervening capital letter

Values and Attributes

- Value is a piece of data.
- Attribute is a named property of a class that describes a value held by each object of the class.

E.g. Attributes: Name, bdate, weight.
Values: JoeSmith, 21 October 1983, 64.
Conventions used (UML):

- List attributes in the 2nd compartment of the class box. Optional details (like default value) may follow each attribute.
- A colon precedes the type, an equal sign precedes default value.
- Show attribute name in regular face, left align the name in the box and use small case for the first letter.
- Similarly we may also include attribute values in the 2nd compartment of object boxes with same conventions

b. Define the purpose of the following terms with suitable example and UML notation with respect to class model a) Multiplicity b) Association class. 10 CO4 L1

Solution:

Multiplicity:

- Multiplicity specifies the number of instances of one class that may relate to a single instance of an associated class.
- Multiplicity constrains the number of related objects. UML conventions:
- UML diagrams explicitly lists multiplicity at the ends of association lines.
- UML specifies multiplicity with an interval, such as

"1" (exactly one).
"1.."(one or more).
"3..5"(three to five, inclusive).
" * " (many, i.e zero or more).

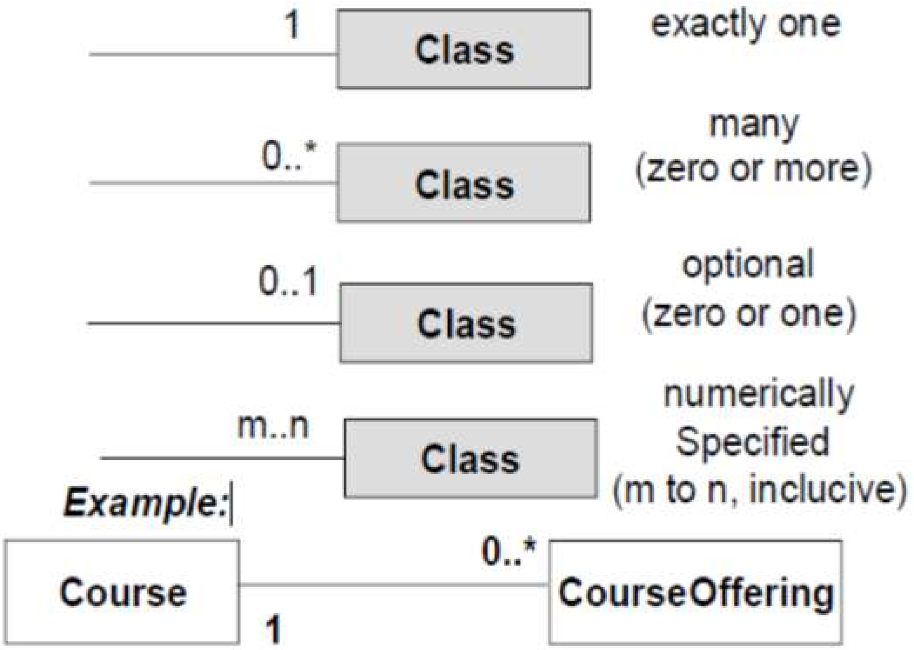
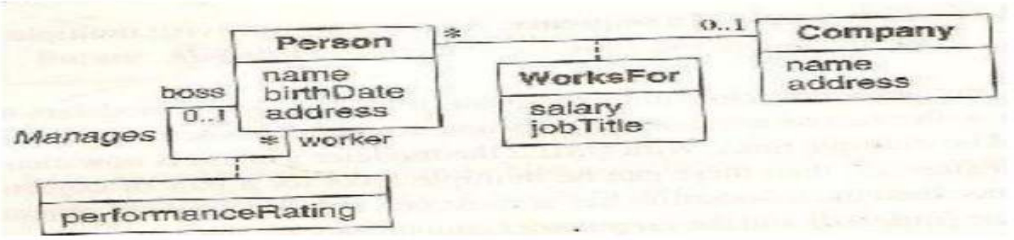


Fig: Multiplicity

UML notation for association class is a box attached to the association by a dashed line.

	 Person name birthDate address boss 0..1 Manages * worker performanceRating * 0..1 WorksFor salary jobTitle Company name address			
6	a. Illustrate Algorithmic cost modeling **Solution:** Algorithmic Cost Modeling Algorithmic cost modeling uses a mathematical formula to predict project costs based on estimates of the project size; the type of software being developed; and other team, process, and product factors. An algorithmic cost model can be built by analyzing the costs and attributes of completed projects, and finding the closest-fit formula to actual experience. Algorithmic models for estimating effort in a software project are mostly based on a simple formula: Effort = A * SizeB* M A is a constant factor which depends on local organizational practices and the type of software that is developed. Size may be either an assessment of the code size of the software or a functionality estimate expressed in function or application points. The value of exponent B usually lies between 1 and 1.5. M is a multiplier made by combining process, product, and development attributes, such as the dependability requirements for the software and the experience of the development team.	10	CO5	L3
	b. Explain COCOMO II model. **Solution:** The COCOMO II Model This is an empirical model that was derived by collecting data from a large number of software projects. These data were analyzed to discover the formulae that were the best fit to the observations. The COCOMO II model takes into account more modern approaches to software development, such as rapid development using dynamic languages, development by component composition, and use of database programming. COCOMO II supports the spiral model of development. The sub-models (Fig 4.9) that are part of the COCOMO II model are: 1. An application-composition model: Models the effort required to develop systems that are created from reusable components, scripting, or database programming. Software size estimates are based on application points, and a simple size/productivity formula is used to estimate the effort required. 2. An early design model: This model is used during early stages of the system design after the requirements have been established. 3. A reuse model: This model is used to compute the effort required to integrate reusable components and/or automatically generated program code. It is normally used in conjunction with the post-architecture model. 4. A post-architecture model: Once the system architecture has been designed, a more accurate estimate of the software size can be made. Again, this model uses the standard formula for cost estimation discussed above.	10	CO5	L2
7	a. Explain Incremental Development process model with a neat block diagram. List its benefits and problems **Solution:** Incremental software development, which is a fundamental part of agile approaches, is better than a waterfall approach for most business, e-commerce, and personal systems. Incremental development reflects the way that we solve	10	CO2	L2

problems. We rarely work out a complete problem solution in advance but move toward a solution in a series of steps, backtracking when we realize that we have made a mistake. By developing the software incrementally, it is cheaper and easier to make changes in the software as it is being developed. Each increment or version of the system incorporates some of the functionality that is needed by the customer. Generally, the early increments of the system include the most important or most urgently required functionality. This means that the customer can evaluate the system at a relatively early stage in the development to see if it delivers what is required. If not, then only the current increment has to be changed and, possibly, new functionality defined for later increments.

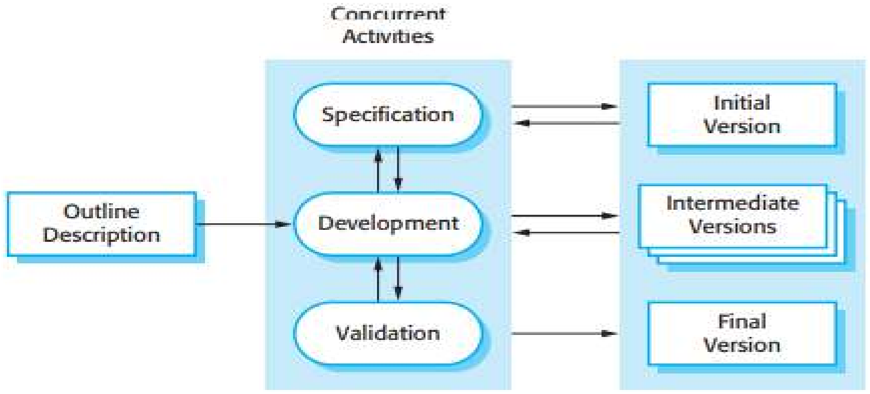
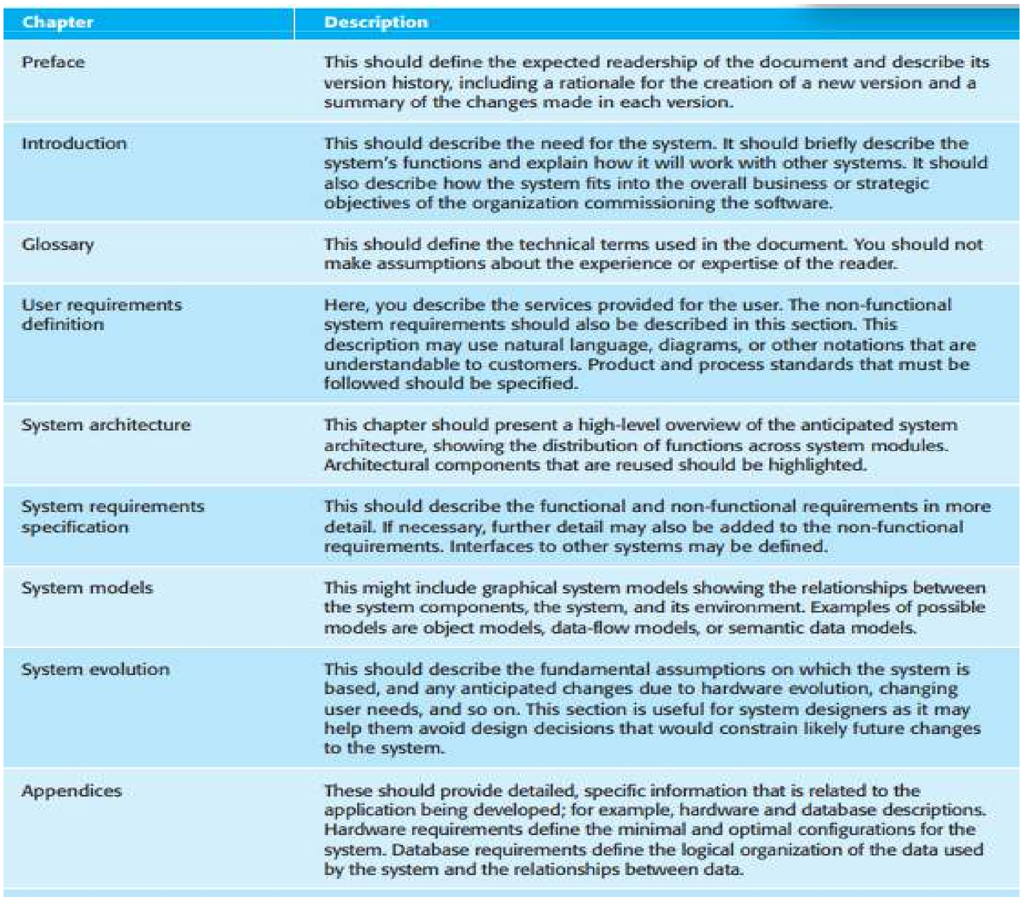


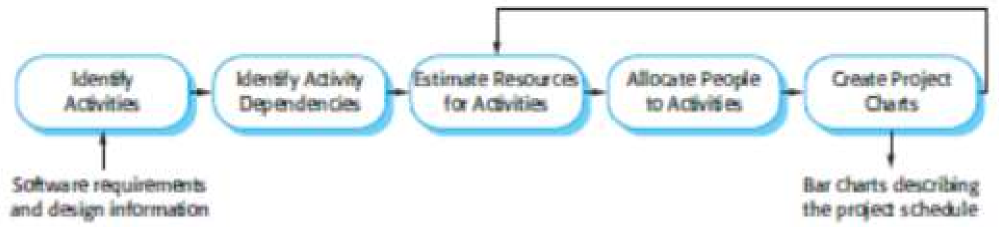
Fig: Incremental process

b. Explain the IEEE standard requirement document with its structure. 10 CO4 L2

Solution:



Chapter	Description
Preface	This should define the expected readership of the document and describe its version history, including a rationale for the creation of a new version and a summary of the changes made in each version.
Introduction	This should describe the need for the system. It should briefly describe the system's functions and explain how it will work with other systems. It should also describe how the system fits into the overall business or strategic objectives of the organization commissioning the software.
Glossary	This should define the technical terms used in the document. You should not make assumptions about the experience or expertise of the reader.
User requirements definition	Here, you describe the services provided for the user. The non-functional system requirements should also be described in this section. This description may use natural language, diagrams, or other notations that are understandable to customers. Product and process standards that must be followed should be specified.
System architecture	This chapter should present a high-level overview of the anticipated system architecture, showing the distribution of functions across system modules. Architectural components that are reused should be highlighted.
System requirements specification	This should describe the functional and non-functional requirements in more detail. If necessary, further detail may also be added to the non-functional requirements. Interfaces to other systems may be defined.
System models	This might include graphical system models showing the relationships between the system components, the system, and its environment. Examples of possible models are object models, data-flow models, or semantic data models.
System evolution	This should describe the fundamental assumptions on which the system is based, and any anticipated changes due to hardware evolution, changing user needs, and so on. This section is useful for system designers as it may help them avoid design decisions that would constrain likely future changes to the system.
Appendices	These should provide detailed, specific information that is related to the application being developed; for example, hardware and database descriptions. Hardware requirements define the minimal and optimal configurations for the system. Database requirements define the logical organization of the data used by the system and the relationships between data.

8	a. With a neat diagram explain project scheduling process. **Solution:** Project Scheduling □ Project scheduling is the process of deciding how the work in a project will be organized as separate tasks, and when and how these tasks will be executed. □ Here there is an estimation of the calendar time needed to complete each task, the effort required, and who will work on the tasks that have been identified. □ It is essential to estimate the resources needed to complete each task, such as the disk space required on a server, the time required on specialized hardware, such as a simulator, and what the travel budget will be. □ Scheduling in plan-driven projects (Fig 4.4) involves breaking down the total work involved in a project into separate tasks and estimating the time required to complete each task. □ Tasks should normally last at least a week, and no longer than 2 months. □ Finer subdivision means that a disproportionate amount of time must be spent on re-planning and updating the project plan. □ The maximum amount of time for any task should be around 8 to 10 weeks. □ If it takes longer than this, the task should be subdivided for project planning and scheduling.  Identify Activities Identify Activity Dependencies Estimate Resources for Activities Allocate People to Activities Create Project Charts Software requirements and design information Bar charts describing the project schedule Fig: Project Scheduling Process	10	CO5	L2
	b. Explain briefly the key stages in the process of product measurement **Solutions:** **The key stages in this component measurement process are:** **1**. Choose measurements to be made: * The questions that the measurement is intended to answer should be formulated and the measurements required to answer these questions defined. Measurements that are not directly relevant to these questions need not be collected. 2. Select components to be assessed: * You may not need to assess metric values for all of the components in a software system. * Sometimes, a representative selection of components can be selected for measurement, allowing to make an overall assessment of system quality. 3. Measure component characteristics: * The selected components are measured and the associated metric values computed. * This normally involves processing the component representation (design, code, etc.) using an automated data collection tool. * This tool may be specially written or may be a feature of design tools that are already in use. 4. Identify anomalous measurements: * After the component measurements have been made, it can be compared with each other and to previous measurements that have been recorded in a measurement database.	10	CO5	L2

5. Analyze anomalous components:

* When the components that have anomalous values for chosen metrics have been identified, it becomes necessary to examine them to decide whether or not these anomalous metric values mean that the quality of the component is compromised.

* An anomalous metric value for complexity

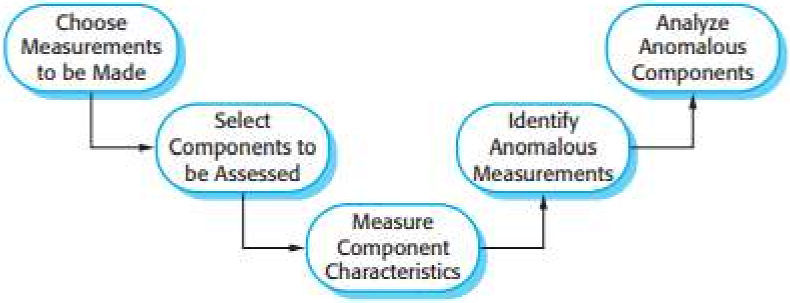
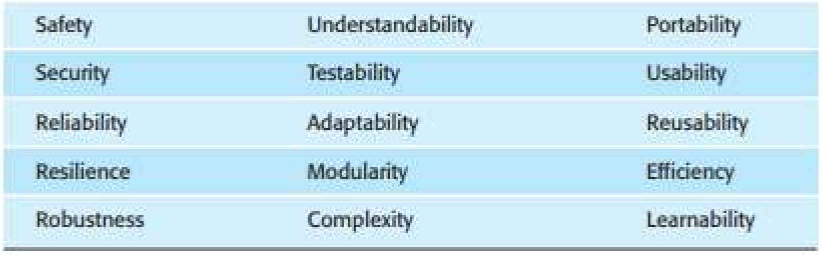

Choose Measurements to be Made
Select Components to be Assessed
Measure Component Characteristics
Identify Anomalous Measurements
Analyze Anomalous Components

Fig: Product Process Model

9 a. Discuss software quality and its attributes. Explain process-based quality. 10 CO4 L2

Solutions:

Software quality attributes



Safety	Understandability	Portability
Security	Testability	Usability
Reliability	Adaptability	Reusability
Resilience	Modularity	Efficiency
Robustness	Complexity	Learnability

There is a clear link between process and product quality in manufacturing because the process is relatively easy to standardize and monitor. Once manufacturing systems are calibrated, they can be run again and again to output high-quality products. However, software is not manufactured—it is designed. In software development, therefore, the relationship between process quality and product quality is more complex. Software development is a creative rather than a mechanical process, so the influence of individual skills and experience is significant. External factors, such as the novelty of an application or commercial pressure for an early product release, also affect product quality irrespective of the process used

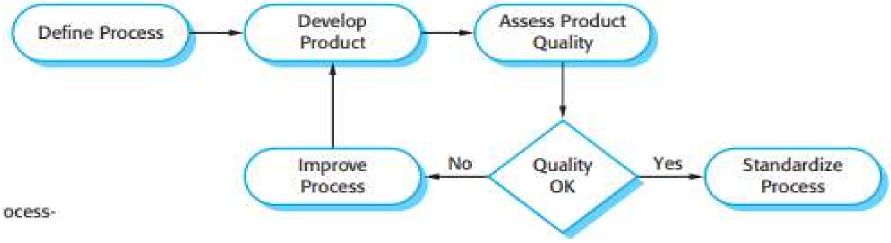


Fig: Process based model

b. Explain software review and inspections of quality assurance. 10 CO4 L2

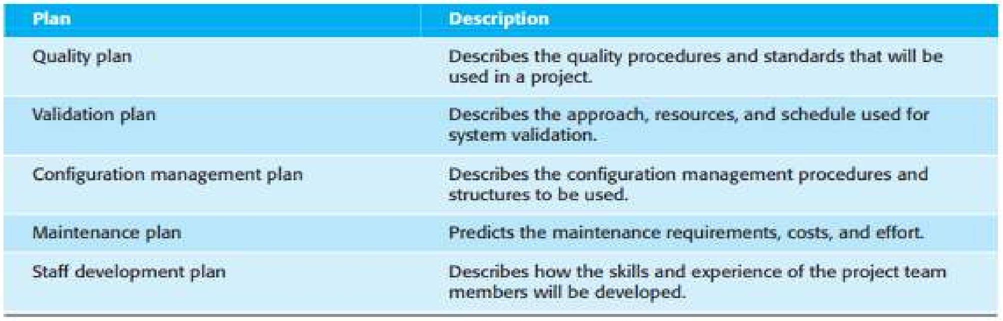
Solutions:

	Reviews and inspections are QA activities that check the quality of project deliverables. This involves examining the software, its documentation and records of the process to discover errors and omissions and to see if quality standards have been followed. During a review, a group of people examine the software and its associated documentation, looking for potential problems and non-conformance with standards. The review team makes informed judgments about the level of quality of a system or project deliverable. Project managers may then use these assessments to make planning decisions and allocate resources to the development process. Quality reviews are based on documents that have been produced during the software development process. Review process is structured into 3 phases: 1. Pre-Review Activities: * These are preparatory activities that are essential for the review to be effective. * Pre-review activities are concerned with review planning and review preparation. * Review planning involves setting up a review team, arranging a time and place for the review, and distributing the documents to be reviewed. * During review preparation, the team may meet to get an overview of the software to be reviewed 2. The Review Meeting: * During the review meeting, an author of the document or program being reviewed should 'walk through' the document with the review team. * The review itself should be relatively short—two hours at most. One team member should chair the review and another should formally record all review decisions and actions to be taken. 3. Post-Review Activities: * After a review meeting has finished, the issues and problems raised during the review must be addressed. * This may involve fixing software bugs, refactoring software so that it conforms to quality standards, or rewriting documents.			
10	a. Explain the various section and supplements of the project plan. **Solutions:** 1. Introduction This briefly describes the objectives of the project and sets out the constraints (e.g., budget, time, etc.) that affect the management of the project 2. Project organization This describes the way in which the development team is organized, the people involved, and their roles in the team. 3. Risk analysis This describes possible project risks, the likelihood of these risks arising, and the risk reduction strategies that are proposed. 4. Hardware and software resource requirements This specifies the hardware and support software required to carry out the development. If hardware has to be bought, estimates of the prices and the delivery schedule may be included. 5. Work breakdown This sets out the breakdown of the project into activities and identifies the milestones and deliverables associated with each activity. Milestones are key stages in the project where progress can be assessed; deliverables are work products that are delivered to the customer. 6. Project schedule This shows the dependencies between activities, the	10	CO5	L2

estimated time required to reach each milestone, and the allocation of people to activities.

7. Monitoring and reporting mechanisms This defines the management reports that should be produced, when these should be produced, and the project monitoring mechanisms to be used.

Project Plan Supplements



Plan	Description
Quality plan	Describes the quality procedures and standards that will be used in a project.
Validation plan	Describes the approach, resources, and schedule used for system validation.
Configuration management plan	Describes the configuration management procedures and structures to be used.
Maintenance plan	Predicts the maintenance requirements, costs, and effort.
Staff development plan	Describes how the skills and experience of the project team members will be developed.

b. Explain requirement elicitation and analysis process

10 CO1 L2

Solutions:

The process activities are:

1.Requirements discovery This is the process of interacting with stakeholders of the system to discover their requirements. Domain requirements from stakeholders and documentation are also discovered during this activity. There are several complementary techniques that can be used for requirements discovery,

2. Requirements classification and organization This activity takes the unstructured collection of requirements, groups related requirements, and organizes them into coherent clusters. The most common way of grouping requirements is to use a model of the system architecture to identify sub-systems and to associate requirements with each sub-system. In practice, requirements engineering and architectural design cannot be completely separate activities.

3. Requirements prioritization and negotiation Inevitably, when multiple stakeholders are involved, requirements will conflict. This activity is concerned with prioritizing requirements and finding and resolving requirements conflicts through negotiation. Usually, stakeholders have to meet to resolve differences and agree on compromise requirements.

4. Requirements specification the requirements are documented and input into the next round of the spiral

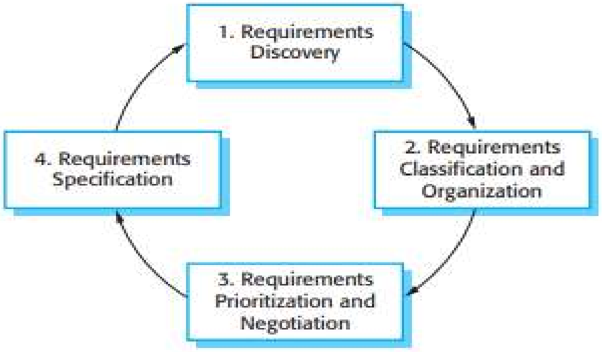

1. Requirements Discovery
2. Requirements Classification and Organization
3. Requirements Prioritization and Negotiation
4. Requirements Specification

Fig: Requirement Process