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

Internal Assessment Test 1 – Sept. 2017

Sub:	Software Engineering					Sub Code:	16MCA34	Branch:	MCA	
Date:	20/9/2017	Duration:	90 mins	Max Marks:	50	Sem / Sec:	III /A		OBE	

	Answer any FIVE FULL Questions/	MARKS	CO	RBT
1(a)	Why is software engineering important? State two reasons.	[4]	CO3	L1
(b)	Brief the essential attributes of good software.	[6]	CO3	L1
2(a)	Explain about the professional and ethical responsibilities of software engineers.	[4]	CO1	L4
(b)	What is software Prototyping? What are its merits?	[6]	CO2	L1
3(a)	Discuss the principles of agile methods.	[5]	CO2	L2
(b)	In brief explain the extreme programming release cycle.	[5]	CO2	L4
4(a)	Define System Design.	[2]	CO1	L1
(b)	Explain the activities involved in design process.	[8]	CO2	L4
5(a)	Why requirements validation process is required?	[3]	CO4	L1
(b)	Explain the different types of checks that should be carried out in this cycle of requirements validation.	[7]	CO4	L4
6	Explain Boehm's spiral model with a neat diagram.	[10]	CO2	L4
7(a)	Explain types of Requirement.	[4]	CO1	L4
(b)	Discuss about Requirement Elicitation and Analysis	[6]	CO1	L2
8	Give a brief description of software engineering diversity	[10]	CO5	L2

Answer Key- I Internal

Q1. a. Why is software engineering important? State two reasons.
b. Brief the essential attributes of good software.

Ans: Software engineering is an engineering discipline that is concerned with all aspects of software production from the early stages of system specification through to maintaining the system after it has gone into use. *It is important for two reasons:*

- ✧ More and more, individuals and society rely on advanced software systems. We need to be able to produce reliable and trustworthy systems economically and quickly.
- ✧ It is usually cheaper, in the long run, to use software engineering methods and techniques for software systems rather than just write the programs as if it was a personal programming project. For most types of system, the majority of costs are the costs of changing the software after it has gone into use.

Attributes of good Software: Good software should deliver the required functionality and performance to the user and should be maintainable, dependable and usable.

Product characteristic	Description
Maintainability	Software should be written in such a way so that it can evolve to meet the changing needs of customers. This is a critical attribute because software change is an inevitable requirement of a changing business environment.
Dependability and security	Software dependability includes a range of characteristics including reliability, security and safety. Dependable software should not cause physical or economic damage in the event of system failure. Malicious users should not be able to access or damage the system.
Efficiency	Software should not make wasteful use of system resources such as memory and processor cycles. Efficiency therefore includes responsiveness, processing time, memory utilisation, etc.
Acceptability	Software must be acceptable to the type of users for which it is designed. This means that it must be understandable, usable and compatible with other systems that they use.

Q2. a. Explain about the professional and ethical responsibilities of software engineers.
b. What is software Prototyping? What are its merits?

- ✧ Ans: Software engineering involves wider responsibilities than simply the application of technical skills. Software engineers must behave in an honest and ethically responsible way if they are to be respected as professionals. Ethical behaviour is more than simply upholding the law but involves following a set of principles that are morally correct.
- ✧ Confidentiality
  - Engineers should normally respect the confidentiality of their employers or clients irrespective of whether or not a formal confidentiality agreement has been signed.

- Competence
  - Engineers should not misrepresent their level of competence. They should not knowingly accept work which is outwith their competence.
- Intellectual property rights
  - Engineers should be aware of local laws governing the use of intellectual property such as patents, copyright, etc. They should be careful to ensure that the intellectual property of employers and clients is protected.
- Computer misuse
  - Software engineers should not use their technical skills to misuse other people's computers. Computer misuse ranges from relatively trivial (game playing on an employer's machine, say) to extremely serious (dissemination of viruses).

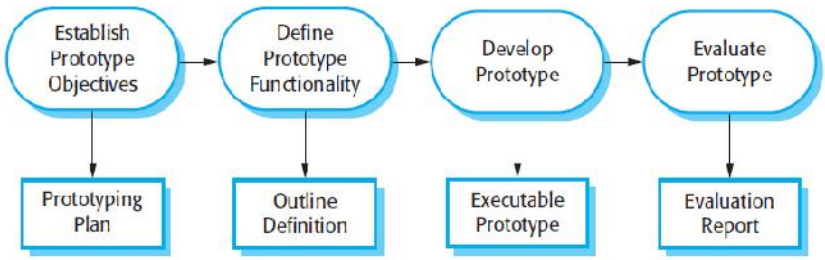
The professional societies in the US have cooperated to produce a code of ethical practice. The Code contains eight Principles related to the behaviour of and decisions made by professional software engineers, including practitioners, educators, managers, supervisors and policy makers, as well as trainees and students of the profession.

- PUBLIC - Software engineers shall act consistently with the public interest.
- 2. CLIENT AND EMPLOYER - Software engineers shall act in a manner that is in the best interests of their client and employer consistent with the public interest.
- 3. PRODUCT - Software engineers shall ensure that their products and related modifications meet the highest professional standards possible.
- 4. JUDGMENT - Software engineers shall maintain integrity and independence in their professional judgment.
- 5. MANAGEMENT - Software engineering managers and leaders shall subscribe to and promote an ethical approach to the management of software development and maintenance.
- 6. PROFESSION - Software engineers shall advance the integrity and reputation of the profession consistent with the public interest.
- 7. COLLEAGUES - Software engineers shall be fair to and supportive of their colleagues.
- 8. SELF - Software engineers shall participate in lifelong learning regarding the practice of their profession and shall promote an ethical approach to the practice of the profession.

Prototyping: A prototype is an initial version of a system used to demonstrate concepts and try out design options.

A prototype can be used in:

- The requirements engineering process to help with requirements elicitation and validation;
- In design processes to explore options and develop a UI design;
- In the testing process to run back-to-back tests.



Benefits of Prototyping:

- Improved system usability.
- A closer match to users' real needs.
- Improved design quality.

- ✧ Improved maintainability.

Reduced development effort

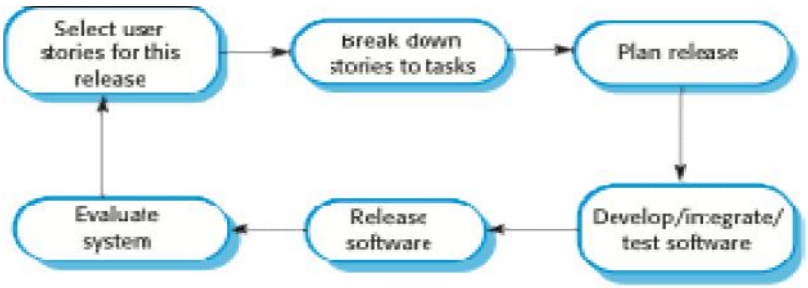
Q3. a. Discuss the principles of agile methods.
b. In brief explain the extreme programming release cycle.

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.

Extreme Programming Life Cycle:

IN XP incremental development is supported through small, frequent system releases. Customer involvement means full-time customer engagement with the team. People not process through pair programming, collective ownership and a process that avoids long working hours. Change supported through regular system releases. Maintaining simplicity through constant refactoring of code.



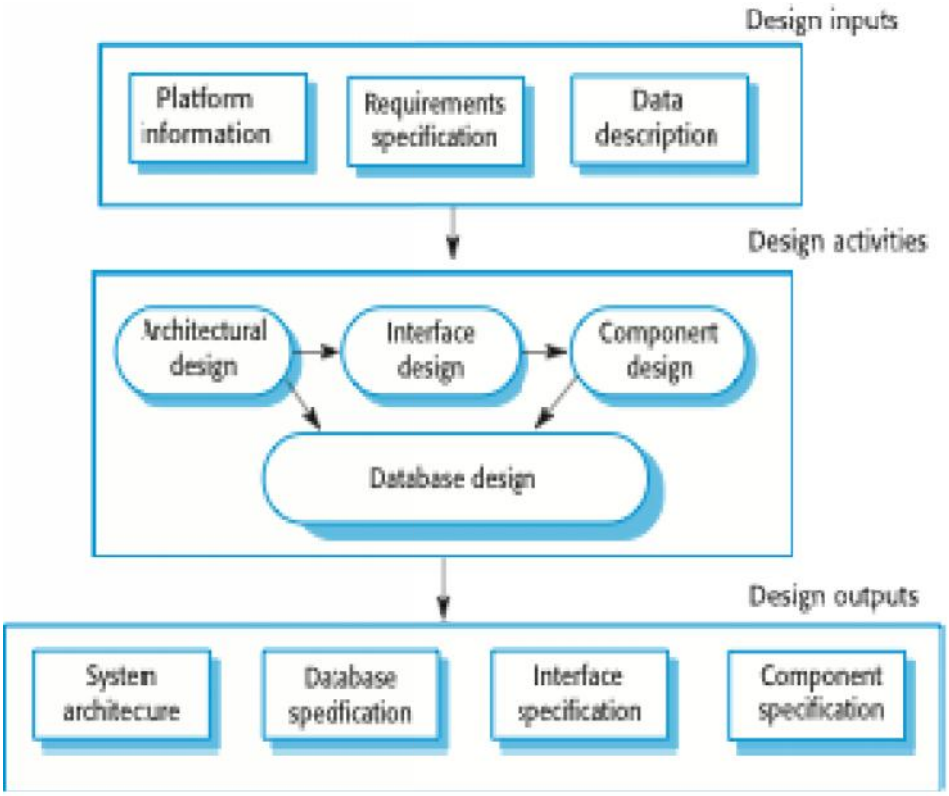
- ✧ In XP, a customer or user is part of the XP team and is responsible for making decisions on requirements.
- ✧ User requirements are expressed as scenarios or user stories.
- ✧ These are written on cards and the development team break them down into implementation tasks. These tasks are the basis of schedule and cost estimates.
- ✧ The customer chooses the stories for inclusion in the next release based on their priorities and the schedule estimates.

Q4. a. Define system design

b. Explain the activities involved in design process.

Software design - Design a software structure that realises the specification;

The process of converting the system specification into an executable system is called system design. This step include design a software structure that realises the specification; The activities of design and implementation are closely related and may be inter-leaved.



Design activities includes following:

- *Architectural design*, where you identify the overall structure of the system, the principal components (sometimes called sub-systems or modules), their relationships and how they are distributed.
- *Interface design*, where you define the interfaces between system components.
- *Component design*, where you take each system component and design how it will operate.
- *Database design*, where you design the system data structures and how these are to be represented in a database.

Q5. a. Why requirements validation process is required?

b. Explain the different types of checks that should be carried out in this cycle of requirements validation.

Requirement validation concerned with demonstrating that the requirements define the system that the customer really wants. Requirements error costs are high so validation is very important fixing a requirements error after delivery may cost up to 100 times the cost of fixing an implementation error.

Requirements checking: Following things should be checked for requirement:

- Validity. Does the system provide the functions which best support the customer's needs?
- Consistency. Are there any requirements conflicts?
- Completeness. Are all functions required by the customer included?

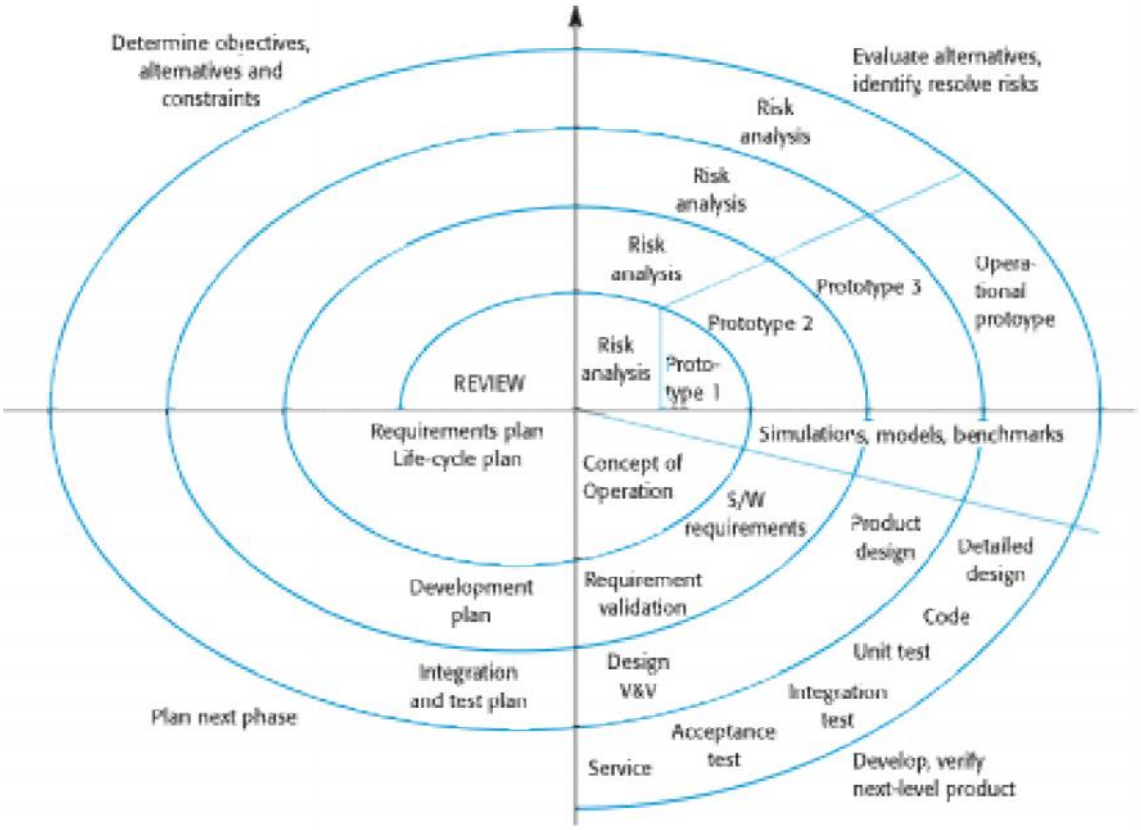
- ✧ Realism. Can the requirements be implemented given available budget and technology
- ✧ Verifiability. Can the requirements be checked?

Requirements validation techniques:

- ✧ Requirements reviews :Systematic manual analysis of the requirements will be done.
- ✧ Prototyping:Using an executable model of the system will check the requirements.
- ✧ Test-case generation: Develop tests for requirements to check testability of the requirements.

Q6: Explain Boehm's spiral model with a neat diagram.

Ans: **Boehm's spiral model**



- ✧ Process is represented as a spiral rather than as a sequence of activities with backtracking.
- ✧ Each loop in the spiral represents a phase in the process.
- ✧ No fixed phases such as specification or design - loops in the spiral are chosen depending on what is required.
- ✧ Risks are explicitly assessed and resolved throughout the process.

Spiral model sectors:

- ✧ Objective setting
  - ✧ Specific objectives for the phase are identified.
- ✧ Risk assessment and reduction
  - ✧ Risks are assessed and activities put in place to reduce the key risks.
- ✧ Development and validation
  - ✧ A development model for the system is chosen which can be any of the generic models.
- ✧ Planning
  - ✧ The project is reviewed and the next phase of the spiral is planned.

Spiral model usage

- ✧ Spiral model has been very influential in helping people think about iteration in software processes and introducing the risk-driven approach to development.
- ✧ In practice, however, the model is rarely used as published for practical software development.

Q7:a. Explain types of Requirement.

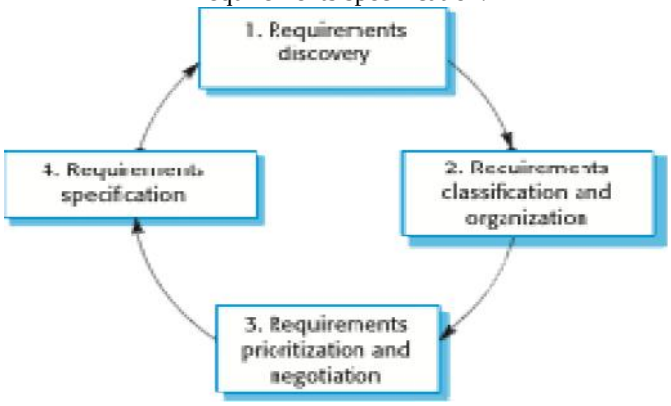
b. Discuss about Requirement Elicitation and Analysis.

Types of requirement

- ✧ User requirements
  - ▪ Statements in natural language plus diagrams of the services the system provides and its operational constraints. Written for customers.
- ✧ System requirements
  - ▪ A structured document setting out detailed descriptions of the system's functions, services and operational constraints. Defines what should be implemented so may be part of a contract between client and contractor.

Requirements elicitation and analysis

- ✧ Sometimes called requirements elicitation or requirements discovery.
- ✧ Involves technical staff working with customers to find out about the application domain, the services that the system should provide and the system's operational constraints.
- ✧ May involve end-users, managers, engineers involved in maintenance, domain experts, trade unions, etc. These are called *stakeholders.*
- ✧ Software engineers work with a range of system stakeholders to find out about the application domain, the services that the system should provide, the required system performance, hardware constraints, other systems, etc.
- ✧ Stages include:
  - ▪ Requirements discovery,
  - ▪ Requirements classification and organization,
  - ▪ Requirements prioritization and negotiation,
  - ▪ Requirements specification.



- ✧ Requirements discovery
  - ▪ Interacting with stakeholders to discover their requirements. Domain requirements are also discovered at this stage.
- ✧ Requirements classification and organisation
  - ▪ Groups related requirements and organises them into coherent clusters.
- ✧ Prioritisation and negotiation
  - ▪ Prioritising requirements and resolving requirements conflicts.
- ✧ Requirements specification
  - ▪ Requirements are documented and input into the next round of the spiral.

Q8: Give a brief description of software engineering diversity

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. How this systematic approach is actually implemented varies dramatically depending on the organization developing the software, the type of software, and the people involved in the development process. There are no universal software engineering methods and techniques that are suitable for all systems and all companies. Rather, a diverse set of software engineering methods and tools has evolved over the past 50 years.

There are many different types of application including:

1.Stand-alone applications - These are application systems that run on a local computer, such as a PC. They include all necessary functionality and do not need to be connected to a network. Examples office applications on a PC, CAD programs, photo manipulation software, etc.

2. 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. Obviously, these include web applications such as e-commerce applications where you can interact with a remote system to buy goods and services.

3. Embedded control systems These are software control systems that control and manage hardware devices. Numerically, there are probably more embedded systems than any other type of system. Examples software in a mobile (cell) phone, software that controls anti-lock braking in a car, and software in a microwave oven to control the cooking process.

4. Batch processing systems These are business systems that are designed to process data in large batches. They process large numbers of individual inputs to create corresponding outputs. Examples of batch systems include periodic billing systems, such as phone billing systems, and salary payment systems.

5. Entertainment systems These are systems that are primarily for personal use and which are intended to entertain the user. Most of these systems are games of one kind or another. The quality of the user interaction offered is the most important distinguishing characteristic of entertainment systems.

6. 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. These are often computationally intensive and require high-performance parallel systems for execution.

7. 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. The software has to interact with sensors and often is installed in a hostile environment such as inside an engine or in a remote location.

8. Systems of systems These are systems that are composed of a number of other software systems. Some of these may be generic software products, such as a spreadsheet program. Other systems in the assembly may be specially written for that environment.