

St. Peter's Engineering College (Autonomous) Dullapally (P), Medchal, Hyderabad – 500100. QUESTION BANK				Dept.	:	CSM
				Academic Year 2024-25		
Subject Code	:	AS22-66ES01	Subject	:	SOFTWARE ENGINEERING	
Class/Section	:	B. Tech.	Year	:	II	Semester : I
Faculty	:	Mrs.D.Divya (B & D sec) , Mr.G.Pavan (A& E Sec) , Mrs.M.Benita Roy (C Sec)				

BLOOMS LEVEL					
Remember	L1	Understand	L2	Apply	L3
Analyze	L4	Evaluate	L5	Create	L6

Q. No	Question (s)	Marks	BL	CO
UNIT – I				
1(a)	Define the software engineering.	1M	L1	C215.1
	Ans: It is a systematic, quantifiable and disciplined approach to operate and maintain a software.			
1(b)	Mention the steps involved in software development life cycle	1M	L1	C215.1
	Ans. The steps followed to develop a software is called as process in Software engineering .The steps are : 1)Software Requirements 2)Analysis 3)Design 4)Implementation 5)Testing 6)Deployment 7)Maintenance.			
1(c)	Define product in the evolving nature of software.	1M	L1	C215.1
	Ans. The software which produces, manages and displays information that software is called as product.			
1(d)	Define system software, application software.	1M	L1	C215.1
	Ans. 1) System Software: The software which provide service to the other software's is known as System Software. Ex: Operating System. 2) Application Software: The software which provides services to the end user known as Application Software. Ex : E-Mail app.			
1(e)	Define software myth.	1M	L1	C215.1
	Ans. Beliefs about software and the process used to built it can be traced to the earliest days of computing myths have a number of attributes that have made them insidious.			
2(a)	Mention the characteristics of software.	3M	L1	C215.1
	Ans. Characteristics of Software: → Software is developed or engineered but not manufactured → Software doesn't wearout → Software is custom built not user built.			
2(b)	List the frame work activities, umbrella activities.	3M	L1	C215.1
	Ans. -> Frame work activities: 1)communication 2)Planning 3)Modeling 4)Construction 5)Deployment -> Umbrella activities: 1)Software project tracking and control 2)Risk management 3)Software quality assurance 4)Formal technical reviews 5)Measurement 6)Software configuration management 7)Reusability management 8)Work product preparation and production			

2(c)	Mention the advantages of waterfall model.	3M	L1	C215.1
	Ans. Waterfall model – Advantages → Simple and easy to understand and use → Easy to manage → Works well for smaller projects → Clearly defined stages → Well - undertood milestones			
2(d)	Mention the advantages of spiral model.	3M	L1	C215.1
	Ans. Spiral model – Advantages → Suitable for large scale product → Development is fast → Efficient cost estimation → Proper risk management → Involves customer feedback			
2(e)	List the phases of agile model.	3M	L1	C215.1
	Ans. Phases of Agile model 1)Requirement gathering 2)Design the requirements 3)Construction / iteration 4)Testing / quality assurance 5)Deployment 6)Feedback			
3(a)	Explain regarding the evolving nature of software, changing nature of software.	5M	L2	C215.1
	<p>Ans Software plays a dual role. Those are 1)Product 2)Vehicle</p> <p><u>PRODUCT:-</u> The software which produces, manages, displays information that is called product</p> <p><u>VEHICLE:-</u> The software which is able to transmit or process the simple data multimedia information from one place to another place is called as vehicle</p> <p>1)Software can control the products 2)software can create new software 3)software can provide different functionalities for different users.</p> <p><u>CHANGING NATURE OF SOFTWARE:-</u> There are 7 types of software those are:- 1)system software 2)Applications software 3)engineering & scientific software 4)embedded software 5)product line software 6)website software 7)AI software 1)<u>System software:</u></p>			

	<p>System software is a collection of programs which are written to service other programs. Some system software processes complex but determinate, information structures. Other system application process largely indeterminate data. Sometimes when, the system software area is characterized by the heavy interaction with computer hardware that requires scheduling, resource sharing, and sophisticated process management.</p> <p>Ex:-OS</p> <p>2) <u>Application software:</u></p> <p>Application software is defined as programs that solve a specific business need. Application in this area process business or technical data in a way that facilitates business operation or management technical decision making. In addition to convention data processing application, application software is used to control business function in real time.</p> <p>Ex:-email</p> <p>3. <u>Engineering and Scientific Software:</u></p> <p>This software is used to facilitate the engineering function and task. however modern application within the engineering and scientific area are moving away from the conventional numerical algorithms. Computer-aided design, system simulation, and other interactive applications have begun to take a real- time and even system software characteristic.</p> <p>Ex:-CAD, Unigraphics & cations</p> <p>4. <u>Embedded Software:</u></p> <p>Embedded software resides within the system or product and is used to implement and control feature and function for the end-user and for the system itself. Embedded software can perform the limited and esoteric function or provided significant function and control capability.</p> <p>Ex:- Micro oven, geysers, heaters</p> <p>5. <u>Product-line Software:</u></p> <p>Designed to provide a specific capability for use by many different customers, product line software can focus on the limited and esoteric marketplace or address the mass consumer market.</p> <p>Ex:- cloud computing in data base management</p> <p>6. <u>Web Application:</u></p> <p>It is a client-server computer program which the client runs on the web browser. In their simplest form, Web apps can be little more than a set of linked hypertext files that present information using text and limited graphics. However, as e-commerce and B2B application grow in importance. Web apps are evolving into a sophisticate computing environment that not only provides a standalone feature, computing function, and content to the end user.</p> <p>Ex:-SPEC website</p> <p>7. <u>Artificial Intelligence Software:</u></p> <p>Artificial intelligence software makes use of a nonnumerical algorithm to solve a complex problem that is not amenable to computation or straightforward analysis. Application within this area includes robotics, expert system, pattern recognition, artificial neural network, theorem proving and game playing.</p> <p>Ex:-PUBG</p>			
3(b)	Explain how the Software Engineering is considered as a Layered Technology.	5M	L2	C215.1
Ans	<ul style="list-style-type: none"> • Software engineering is a fully layered technology • To develop a software, we need to go from one layer to another. 			

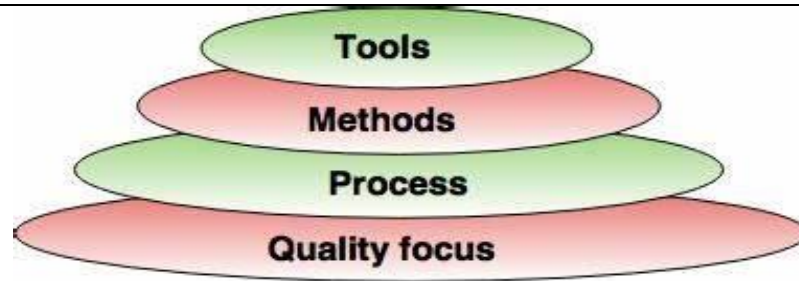


Fig. - Software Engineering Layers

• All these layers are related to each other and each layer demands the fulfillment of the previous layer. The layered technology consists of:

1. Quality focus :The characteristics of good quality software are:

- Correctness of the functions required to be performed by the software.
- Maintainability of the software
- Integrity i.e. providing security so that the unauthorized user cannot access information or data.
- Usability i.e. the efforts required to use or operate the software.

2. Process:

- It is the base layer or foundation layer for the software engineering.
- The software process is the key to keep all levels together.
- It defines a framework that includes different activities and tasks.
- In short, it covers all activities, actions and tasks required to be carried out for software development.

3. Methods:

- The method provides the answers of all 'how-to' that are asked during the process.
- It provides the technical way to implement the software.
- It includes collection of tasks starting from communication, requirement analysis, analysis and design modelling, program construction, testing and support.

4. Tools :

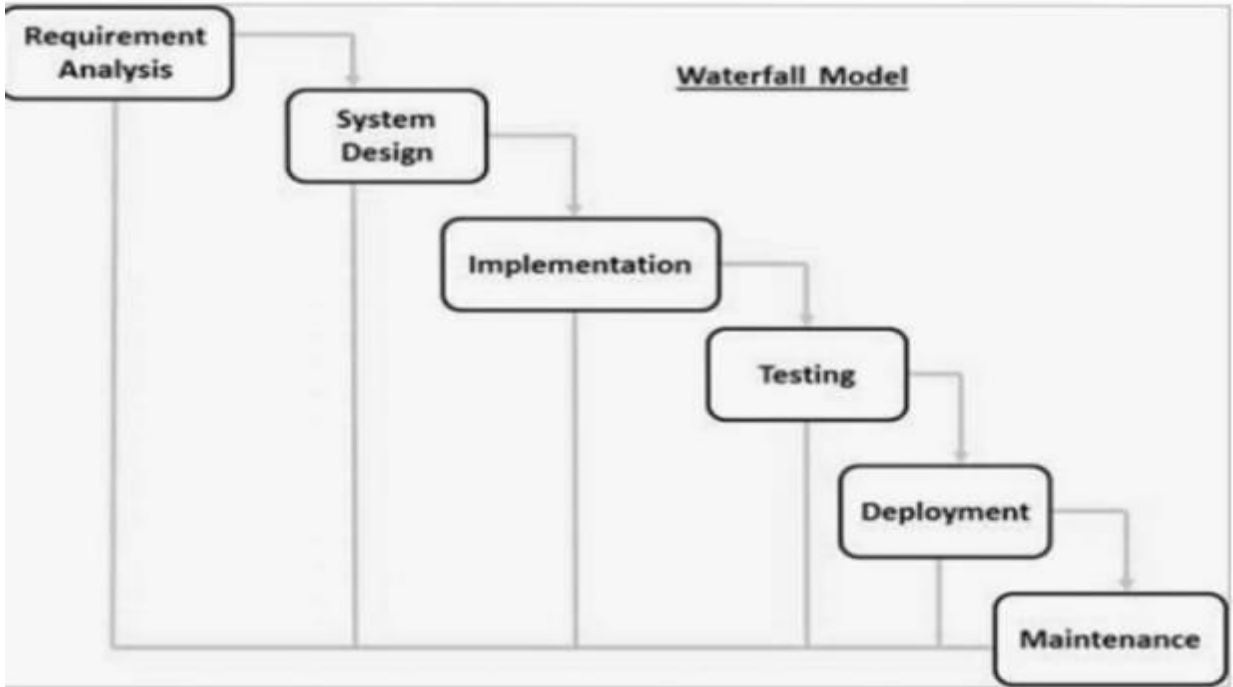
- The software engineering tool is an automated support for the software development.
- The tools are integrated i.e the information created by one tool can be used by the other tool.

For example: The Microsoft publisher can be used as a web designing tool

3(c)	Describe about Software MYTHS in detail.	5M	L2	C215.1
	<p>Ans: Beliefs about software and the process used to build it- can be traced to the earliest days of computing myths have a number of attributes that have made them insidious.</p> <p>Management myths: Managers with software responsibility, like managers in most disciplines, are often under pressure to maintain budgets, keep schedules from slipping, and improve quality.</p> <p>Myth: We already have a book that's full of standards and procedures for building software - Won't that provide my people with everything they need to know?</p> <p>Reality: The book of standards may very well exist but, is it used? Are software practitioners aware of its existence? Does it reflect modern software engineering practice?</p> <p>Myth: If we get behind schedule, we can add more programmers and catch up.</p> <p>Reality: Software development is not a mechanistic process like manufacturing. As new people are added, people who were working must spend time educating the new comers, thereby reducing the amount of time spent on productive development effort. People can be added but only in a planned and well co-ordinated manner.</p>			

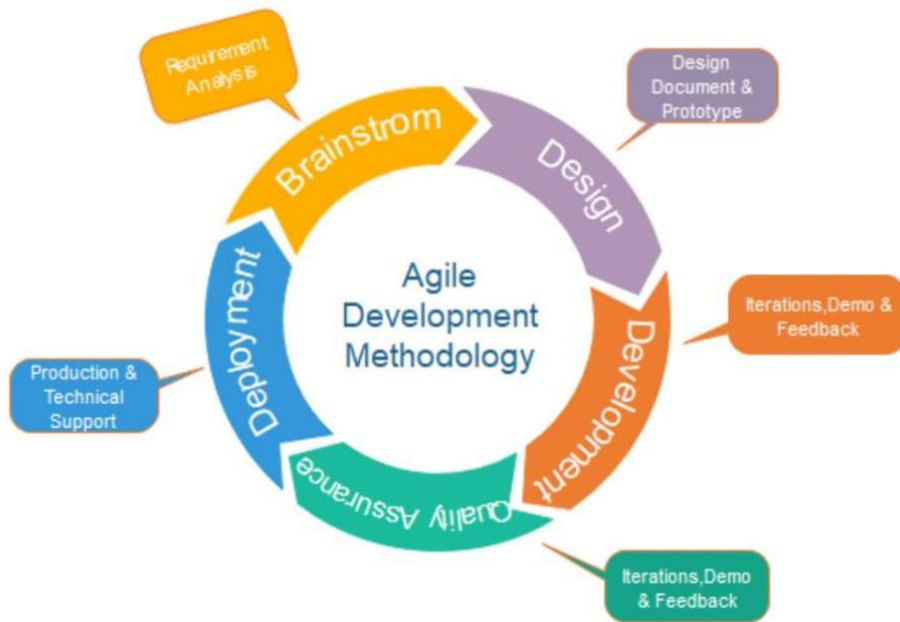
	<p>Myth: If I decide to outsource the software project to a third party, I can just relax and let that firm built it.</p> <p>Reality: If an organization does not understand how to manage and control software projects internally, it will invariably struggle when it outsources software projects.</p> <p>Customer myths: The customer believes myths about software because software managers and practitioners do little to correct misinformation.</p> <p>Myths lead to false expectations and ultimately, dissatisfaction with the developer.</p> <p>Myth: A general statement of objectives is sufficient to begin with writing programs - we can fill in the details later.</p> <p>Reality: Although a comprehensive and stable statement of requirements is not always possible, an ambiguous statement of objectives is recipe for disaster.</p> <p>Myth: Project requirements continually change, but change can be easily accommodated because software is flexible.</p> <p>Reality: It is true that software requirements change, but the impact of change varies with the time at which it is introduced and change can cause upheaval that requires additional resources and major design modification.</p> <p>Developer's myths: Myths that are still believed by software practitioners: during the early days of software, programming was viewed as an art from old ways and attitudes die hard.</p> <p>Myth: Once we write the program and get it to work, our jobs are done.</p> <p>Reality: Someone once said that the sooner you begin writing code, the longer it'll take you to get done. Industry data indicate that between 60 and 80 percent of all effort expended on software will be expended after it is delivered to the customer for the first time.</p> <p>Myth: The only deliverable work product for a successful project is the working program.</p> <p>Reality: A working program is only one part of a software configuration that includes many elements. Documentation provides guidance for software support.</p> <p>Myth: software engineering will make us create voluminous and unnecessary documentation and will invariably slows down.</p> <p>Reality: software engineering is not about creating documents. It is about creating quality. Better quality leads to reduced rework. And reduced rework results in faster delivery times.</p>	5M	L2	C215.1
3(d)	<p>Describe the framework activities in software engineering briefly.</p> <p>Ans. process framework:</p> <ul style="list-style-type: none"> • Establish the foundation for a complete software process. • Identifies a small number of framework activities. • Applies to all software projects, regardless of size/complexity. • Also, set of umbrella activities. • Application across entire software process. • Each framework activities has set of software engineering actions. • Each software engineering action has collection of related tasks called task sets or work tasks. <p>Generic process of framework:</p> <p>It is applicable to that vast majority of software projects.</p> <p>1) Communication:</p> <p>This framework activity involves heavy communication and collaboration with the customer</p>			

	<p>and encompasses requirements gathering and other related activities.</p> <p>2) Planning:</p> <p>This activity establishes a plan for the software engineering work that follows. It required, the work products to be produced , and a work schedule.</p> <p>3) Modeling:</p> <p>This activity encompasses the creation of models that allow the developer and customer to better understand software requirements and the design that will achieve those requirements . the modeling activity is composed of two software engineering action analysis and design.</p> <ul style="list-style-type: none"> • Analysis encompasses a set of work tasks. • Design encompasses work tasks that create a design model. <p>4) Construction:</p> <p>This activity combines code generation and the testing that is required to uncover the errors in the code.</p> <p>5) Deployment:</p> <p>The software is delivered to the customer who evaluates the delivered product and provides feedback based on the evaluation .</p> <p>These five generic framework activities can be used during the development of small programs the creation of large web applications and for the engineering of large , complex computer based system.</p>			
3(e)	Explain in detail the capability Maturity Model Integration (CMMI)?	5M	L2	C215.1
	<p>Ans. THE CAPABILITY MATURITY MODEL INTEGRATION (CMMI):</p> <p>The CMMI represents a process meta-model in two different ways: • As a continuous model • As a staged model. Each process area is formally assessed against specific goals and practices and is rated according to the following capability levels.</p> <p>Level 0: Incomplete: The process area is either not performed or does not achieve all goals and objectives defined by CMMI for level 1 capability.</p> <p>Level 1: Performed: All of the specific goals of the process area have been satisfied. Work tasks required to produce defined work products are being conducted.</p> <p>Level 2: Managed: All 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;</p> <p>Level 3: Defined: All level 2 criteria have been achieved. In addition, the process is “tailored from the organizations set of standard processes according to the organizations tailoring guidelines, and contributes and work products, measures and other process-improvement information to the organizational process assets”.</p> <p>Level 4: Quantitatively managed: All 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”</p> <p>Level 5: Optimized: All level 4 criteria have been achieved. In addition, the process area is adapted and optimized using quantitative means to meet changing customer needs and to continually improve the efficacy of the process area under consideration”</p>			

4(a)	Briefly explain Waterfall Model. Also write down advantages and disadvantages of the water fall model.	10M	L2	C215.1
	<p>Ans. Waterfall Model :</p> <p>Waterfall approach was first SDLC Model to be used widely in Software Engineering. In "The Waterfall" approach, the whole process of software development is divided into separate phases. In this Waterfall model, typically, the outcome of one phase acts as the input for the next phase sequentially. The following illustration is a representation of the different phases of the Waterfall Model. The sequential phases in Waterfall model are –</p> <ul style="list-style-type: none"> • Requirement Gathering and analysis – All possible requirements of the system to be developed are captured in this phase and documented in a requirement specification document. • System Design – The requirement specifications from first phase are studied in this phase and the system design is prepared. This system design helps in specifying hardware and system requirements and helps in defining the overall system architecture. • Implementation – With inputs from the system design, the system is first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality, which is referred to as Unit Testing. • Integration and Testing – All the units developed in the implementation phase are integrated into a system after testing of each unit. Post integration the entire system is tested for any faults and failures. • Deployment of system – Once the functional and non-functional testing is done; the product is deployed in the customer environment or released into the market. • Maintenance – There are some issues which come up in the client environment. To fix those issues, patches are released. Also to enhance the product some better versions are released. Maintenance is done to deliver these changes in the customer environment. <p>All these phases are cascaded to each other in which progress is seen as flowing steadily downwards (like a waterfall) through the phases. The next phase is started only after the defined set of goals are achieved for previous phase and it is signed off, so the name "Waterfall Model".</p>  <p>Waterfall Model – Application</p>			

	<ul style="list-style-type: none"> • Requirements are very well documented, clear and fixed. • Product definition is stable. • Technology is understood and is not dynamic. • There are no ambiguous requirements. • The project is short. <p>Waterfall Model – Advantages</p> <ul style="list-style-type: none"> • Simple and easy to understand and use • Easy to manage • Phases are processed and completed one at a time. • Works well for smaller projects • Clearly defined stages. • Well understood milestones. <p>Waterfall Model – Disadvantages</p> <ul style="list-style-type: none"> • No working software is produced until late during the life cycle. • High amounts of risk and uncertainty. • Not a good model for complex and object-oriented projects. • Poor model for long and ongoing projects. 			
4(b)	Explain Spiral Model with neat diagram, write advantages and disadvantages.	10M	L2	C215.1
	<p>It is the combination of waterfall model and iterative model. A scientist named Boehm proposed this Spiral model in the year 1986.</p> <p>The Spiral model is also named as metal model it is a risk driven model.</p> <p>The structure of Spiral model is represented as</p> <div data-bbox="305 1129 1200 1717" data-label="Diagram"> </div> <ul style="list-style-type: none"> • The communication phase represent requirement analysis and planning represents risk analysis. • Modeling and construction phase represents the prototype building. Deployment phase represents the Evaluation of feedback/performance Evaluation. <p>1. Requirement Analysis :- In this phase it analysis and expands the reason for what it has to bedone and additionally solutions are identified if incase there is a failure in the attempted version.</p> <p>2. Risk Analysis :-</p>			

	<p>This phase focuses on the risk and it analysis the risks of all the possible solutions .This is performed to identify the faults are occurring in the process each risk is resolved using the most efficient strategy.</p> <p>3. Prototype Building :- This phase is focuses o the building or developing the actual software and testing the software. In this phase the programmers create an architectural design, modules, physical product design and finalizes the design it takes proposal from the first two phases and turn it into usable software.</p> <p>4.Evaluation feedback/performance Evaluation :- This is the final phase where the performance of the newly developed software gets tested and Evaluated programmers analyse their past work to learn before starting a new project. After this step they can start planning for the next phase and they can repeat the cycle in the end the software company releases the feedback into the market.</p> <p>Advantages of Spiral Model :-</p> <ul style="list-style-type: none"> • Suitable for large scale product. • Development is fast and efficient cost estimator. • Proper risk management. • Involves customer feedback. <p>Disadvantages of Spiral Model:-</p> <ul style="list-style-type: none"> • Spiral way goon indefinitely and end of the project may not be known early. • more complex and expensive than other SDLC Models 	10M	L2	C215.1
4(c)	<p>Explain SDLC - Agile Model in brief and mention the advantages</p> <p>Agile Method for SDLC :- Agile SDLC model is a combination of iterative and incremental process models with focus on process adaptability and customers and customers satisfaction by rapid delivery of working software product.</p> <p>Agile methods break the product into small incremental builds .These builds are provided in iterations. Each iteration typically lasts from about one to three weeks. Every iteration involves cross functional tzams working simultaneously on various areas like</p> <ul style="list-style-type: none"> • Planning • Requirement analysis • Design • Coding • Unit testing and • Acceptance testing <p>At the end of the iteration, a working product is display to the customer and the all important stake holders.</p>			



Phases of Agile Model :-

Following are the phases in the Agile model are:-

- a. Requirements gathering
- b. Design the requirements
- c. Construction / iteration
- d. Testing /Quality Assurance
- e. Deploying
- f. Feedback

1.Requirements Gathering:-

In this phase define the requirements you should business opportunities and plan the time and effort needed to build the project. Based on this information , you can evaluate technical and economic feasibility.

2.Design the requirements:-

When you have identified the project work with stakeholders to define the requirements you can use the user flow diagram or the high level UML diagram to show the work of new features and show how it will apply to your existing system.

3.Construction /iteration :-

When the team defines the requirements , the work begins. Designers and the developers start working on their projects , which aims to deploy a working product. The product will undergo various stages of improvement. So it includes simple, minimal functionality.

4.Testing:-

In this phase ,the quality assurance team examines the products performance and looks for the bug.

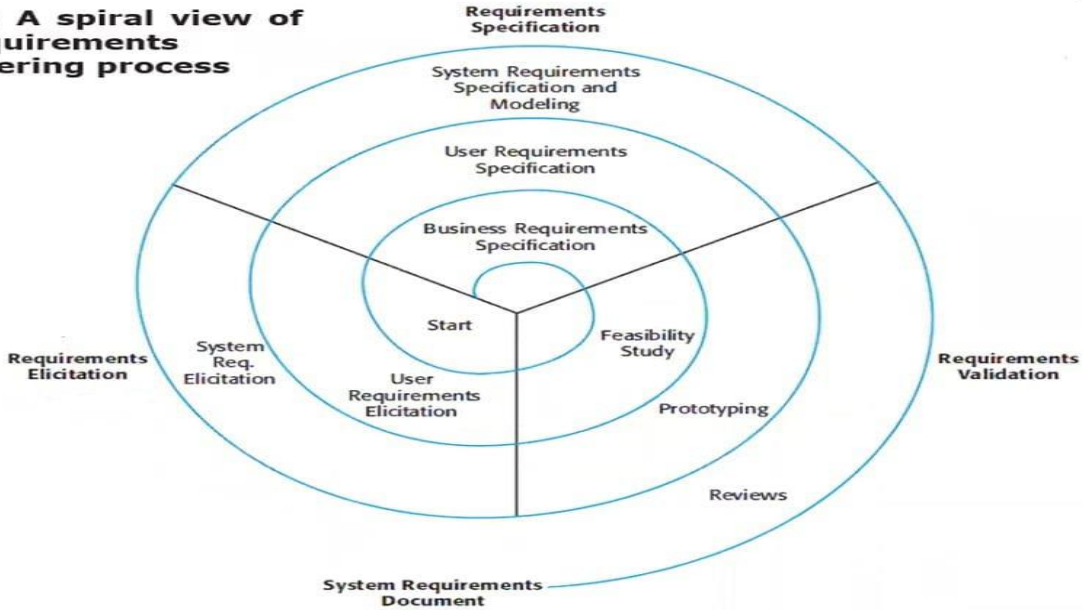
5.Deployment:-

In this phase , the team issues a product for the users work environment .

6.Feedback:-

After realizing the product , the last step is feedback. In this , the team receives feedback about the product and works through the feedback.

Q. No	Question (s)	Marks	BL	CO																
UNIT - II																				
1(a)	Define software requirements.	1M	L1	C215.2																
	Ans. The Features or Functions or Services that are expected by user in the software or product or system are called software requirements.																			
1(b)	List the types of software requirements.	1M	L1	C215.2																
	Ans. Depending upon the functionality or working the software requirements are classified into two types those are 1. Functional requirements 2. Non Functional requirements. Depending upon the peoples readability those are classified as two types. Those are, 1. User requirements 2. System requirements																			
1(c)	List the types of Non-functional requirements.	1M	L1	C215.2																
	Ans. The Non Functional requirements are classified into 3 types those are: 1. Product requirements 2. Organization requirements 3. External requirements.																			
1(d)	Define requirement engineering process.	1M	L1	C215.2																
	Ans. The process involved in gathering requirements, documenting requirements ,validating Requirements and managing requirements is called as requirement engineering process																			
1(e)	Mention the types of feasibility.	1M	L1	C215.2																
	Ans. The Feasibility study is classified into five types .those are 1. Economical feasibility 2. Operational feasibility 3. Legal feasibility 4. Technical feasibility 5. Schedule feasibility.																			
2(a)	Mention the main differences between the functional & non-functional requirements.	3M	L2	C215.2																
	<table><tr><th>Functional Testing</th><th>Non - Functional Testing</th></tr><tr><td>It is executed to analyze the functionality of components of an application as per the client's requirements.</td><td>It is executed to check the performance, reliability, scalability, and other non-functional aspects of an application.</td></tr><tr><td>It is executed in the early stages of development.</td><td>It is generally performed after functional testing.</td></tr><tr><td>Can be performed both manually and with automation tools.</td><td>Requires automation tools for effective testing.</td></tr><tr><td>Focuses on user requirements.</td><td>Focuses on user expectations.</td></tr><tr><td>Determines what the product is capable of</td><td>Determines how effectively the product works</td></tr><tr><td>Business requirements are the inputs of functional testing.</td><td>Parameters like speed, scalability are the inputs of non-functional testing.</td></tr><tr><td>Examples of Functional Testing: Unit Testing, White Box Testing, Smoke Testing, Sanity Testing, Usability Testing, Regression Testing.</td><td>Examples of Non-Functional Testing: Performance Testing, Load Testing, Stress Testing, Security Testing, Installation Testing, Cross Browser Compatibility Testing.</td></tr></table>				Functional Testing	Non - Functional Testing	It is executed to analyze the functionality of components of an application as per the client's requirements.	It is executed to check the performance, reliability, scalability, and other non-functional aspects of an application.	It is executed in the early stages of development.	It is generally performed after functional testing.	Can be performed both manually and with automation tools.	Requires automation tools for effective testing.	Focuses on user requirements.	Focuses on user expectations.	Determines what the product is capable of	Determines how effectively the product works	Business requirements are the inputs of functional testing.	Parameters like speed, scalability are the inputs of non-functional testing.	Examples of Functional Testing: Unit Testing, White Box Testing, Smoke Testing, Sanity Testing, Usability Testing, Regression Testing.	Examples of Non-Functional Testing: Performance Testing, Load Testing, Stress Testing, Security Testing, Installation Testing, Cross Browser Compatibility Testing.
Functional Testing	Non - Functional Testing																			
It is executed to analyze the functionality of components of an application as per the client's requirements.	It is executed to check the performance, reliability, scalability, and other non-functional aspects of an application.																			
It is executed in the early stages of development.	It is generally performed after functional testing.																			
Can be performed both manually and with automation tools.	Requires automation tools for effective testing.																			
Focuses on user requirements.	Focuses on user expectations.																			
Determines what the product is capable of	Determines how effectively the product works																			
Business requirements are the inputs of functional testing.	Parameters like speed, scalability are the inputs of non-functional testing.																			
Examples of Functional Testing: Unit Testing, White Box Testing, Smoke Testing, Sanity Testing, Usability Testing, Regression Testing.	Examples of Non-Functional Testing: Performance Testing, Load Testing, Stress Testing, Security Testing, Installation Testing, Cross Browser Compatibility Testing.																			

2(b)	List the readers of user requirements, system requirements, software requirement specification document.	3M	L1	C215.2
	<p>Ans. <u>Readers of user requirements:</u> The readers of user requirements are, ->System manager -> Client managers -> System architects -> Client engineers <u>Readers of system requirements:</u> The readers of system requirements are, -> Client manager ->Contract manager -> System architects ->Software developers. <u>Readers of software requirement specification document:</u> The readers of software requirement specification document are, -> System customers -> Managers -> System Engineers -> System Test Engineers -> System Maintenance Engineers</p>			
2(c)	Draw the structure of spiral view of requirement engineering process.	3M	L1	C215.2
	<p>Ans. In the spiral view of requirement engineering process we are having three phases those are:</p> <ol style="list-style-type: none"> 1. Requirement elicitation 2. Requirement specification 3. Requirement validation <p>The structure of spiral model of requirement engineering process is represented below</p> <p>Figure: A spiral view of the requirements engineering process</p> 			

2(d)	What are the goals of feasibility study?	3M	L1	C215.2																								
	Ans. Goals of Feasibility Study: 1. Feasibility study is the process of accessing the strengths and weakness of the software or project or system. 2. It prevents the directions of activities that will improve a projector software or system and achieve the desired output. 3. It determines whether the projects is technically financially, legally and operationally feasible with in the estimated cost and time.																											
2(e)	Mention the process activities involved in requirement elicitation & analysis.	3M	L1	C215.2																								
	Ans. There are four process activities involved in requirement elicitation & analysis. Those are, 1. Requirement Discovery 2. Requirement Classification & Organization 3. Requirement Prioritization & Negotiation 4. Requirement Specification																											
3(a)	Compare and contrast functional and non-functional requirements in software engineering.	5M	L2	C215.2																								
	<table><tr><th>S. No</th><th>Functional Requirements</th><th>Non – Functional Requirements</th></tr><tr><td>1.</td><td>The Functional Requirements concentrates on what to do must in a software or product or system.</td><td>The Non – Functional Requirements concentrates on quality attributes of the product or system or software.</td></tr><tr><td>2.</td><td>The Functional Requirements helps in identify the functions or features or services provided by the software or product or system.</td><td>The Non – Functional Requirements helps in identify performance, security, privacy, usability and scalability parameters.</td></tr><tr><td>3.</td><td>Easy to maintain the functional requirements of a software or product or system.</td><td>Not easy to maintain i.e. difficult to maintain the non – functional requirements of a software or product or system.</td></tr><tr><td>4.</td><td>These are the mandatory requirements.</td><td>These are the non – mandatory requirements.</td></tr><tr><td>5.</td><td>These are designed depending on the users requirements.</td><td>These are designed depending on the expectation of the users.</td></tr><tr><td>6.</td><td>These are the requirements specified by the users.</td><td>These are the requirements specified by the software developers, software architects, technical persons.</td></tr><tr><td>7.</td><td>It describes what the product does.</td><td>It describes how the product does.</td></tr></table>				S. No	Functional Requirements	Non – Functional Requirements	1.	The Functional Requirements concentrates on what to do must in a software or product or system.	The Non – Functional Requirements concentrates on quality attributes of the product or system or software.	2.	The Functional Requirements helps in identify the functions or features or services provided by the software or product or system.	The Non – Functional Requirements helps in identify performance, security, privacy, usability and scalability parameters.	3.	Easy to maintain the functional requirements of a software or product or system.	Not easy to maintain i.e. difficult to maintain the non – functional requirements of a software or product or system.	4.	These are the mandatory requirements.	These are the non – mandatory requirements.	5.	These are designed depending on the users requirements.	These are designed depending on the expectation of the users.	6.	These are the requirements specified by the users.	These are the requirements specified by the software developers, software architects, technical persons.	7.	It describes what the product does.	It describes how the product does.
S. No	Functional Requirements	Non – Functional Requirements																										
1.	The Functional Requirements concentrates on what to do must in a software or product or system.	The Non – Functional Requirements concentrates on quality attributes of the product or system or software.																										
2.	The Functional Requirements helps in identify the functions or features or services provided by the software or product or system.	The Non – Functional Requirements helps in identify performance, security, privacy, usability and scalability parameters.																										
3.	Easy to maintain the functional requirements of a software or product or system.	Not easy to maintain i.e. difficult to maintain the non – functional requirements of a software or product or system.																										
4.	These are the mandatory requirements.	These are the non – mandatory requirements.																										
5.	These are designed depending on the users requirements.	These are designed depending on the expectation of the users.																										
6.	These are the requirements specified by the users.	These are the requirements specified by the software developers, software architects, technical persons.																										
7.	It describes what the product does.	It describes how the product does.																										

	8.	After completion of these functional requirements only the system allows to frame the non – functional requirements.	Without completion of functional requirements the system doesn't allow the non – functional requirements.	
3(b)	Explain regarding the user requirements & system requirements.		5M	L2 C215.2
	<p>Ans. Depending on the persons readability the software requirements are classified into two types. Those are,</p> <ol style="list-style-type: none"> 1. User requirements 2. System requirements <p>1 User requirements: User requirements are the statements in a natural language plus, diagrams of what services the system is expected to provide the users.</p> <p>Readers of the user requirements:</p> <ul style="list-style-type: none"> ->System manager -> Client managers -> System architects -> Client engineers <p>2 System requirements: These are more detailed specifications of the software system functions or features or services. The system requirements document should be defined exactly what is to be implemented.</p> <p>Readers of the system requirements:</p> <ul style="list-style-type: none"> -> Client manager ->Contract manager -> System architects ->Software developers 			
3(c)	Explain the interface specifications briefly.		5M	L2 C215.2
	<p>Ans . -INTERFACE SPECIFICATION: Most systems must operate with other systems and the operating interfaces must be specified as part of the requirements. Three types of interface may have to be defined.</p> <ol style="list-style-type: none"> 1. Procedural interfaces: Where existing programs or sub-systems offer a range of services that are accessed by calling interface procedures. These interfaces are sometimes called Application Programming Interfaces (APIs) 2. Data Structures that are exchanged: that are passed from one sub-system to another. Graphical data models are the best notations for this type of description. 3. Data representations: that have been established for an existing sub-system. 			
3(d)	Explain regarding the feasibility study in detail.		5M	L2 C215.2
	<p>Ans. <u>Feasibility study:</u></p> <p>Feasibility study in requirement engineering process of Software engineering is a study that represents the evaluation of feasibility of the proposed software (or) project (or) system.</p> <p>Goals of Feasibility study:</p> <p>→Feasibility study is the process of accessing the strengths & Weakness of the software (or)</p>			

project(or) System.

→It presents the directions of activities that will improve a project (3) software (or) System & achieve the desired output.

→ It determines whether the project is technically, financially, legally & operationally feasible within the estimated cost & time

Types of feasibility:

The Feasibility study is classified into five types. Those are,

1. Economical Feasibility
2. Operational Feasibility
3. Legal Feasibility
4. Technical Feasibility
5. Schedule Feasibility

1) Economical Feasibility:

→The Feasibility which deals with whether the system (or) project (or) Software Can be developed within the budget (or) not

2) operational Feasibility:

→This operational Feasibility is used to determine that how will the proposed system will be developed to solve the current problems.

3) Legal Feasibility:

→This legal Feasibility deals with the legal issues related to the system (or) project (or) Software such as cyber laws & copy rights.

4) Technical Feasibility:

→The Technical Feasibility deals with the level & type of technology needed for the system (or) Software (or) project.

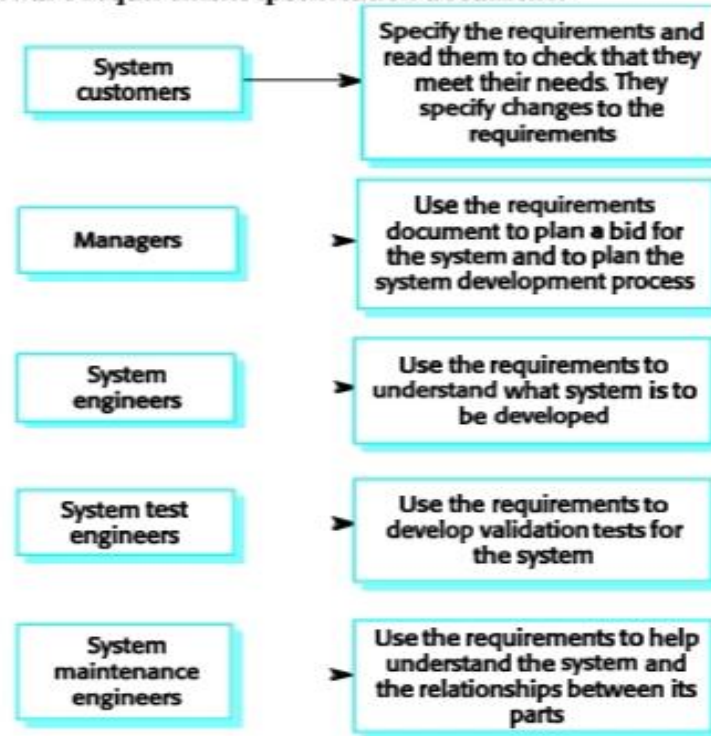
5) Schedule Feasibility:

→The Schedule Feasibility deals with whether the System (or) project (or) Software can be developed within the given time period (or) not.

3(e)	Explain regarding the viewpoints in detail.	5M	L2	C215.2
	<p>Ans.</p> <p>Viewpoints:</p> <p>Viewpoints are a way of structuring the requirements to represent the perspectives of different stakeholders. There are three types of viewpoints in the requirement elicitation techniques.</p> <p>Those are,</p> <ol style="list-style-type: none"> 1. Direct Viewpoints or Interactor Viewpoints 2. Indirect Viewpoints 3. Domain Viewpoints <p>1. Direct Viewpoints: Direct viewpoints involve stakeholders who will directly interact with the system. These individuals provide firsthand requirements based on their experiences and needs.</p> <p>2. Indirect Viewpoints: Indirect viewpoints represent stakeholders who do not use the system but influence the requirements. Their insights are crucial for understanding Broader needs and constraints.</p> <p>3. Domain Viewpoints: Domain viewpoints reflect the characteristics and constraints of the Environment in which the system will operate. These include considerations from existing systems or manual processes being replaced.</p>			
4(a)	Explain the purpose and key components of a Software Requirements Specification (SRS) document. Identify the different readers of an SRS and their concerns. And also describe the relevant IEEE standards for SRS documents.	10M	L2	C215.2
	<p>Ans. THE SOFTWARE REQUIREMENT SPECIFICATION DOCUMENT (SRS):</p> <ul style="list-style-type: none"> <input type="checkbox"/> A software requirement specifications (SRS) document is a detailed description of what a software will do and how it should perform. <input type="checkbox"/> A software requirement specifications (SRS) document lists the requirements, expectations, design, and standards for a future project. These include the high-level 			

business requirements dictating the goal of the project, end-user requirements and needs, and the product's functionality in technical terms. To put it simply, an SRS provides a detailed description of how a software product should work and how your development team should make it work.

Readers of software requirement specification document:



IEEE requirements standard defines a generic structure for a requirement specification document that must be instantiated for each specific system.

1. Introduction.

- i) Purpose of the requirements document
- ii) Scope of the project
- iii) Definitions, acronyms and abbreviations
- iv) References
- v) Overview of the remainder of the document

2. General description.

- i) Product perspective

	ii) Product functions iii) User characteristics iv) General constraints v) Assumptions and dependencies 3. Specific requirements cover functional, non-functional and interface requirements. The requirements may document external interfaces, describe system functionality and performance, specify logical database requirements, design constraints, emergent system properties and quality characteristics. 4. Appendices. 5. Index			
4(b)	Explain about the requirements elicitation techniques in detail.	10M	L2	C215.2
Ans	<p>Requirements Elicitation techniques:</p> <p>The requirements elicitation techniques are,</p> <ul style="list-style-type: none"> ➤ Viewpoints ➤ Interviews ➤ Scenarios ➤ Use Cases ➤ Surveys and Questionnaires ➤ Brainstorming and Prototyping <p><u>Viewpoints:</u></p> <p>Viewpoints are a way of structuring the requirements to represent the perspectives of different stakeholders. There are three types of viewpoints in the requirement elicitation techniques.</p> <p>Those are,</p> <p>1. Direct Viewpoints or Interactor Viewpoints</p>			

2. Indirect Viewpoints

3. Domain Viewpoints

1. **Direct Viewpoints:** Direct viewpoints involve stakeholders who will directly interact with the system. These individuals provide firsthand requirements based on their experiences and needs.

2. **Indirect Viewpoints:** Indirect viewpoints represent stakeholders who do not use the system but influence the requirements. Their insights are crucial for understanding broader needs and constraints.

3. **Domain Viewpoints:** Domain viewpoints reflect the characteristics and constraints of the environment in which the system will operate. These include considerations from existing systems or manual processes being replaced.

Interviewing:

In formal or informal interviewing, the Requirements Elicitation team puts questions to stakeholders about the system that they use and the system to be developed.

There are two types of interviews.

Closed interviews where a pre-defined set of questions are answered.

Open interviews where there is no pre-defined agenda and a range of issues are explored with stakeholders.

Scenarios:

Scenarios are real-life examples of how a system can be used.

They should include

A description of the starting situation;

A description of the normal flow of events;

A description of what can go wrong;

Information about other concurrent activities;

A description of the state when the scenario finishes.

Use cases:

Use-cases are a scenario based technique in the UML which identify the actors in an interaction and which describe the interaction itself.

	<p>A set of use cases should describe all possible interactions with the system.</p> <p>Sequence diagrams may be used to add detail to use-cases by showing the sequence of event processing in the system.</p>			
4(c)	Define requirements validation in the requirements engineering process. Explain its importance and briefly describe techniques used to validate requirements.	10M	L2	C215.2
<p>Ans.</p> <p><u>Requirements Validation:</u></p> <ul style="list-style-type: none"> • Concerned with demonstrating that the requirements define the system that the customer really wants. • Requirements error costs are high so validation is very important <ul style="list-style-type: none"> - Fixing a requirements error after delivery may cost up to 100 times the cost of fixing an implementation error. <p><u>Requirements checking:</u></p> <p>Validity: Does the system provide the functions which best support the customer's needs?</p> <p>Consistency: Are there any requirements conflicts?</p> <p>Completeness: Are all functions required by the customer included?</p> <p>Realism: Can the requirements be implemented given available budget and technology</p> <p>Verifiability. Can the requirements be checked?</p> <p><u>Requirements validation techniques:</u></p> <p>Requirements reviews:</p> <p>1) Systematic manual analysis of the requirements.</p> <p>Prototyping:</p> <p>1) Using an executable model of the system to check requirements.</p> <p>Test-case generation:</p> <p>1) Developing tests for requirements to check testability.</p> <p><u>Requirements reviews:</u></p> <p>1) Regular reviews should be held while the requirements definition is being formulated.</p>				

	<p>2)Both client and contractor staff should be involved in reviews.</p>
--	--

	<p>3) Reviews may be formal (with completed documents) or informal. Good communications between developers, customers and users can resolve problems at an early stage.</p>
--	---

Q. No	Question (s)	Marks	BL	CO
UNIT – III (Part-A)				
1(a)	Define design engineering.	1M	L1	C215.3
	Ans. It encompasses the set of principles, concepts, and practices that lead to the development of a high- quality system or product.			
1(b)	What is the goal of design engineering?	1M	L1	C215.3
	Ans. The goal of design engineering is to produce a model or representation that exhibits firmness, commodity.			
1(c)	List out Quality attributes.	1M	L1	C215.3
	Ans. The quality attributes are, 1. Functionality 2. Usability 3. Reliability 4. Performance 5. Supportability			
2(a)	List out any four design concepts.	3M	L1	C215.3
	Ans.(i) Abstraction (ii) Architecture (iii) Patterns (iv) Modularity			
2(b)	Define the concept of “use case”.	3M	L1	C215.3
	Ans. A use case describes the interactions between one or more Actors and the system in order to provide an observable result.			
3(a)	Briefly Explain Quality attributes.	5M	L2	C215.3
	Ans. The FURPS quality attributes represent a target for all software design: <ul style="list-style-type: none"> ➤ Functionality is assessed by evaluating the feature set and capabilities of the program, the generality of the functions that are delivered, and the security of the overall system. ➤ Usability is assessed by considering human factors, overall aesthetics, consistency and documentation. ➤ Reliability is evaluated by measuring the frequency and severity of failure, the accuracy of output results, and the mean – time –to- failure (MTTF), the ability to recover from failure, and the predictability of the program. ➤ Performance is measured by processing speed, response time, resource consumption, throughput, and efficiency ➤ Supportability combines the ability to extend the program (extensibility), adaptability, serviceability- these three attributes represent a more common term maintainability Not every software quality attribute is weighted equally as the software design is developed.			

One application may stress functionality with a special emphasis on security.

Another may demand performance with particular emphasis on processing speed

A third might focus on reliability

3(b) Explain regarding the Design Model.

5M

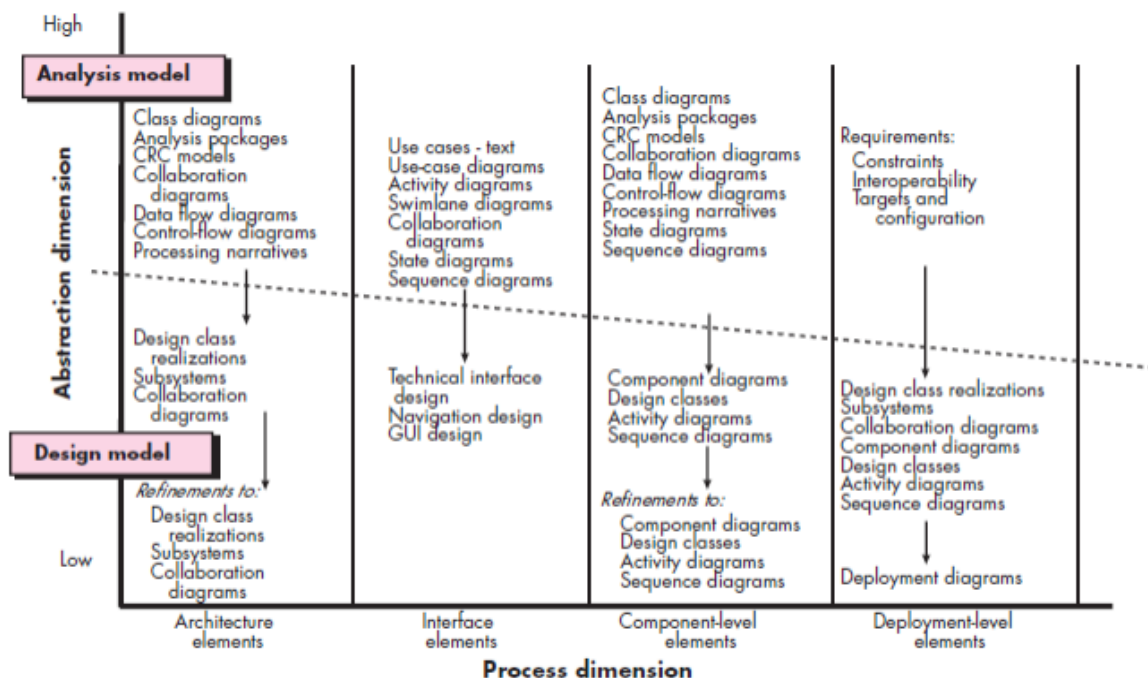
L2

C215.3

Ans. **THE DESIGN MODEL:**

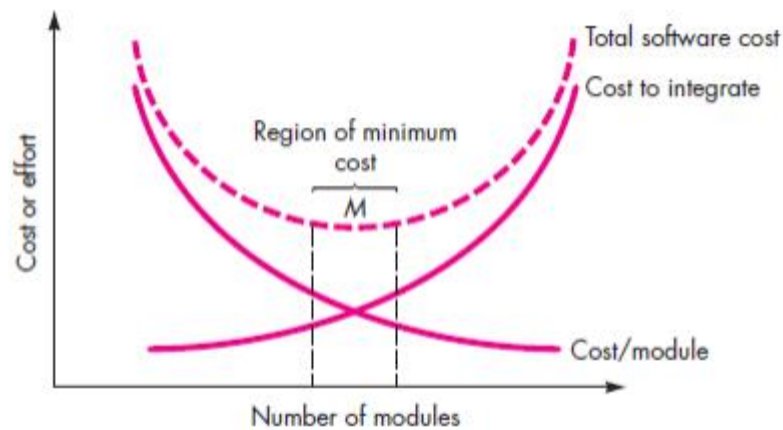
The design model can be viewed in two different dimensions as illustrated in Figure 8.4. The process dimension indicates the evolution of the design model as design tasks are executed as part of the software process. The abstraction dimension represents the level of detail as each element of the analysis model is transformed into a design equivalent and then refined iteratively. Referring to below Figure, the dashed line indicates the boundary between the analysis and design models. The analysis model slowly blends into the design and a clear distinction is less obvious. The elements of the design model use UML diagrams, that were used in the analysis model. The difference is that these diagrams are refined and elaborated as part of design; more implementation-specific detail is provided, and architectural structure and style, components that reside within the architecture, and interfaces between the components and with the outside world are all emphasized.

You should note, however, that model elements indicated along the horizontal axis are not always developed in a sequential fashion. The deployment model is usually delayed until the design has been fully developed.



4(a)	Explain regarding the following design concepts briefly. (i) Abstraction, (ii) Architecture, (iii) Patterns, (iv) Modularity	10M	L2	C215.3
	<p>Ans. Design concepts has evolved over the history of software engineering. Each concept provides the software designer with a foundation from which more sophisticated design methods can be applied. Some of the software design concepts that span both traditional and object-oriented software development is given below.</p> <p>(i) Abstraction: When you consider a modular solution to any problem, many levels of abstraction can be posed. At the highest level of abstraction, a solution is stated in broad terms using the language of the problem environment. At lower levels of abstraction, a more detailed description of the solution is provided. Finally, at the lowest level of abstraction, the solution is stated in a manner that can be directly implemented. A procedural abstraction refers to a sequence of instructions that have a specific and limited function. The name of a procedural abstraction implies these functions, but specific details are suppressed. A data abstraction is a named collection of data that describes a data object.</p> <p>(ii) Architecture: Software architecture alludes to “the overall structure of the software and the ways in which that structure provides conceptual integrity for a system”. In its simplest form, architecture is the structure or organization of program components (modules), the manner in which these components interact, and the structure of data that are used by the components. One goal of software design is to derive an architectural rendering of a system. A set of architectural patterns enables a software engineer to solve common design problems.</p> <p>Shaw and Garlan describe a set of properties as part of an architectural design:</p> <p>Structural properties: This aspect of the architectural design representation defines the components of a system (e.g., modules, objects, filters) and the manner in which those components are packaged and interact with one another. For example, objects are packaged to encapsulate both data and the processing that manipulates the data and interact via the invocation of methods.</p> <p>Extra-functional properties: The architectural design description should address how the design architecture achieves requirements for performance, capacity, reliability, security, adaptability, and other system characteristics.</p> <p>Families of related systems: The architectural design should draw upon repeatable patterns that are commonly encountered in the design of families of similar systems. In essence, the design should have the ability to reuse architectural building blocks.</p> <p>(iii) Patterns: A pattern is a named nugget of insight which conveys the essence of a proven solution to a recurring problem within a certain context amidst competing concerns. Stated A design pattern describes a design structure that solves a particular design problem within a specific context and amid “forces” that may have an impact on the manner in which the pattern is applied and used.</p> <p>The intent of each design pattern is to provide a description that enables a designer to determine</p> <ol style="list-style-type: none"> (1) whether the pattern is applicable to the current work (2) whether the pattern can be reused (hence, saving design time) (3) whether the pattern can serve as a guide for developing a similar, but functionally or structurally different pattern. <p>(iv) Modularity: Modularity is the most common manifestation of separation of concerns. Software is divided into separately named and addressable components, sometimes called modules, that are integrated to satisfy problem requirements. It has been stated that “modularity is the single attribute of software that allows a program to be intellectually manageable”. The number of control paths, span of reference, number of variables, and overall complexity would make understanding close to impossible. In almost all instances, you should break the design into many</p>			

modules, hoping to make understanding easier and, as a consequence, reduce the cost required to build the software. If you subdivide software indefinitely the effort required to develop it will become negligibly small! Unfortunately, other forces come into play, causing this conclusion to be (sadly) invalid. Referring to below figure, the effort (cost) to develop an individual software module does decrease as the total number of modules increases.



Given the same set of requirements, more modules means smaller individual size. However, as the number of modules grows, the effort (cost) associated with integrating the modules also grows. These characteristics lead to a total cost or effort curve shown in the figure. There is a number, M , of modules that would result in minimum development cost, but we do not have the necessary sophistication to predict M with assurance.

The curves shown in above figure do provide useful qualitative guidance when modularity is considered. You should modularize, but care should be taken to stay in the vicinity of M . **Undermodularity** or **overmodularity** should be avoided.

You modularize a design (and the resulting program) so that development can be more easily planned; software increments can be defined and delivered; changes can be more easily accommodated; testing and debugging can be conducted more efficiently, and long-term maintenance can be conducted without serious side effects.