SOFTWARE ENGINEERING Final NOTES

What is Requirements engineering?
Ans: Requirement's engineering refers to the process of defining, documenting, and maintaining requirements in the engineering design process. Requirement engineering provides the appropriate mechanism to understand what the customer desires, analyzing the need, and assessing feasibility, negotiating a reasonable solution, specifying the solution clearly, validating the specifications and managing the requirements as they are transformed into a working system. Thus, requirement engineering is the disciplined application of proven principles, methods, tools, and notation to describe a proposed system's intended behavior and its associated constraints.

Types Of Requirements:

User Requirements:

- User requirements are statements, in a natural language plus diagrams, of what services the system is expected to provide and the constraints under which it must operate.

System Requirements:

- System requirements set out the system's functions, services and operational constraints in detail. The system requirements document (sometimes called a functional specification) should be precise. It should define exactly what is to be implemented. It may be part of the contract between the system buyer and the software developers.

    - Functional requirements
    - Non-functional requirements
    - Domain requirements

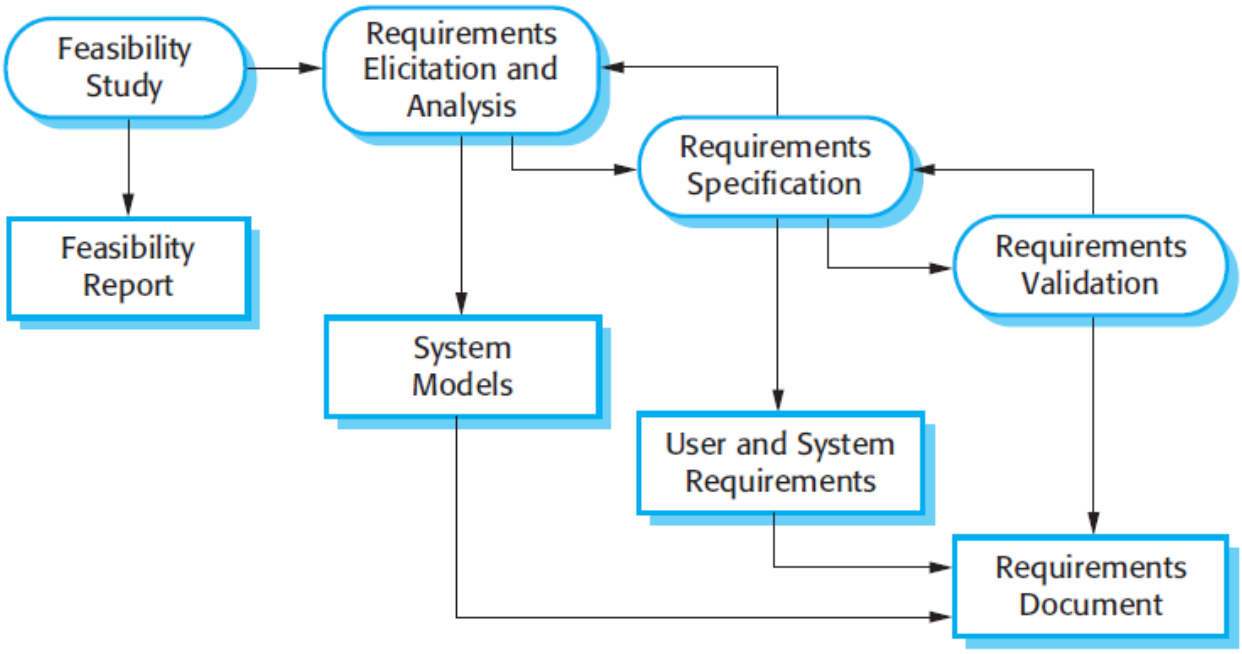
REQUIREMENTS ENGINEERING PHASES

- **Inception:** Inception is a task where the requirement engineering asks a set of questions to establish a software process.
- **Elicitation**: Elicitation means to find the requirements from anybody.
  The requirements are difficult because the following problems occur in elicitation.
    - **Problem of scope:** The customer gives the unnecessary technical detail rather than clarity of the overall system objective.
    - **Problem of understanding:** Poor understanding between the customer and the developer regarding various aspect of the project like capability, limitation of the computing environment.
    - **Problem of volatility:** In this problem, the requirements change from time to time and it is difficult while developing the project.
- **Analysis and Elaboration:** The information taken from user during inception and elaboration and are expanded and refined in elaboration.

Its main task is developing pure model of software using functions, feature and constraints of a software.

- **Negotiation:** In negotiation task, a software engineer decides the how will the project be achieved with limited business resources. | The project cost and delivery time.
- **Specification**: In this task, the requirement engineer constructs a final work product. The work product is in the form of software requirement specification. In this task, formalize the requirement of the proposed software such as informative, functional and behavioral. The requirement is formalized in both graphical and textual formats. IT WILL BE A SRS DOCUMENT.
- **Validation:** The work product is built as an output of the requirement engineering and that is accessed for the quality through a validation step.
- **Requirements Management:** It is a set of activities that help the project team to identify, control and track the requirements and changes can be made to the requirements at any time of the ongoing project.

Requirement Engineering Process:



What is Software design?

Ans: Software design is a process to transform user requirements into some suitable form, which helps the programmer in software coding and implementation.

Software Design Levels

Software design yields three levels of results:

- **Architectural Design** - The architectural design is the highest abstract version of the system. It identifies the software as a system with many components interacting with each other. At this level, the designers get the idea of proposed solution domain.
- **High-level Design-** The high-level design breaks the 'single entity-multiple component' concept of architectural design into less-abstracted view of sub-systems and modules and depicts their interaction with each other. High-level design focuses on how the system along with all of its components can be implemented in forms of modules. It recognizes modular structure of each sub-system and their relation and interaction among each other.
- **Detailed Design-** Detailed design deals with the implementation part of what is seen as a system and its sub-systems in the previous two designs. It is more detailed towards modules and their implementations. It defines logical structure of each module and their interfaces to communicate with other modules.

What is Modularity?

Ans: Modularity is an attribute of software that allows a program to be intellectually manageable into distinct logical parts.

What is Modularization?

Ans: Modularization is a technique to divide a software system into multiple discrete and independent modules, which are expected to be capable of carrying out task(s) independently.

FUNCTIONAL INDEPENDENCE

- **Cohesion** is an indication of the relative functional strength of a module. A cohesive module performs a single task, requiring little interaction with other components in other parts of a program. Stated simply, a cohesive module should (ideally) do just one thing.

- **Coupling** is an indication of the relative interdependence among modules. Coupling depends on the interface complexity between modules, the point at which entry or reference is made to a module, and what data pass across the interface.

- **Aspect** is a representation of a cross-cutting concern. Consider two requirements, A and B. Requirement A crosscuts requirement B "if a software decomposition [refinement] has been chosen in which B cannot be satisfied without taking A into account.

- **Refactoring** is the process of changing a software system in such a way that it does not alter the external behavior of the code [design] yet improves its internal structure (sort algorithm)

What is Software testing?

Ans: Software testing is a process of identifying the correctness of software by considering its all attributes (Reliability, Scalability, Portability, Re-usability, Usability) and evaluating the execution of software components to find the software bugs or errors or defects.

Testing Shows

- ▪ Error
- ▪ Requirements Conformance
- ▪ Performance
- ▪ An indication of quality

What is Validation?

Ans: Validation refers to a different set of tasks that ensure that the software that has been built is traceable to customer requirements.

What is Verification?

Ans: Verification refers to the set of tasks that ensure that software correctly implements a specific function/process

What is UNIT TESTING?

Ans: UNIT TESTING is a type of software testing where individual units or components of a software are tested. The purpose is to validate that each unit of the software code performs as expected.

What is Stub?

Ans: it is the behavior of the lower-level modules that are under development and not yet integrated to other modules.

What is System integration testing?

Ans: System integration testing (SIT) is a systematic technique for assembling a software system while

conducting tests to uncover errors associated with interfacing the modules.

What is the “big bang” approach?

Ans: Big Bang Integration Testing is an integration testing strategy where all

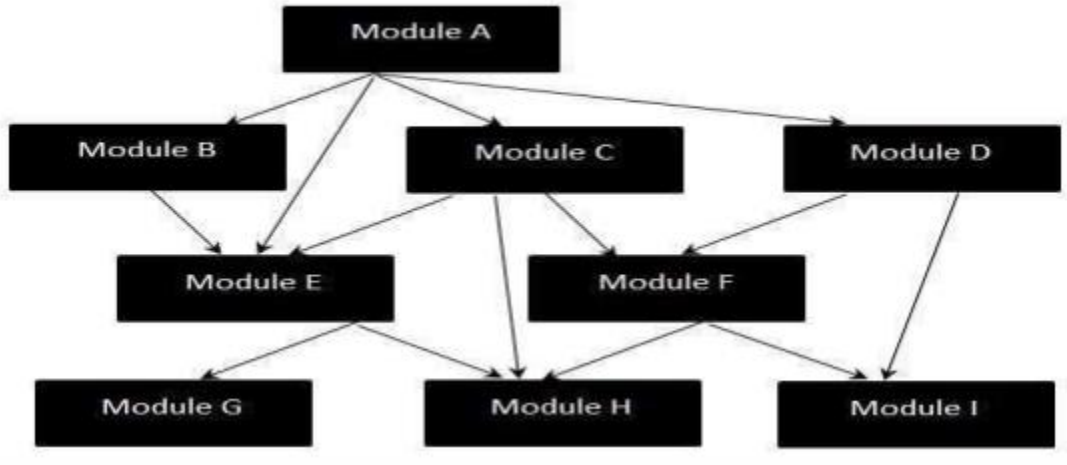
units are linked at once, resulting in a complete system.

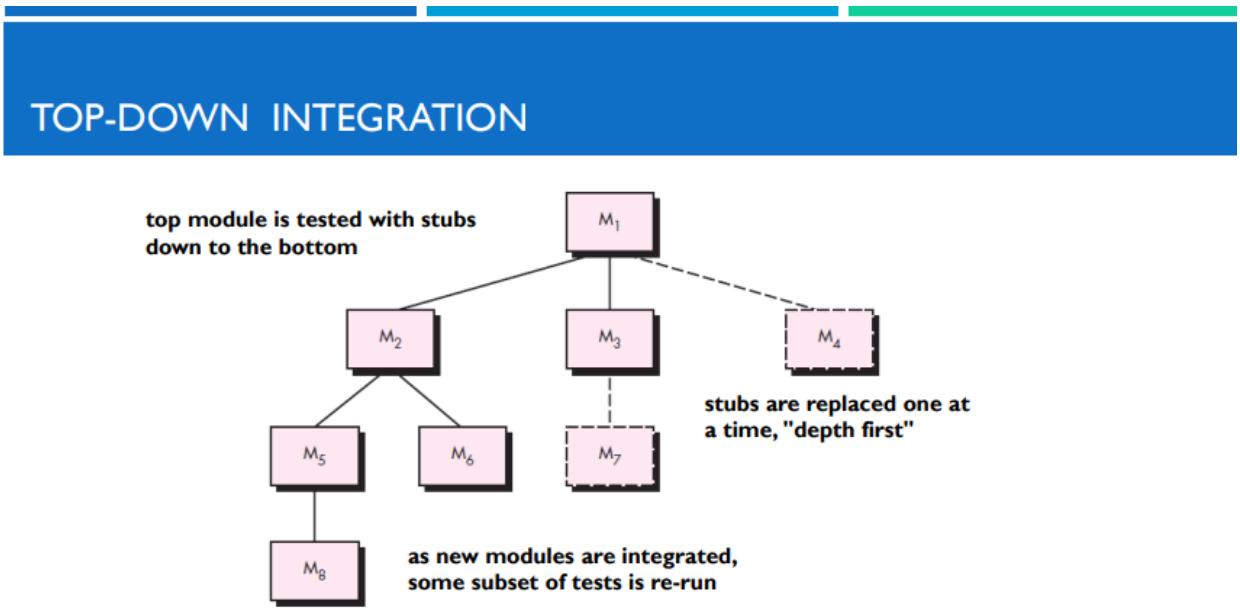
What is black box testing?

Ans: Black Box Testing is a software testing method in which the functionalities of software applications are tested without having knowledge of internal code structure

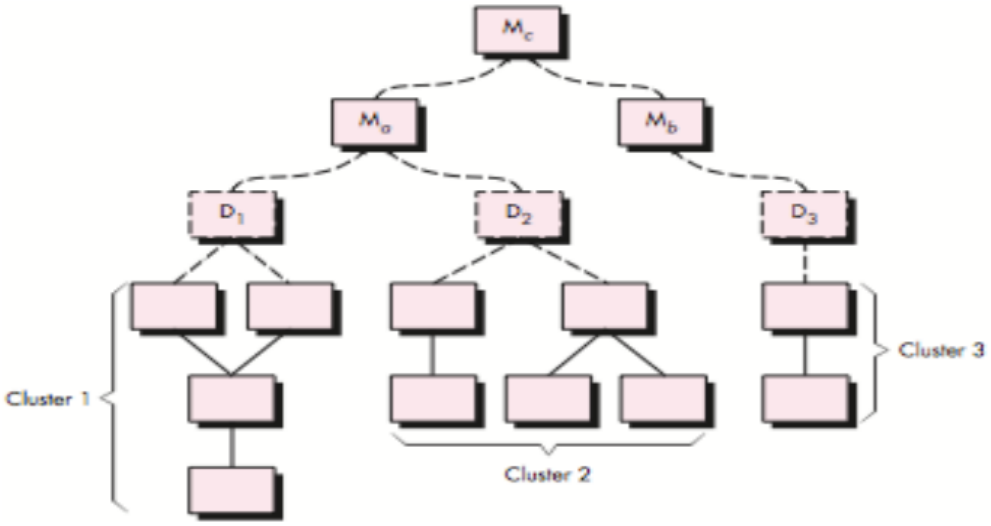
What is White box testing?

Ans: When the tester knows the code behind functionality and uses that knowledge for testing purposes, it is called white box testing. Unit test is considered as white box testing.





BOTTOM-UP INTEGRATION



What is Software Quality Attributes?

Ans: Software Quality Attributes are features that facilitate the measurement of performance of a software product by Software Testing.

- **Availability:** This attribute is indicative as to whether an application will execute the tasks it is assigned to perform. Availability also includes certain concepts that relate to software security, performance, integrity, reliability, dependability, and confidentiality.
- **Performance:** Performance requirements define how well or how rapidly the system must perform specific functions.
- **EFFICIENCY:** Efficiency is a measure of how well the system utilizes processor capacity, disk space, memory, or communication bandwidth
- **FLEXIBILITY:** Flexibility measures how easy it is to add new capabilities to the product
- **INTEGRITY:** which encompasses security, deals with blocking unauthorized access to system functions, preventing information loss, ensuring that the software is protected from virus infection, and protecting the privacy and safety of data entered into the system. Integrity is a major issue with Internet software.
- **INTEROPERABILITY:** Interoperability indicates how easily the system can exchange data or services with other systems
- **RELIABILITY:** The probability of the software executing without failure for a specific period of time is known as reliability.
- **ROBUSTNESS:** Robustness is the degree to which a system continues to function properly when confronted with invalid inputs, defects in connected software or hardware components, or unexpected operating conditions

- **USABILITY:** Usability measures the effort required to prepare input for, operate, and interpret the output of the product
- **MAINTAINABILITY:** Maintainability indicates how easy it is to correct a defect or modify the software
- **REUSABILITY:** Reusability indicates the relative effort involved to convert a software component for use in other applications
- **TESTABILITY:** Testability refers to the ease with which software components or the integrated product can be tested to look for defects

What is Software product metrics.?

Ans Software product metrics are measures of software products such as source code and design documents.

- Size: size is defined in terms of volume, length, and functionality
- Complexity: how classes of an OO design are interrelated to one another
- Coupling: the physical connections between elements of the OO design
- Cohesion: the degree to which all operations working together to achieve a single, well-defined purpose
- Sufficiency: the degree to which an abstraction possesses the features required of it, or the degree to which a design component possesses features in its abstraction, from the point of view of the current application. (e.g. deals with interface and hide internals to the users)
- Completeness: an indirect implication about the degree to which the abstraction or design component can be reused
- Primitiveness: applied to both operations and classes, the degree to which an operation is atomic
- Similarity: the degree to which two or more classes are similar in terms of their structure, function, behavior, or purpose
- Volatility: measures the likelihood that a change will occur

What is Software Configuration Management (SCM).?

Ans: Software Configuration Management (SCM) is a process to systematically manage, organize, and control the changes in the documents, codes, and other entities during the Software Development Life Cycle.

What is SOFTWARE PROJECT ESTIMATION?

Ans: project estimation is a complex process that revolved around predicting the time, cost, and scope that a project requires to be deemed finished. But in terms of software development or software engineering, it also takes the experience of the software development company, the technique they have to utilize, the process they need to follow in order to finish the project (Software Development Life Cycle). Project Estimation requires the use of complex tools & good mathematical as well as knowledge about planning

What is a project schedule?

Ans: A project schedule is a timetable that shows the start and end date of all project tasks, how the tasks relate to each other and usually which team members or other resources are responsible for delivery.

What is a Risk Management?

Ans: Risk Management is the system of identifying addressing and eliminating these problems before they can damage the project.

There are three main classifications of risks which can affect a software project:

- **Project risks**
- **Technical risks**
- **Business risks**