CMR
INSTIT UTE OF
TECHNLOGY

USN | | | | | | | | | | |

Third Internal Test-May 2019

Sub:	**Software Engineering**					Sub Code:	17CS45	Branch:	**ISE**
Date:	15/05/2019	Duration:	90 min's	Max Marks:	50	Sem / Sec:	IV- A		

Scheme and Solutions

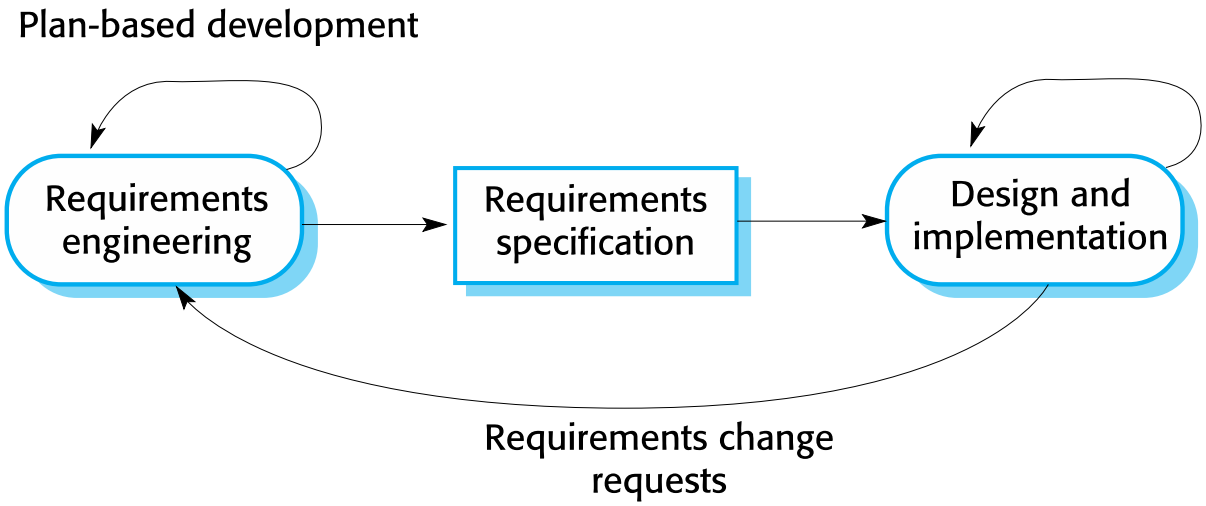
1 (a). State the principles of agile methods.

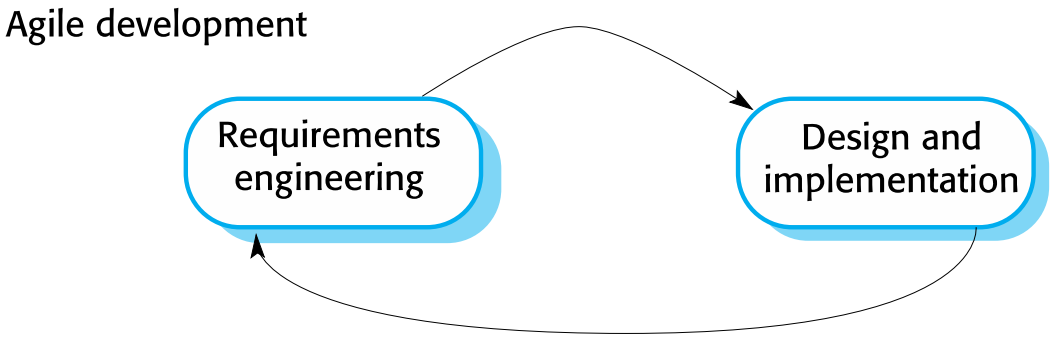
	Principle	**Description**
	Customer involvement	Customers should be closely involved throughout the development process. Their role is provide and prioritize new system requirements and to evaluate the iterations of the system.
	Incremental delivery	The software is developed in increments with the customer specifying the requirements to be included in each increment.
	People not process	The skills of the development team should be recognized and exploited. Team members should be left to develop their own ways of working without prescriptive processes.
	Embrace change	Expect the system requirements to change and so design the system to accommodate these changes.
	Maintain simplicity	Focus on simplicity in both the software being developed and in the development process. Wherever possible, actively work to eliminate complexity from the system.

1 (b). Explain plan driven agile development process

- ✧ Program specification, design and implementation are inter-leaved
- ✧ The system is developed as a series of versions or increments with stakeholders involved in version specification and evaluation
- ✧ Frequent delivery of new versions for evaluation
- ✧ Extensive tool support (e.g. automated testing tools) used to support development.
- ✧ Minimal documentation – focus on working code
- ✧
- ✧ Plan-driven development
  - A plan-driven approach to software engineering is based around separate development stages with the outputs to be produced at each of these stages planned in advance.
  - Not necessarily waterfall model – plan-driven, incremental development is possible
  - Iteration occurs within activities.
- ✧ Agile development

- Specification, design, implementation and testing are inter-leaved and the outputs from the development process are decided through a process of negotiation during the software development process.





2. Explain the practices involved in extreme programming

Principle or practice	**Description**
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. The developers break these stories into development 'Tasks'
Small releases	The minimal useful set of functionality that provides business value is developed first. Releases of the system are frequent and incrementally add functionality to the first release.
Simple design	Enough design is carried out to meet the current requirements and no more.
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.
Refactoring	All developers are expected to refactor the code continuously as soon as possible code improvements are found. This keeps the code simple and maintainable.

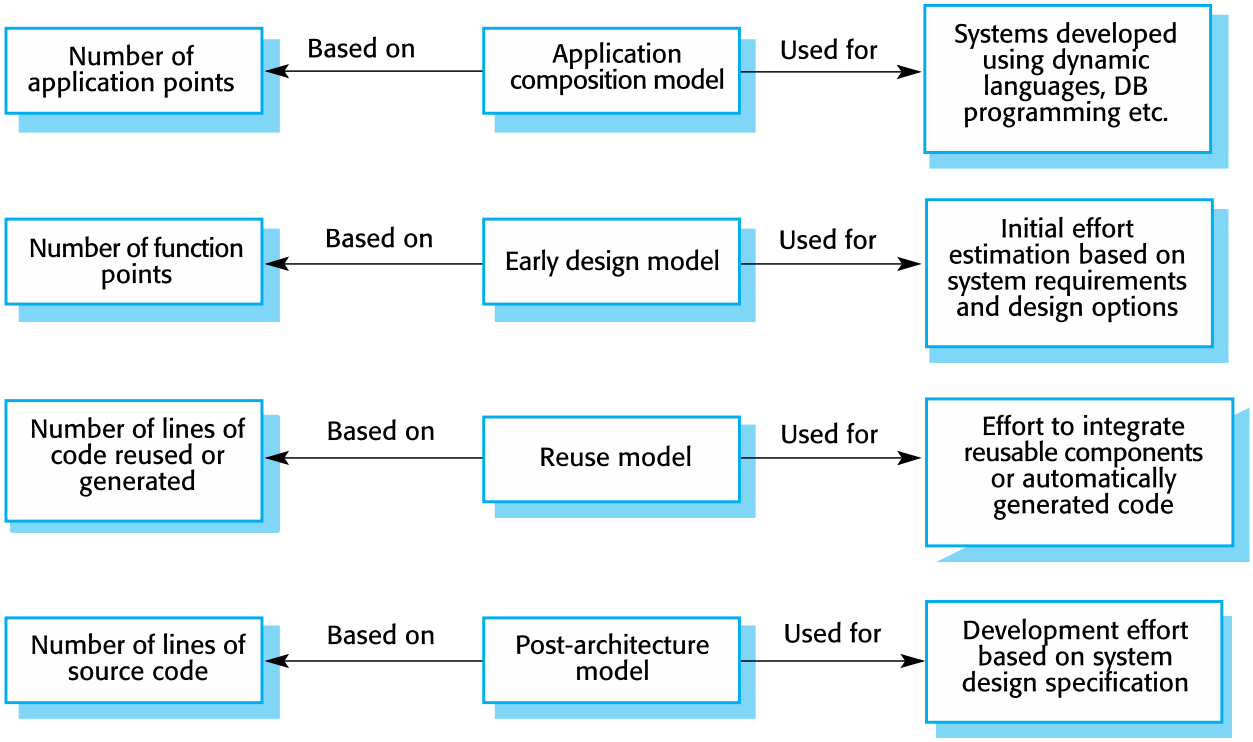
Pair programming	Developers work in pairs, checking each other's work and providing the support to always do a good job.
Collective ownership	The pairs of developers work on all areas of the system, so that no islands of expertise develop and all the developers take responsibility for all of the code. Anyone can change anything.
Continuous integration	As soon as the work on a task is complete, it is integrated into the whole system. After any such integration, all the unit tests in the system must pass.
Sustainable pace	Large amounts of overtime are not considered acceptable as the net effect is often to reduce code quality and medium term productivity
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. In an extreme programming process, the customer is a member of the development team and is responsible for bringing system requirements to the team for implementation.

3. List and explain the various COCOMO cost estimation models

- COCOMO 2 incorporates a range of sub-models that produce increasingly detailed software estimates.
- The sub-models in COCOMO 2 are:
  - Application composition model. Used when software is composed from existing parts.
    - Supports prototyping projects and projects where there is extensive reuse.
    - Based on standard estimates of developer productivity in application (object) points/month.
    - Takes software tool use into account.
    - Formula is
    - PM = (NAP ´ (1 - %reuse/100)) / PROD
    - PM is the effort in person-months, NAP is the number of application points and PROD is the productivity.
  - Early design model. Used when requirements are available but design has not yet started.
    - Estimates can be made after the requirements have been agreed.
    - Based on a standard formula for algorithmic models
    - PM = A ´ $Size^B$ ´ M where
    - M = PERS ´ RCPX ´ RUSE ´ PDIF ´ PREX ´ FCIL ´ SCED;
    - A = 2.94 in initial calibration,
    - Size in KLOC,
    - B varies from 1.1 to 1.24 depending on novelty of the project, development flexibility, risk management approaches and the process maturity.
  - Reuse model. Used to compute the effort of integrating reusable components.
    - For generated code:
    - PM = (ASLOC * AT/100)/ATPROD
    - ASLOC is the number of lines of generated code
    - AT is the percentage of code automatically generated.
    - ATPROD is the productivity of engineers in integrating this code.
    - When code has to be understood and integrated:
    - ESLOC = ASLOC * (1-AT/100) * AAM.
    - ASLOC and AT as before.

- ✧ AAM is the adaptation adjustment multiplier computed from the costs of changing the reused code, the costs of understanding how to integrate the code and the costs of reuse decision making.

- Post-architecture model. Used once the system architecture has been designed and more information about the system is available.
  - ✧ The code size is estimated as:
  - ✧ Number of lines of new code to be developed;
  - ✧ Estimate of equivalent number of lines of new code computed using the reuse model;
  - ✧ An estimate of the number of lines of code that have to be modified according to requirements changes.
  - ✧



4. What are product metrics? Explain the different type's software product metrics.

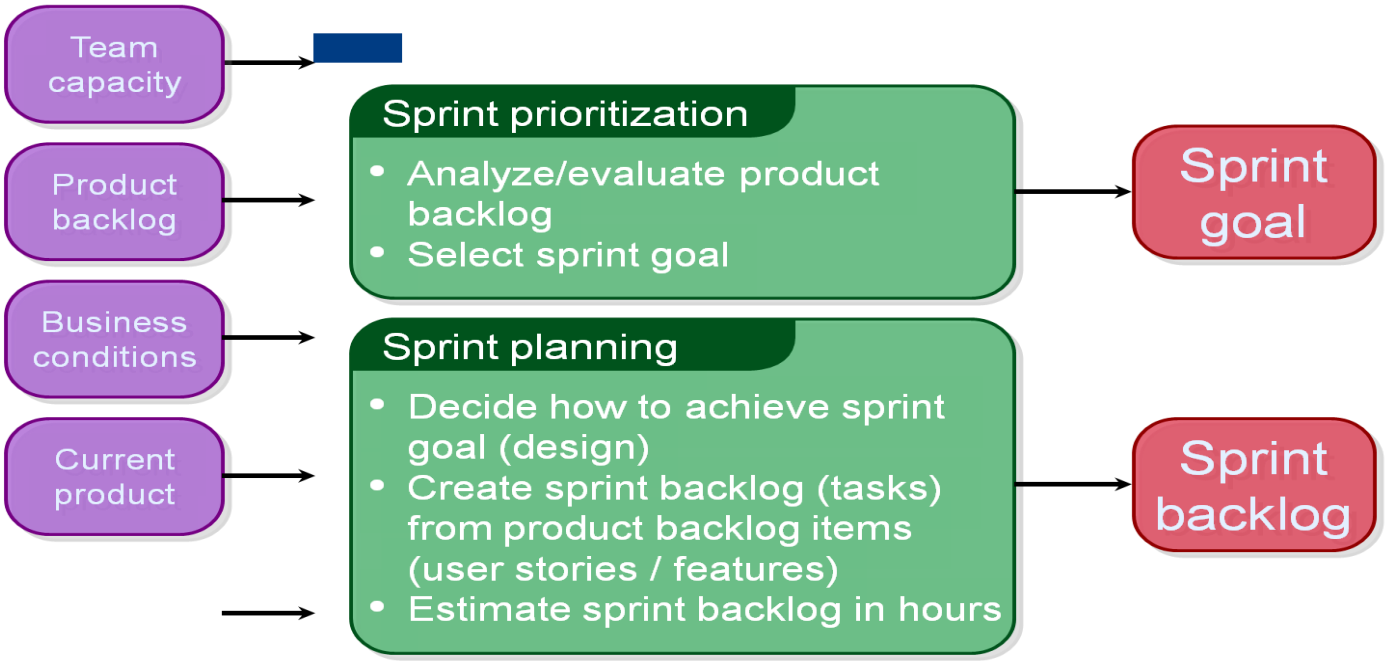
- ✧ A quality metric should be a predictor of product quality.
- ✧ Classes of product metric
  - Dynamic metrics which are collected by measurements made of a program in execution;
  - Static metrics which are collected by measurements made of the system representations;
  - Dynamic metrics help assess efficiency and reliability
  - Static metrics help assess complexity, understandability and maintainability.
- ✧ Dynamic metrics are closely related to software quality attributes
  - It is relatively easy to measure the response time of a system (performance attribute) or the number of failures (reliability attribute).
- ✧ Static metrics have an indirect relationship with quality attributes
  - You need to try and derive a relationship between these metrics and properties such as complexity, understandability and maintainability.

Software metric	**Description**
Fan-in/Fan-out	Fan-in is a measure of the number of functions or methods that call another function or method (say X). Fan-out is the number of functions that are called by function X. A high value for fan-in

	means that X is tightly coupled to the rest of the design and changes to X will have extensive knock-on effects. A high value for fan-out suggests that the overall complexity of X may be high because of the complexity of the control logic needed to coordinate the called components.
Length of code	This is a measure of the size of a program. Generally, the larger the size of the code of a component, the more complex and error-prone that component is likely to be. Length of code has been shown to be one of the most reliable metrics for predicting error-proneness in components.
Software metric	Description
Cyclomatic complexity	This is a measure of the control complexity of a program. This control complexity may be related to program understandability. I discuss cyclomatic complexity in Chapter 8.
Length of identifiers	This is a measure of the average length of identifiers (names for variables, classes, methods, etc.) in a program. The longer the identifiers, the more likely they are to be meaningful and hence the more understandable the program.
Depth of conditional nesting	This is a measure of the depth of nesting of if-statements in a program. Deeply nested if-statements are hard to understand and potentially error-prone.
Fog index	This is a measure of the average length of words and sentences in documents. The higher the value of a document's Fog index, the more difficult the document is to understand.

5. Draw and explain the block diagram of SCRUM process and list the characteristics of SCRUM process.

The Product Owner is responsible for maximizing return on investment (ROI) by identifying product features, translating these into a prioritized list, deciding which should be at the top of the list for the next Sprint, and continually re-prioritizing and refining the list.



Scrum calls for four ceremonies that bring structure to each sprint:

- Sprint planning: A team planning meeting that determines what to complete in the coming sprint.
- Daily stand-up: Also known as a Daily Scrum, a 15- minute mini-meeting for the software team to sync.
- Sprint demo: A sharing meeting where the team shows what they've shipped in that sprint.
- Sprint retrospective: A review of what did and didn't go well with actions to make the next sprint better

The Project Owner:

- creates a prioritized wish list called a product backlog
- closely partners with the business and the team to ensure everyone understands the work items in the product backlog
- gives the team clear guidance on which features to deliver next
- decides when to ship the product with a preference towards more frequent delivery

6. Explain how the reviews and inspections are used to check the quality of project delivery with checklist.

- A group examines part or all of a process or system and its documentation to find potential problems.
- Software or documents may be 'signed off' at a review which signifies that progress to the next development stage has been approved by management.
- There are different types of review with different objectives
  - Inspections for defect removal (product);
  - Reviews for progress assessment (product and process);
  - Quality reviews (product and standards).
- Pre-review activities
  - Pre-review activities are concerned with review planning and review preparation

- ✧ 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.
- ✧ Post-review activities
  - These address the problems and issues that have been raised during the review meeting.

Fault class	Inspection check
Data faults	• Are all program variables initialized before their values are used? • Have all constants been named? • Should the upper bound of arrays be equal to the size of the array or Size - 1KB • If character strings are used, is a delimiter explicitly assigned? • Is there any possibility of buffer overflow?
Control faults	• For each conditional statement, is the condition correct? • Is each loop certain to terminate? • Are compound statements correctly bracketed? • In case statements, are all possible cases accounted for? • If a break is required after each case in case statements, has it been included?
Input/output faults	• Are all input variables used? • Are all output variables assigned a value before they are output? • Can unexpected inputs cause corruption?
Fault class	Inspection check
Interface faults	• Do all function and method calls have the correct number of parameters? • Do formal and actual parameter types match? • Are the parameters in the right order? • If components access shared memory, do they have the same model of the shared memory structure?
Storage management faults	• If a linked structure is modified, have all links been correctly reassigned? • If dynamic storage is used, has space been allocated correctly? • Is space explicitly deallocated after it is no longer required?
Exception management faults	• Have all possible error conditions been taken into account?

7 Explain ISO 9001 standards framework process and quality management

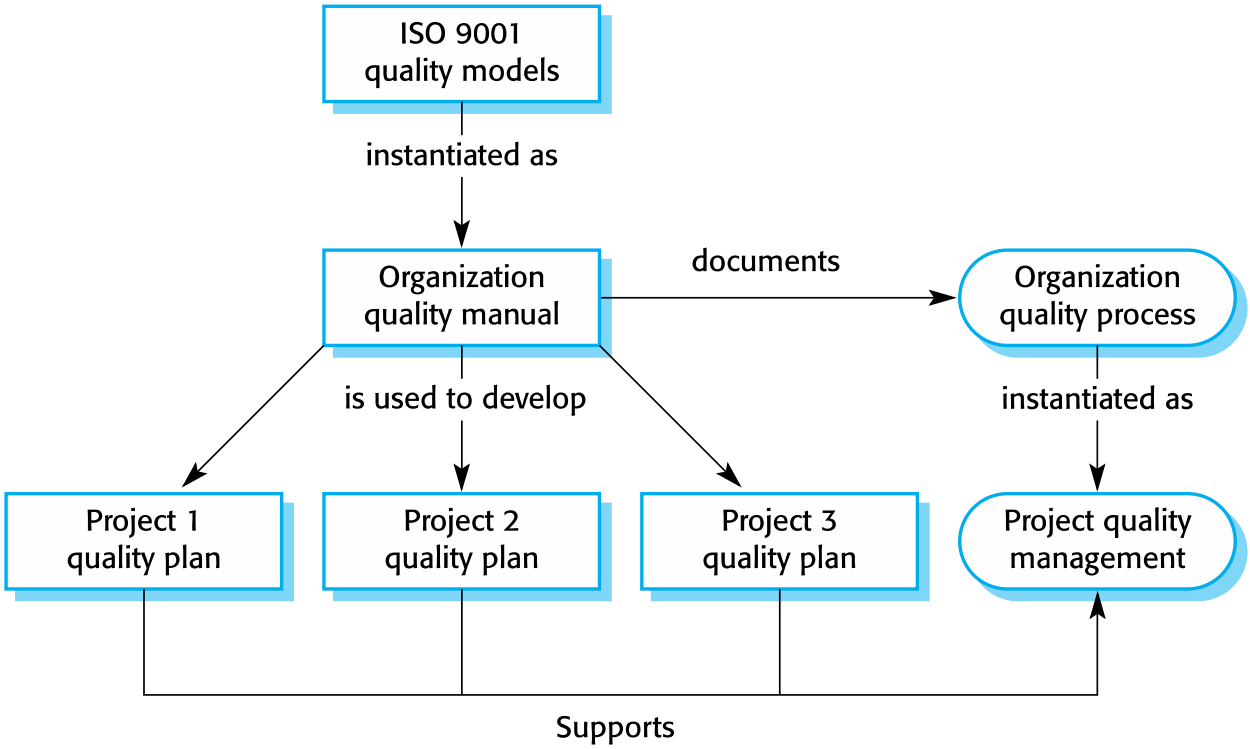
- ✧ An international set of standards that can be used as a basis for developing quality management systems.
- ✧ ISO 9001, the most general of these standards, applies to organizations that design, develop and maintain products, including software.
- ✧ The ISO 9001 standard is a framework for developing software standards.
  - It sets out general quality principles, describes quality processes in general and lays out the organizational standards and procedures that should be defined. These should be documented in an organizational quality manual.

- Quality standards and procedures should be documented in an organisational quality manual.
- An external body may certify that an organisation's quality manual conforms to ISO 9000 standards.
- Some customers require suppliers to be ISO 9000 certified although the need for flexibility here is increasingly recognised.

✧ The ISO 9001 certification is inadequate because it defines quality to be the conformance to standards.

✧ It takes no account of quality as experienced by users of the software. For example, a company could define test coverage standards specifying that all methods in objects must be called at least once.

✧ Unfortunately, this standard can be met by incomplete software testing that does not include tests with different method parameters. So long as the defined testing procedures are followed and test records maintained, the company could be ISO 9001 certified.



8a Explain the key features of testing XP

✧ Extreme programming has a technical focus and is not easy to integrate with management practice in most organizations.

✧ Consequently, while agile development uses practices from XP, the method as originally defined is not widely used.

✧ Key practices

- User stories for specification
- Refactoring
- Test-first development
- Pair programming

8b Write a short note software quality

✧ Concerned with ensuring that the required level of quality is achieved in a software product.

✧ Three principal concerns:

- At the organizational level, quality management is concerned with establishing a framework of organizational processes and standards that will lead to high-quality software.
- At the project level, quality management involves the application of specific quality processes and checking that these planned processes have been followed.

- At the project level, quality management is also concerned with establishing a quality plan for a project. The quality plan should set out the quality goals for the project and define what processes and standards are to be used.

✧ Quality management provides an independent check on the software development process.

✧ The quality management process checks the project deliverables to ensure that they are consistent with organizational standards and goals

✧ The quality team should be independent from the development team so that they can take an objective view of the software. This allows them to report on software quality without being influenced by software development issues.

7	Explain ISO 9001 standards framework process and quality management	[10]	CO5	L3
8a	Explain the key features of testing XP	[05]	CO4	L3
8b	Write a short note software quality	[05]	CO5	L1

Course Outcomes		Modules covered	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3	PSO4
CO1	Design a software system, component, or process to meet desired needs within realistic constraints.	**1,2,5**	2	1	3	-	-	-	-	-	-	1	-	-	1	-	-	-
CO2	Assess professional and ethical responsibility	**1**	-	-	-	-	-	-	-	3	-	-	-	-	-	-	-	-
CO3	Function on multi-disciplinary teams	**4, mini-project**	-	1	-	1	-	-	-	-	3	1	1	2	2	-	-	-
CO4	Use the techniques, skills, and modern engineering tools necessary for engineering practice	**1,2,3,4,5**	2	-	-	-	2	-	-	-	-	-	-	-	-	-	-	-
CO5	Analyze, design, implement, verify, validate, implement, apply, and maintain software systems or parts of software systems.	**1,2,3,4**	2	1	2	1	-	-	-	-	-	1	-	-	1	-	-	-
CO6	Demonstrate an understanding of and apply current theories, models, and techniques that provide a basis for the software lifecycle	**1,2**	2	3	3	2	1	-	-	-	-	-	2		3	-	2	-

COGNITIVE LEVEL	REVISED BLOOMS TAXONOMY KEYWORDS
L1	List, define, tell, describe, identify, show, label, collect, examine, tabulate, quote, name, who, when, where, etc.
L2	summarize, describe, interpret, contrast, predict, associate, distinguish, estimate, differentiate, discuss, extend
L3	Apply, demonstrate, calculate, complete, illustrate, show, solve, examine, modify, relate, change, classify, experiment, discover.
L4	Analyze, separate, order, explain, connect, classify, arrange, divide, compare, select, explain, infer.
L5	Assess, decide, rank, grade, test, measure, recommend, convince, select, judge, explain, discriminate, support, conclude, compare, summarize.

PROGRAM OUTCOMES (PO), PROGRAM SPECIFIC OUTCOMES (PSO)				CORRELATION LEVELS	
PO1	Engineering knowledge	PO7	Environment and sustainability	0	No Correlation
PO2	Problem analysis	PO8	Ethics	1	Slight/Low
PO3	Design/development of solutions	PO9	Individual and team work	2	Moderate/ Medium
PO4	Conduct investigations of complex problems	PO10	Communication	3	Substantial/ High
PO5	Modern tool usage	PO11	Project management and finance		
PO6	The Engineer and society	PO12	Life-long learning		
PSO1	Implement and maintain enterprise solutions using latest technologies.				
PSO2	Develop and simulate *wired & wireless network protocols* for various network applications using modern tools.				
PSO3	Apply the knowledge of Information technology and software testing to maintain legacy systems.				
PSO4	Apply knowledge of web programming and design to develop *web based applications* using				

	database and other technologies.