



CMR INSTITUTE OF TECHNOLOGY		USN <input type="text"/>							
Internal Assessment Test - I									
Sub:	Software Engineering						Code:	17MCA34	
Date:	08-09-2018	Duration:	90 mins	Max Marks:	50	Sem:	III	Branch:	MCA
Answer ONE FULL QUESTION from each part									
								Marks	OBE CO RBT
Part - I									
1 (a)	What is software engineering? Explain the essential attributes of good software						06	CO1	L2
(b)	Distinguish the difference between generic product and customized product						04	CO1	L2
(OR)									
2	What do you mean by software engineering diversity? Explain in detail.						10	CO1	L1
Part - II									
3 (a)	Discuss the four fundamental activities that are common to all software processes						04	CO2	L2
(b)	Explain the phases of RUP with a neat diagram.						06	CO2	L1
(OR)									
4	Discuss the process involved in waterfall and incremental development model.						10	CO2	L2

CMR INSTITUTE OF TECHNOLOGY		USN <input type="text"/>							
Internal Assessment Test - I									
Sub:	Software Engineering						Code:	17MCA34	
Date:	08-09-2018	Duration:	90 mins	Max Marks:	50	Sem:	III	Branch:	MCA
Answer ONE FULL QUESTION from each part									
								Marks	OBE CO RBT
Part - I									
1 (a)	What is software engineering? Explain the essential attributes of good software						06	CO1	L2
(b)	Distinguish the difference between generic product and customized product						04	CO1	L2
(OR)									
2	What do you mean by software engineering diversity? Explain in detail.						10	CO1	L1
Part - II									
3 (a)	Discuss the four fundamental activities that are common to all software processes						04	CO2	L2
(b)	Explain the phases of RUP with a neat diagram.						06	CO2	L1
(OR)									
4	Discuss the process involved in waterfall and incremental development model.						10	CO2	L2

PART - III					
5	(a)	What is meant by pair programming? Mention the advantages of pair programming	05	CO2	L2
	(b)	What is scrum? Discuss the various phases involved in scrum process.	05	CO2	L2
(OR)					
6	(a)	Explain the principles of agile methods.	06	CO2	L2
	(b)	Extreme programming expresses user requirements as stories, with each story written on a card. Write two examples of such story card of your own application.	04	CO2	L3
Part – IV					
7		Explain in detail about extreme programming.	10	CO2	L2
(OR)					
8		Explain with neat diagram, the various activities in requirement engineering process.	10	CO1	L2
Part – V					
9	(a)	Define: (i) Functional requirements (ii) Non-functional requirements	04	CO1	L1
	(b)	Explain the different metrics of non-functional requirements.	06	CO1	L2
(OR)					
10	(a)	What is requirement specification? Explain various ways of writing system requirements	06	CO2	L1
	(b)	What are the problems in domain requirements?	04	CO2	L1

PART - III					
5	(a)	What is meant by pair programming? Mention the advantages of pair programming	05	CO2	L2
	(b)	What is scrum? Discuss the various phases involved in scrum process.	05	CO2	L2
(OR)					
6	(a)	Explain the principles of agile methods.	06	CO2	L2
	(b)	Extreme programming expresses user requirements as stories, with each story written on a card. Write two examples of such story card of your own application.	04	CO2	L3
Part – IV					
7		Explain in detail about extreme programming.	10	CO2	L2
(OR)					
8		Explain with neat diagram, the various activities in requirement engineering process.	10	CO1	L2
Part – V					
9	(a)	Define: (i) Functional requirements (ii) Non-functional requirements	04	CO1	L1
	(b)	Explain the different metrics of non-functional requirements.	06	CO1	L2
(OR)					
10	(a)	What is requirement specification? Explain various ways of writing system requirements	06	CO2	L1
	(b)	What are the problems in domain requirements?	04	CO2	L1

1. (a) What is software engineering? Explain the essential (6) attributes of good software

- **Software Engineering:** an engineering discipline that is concerned with all aspects of software production. (or) Software engineering is a systematic approach to the production of software that takes into account practical cost, schedule, and dependability issues, as well as the needs of software customers and producers.

- **Attributes of good software:**

- Maintainability
- Dependability and security
- Efficiency
- Acceptability

1. (b) Distinguish the difference between generic product and (4) customized product

i. Generic products: These are stand-alone systems that are produced by a development organization and sold on the open market to any customer who is able to buy them.

ii. Customized (or bespoke) products: These are systems that are commissioned by a particular customer.

(OR)

2. What do you mean by software engineering diversity? (10) Explain in detail.

- a diverse set of software engineering methods and tools has evolved over the past 50 years, as there are no universal software engineering

methods and techniques that are suitable for all systems and all companies

- Stand-alone applications:** These are application systems that run on a local computer, such as a PC.
- Interactive transaction-based applications:** These are applications that execute on a remote computer and that are accessed by users from their own PCs or terminals.
- Embedded control systems:** These are software control systems that control and manage hardware devices.
- Batch processing systems:** These are business systems that are designed to process data in large batches.
- Entertainment systems:** These are systems that are primarily for personal use and which are intended to entertain the user.
- Systems for modeling and simulation:** These are systems that are developed by scientists and engineers to model physical processes or situations, which include many, separate, interacting objects.
- Data collection systems:** These are systems that collect data from their environment using a set of sensors and send that data to other systems for processing.
- Systems of systems:** These are systems that are composed of a number of other software systems.

3. (a) Discuss the four fundamental activities that are common (4) to all software processes

- Software specification:** where customers and engineers define the software that is to be produced and the constraints on its operation.

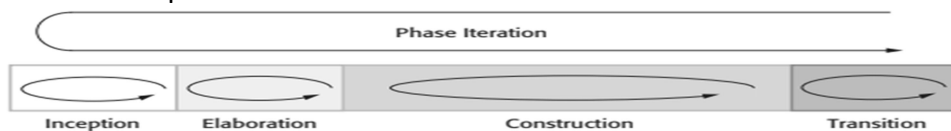
ii. Software development: where the software is designed and programmed.

iii. Software validation: where the software is checked to ensure that it is what the customer requires.

iv. Software evolution: where the software is modified to reflect changing customer and market requirements.

3. (b) Explain the phases of RUP with a neat diagram (6)

- an example of a modern process model that has been derived from the UML
- recognizes that conventional process models present a single view of the process
- The RUP is a phased model that identifies four discrete phases in the software process:



(i) Inception:

- Establish a business case for the system
- Identify the external entities
- Assess the contribution that the system makes to the business

(ii) Elaboration:

- Develop an understanding of the problem domain
- Establish an architectural framework for the system
- Develop the product plan
- Identify key project risks

(iii) Construction:

- Involves system design, programming and testing
- Parts of the system are developed in parallel and integrated

(iv) Transition:

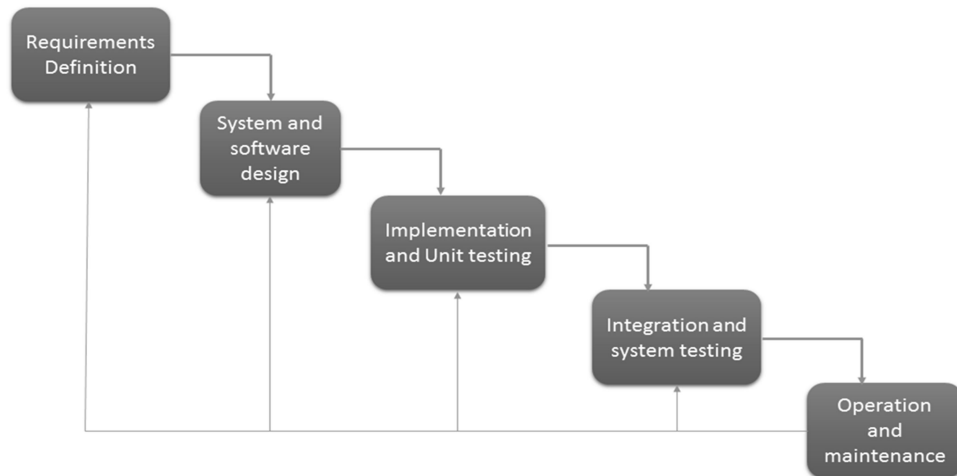
- Moving the system from the development community to the user community and making it work in a real environment
- Should have a documented software system that is working correctly in its operational environment

(OR)

4. Discuss the process involved in waterfall and incremental (10) development model.

(i) Waterfall model:

- It is an example of a plan-driven process; you must plan and schedule all of the process activities before starting work on them.
- The principal stages of the waterfall model directly reflect the fundamental development activities:



1. Requirements analysis and definition: The system's services, constraints, and goals are established by consultation with system users.

2. System and software design:

- The systems design process allocates the requirements to either hardware or software systems by establishing an overall system architecture.
- Software design involves identifying and describing the fundamental software system abstractions and their relationships.

3. Implementation and unit testing: During this stage, the software design is realized as a set of programs or program units.

4. 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.

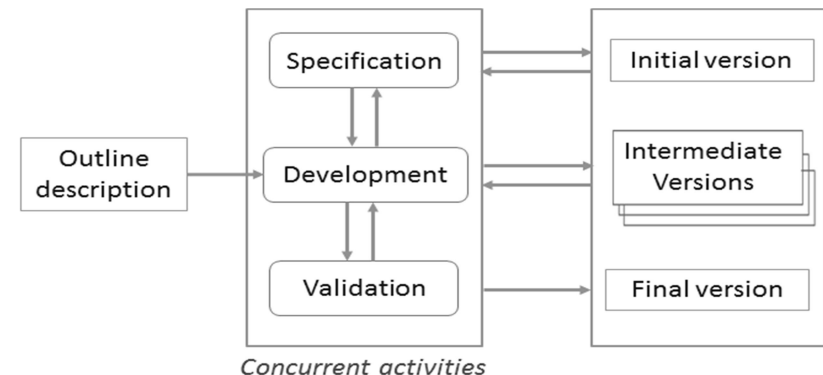
5. Operation and maintenance:

- The system is installed and put into practical use.

- 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.

(ii) Incremental Development model:

- based on the idea of:
 - developing an initial implementation
 - exposing this to user comment
 - evolving it through several versions until an adequate system has been developed
- Each increment or version of the system incorporates some of the functionality that is needed by the customer.
- It is a fundamental part of agile approaches

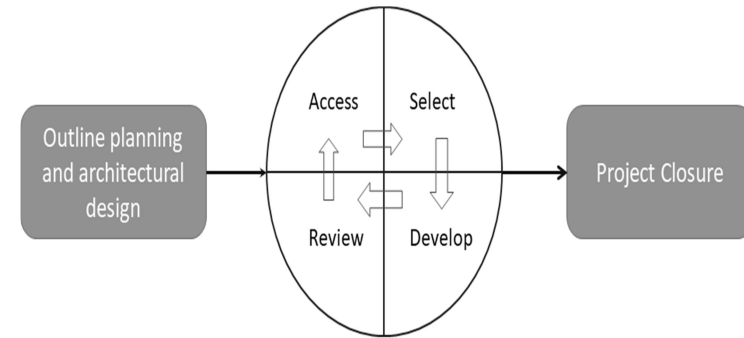


5. (a) What is meant by pair programming? Mention the (5) advantages of pair programming

- Pair programming involves two programmers, when one does coding the other one will do unit testing.
- **Advantages:**
 - i. The time consumption of producing the deliverable is reduced.
 - ii. It offers more productivity.
 - iii. The quality of the product delivery is enhanced.
 - iv. The effort produced by the individuals are improved.

5. (b) What is scrum? Discuss the various phases involved in (5) scrum process.

- Scrum approach is a general agile method but its focus is on managing iterative development
- There are three phases in Scrum.
 - i.* The **initial phase** is an outline planning phase where you establish the general objectives for the project and design the software architecture.
 - ii.* This is followed by a series of **sprint cycles**, where each cycle develops an increment of the system.
 - iii.* The project **closure phase** wraps up the project, completes required documentation such as system help frames and user manuals and assesses the lessons learned from the project.



(OR)

6. (a) Explain the principles of agile methods (6)

- i.* **Customer involvement:** Customers should be closely involved throughout the development process.
- ii.* **Incremental delivery:** The software is developed in increments with the customer specifying the requirements to be included in each increment.
- iii.* **People not process:** The skills of the development team should be recognized and exploited.
- iv.* **Embrace change:** Expect the system requirements to change and so design the system to accommodate these changes.
- v.* **Maintain simplicity:** Focus on simplicity in both the software being developed and in the development process.

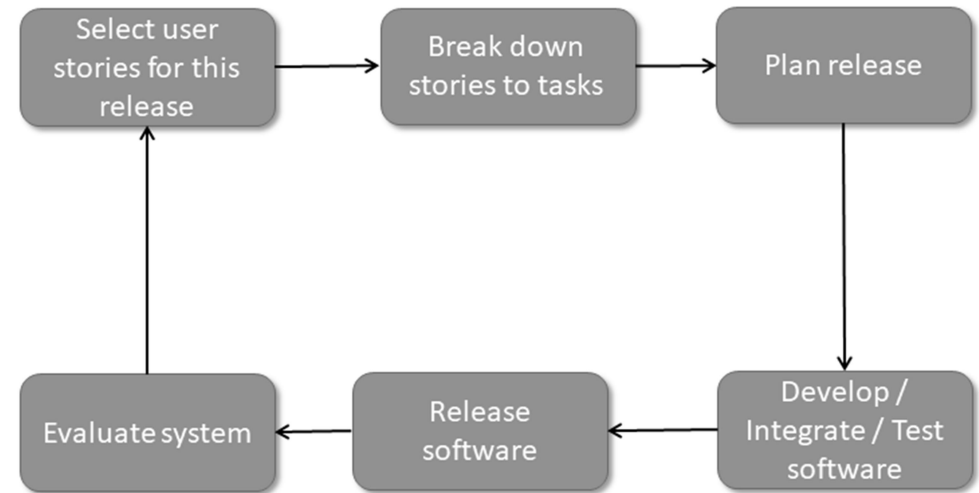
6. (b) Extreme programming expresses user requirements as (4) stories, with each story written on a card. Write two examples of such story card of your own application.

- i. The customer wants to search for the flights between two cities with the best price
- ii. The client can download the content of the page in pdf format after paying the necessary payment
- iii. A user who wants to access the system should provide the user's First name, last name, and E-mail address
- iv. Create a new booking from within the Customer object, where the customer and the preferred payment method, and telephone are copied in automatically.

N.B.: The students can write their own story cards

7. Explain in detail about extreme programming. (10)

- perhaps the best known and most widely used of the agile methods
- the approach was developed by pushing recognized good practice, such as iterative development, to 'extreme' levels.
- People not process through pair programming, collective ownership and a process that avoids long working hours.
- Maintaining simplicity through constant refactoring of code.
- requirements are expressed as scenarios (called user stories)



XP Release cycle

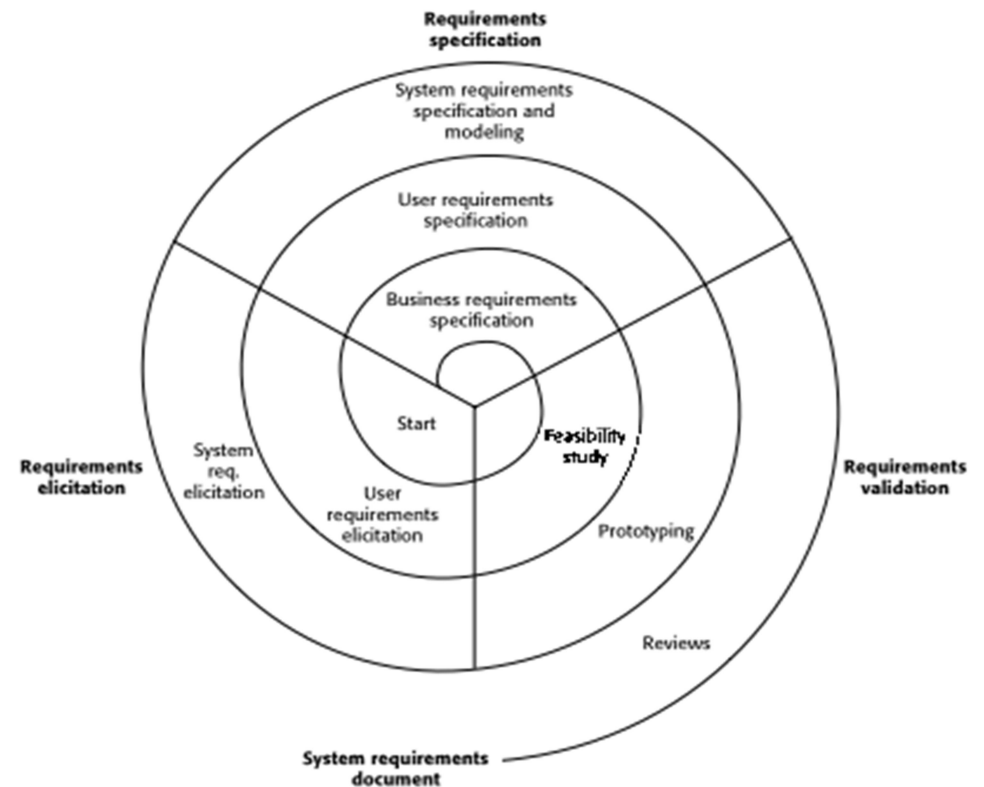
- Extreme programming involves a number of practices
 - i. **Incremental planning:** Requirements are recorded on Story Cards and the Stories to be included in a release are determined by the time available and their relative priority.
 - ii. **Small releases:** Releases of the system are frequent and incrementally add functionality to the first release.
 - iii. **Simple design:** Enough design is carried out to meet the current requirements.
 - iv. **Test-first development:** An automated unit test framework is used to write tests for a new piece of functionality before that functionality itself is implemented.
 - v. **Refactoring:** All developers are expected to refactor the code continuously as soon as possible code improvements are found to keep the code simple and maintainable.

- vi. **Pair programming:** Developers work in pairs, checking each other's work and providing the support to always do a good job.
- vii. **Collective ownership:** The pairs of developers work on all areas of the system, so that no islands of expertise develop and all the developers take responsibility for all of the code.
- viii. **Continuous integration:** As soon as the work on a task is complete, it is integrated into the whole system.
- ix. **Sustainable pace:** Large amounts of overtime are not considered acceptable as the net effect is often to reduce code quality and medium term productivity
- x. **On-site customer:** A representative of the end-user of the system (the Customer) should be available full time for the use of the XP team.

(OR)

8. Explain with neat diagram, the various activities in (10) requirement engineering process.

- include four high-level activities.
 - **feasibility study:** the system is useful to the business
 - **elicitation and analysis:** discovering requirements
 - **Validation:** checking that the requirements actually define the system that the customer wants
 - **Specification:** Converting these requirements into some standard form



- Spiral view of requirements engineering process:
 - Requirements engineering is an iterative process, around a spiral, in which the activities are interleaved.
 - The number of iterations around the spiral can vary so the spiral can be exited after some or all of the user requirements have been elicited.

- The set of models describes the behavior of the system and is annotated with additional information describing the system's required performance or reliability.

9. (a) Define: (i) Functional requirements (ii) Non-functional requirements (4)

- **Functional requirements:** Statements of services the system should provide; it may state what the system should do
- **Non-functional requirements:** Constraints on the services or functions offered by the system; these are applied to the whole system

9. (b) Explain the different metrics of non-functional requirements. (6)

- Speed
- Size
- Ease of use
- Reliability
- Robustness
- Portability

(OR)

10. (a) What is requirement specification? Explain various ways of writing system requirements (6)

- the process of writing down the user and system requirements in a requirements document.

- the user and system requirements should be clear, unambiguous, easy to understand, complete, and consistent.

● **Various ways of writing system requirements:**

(i) Natural language: requirements are written using numbered sentences in natural language

(ii) Structural natural language: The requirements are written in natural language on a standard form or template.

(iii) Design description languages: This approach uses a language like a programming language, but with more abstract features to specify the requirements

(iv) Graphical notations: Graphical models, such as UML diagrams, supplemented by text annotations, are used to define the functional requirements for the system

(v) Mathematical specifications: These notations are based on mathematical concepts such as finite-state machines or sets.

10. (b) What are the problems in domain requirements? (4)

i. Understandability: Requirements are expressed in the language of the application domain; this is often not understood by software engineers developing the system.

ii. Implicitness: Domain specialists understand the area so well that they do not think of making the domain requirements explicit.