**2 MARKS**

**1.Write the importance of software engineering?**

Software engineering is important because **specific software is needed in almost every industry, in every business, and for every function**. It becomes more important as time goes on – if something breaks within your application portfolio, a quick, efficient, and effective fix needs to happen as soon as possible.

**2.What is the prime objective of software engineering?**

The main purpose of software engineering is **to deal with software applications' design, development, testing, and maintenance**.

**3.Define Software Crisis?**

**The cost of owning and maintaining software was as expensive as developing the software**. **At that time** Projects were running over-time. At that time Software was very inefficient. The quality of the software was low quality.

**4.What is SRS?**

A software requirements specification (SRS) is **a document that describes what the software will do and how it will be expected to perform**. It also describes the functionality the product needs to fulfill all stakeholders (business, users) needs.

**5.Define pseudo code?**

Pseudocode is **an artificial and informal language that helps programmers develop algorithms**. Pseudocode is a "text-based" detail (algorithmic) design tool. The rules of Pseudocode are reasonably straightforward. All statements showing "dependency" are to be indented.

**6.What is modularization?**

Modularization is **a practice of organizing a codebase into loosely coupled and self contained parts**. Each part is a module. Each module is independent and serves a clear purpose.

**7.What is structural testing?**

Structural testing is a type of software testing which **uses the internal design of the software for testing** or in other words the software testing which is performed by the team which knows the development phase of the software, is known as structural testing.

**8.Define software testing?**

Software testing is **the process of evaluating and verifying that a software product or application does what it is supposed to do**. The benefits of testing include preventing bugs, reducing development costs and improving performance.

**9.Define SRS document?**

What Is a Software Requirements Specification (SRS) Document? A software requirements specification (SRS) is **a document that describes what the software will do and how it will be expected to perform**. It also describes the functionality the product needs to fulfill all stakeholders (business, users) needs.

**10.What do you mean by Halstead software science?**

**Halstead's Software Science:**

**Halstead's complexity measurement was developed to measure a program module's complexity directly from source code, with emphasis on computational complexity**. The Halstead's measures are based on four scalar number derived directly from a program's.

**5 MARKS**

**1.Describe about Feasibility study?**

**Feasibility Study** in [Software Engineering](https://www.geeksforgeeks.org/software-engineering-introduction-to-software-engineering/) is a study to evaluate feasibility of proposed project or system. Feasibility study is one of stage among important four stages of [Software Project Management Process](https://www.geeksforgeeks.org/software-engineering-project-management-process/). As name suggests feasibility study is the feasibility analysis or it is a measure of the software product in terms of how much beneficial product development will be for the organization in a practical point of view.

**Types of Feasibility Study :**

**Technical Feasibility –**   
In Technical Feasibility current resources both hardware software along with required technology are analyzed/assessed to develop project. This technical feasibility study gives report whether there exists correct required resources.

**Operational Feasibility –**   
In Operational Feasibility degree of providing service to requirements is analyzed along with how much easy product will be to operate and maintenance after deployment.

**Economic Feasibility –**   
In Economic Feasibility study cost and benefit of the project is analyzed. Means under this feasibility study a detail analysis is carried out what will be cost of the project for development which includes all required cost for final development like hardware and software etc.

**Legal Feasibility –**   
In Legal Feasibility study project is analyzed in legality point of view. This includes analyzing barriers of legal implementation of project, data protection, project certificate, license, copyright etc.

**Schedule Feasibility –**   
In Schedule Feasibility Study mainly timelines/deadlines is analyzed for proposed project which includes how many times teams will take to complete final project which has a great impact on the organization as purpose of project may fail if it can’t be completed on time.

**2.Define Software crisis? What are the causes of software crisis?**

Software crisis is a term used in the early days of computing science for the difficulty of writing useful and efficient computer programs in the required time. The software crisis was due to the **rapid increases in computer power and the complexity of the problems that could not be tackled**.

**Causes of Software Crisis:**

* The cost of owning and maintaining software was as expensive as developing the software
* At that time Projects were running over-time
* At that time Software was very inefficient
* The quality of the software was low quality
* Software often did not meet user requirements
* The average software project overshoots its schedule by half
* At that time Software was never delivered
* Non-optimal resource utilization.
* Difficult to alter, debug, and enhance.
* The software complexity is harder to change.

**3.Explain about Software Requirement Specification?**

The production of the requirements stage of the software development process is **Software Requirements Specifications (SRS)** (also called a **requirements document**). This report lays a foundation for software engineering activities and is constructing when entire requirements are elicited and analyzed. **SRS** is a formal report, which acts as a representation of software that enables the customers to review whether it (SRS) is according to their requirements. Also, it comprises user requirements for a system as well as detailed specifications of the system requirements.

**Following are the features of a good SRS document:**

**1. Correctness:** User review is used to provide the accuracy of requirements stated in the SRS. SRS is said to be perfect if it covers all the needs that are truly expected from the system.

**2. Completeness:** The SRS is complete if, and only if, it includes the following elements:

**I.** All essential requirements, whether relating to functionality, performance, design, constraints, attributes, or external interfaces.

**II.** Definition of their responses of the software to all realizable classes of input data in all available categories of situations.

**3. Consistency:** The SRS is consistent if, and only if, no subset of individual requirements described in its conflict.

**4. Modifiability:** SRS should be made as modifiable as likely and should be capable of quickly obtain changes to the system to some extent.

**5.Verifiability:** SRS is correct when the specified requirements can be verified with a cost-effective system to check whether the final software meets those requirements. The requirements are verified with the help of reviews.

**6.Traceability:** The SRS is traceable if the origin of each of the requirements is clear and if it facilitates the referencing of each condition in future development or enhancement documentation.

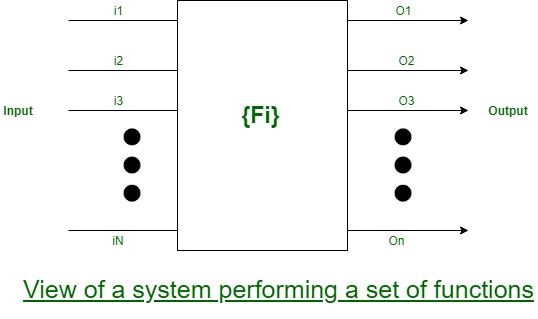
**4.Mention the values of good SRS. Also Explain the components of SRS?**

**GOOD VALUE OF SRS**

* **Breaks Down the Problem.** A good SRS will break down the problem into chunks that can be solved more readily. This also helps to increase understanding of issues and makes them easier to tackle.
* **Offers Design Input.** Your SRS should contain design details to assist with implementation and deployment.
* **Considers Components for Feedback.** A meaningful quality to users of the finished software is the opportunity to provide feedback. This should be a consideration when developing a strong SRS.
* **Includes Validation Strategies.** Validation strategies should be implemented to ensure requirements are stated correctly and function the way they are intended to.
* **Complete, Concise, and Modifiable.** The finished product should offer a total picture of the development project as concisely as possible to promote understanding.

**components of SRS**

1. **Functional Requirements:**  
   The purposeful requirements part discusses the functionalities needed from the system, The system is taken into account to perform a group of high- level functions Fi. The functional view of the system is shown in below diagram Each function Fi of the system can be considered as a transformation of a set of input data Ii to the corresponding set of output knowledge Oi.



**2.Non-functional Requirements:**  
Non-functional necessities accommodate the characteristics of the system which may not be expressed as functions – like the maintainability of the system, movability of the system, the usability of the system, etc. Non-functional requirements may include:

1. Reliability issues
2. Accuracy of results
3. Human-computer interface issues

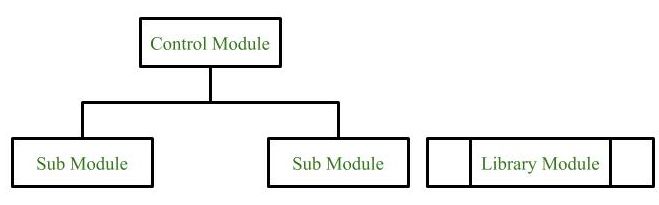
**3.Goals of Implementation:**  
The goals of implementation part documents some general suggestions relating to development. These suggestions guide trade-off among style goals. The goals of the implementation section would possibly document problems like revisions to the system functionalities that will be needed within the future, new devices to be supported within the future, reusability problems, etc.

**5.Explain structure charts briefly?**

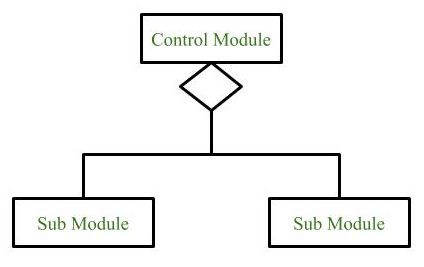
**Structure Chart** represent hierarchical structure of modules. It breaks down the entire system into lowest functional modules, describe functions and sub-functions of each module of a system to a greater detail. Structure Chart partitions the system into black boxes (functionality of the system is known to the users but inner details are unknown). Inputs are given to the black boxes and appropriate outputs are generated.

**Symbols used in construction of structured chart:**

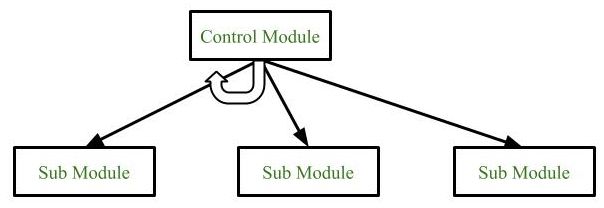
**1.Module:-Control module ,Sub module ,Library module this all 3 part odf module .**



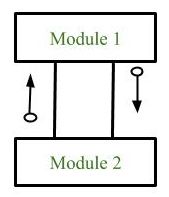
**2.Conditional Call**  
It represents that control module can select any of the sub module on the basis of some condition.



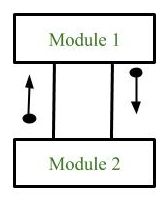
**3.Loop (Repetitive call of module)**  
It represents the repetitive execution of module by the sub module.  
A curved arrow represents loop in the module.



**4.Data Flow**  
It represents the flow of data between the modules. It is represented by directed arrow with empty circle at the end.

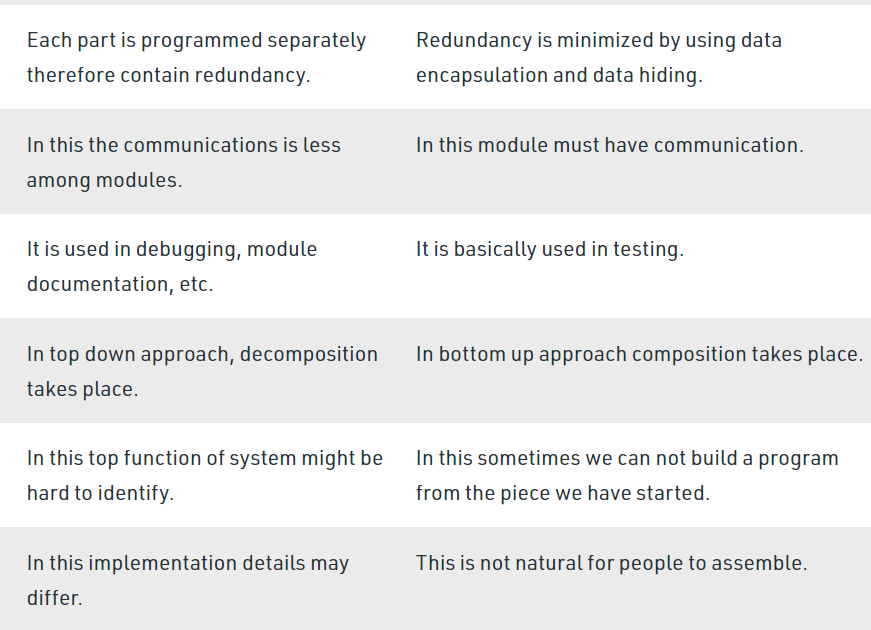


**5.Control Flow**  
It represents the flow of control between the modules. It is represented by directed arrow with filled circle at the end.



**6.Define top down and bottom-up approach in software design?**

**top down bottom-up**

****

**7.What are the different methods of testing?**

### 1. Accessibility Testing

[Accessibility testing](https://www.perfecto.io/accessibility-testing) is the practice of ensuring your mobile and web apps are working and usable for users without and with disabilities such as vision impairment, hearing disabilities, and other physical or cognitive conditions.

### 2. Acceptance Testing

Acceptance testing ensures that the end-user (customers) can achieve the goals set in the business requirements, which determines whether the software is acceptable for delivery or not. It is also known as user acceptance testing (UAT).

### 3. Black Box Testing

Black box testing involves testing against a system where the code and paths are invisible.

### 4. End to End Testing

End to end testing is a technique that tests the application’s workflow from beginning to end to make sure everything functions as expected.

### 5. Functional Testing

[Functional testing](https://www.perfecto.io/functional-testing-web-mobile-apps) checks an application, website, or system to ensure it’s doing exactly what it’s supposed to be doing.

### 6. Interactive Testing

Also known as manual testing, [interactive testing](https://www.perfecto.io/solutions/interactive-testing) enables testers to create and facilitate manual tests for those who do not use automation and collect results from external tests.

### 7. Integration Testing

Integration testing ensures that an entire, integrated system meets a set of requirements. It is performed in an integrated hardware and software environment to ensure that the entire system functions properly.

### 8. Load Testing

This type of non-functional software testing process determines how the software application behaves while being accessed by multiple users simultaneously.

### 9. Non Functional Testing

[Non functional testing](https://www.perfecto.io/blog/what-is-non-functional-testing) verifies the readiness of a system according to nonfunctional parameters (performance, accessibility, UX, etc.)  which are never addressed by functional testing.

### 10. Performance Testing

[Performance testing](https://www.blazemeter.com/blog/performance-testing-vs-load-testing-vs-stress-testing) examines the speed, stability, reliability, scalability, and resource usage of a software application under a specified workload.

**8.What are the different level of testing?**

[Software Testing](https://www.geeksforgeeks.org/software-testing-basics/) is an activity performed to identify errors so that errors can be removed to obtain a product with greater quality. To assure and maintain the quality of software and to represents the ultimate review of specification, design, and coding, Software testing is required.

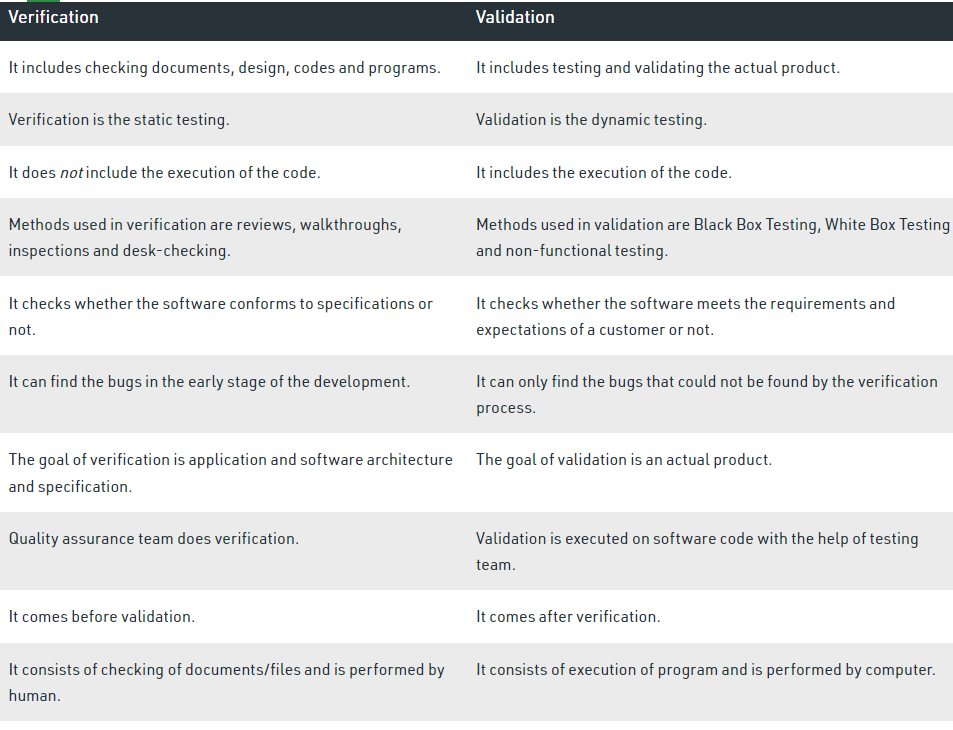
There are different levels of testing :

1. [Unit Testing](https://www.geeksforgeeks.org/unit-testing-software-testing/)**:**  
   In this type of testing, errors are detected individually from every component or unit by individually testing the components or units of software to ensure that if they are fit for use by the developers. It is the smallest testable part of the software.
2. [Integration Testing](https://www.geeksforgeeks.org/software-engineering-integration-testing/)**:**  
   In this testing, two or more modules which are unit tested are integrated to test i.e. technique interacting components and are then verified if these integrated modules work as per the expectation or not and interface errors are also detected.
3. [System Testing](https://www.geeksforgeeks.org/system-testing/)**:**  
   In system testing, complete and integrated Softwares are tested i.e. all the system elements forming the system is tested as a whole to meet the requirements of the system.
4. [Acceptance Testing](https://www.geeksforgeeks.org/acceptance-testing-software-testing/)**:**  
   It is a kind of testing conducted to ensure whether the requirement of the users are fulfilled prior to its delivery and the software works correctly in the user’s working environment.

**9.Explain waterfall and spiral model for software life cycle and discuss various activities in**

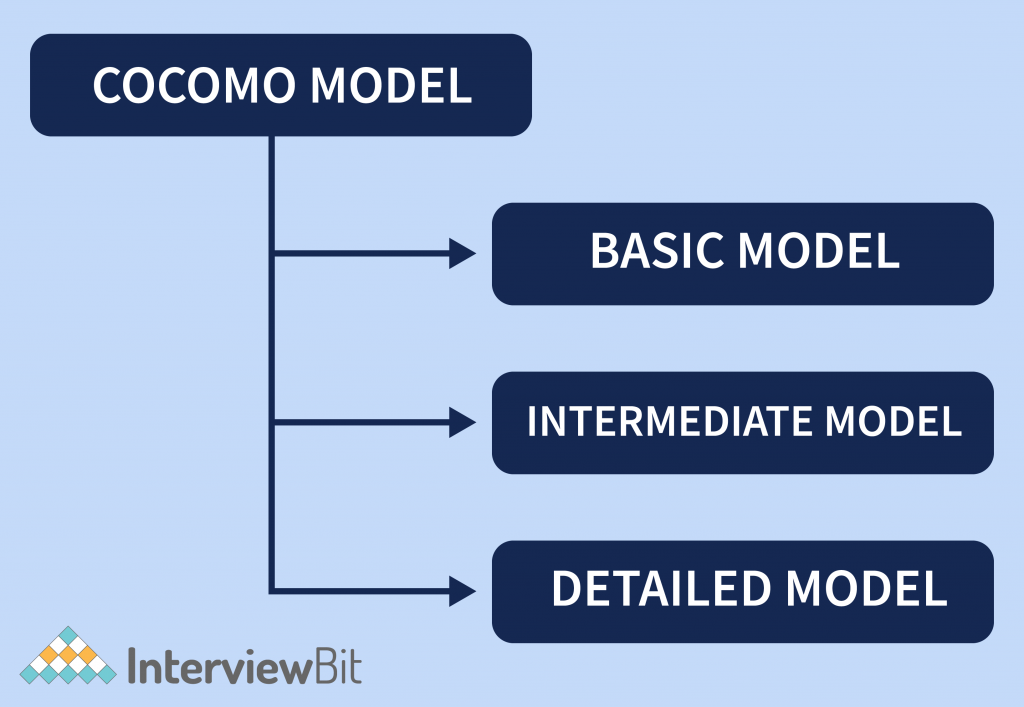
**each phase?**

**10.What are the differences between validation and verification in software development?**

****

**11.Define cocomo model? What are the type of cocomo model**COCOMO or **Constructive Cost Estimation Model** is a model that estimates the effort and time taken to complete the model based on the size of the source code. It includes 15 multiplying factors from different attributes of the project, and finally calculates time and effort using this information

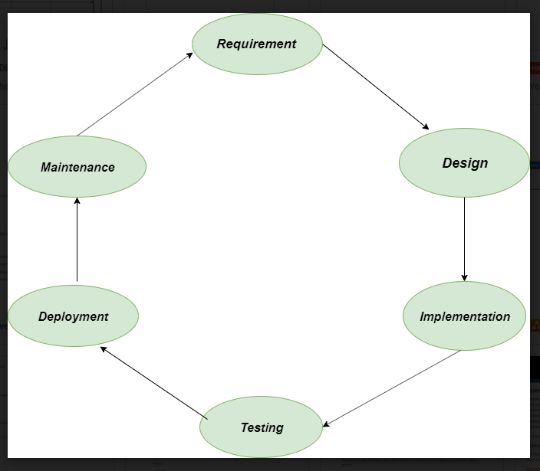
**TYPES OF COCOMO**



1. **Organic –** A software project is said to be an organic type if the team size required is adequately small, the problem is well understood and has been solved in the past and also the team members have a nominal experience regarding the problem.
2. **Semi-detached –** A software project is said to be a Semi-detached type if the vital characteristics such as team size, experience, and knowledge of the various programming environment lie in between that of organic and Embedded.
3. **Embedded –** A software project requiring the highest level of complexity, creativity, and experience requirement fall under this category. Such software requires a larger team size than the other two models.

**8 MARKS**

**1.Explain the concept of SDLC Models?**

**Software Development Life Cycle (SDLC)** is a framework that defines activities that are performed during the software development process. There are 6 phases in SDLC model as given below. 

**1. Requirement:** In this phase, all the requirements are collected from the customer/client. They are provided in a document called Businessmen requirement specification (BRS) and System requirement specification (SRS). All the details are discussed with the customer/client in detail.

**2. Design:** It has two steps:

* **High-level design (HLD):** It gives the architecture of software products.
* **Low-level design (LLD):** It describes how each and every feature in the product should work and every component.

**3. Implementation:**

* This is the longest phase.
* This phase consists of Front end + Middleware + Back-end.
* **In front-end:**Development of coding is done even SEO settings are done.
* **In Middleware:** They connect both the front end and back end.
* **In the back-end:** A database is created.

**4. Testing:** Testing is carried out to verify the entire system. The aim of the tester is to find out the gaps and defects within the system and also to check whether the system is running according to the requirement of the customer/client.

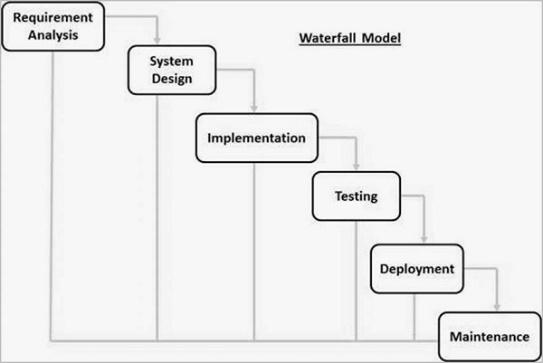
**5. Deployment:** After successful testing, the product is delivered/deployed to the client, and even clients are trained on how to use the product.

**6. Maintenance:** Once the product has been delivered to the client a task of maintenance starts as when the client will come up with an error the issue should be fixed from time to time.

**2.Explain waterfall and spiral model for software life cycle and discuss various activities in each phase?**

## **Waterfall Model - Design**

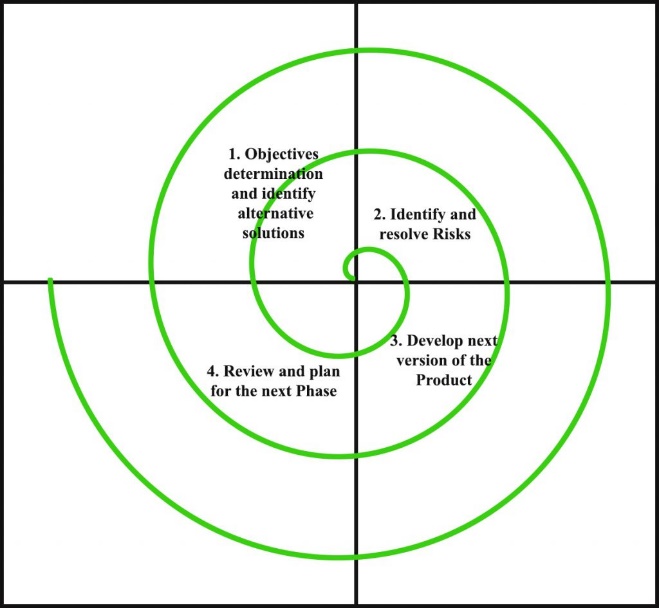
Waterfall approach was first SDLC Model to be used widely in Software Engineering to ensure success of the project. In "The Waterfall" approach, the whole process of software development is divided into separate phases. In this Waterfall model, typically, the outcome of one phase acts as the input for the next phase sequentially.



* **Requirement Gathering and analysis** − All possible requirements of the system to be developed are captured in this phase and documented in a requirement specification document.
* **System Design** − The requirement specifications from first phase are studied in this phase and the system design is prepared. This system design helps in specifying hardware and system requirements and helps in defining the overall system architecture.
* **Implementation** − With inputs from the system design, the system is first developed in small programs called units, which are integrated in the next phase.
* **Integration and Testing** − All the units developed in the implementation phase are integrated into a system after testing of each unit.
* **Deployment of system** − Once the functional and non-functional testing is done; the product is deployed in the customer environment or released into the market.
* **Maintenance** − There are some issues which come up in the client environment. To fix those issues, patches are released.

**Spiral model** is one of the most important Software Development Life Cycle models, which provides support for **Risk Handling**. In its diagrammatic representation, it looks like a spiral with many loops. The exact number of loops of the spiral is unknown and can vary from project to project. Each loop of the spiral is called a **Phase of the software development process.**

**The below diagram shows the different phases of the Spiral Model: –**



1. **Objectives determination and identify alternative solutions:** Requirements are gathered from the customers and the objectives are identified, elaborated, and analyzed at the start of every phase. Then alternative solutions possible for the phase are proposed in this quadrant.
2. **Identify and resolve Risks:** During the second quadrant, all the possible solutions are evaluated to select the best possible solution. Then the risks associated with that solution are identified and the risks are resolved using the best possible strategy. At the end of this quadrant, the Prototype is built for the best possible solution.
3. **Develop next version of the Product:** During the third quadrant, the identified features are developed and verified through testing. At the end of the third quadrant, the next version of the software is available.
4. **Review and plan for the next Phase:** In the fourth quadrant, the Customers evaluate the so far developed version of the software. In the end, planning for the next phase is started.

**3.Illustrate about Requirement Engineering Process? Requirements engineering (RE)** 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.

## **Requirement Engineering Process**

It is a four-step process, which includes -

1. Feasibility Study
2. Requirement Elicitation and Analysis
3. Software Requirement Specification
4. Software Requirement Validation
5. Software Requirement Management

### 1. Feasibility Study:

The objective behind the feasibility study is to create the reasons for developing the software that is acceptable to users, flexible to change and conformable to established standards.

### 2.Requirement Elicitation and Analysis:

This is also known as the **gathering of requirements**. Here, requirements are identified with the help of customers and existing systems processes, if available.

Analysis of requirements starts with requirement elicitation. The requirements are analyzed to identify inconsistencies, defects, omission, etc. We describe requirements in terms of relationships and also resolve conflicts if any.

### 3.Software Requirement Specification:

Software requirement specification is a kind of document which is created by a software analyst after the requirements collected from the various sources - the requirement received by the customer written in ordinary language. It is the job of the analyst to write the requirement in technical language so that they can be understood and beneficial by the development team.

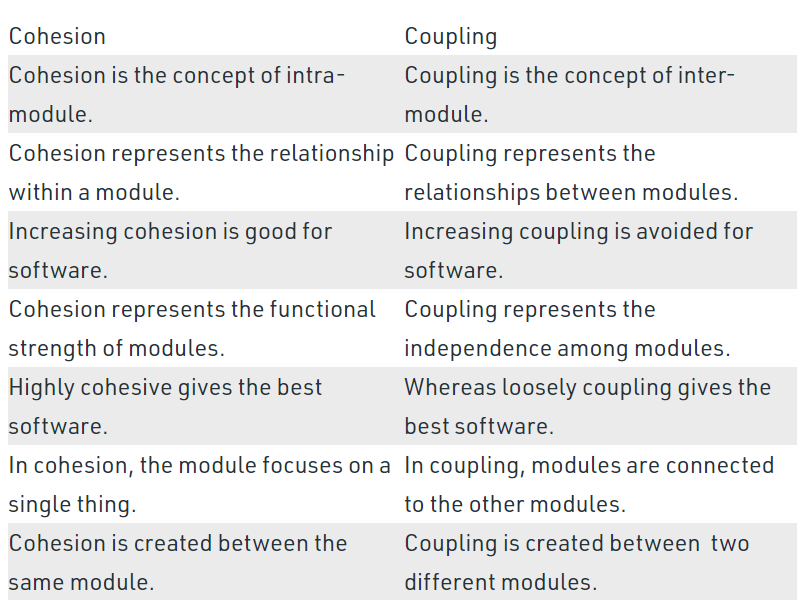
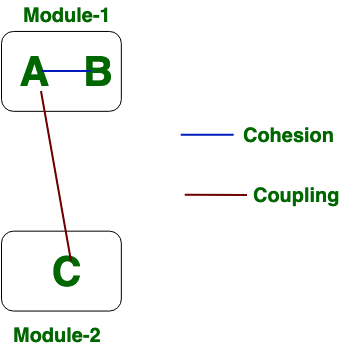
4.Software Requirement Validation:

After requirement specifications developed, the requirements discussed in this document are validated. The user might demand illegal, impossible solution or experts may misinterpret the needs. Requirements can be the check against the following conditions -

* If they can practically implement
* If they are correct and as per the functionality and specially of software
* If there are any ambiguities
* If they are full
* If they can describe

**4.Discuss the use case scenario with an example?**

**5.What is cohesion and coupling? Also explain types of coupling**

**** 

**6.Explain the following term**

**1. Halstead software science:** A computer program is an implementation of an algorithm considered to be a collection of tokens which can be classified as either operators or operand.

**Advantages:**

* Predicts error rate.
* Predicts maintenance effort
* Simple to calculate
* Measure overall quality
* Used for any language

**Disadvantages:**

* Depends on complete code
* Complexity increases as program level decreases
* Difficult to compute

**2. Function Oriented Design:** Function oriented design is the result of focusing attention to the function of the program. This is based on the stepwise refinement. Stepwise refinement is based on the iterative procedural decomposition. Stepwise refinement is a top-down strategy where a program is refined as a hierarchy of increasing levels of details. , We start with a high level description of what the program does. Then, in each step, we take one part of our high level description and refine it. Refinement is actually a process of elaboration. The process should proceed from a highly conceptual model to lower level details. The refinement of each module is done until we reach the statement level of our programming language.

**3. Object Oriented Design:** Object oriented design is the result of focusing attention not on the function performed by the program, but instead on the data that are to be manipulated by the program. Thus, it is orthogonal to function -oriented design. Object-oriented design begins with an examination of the real world “things”. These things are characteristics individually in terms of their attributes and behavior.

Objects are independent entities that may readily be changed because all state and representation information is held within the object itself. Object may be distributed and may execute sequentially or in parallel.

**7.Define functional and non-functional testing. What are the difference between the black**

**box testing and white box testing?**

**8.Define alpha and beta testing. Also define static Testing Strategies?**

**9.What is cohesion and coupling? Also define the types of coupling and cohesion?**

**10.Define software testing? Define functional and non-functional testing. What are the**

**difference between the black box testing and white box testing?**

**11.What are CASE tools? What are various types of software maintenance?**

**12.Define COCOMO model? What are the types of COCOMO model?**