UnitI
PartA(MultipleChoiceQuestions)
Remembering

1. IEEE defines ___________as the collection of computer programs, procedures, rules and associated documentation and data. Identify the same
A. SOFTWARE Engineering
B. **software**
C. software model
D. SRS

2. IEEE defines ___________is a systematic approach to the development, operation, maintenance and requirement of the software. Identify the same
A. **SOFTWARE Engineering**
B. requirement specification
C. coding
D. SRS

3. Quote from memory that the fundamental goal of software engineering is to produce _______
A. SRS
B. Review report
C. Cost effective design
D. <u>**High quality software product**</u>

4. Quote from the memory that ___________is the effort required to couple system with other system
A. Maintainability
B. <u>**Inter-operability**</u>
C. Portability
D. Quality

5. Quote from memory that ___________is the effort required to transfer the software from one hardware configuration to other.
A. Maintainability
B. Inter-operability
C. <u>**Portability**</u>
D. Reliability

6. State that ___________is the effort required to locate and fix the errors in the program.
A. <u>**Maintainability**</u>
B. Inter-operability
C. Portability
D. Testing

7. State that ________is one of the quality attributes in product revision

A. Portability
B. Efficiency
C. Integrity
D. Testability

8._____is a risk driven model. Identify
A. Waterfall
B. **Spiral**
C. Iterative enhancement
D. Prototype

9. Blocking states is found in ________model. Recognize.
A. W**aterfall**
B. Spiral
C. Iterative enhancement
D. Prototype

10. Freezing of requirement before development process is a limitation in ________, identify
A. **Waterfall**
B. Spiral
C. Iterative enhancement
D. Prototype

Understanding

11. Throw away models are used in _______model of development
A. Waterfall
B. Spiral
C. Iterative enhancement
D. Prototype

12. In which model , software is built in increments
A. Waterfall
B. Spiral
C. Iterative enhancement
D. Prototype

13. Evaluating a software process and identifying the loop holes increases _______of a software
A. Predictability
B. Scalability
C. Cost
D. Quality

14. Development process of software has _____phases
A. Three

B. Four
C. Five
D. Two

15. Software process components are development process, management process and ________process
A. Monitoring
B. SCM
C. Termination analysis
D. Quality analysis

16. Planning, monitoring and control and termination analysis are part of_____
A. Development process
B. Quality analysis
C. SCM
D. Management process

17.____provide quantifiable measures used to measure different characteristics of software product.
A. **Software Metric**
B. Software Model
C. Measurement
D. Mass

18. Expand SCM
A. Software Configuration Model
B. Software Code Management
C. **Software Configuration Management**
D. Software Code Maintenance

19.____is independent of development process
A. design
B. management
C. **SCM**
D. testing

20. Terms like Change Request, Status accounting appear in______
A. management process
B. development process
C. quality analysis process
D. SCM process

21. Baseline in SCM process has set of _____
A. work products
B. Change Request (CR)
C. **software configuration Item**

D. Change Control Procedures

22. CMM, QIP, GQM are part of___
A. SCM process
B. <u>Process improvement and maturity model</u>
C. status accounting
D. estimation models

23. QIP has __basic steps.
A. four
B. five
C. **<u>six</u>**
D. Three

24. Initial, repeatable, defined, managed and optimizing are levels of____model
A. QIP
B. GQM
C. **<u>CMM</u>**
D. SCI life cycle

25. In SCI life cycle, CR request is approved by_____
a) **<u>Configuration Manager</u>**
b) CCB
c) Project manager
d) developer

Part B (4 marks) Remember

1. **Define Software Engineering. Explain various problem faced in Software Engineering**

<u>Software Engineering is a systematic approach to the development, operation, maintenance and retirement of the software.</u> There is another definition for s/w engineering, which states that “Software engineering is an application of science and mathematics by which the capabilities of computer equipments are made useful to man via computer programs, procedures and associated documentation”.

Problem of Scale

A fundamental problem of software engineering is the problem of scale. Development of a very large system requires very different set of methods compared to developing a small system. In other words, the methods that are used for developing small systems generally do not scale up to large systems. For example: consider the problem of counting people in a room versus taking the census of a country. Both are counting problems but the methods used are totally different. A different set of methods have to be used for developing large software. Any large project involves the use of technology and project management. In small projects, informal methods for development and management can be used. However, for large projects both have to be much more formal. When dealing with small software project, the technology and project management requirement is low. However, when the scale changes to the larger systems, we have to follow formal methods. For example: if we have 50 bright programmers without formal management and development procedures and ask them to develop a large project, they will produce anything of no use.

Cost, Schedule and Quality

The cost of developing a system is the cost of resources used for the system, which in the case of software are, the manpower, hardware, software and other support resources. The manpower component is predominant as the software development is highly labor-intensive.

Schedule is an important factor in many projects. For some business systems, it is required to build a software with small cycle of time. The developing methods that produce high quality software is another fundamental goal of software engineering. We can view the quality of a software product having three dimensions: Product Operation, Product Transition and Product Revision.

The Product operation deals with the quality factors such as correctness reliability and efficiency. Product transition deals with quality factors such as portability, interoperability. Product revision deals with aspects related to modification of programs, including factors like maintainability and testability.

The Problem of Consistency

For an organization there is another goal i.e. consistency. An organization involved in software development does not just want low cost and high quality for a project but it wants these consistently. Consistency of performance is an important factor for any organization; it allows an organization to predict the outcome of the project with reasonable accuracy and to improve its processes to produce higher-quality products. To achieve consistency, some standardized procedures must be followed.

2. **Discover the Quality attributes of Software Engineering**

The Product operation deals with the quality factors such as correctness reliability and efficiency Product transition deals with quality factors such as portability, interoperability. Product revision deals with aspects related to modification of programs, including factors like maintainability and testability.

Correctness is the extent to which a program satisfies its specifications. **Reliability** is the property that defines how well the software meets its requirements. **Efficiency** is the factor in all issues rating to the execution of the software. It includes considerations such as response time, memory requirements and throughput. **Usability** is the effort required to learn and operate the software properly.

Maintainability is the effort required to locate and fix errors in the programs. **Testability** is the effort required to test and check that symbol or module performs correct operation or not. **Flexibility** is the effort required to modify an operational program (functionality).

Portability is the effort required to transfer the software from one hardware configuration to another. **Reusability** is the extent to which parts of software can be used in other related applications. **Inter-operability** is the effort required to couple the system with other systems.

3. **Discover different phases of development process**

Phased Development Process

A development process consists of various phases, each phase ending with a predefined output. Software engineering must consist of these activities:

- Requirement specification for understanding and clearly stating the problem.
- Design for deciding a plan for the solution.
- Coding for implementing the planned solution.
- Testing for verifying the programs.

Requirement Analysis

Requirement analysis is done in order to understand the problem to be solved. In this phase, collect the requirement needed for the software project.

The goal of software requirement specification phase is to produce the **software requirement specification document**. The person who is responsible for requirement analysis is called as **analyst.** In problem analysis, the analyst has to understand the problem. Such analysis requires a thorough understanding of the existing system. This requires interaction with the client and end- users as well as studying the existing manuals and procedures. Once the problem is analyzed, the requirements must be specified in the requirement specification document.

Software Design

The purpose of design phase is to plan a solution for the problem specified by the requirement document. The output of this phase is the design document which is the blue-pint or plan for the solution and used later during implementation, testing and maintenance.

Design activity is divided into two phases- **System design** and **detailed design.** System design aims to identify the module that should be included in the system. During detailed design, the internal logic of each of the modules specified during system design is decided.

Coding

The goal of coding is to translate the design into code in a given programming language. The aim is to implement the design in the best possible manner. Testing and maintenance costs are much higher than the coding cost, therefore, the goal of should be to reduce testing and maintenance efforts. Hence the programs should be easy to read and understand.

Testing

After coding phase computer programs are available, which can be executed for testing purpose. Testing not only has to uncover errors introduced during coding, but also errors introduced during previous phases.

The starting point of testing is **unit testing.** Here each module is tested separately. After this, the module is integrated to form sub-systems and then to form the entire system. During integration of modules, **integration testing** is done to detect design errors. After the system is put together, **system testing** is performed. Here, the system is tested against the requirements to see whether all the requirements are met or not. Finally, **acceptance testing** is performed by giving user's real-world data to demonstrate to the user.

4. **State and explain the working of waterfall model with the help of a diagram.**

Waterfall Model

Waterfall model is the simplest model which states that the phases are organized in a linear order. In this model, a project begins with feasibility analysis. On successfully demonstrating the

feasibility of a project, the requirement analysis and project planning begins. The design starts after the requirement analysis is complete and the coding begins after the design is complete,

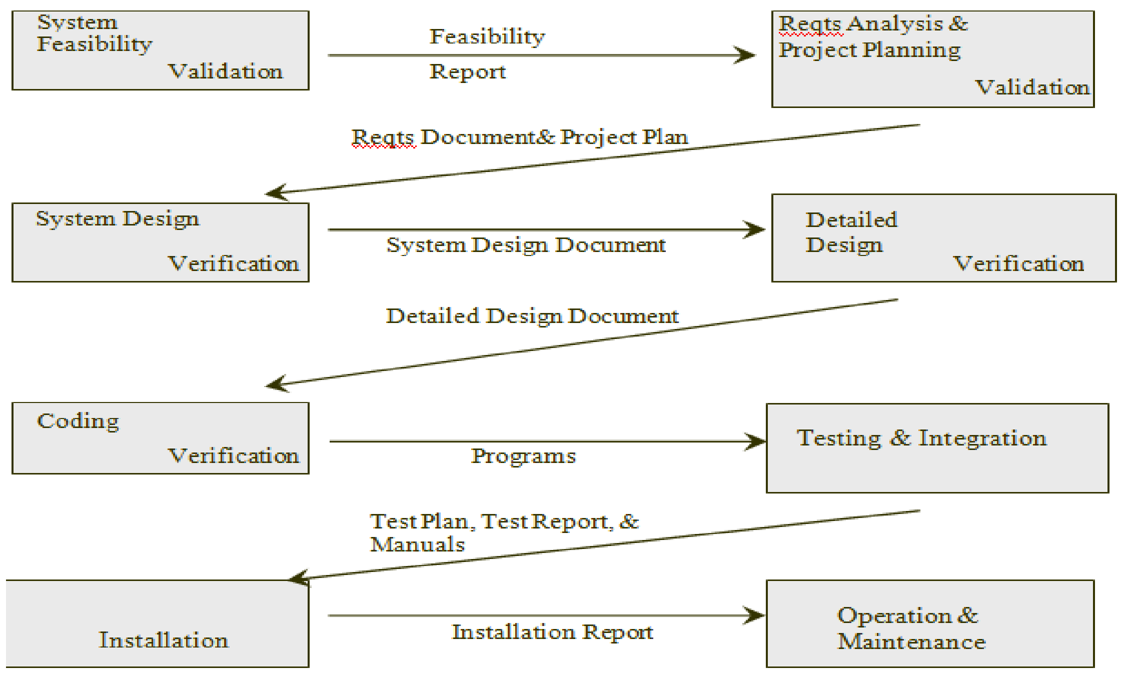
once the programming is complete, the code is integrated and testing is done. On successful completion of testing, the system is installed. After this, the regular operations and maintenance take place as shown in the figure (next page).

Each phase begins soon after the completion of the previous phase. Verification and validation activities are to be conducted to ensure that the output of a phase is consistent with the overall requirements of the system. At the end of every phase there will be an output. Outputs of earlier phases can be called as work products and they are in the form of documents like requirement document and design document. The output of the project is not just the final program along with the user manuals but also the requirement document, design document, project plan, test plan and test results.

Project Outputs of the Waterfall Model

- Requirement document
- Project plan
- System design document
- Detailed design document
- Test plan and test report
- Final code
- Software manuals
- Review report.

Reviews are formal meetings to uncover deficiencies in a product. The review reports are the outcomes of these reviews.



Waterfall Model

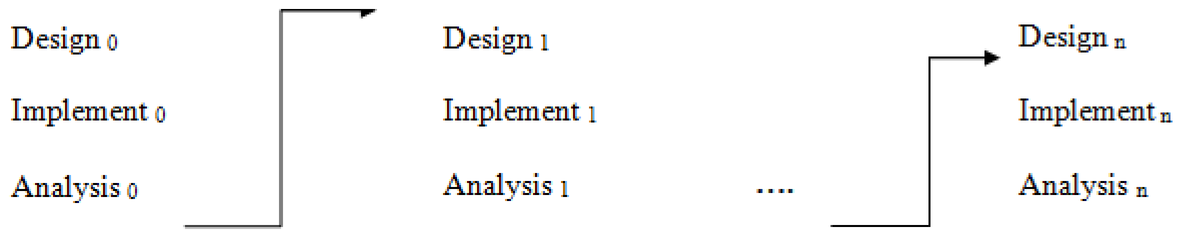
5. **List out the limitation of water fall model**

1. Waterfall model assumes that requirements of a system can be frozen before the design begins. It is difficult to state all the requirements before starting a project.
2. Freezing the requirements usually requires choosing the hardware. A large project might take a few years to complete. If the hardware stated is selected early then due to the speed at which the hardware technology is changing, it will be very difficult to accommodate the technological changes.
3. Waterfall model stipulates that the requirements be completely specified before the rest of the development can proceed. In some situations, it might be desirable to produce a part of the system and then later enhance the system. This can't be done if waterfall model is used.
4. It is a document driven model which requires formal documents at the end of each phase. This approach is not suitable for interactive applications.
5. In an interesting analysis it is found that, the linear nature of the life cycle leads to "blocking states" in which some project team members have to wait for other team members to

complete the dependent task. The time spent in waiting can exceed the time spent in productive work.
6. Client gets a feel about the software only at the end.

Understand

6. **Explain the working of iterative enhancement model**



Iterative Enhancement Model

This model tries to combine the benefits of both prototyping and waterfall model. The basic idea is, software should be developed in increments, and each increment adds some functional capability to the system. This process is continued until the full system is implemented. An advantage of this approach is that, it results in better testing because testing each increment is likely to be easier than testing the entire system.As prototyping ,the increments provide feedback from the client, which will be useful for implementing the final system. It will be helpful for the client to state the final requirements.

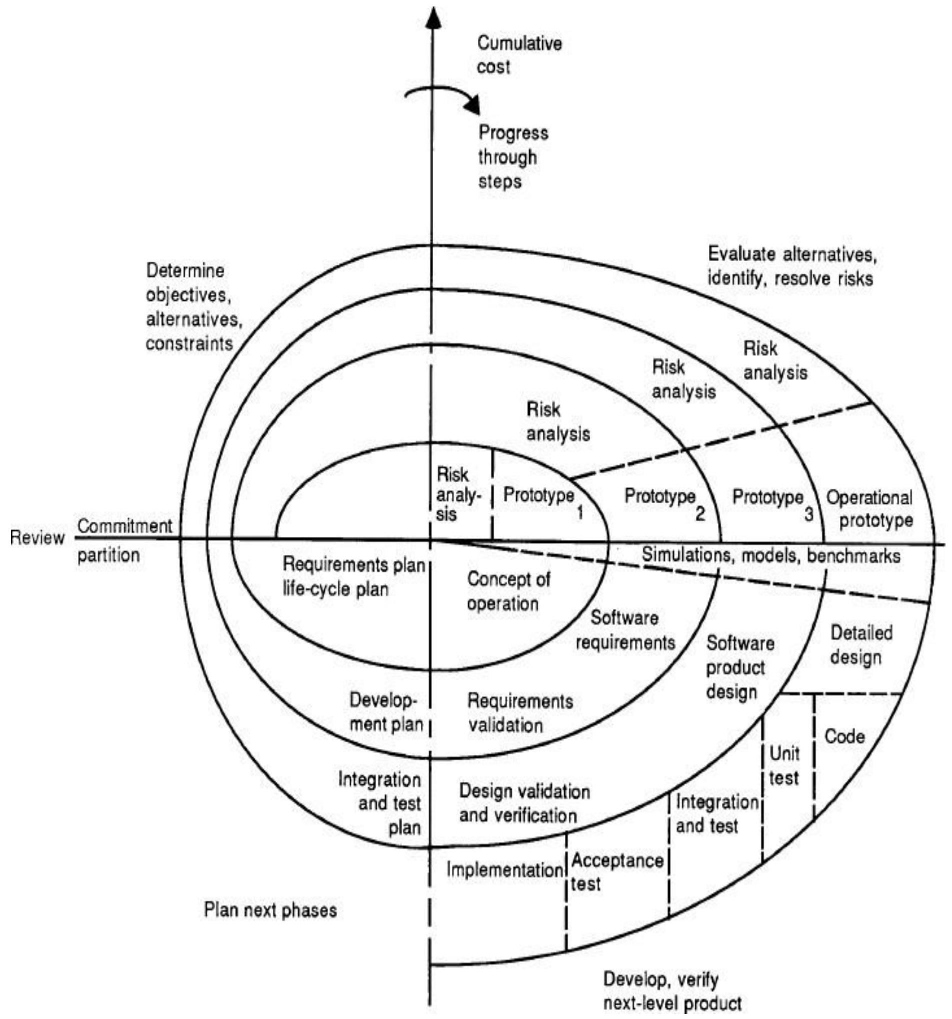
Here a project control list is created. It contains all tasks to be performed to obtain the final implementation and the order in which each task is to be carried out. Each step consists of removing the next task from the list, designing, coding, testing and implementation and the analysis of the partial system obtained after the step and updating the list after analysis. These three phases are called design phase, implementation phase and analysis phase. The process is iterated until the project control list becomes empty. At this moment, the final implementation of the system will be available.

The first version contains some capability. Based on the feedback from the users and experience with the current version, a list of additional features is generated. And then more features are added to the next versions. This type of process model will be helpful only when the system development can be broken down into stages.

7. **Describe the spiral model with the help of diagram**

As the name suggests, the activities of this model can be organized like a spiral that has many cycles as shown in the above figure. Each cycle in the spiral begins with the identification of objectives for that cycle; the different alternatives that are possible for achieving the objectives and the constraints that exist. This is the first quadrant of the cycle. The next step is to evaluate different alternatives based on the objectives and constraints. The focus is based on the risks. Risks reflect the chances that some of the objectives of the project may not be met. Next step is to develop strategies that resolve the uncertainties and risks. This step may involve activities like prototyping.

The risk-driven nature of the spiral model allows it to suit for any applications. The important feature of spiral model is that, each cycle of spiral is completed by a review that covers all the products developed during that cycle; including plans for the next cycle. In a typical application of spiral model, one might start with an extra round-zero, in which the feasibility of the basic project objectives is studied. In round-one a concept of operation might be developed. The risks are typically whether or not the goals can be met within the constraints. In round-2, the top-evel requirements are developed. In succeeding rounds the actual development may be done. In a project, where risks are high, this model is preferable.



8. **Describe the SCM life cycle of an item**

Configuration Identification:

When a change is done, itshould be clear,*to what*,the change has been applied. This requires a **baseline** to be established. A baseline forms a reference point in the development of a system and is generally defined after the major phases in the development process. A softwar baseline represents the software in a most recent state.Some baselines are requirement baseline, design baseline and the product baseline or system baseline.

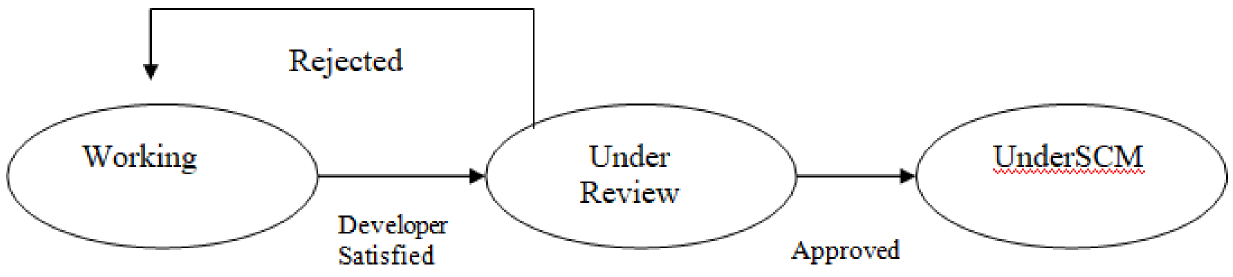
Though the goal of SCM is to control the establishment and changes to these baselines, treating each base line as a single entity for the change is undesirable, because the change may be limited to a very small portion of the baseline. For this reason, a baseline can consist of many software configuration items. [SCI's] A baseline is a set of SCIs and their relations.

Because a baseline consists of SCIs and SCI is the basic unit for change control, the SCM process starts with identification of the configuration items. Once the SCI is identified, it is given a name and becomes the unit of change control.

Change Control:

Once the SCIs are identified, and their dependencies are understood, the change control procedures of SCM can be applied.The decisions regarding the change are generally taken by the configuration control board [CCB] headed by configuration manager [CM]

When a SCI is under development, it has considered being in working state. It is not under SCM and can be changed freely.Once the developer is satisfied with the SCI,then it is given to CM for review and the item enters to 'under review' state. The CM reviews the SCI and if it is approved, it enters into a library after which the item is formally under SCM. If the item is not approved, the item is given back to the developer. This cycle of a SCI is given in the figure below.



Once the SCI is in the library, it can not be modified, even without the permission of the CM. an SCI under SCM can be changed only if the change has been approved by the CM. A change is initiated by a change request (CR). The reason for change can be anything. The CM evaluates the CR primarily by considering the effect of change on the cost schedule and quality of the project and the benefits likely to come due to this change. Once the CR is accepted, project manager will take over the plan and then CR is implemented by the programmer.

Status Accounting and Auditing

The aim of status accounting is to answer the question like what is the status of the CR (approved/rejected), what is the average and effort for fixing a CR and what is the number of CR. For status accounting, the main source of information is CR. Auditing has a different role.

9. **Explain various activities of Software configuration Management Process**

Same as above