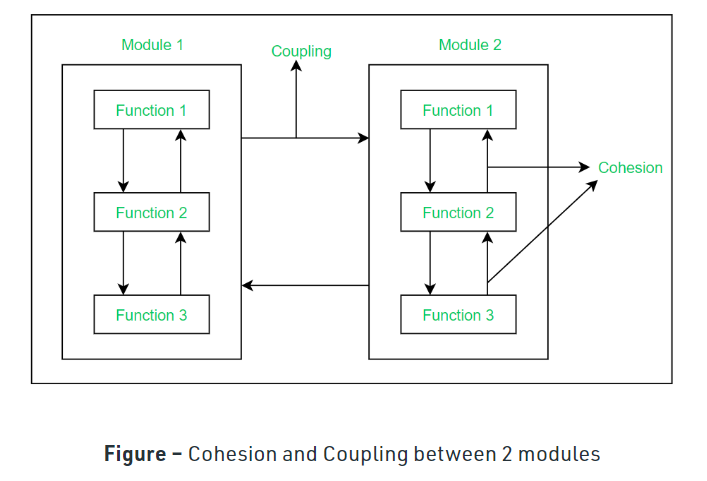
1. **What is modularization?**

The process of breaking down a software into multiple independent modules where each module is developed separately is called **Modularization**. Effective modular design can be achieved if the partitioned modules are separately solvable, modifiable as well as compilable. Here separate compilable modules means that after making changes in a module there is no need of recompiling the whole software system.

Benefits of modularization are:

* Faster Project Delivery
* Better Quality Assurance
* Safer Working Environments
* Cost Savings for Owners

Independence of modules of a software system can be measured using 2 criteria : [Cohesion, and Coupling](https://www.geeksforgeeks.org/software-engineering-coupling-and-cohesion/). These are explained as following below:



1. **Define software testing?**

Software testing can be stated as the process of verifying and validating whether a software or application is bug-free, meets the technical requirements as guided by its design and development, and meets the user requirements effectively and efficiently by handling all the exceptional and boundary cases. The process of software testing aims not only at finding faults in the existing software but also at finding measures to improve the software in terms of efficiency, accuracy, and usability. It mainly aims at measuring the specification, functionality, and performance of a software program or application.

**Software testing can be divided into two steps:**   
 **Verification:** it refers to the set of tasks that ensure that the software correctly implements a specific function.

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

Software Testing can be broadly classified into two types:

. **Manual Testing:** Manual testing includes testing software manually, i.e., without using any automation tool or any script. In this type, the tester takes over the role of an end-user and tests the software to identify any unexpected behavior or bug. There are different stages for manual testing such as unit testing, integration testing, system testing, and user acceptance testing.

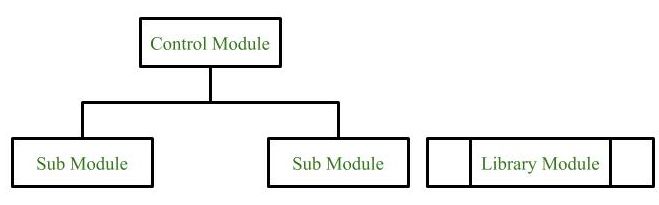
**Automation Testing:** Automation testing, which is also known as Test Automation, is when the tester writes scripts and uses another software to test the product. This process involves the automation of a manual process. Automation Testing is used to re-run the test scenarios quickly and repeatedly, that were performed manually in manual testing.

1. **Explain structure charts briefly.**

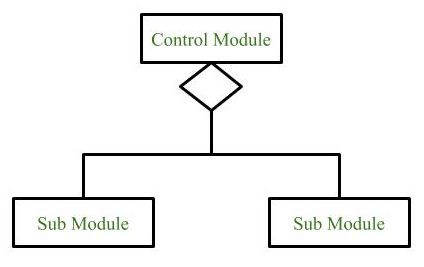
**Structure Chart** represent hierarchical structure of modules. It breaks down the entire system into lowest functional modules, describe functions and sub-functions of each module of a system to a greater detail. Structure Chart partitions the system into black boxes (functionality of the system is 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**

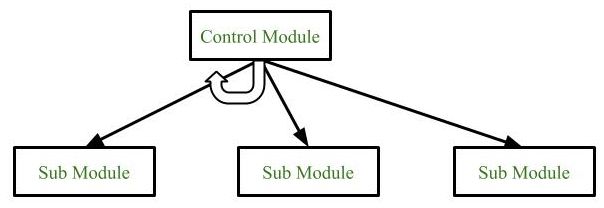
* + **Module**  
    It represents the process or task of the system. It is of three types.
  + **Control Module**  
    A control module branches to more than one sub module.
  + **Sub Module**  
    Sub Module is a module which is the part (Child) of another module.
  + **Library Module**  
    Library Module are reusable and invokable from any module.



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

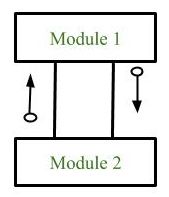


1. **Loop (Repetitive call of module)**  
   It represents the repetitive execution of module by the sub module.  
   A curved arrow represents loop in the module.

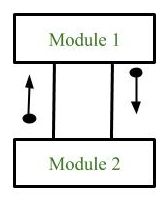


All the sub modules cover by the loop repeat execution of module.

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



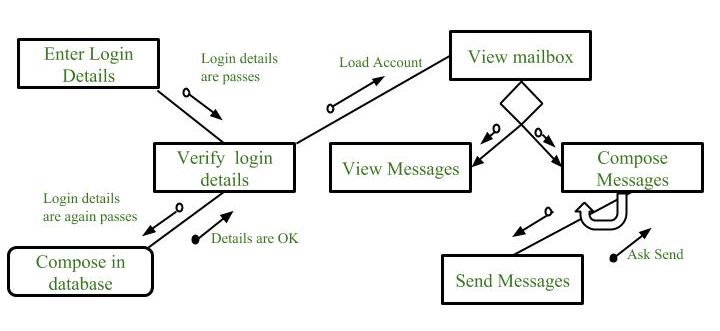
1. **Control Flow**  
   It represents the flow of control between the modules. It is represented by directed arrow with filled circle at the end.



1. **Physical Storage**  
   Physical Storage is that where all the information are to be stored.



**Example : Structure chart for an Email server**



**Types of Structure Chart:**

1. **Transform Centered Structured:**  
   These type of structure chart are designed for the systems that receives an input which is transformed by a sequence of operations being carried out by one module.
2. **Transaction Centered Structure:**  
   These structure describes a system that processes a number of different types of transaction.
3. **What are the different methods of testing.**

Testing is the process of executing a program to find errors. To make our software perform well it should be error-free. If testing is done successfully it will remove all the errors from the software.

### Types of Testing:-

#### 1. Unit Testing

It focuses on the smallest unit of software design. In this, we test an individual unit or group of interrelated units. It is often done by the programmer by using sample input and observing its corresponding outputs.

#### 2. Integration Testing

The objective is to take unit-tested components and build a program structure that has been dictated by design. Integration testing is testing in which a group of components is combined to produce output.

#### 3. Regression Testing

Every time a new module is added leads to changes in the program. This type of testing makes sure that the whole component works properly even after adding components to the complete program.

#### 4. Smoke Testing

This test is done to make sure that the software under testing is ready or stable for further testing   
It is called a smoke test as the testing of an initial pass is done to check if it did not catch the fire or smoke in the initial switch on.

#### 5. Alpha Testing

This is a type of validation testing. It is a type of *acceptance testing*which is done before the product is released to customers. It is typically done by QA people.

#### 6. Beta Testing

The beta test is conducted at one or more customer sites by the end-user of the software. This version is released for a limited number of users for testing in a real-time environment

#### 7. System Testing

This software is tested such that it works fine for the different operating systems. It is covered under the black box testing technique. In this, we just focus on the required input and output without focusing on internal working.   
In this, we have security testing, recovery testing, stress testing, and performance testing

#### 8. Stress Testing

In this, we give unfavorable conditions to the system and check how they perform in those conditions.

#### 9. Performance Testing

It is designed to test the run-time performance of software within the context of an integrated system. It is used to test the speed and effectiveness of the program. It is also called load testing. In it we check, what is the performance of the system in the given load.

#### 10. Object-Oriented Testing

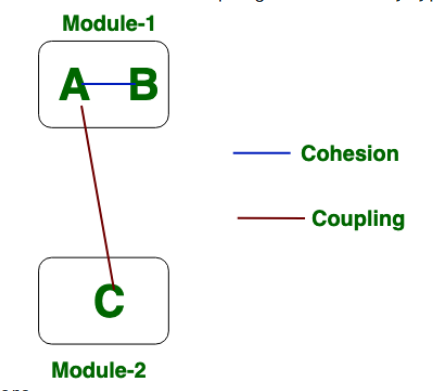
This testing is a combination of various testing techniques that help to verify and validate object-oriented software.

#### 11. Acceptance Testing

Acceptance testing is done by the customers to check whether the delivered products perform the  desired tasks or not, as stated in requirements.

1. **What is cohension and coupling? Also explain types of coupling.**

**Cohesion:** Cohesion is the indication of the relationship within the module. It is the concept of intra-module. Cohesion has many types but usually, high cohesion is good for software.

**Coupling:** Coupling is also the indication of the relationships between modules. It is the concept of the Inter-module. The coupling has also many types but usually, the low coupling is good for software. 

**Types of Coupling:**

* **Data Coupling:** If the dependency between the modules is based on the fact that they communicate by passing only data, then the modules are said to be data coupled. In data coupling, the components are independent of each other and communicate through data. Module communications don’t contain tramp data. Example-customer billing system.
* **Stamp Coupling** In stamp coupling, the complete data structure is passed from one module to another module. Therefore, it involves tramp data. It may be necessary due to efficiency factors- this choice was made by the insightful designer, not a lazy programmer.
* **Control Coupling:** If the modules communicate by passing control information, then they are said to be control coupled. It can be bad if parameters indicate completely different behavior and good if parameters allow factoring and reuse of functionality. Example- sort function that takes comparison function as an argument.
* **External Coupling:** In external coupling, the modules depend on other modules, external to the software being developed or to a particular type of hardware. Ex- protocol, external file, device format, etc.
* **Common Coupling:** The modules have shared data such as global data structures. The changes in global data mean tracing back to all modules which access that data to evaluate the effect of the change. So it has got disadvantages like difficulty in reusing modules, reduced ability to control data accesses, and reduced maintainability.
* **Content Coupling:** In a content coupling, one module can modify the data of another module, or control flow is passed from one module to the other module. This is the worst form of coupling and should be avoided.

1. **Define functional and non functional testing. What are the difference between the black box testing and white box testing.**

**Functional Testing:**  
Functional testing is a type of software testing in which the system is tested against the functional requirements and specifications. Functional testing ensures that the requirements or specifications are properly satisfied by the application. This type of testing is particularly concerned with the result of processing. It focuses on simulation of actual system usage but does not develop any system structure assumptions.

**Non-functional Testing:**  
Non-functional testing is a type of software testing that is performed to verify the non-functional requirements of the application. It verifies whether the behavior of the system is as per the requirement or not. It tests all the aspects which are not tested in functional testing.

**Differences between Black Box Testing vs White Box Testing:**

| S. No. | Black Box Testing | White Box Testing |
| --- | --- | --- |
| 1. | It is a way of software testing in which the internal structure or the program or the code is hidden and nothing is known about it. | It is a way of testing the software in which the tester has knowledge about the internal structure or the code or the program of the software. |
| 2. | Implementation of code is not needed for black box testing. | Code implementation is necessary for white box testing. |
| 3. | It is mostly done by software testers. | It is mostly done by software developers. |
| 4. | No knowledge of implementation is needed. | Knowledge of implementation is required. |
| 5. | It can be referred to as outer or external software testing. | It is the inner or the internal software testing. |
| 6. | It is a functional test of the software. | It is a structural test of the software. |
| 7. | This testing can be initiated based on the requirement specifications document. | This type of testing of software is started after a detail design document. |
| 8. | No knowledge of programming is required. | It is mandatory to have knowledge of programming. |
| 9. | It is the behavior testing of the software. | It is the logic testing of the software. |
| 10. | It is applicable to the higher levels of testing of software. | It is generally applicable to the lower levels of software testing. |
| 11. | It is also called closed testing. | It is also called as clear box testing. |
| 12. | It is least time consuming. | It is most time consuming. |
| 13. | It is not suitable or preferred for algorithm testing. | It is suitable for algorithm testing. |

1. **Define pseudo code?**

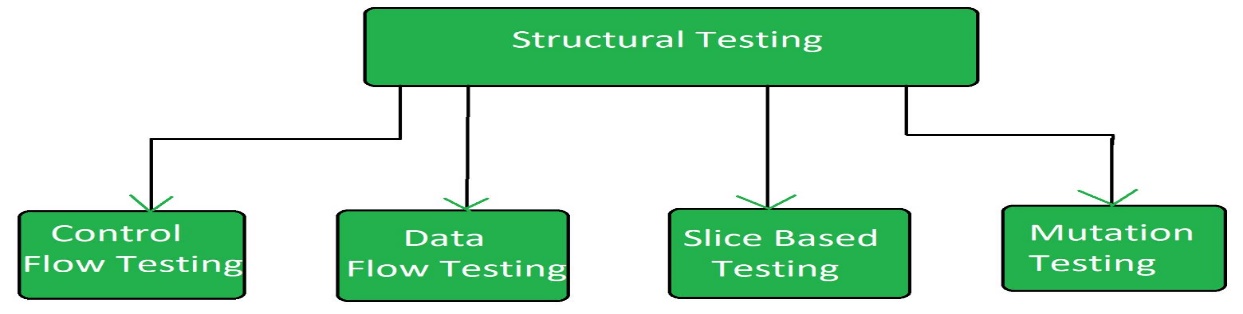
Pseudocode is an informal way of programming description that does not require any strict programming language syntax or underlying technology considerations. It is used for creating an outline or a rough draft of a program. Pseudocode summarizes a program’s flow, but excludes underlying details. System designers write pseudocode to ensure that programmers understand a software project's requirements and align code accordingly. Pseudocode is not an actual programming language. So it cannot be compiled into an executable program. It uses short terms or simple English language syntaxes to write code for programs before it is actually converted into a specific programming language.

1. What is structural testing?

**Structural testing** is a type of [software testing](https://www.geeksforgeeks.org/software-testing-basics/) 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.

Structural testing is basically related to the internal design and implementation of the software i.e. it involves the development team members in the testing team.

**Types of Structural Testing:**  
There are 4 types of Structural Testing:



1. **Define top down and bottom up approach in software design.**

**Top-Down Design Model:**   
In the top-down model, an overview of the system is formulated without going into detail for any part of it. Each part of it then refined into more details, defining it in yet more details until the entire specification is detailed enough to validate the model. if we glance at a haul as a full, it’s going to appear not possible as a result of it’s so complicated.

**Advantages:**

* Breaking problems into parts help us to identify what needs to be done.
* At each step of refinement, new parts will become less complex and therefore easier to solve.
* Parts of the solution may turn out to be reusable.
* Breaking problems into parts allows more than one person to solve the problem.
* **Bottom-Up Design Model:**   
  In this design, individual parts of the system are specified in detail. The parts are linked to form larger components, which are in turn linked until a complete system is formed. Object-oriented language such as C++ or java uses a bottom-up approach where each object is identified first.
* **Advantage:**

Easy to create test conditions.

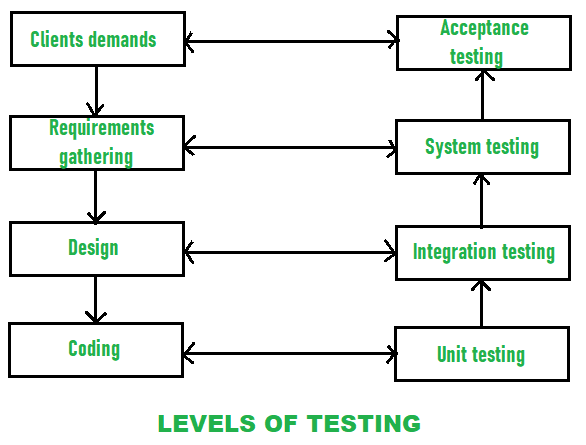
* Test results are easy to observe
* It is suited if defects occur at the bottom of the program.

1. **What are the different level of testing.**

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.

These testing can be conducted at various stages of software development. The levels of testing along with the corresponding software development phase is shown by the following diagram –



**11 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 operands. **Halstead’s metrics** are included in a number of current commercial tools that count software lines of code. By counting the tokens and determining which are operators and which are operands.

**Advantages of Halstead Metrics:**

* It is simple to calculate.
* It measures overall quality of the programs.
* It predicts the rate of error.
* It predicts maintenance effort.
* It does not require the full analysis of programming structure.
* It is useful in scheduling and reporting projects.
* It can be used for any programming language.

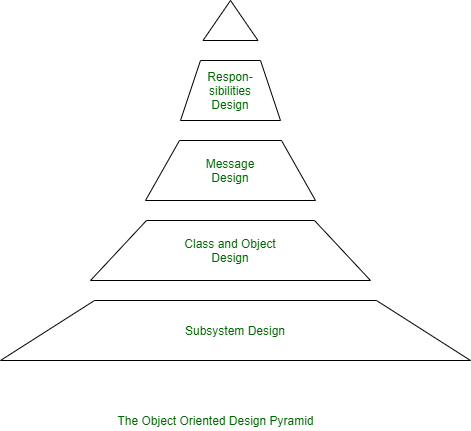
1. **Function Oriented Design** - **Function Oriented Design** is an approach to software design where the design is decomposed into a set of interacting units where each unit has a clearly defined function.

Function Oriented Design Strategies are as follows:

1. [Data Flow Diagram (DFD)](https://www.geeksforgeeks.org/levels-in-data-flow-diagrams-dfd/)**:**  
   A data flow diagram (DFD) maps out the flow of information for any process or system. It uses defined symbols like rectangles, circles and arrows, plus short text labels, to show data inputs, outputs, storage points and the routes between each destination.
2. **Data Dictionaries:**  
   Data dictionaries are simply repositories to store information about all data items defined in DFDs. At the requirement stage, data dictionaries contains data items. Data dictionaries include Name of the item, Aliases (Other names for items), Description / purpose, Related data items, Range of values, Data structure definition / form.
3. **Structure Charts:**  
   It is the hierarchical representation of system which partitions the system into black boxes

**Object Oriented Design** - **bject-Oriented Analysis (OOA):** Object-Oriented Analysis (OOA) is the first technical activity performed as part of object-oriented software engineering. OOA introduces new concepts to investigate a problem. It is based on a set of basic principles, which are as follows-

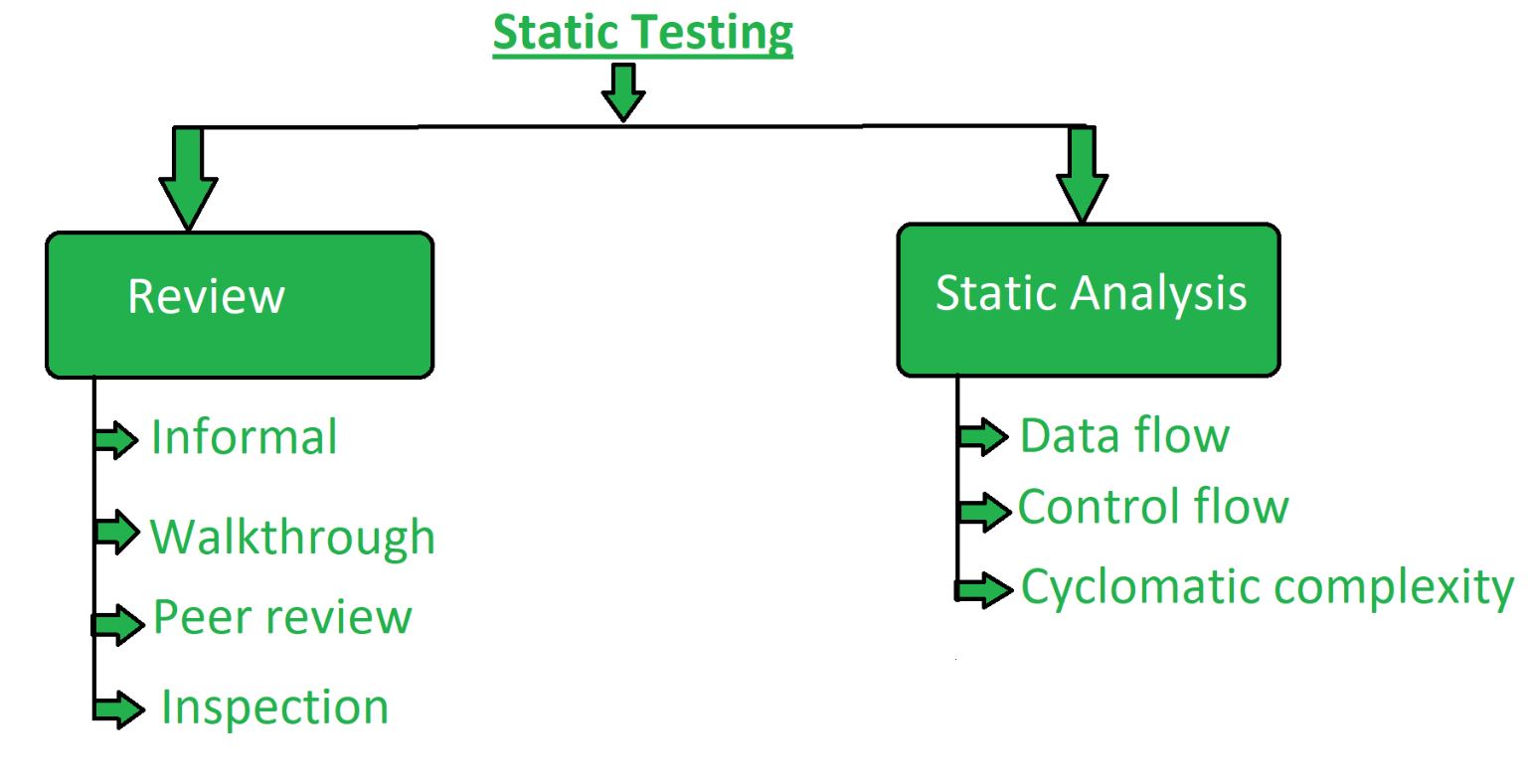
1. The information domain is modeled.
2. Behavior is represented.
3. The function is described.
4. Data, functional, and behavioral models are divided to uncover greater detail.
5. Early models represent the essence of the problem, while later ones provide implementation details.
6. It is having the following four layers.



1. **The Subsystem Layer :** It represents the subsystem that enables software to achieve user requirements and implement technical frameworks that meet user needs.
2. **The Class and Object Layer :** It represents the class hierarchies that enable the system to develop using generalization and specialization. This layer also represents each object.
3. **The Message Layer :** It represents the design details that enable each object to communicate with its partners. It establishes internal and external interfaces for the system.
4. **The Responsibilities Layer :** It represents the data structure and algorithmic design for all the attributes and operations for each object.
5. **Define alpha and beta testing. Also define static Testing Strategies.**

* **Alpha Testing** is a type of software testing performed to identify bugs before releasing the product to real users or to the public. Alpha Testing is one of the **user acceptance testings**. This is referred to as alpha testing only because it is done early on, near the end of the development of the software. Alpha testing is commonly performed by homestead software engineers or quality assurance staff. It is the last testing stage before the software is released into the real world.
* **Beta Testing** is performed by real users of the software application in a real environment. Beta testing is one of the types of **User Acceptance Testing**. A Beta version of the software, whose feedback is needed, is released to a limited number of end-users of the product to obtain feedback on the product quality. Beta testing helps in minimization of product failure risks and it provides increased quality of the product through customer validation. It is the last test before shipping a product to the customers. One of the major advantages of beta testing is direct feedback from customers.
* **Static Testing** is a type of a [Software Testing](https://www.geeksforgeeks.org/software-testing-basics/) method which is performed to check the defects in software without actually executing the code of the software application. Static testing is performed in early stage of development to avoid errors as it is easier to find sources of failures and it can be fixed easily. The errors that can’t not be found using Dynamic Testing, can be easily found by Static Testing.

**Static Testing Techniques:**  
There are mainly two type techniques used in Static Testing:



1. **What are the four types of changes are encountered during the support  
   phase?**

The support phasefocuses on “change”associated with error correction, adaptations required as the software’s environment evolves, and changes due to enhancements brought about by changing customer requirements. Four types of change are encountered during the support phase.

* **Correction:**

Even with the best quality assurance activities, it is likely that the customer will uncover defects in the software. Corrective maintenance changes the software to correct defects.

* **Adaptation:**

Over time, the original environment, that is,  CPU, [operating system](https://onlineclassnotes.com/2016/04/what-is-operating-system-describe.html), business rules etc for which the software was developed is likely to change. Adaptive maintenance results in modification to the software to accommodate changes to its external environment.

* **Enhancement:**

As software is used, the customer/user will recognize additional functions that will provide benefit. Perfectible maintenanceextends the software beyond its original functional requirements.

* **Prevention:**

Computer software deteriorates due to change, and because of this, preventive maintenance,often called software re-engineering,must be conducted to enable the software to serve the needs of its end users.

**14. State the System Engineering Hierarchy.**

System Engineering encompasses a collection of top- down and methods to navigate the hierarchy .The system engineering process usually begins with a “world view”. That is, the entire business or product domain is examined to ensure that the proper business or technology context can be established. The world view is refined to focus more fully on a specific domain of interest. Within a specific domain, the need for targeted system elements is analysed. Finally, the analysis, design, and construction of a targeted system elements are initiated. At the top of the hierarchy, a very broad context is established and, at the bottom, detailed activities, performed by the relevant engineering disciplines are conducted.

**15. Mention some of the factors to be considered during System Modeling.**

The software modeling community is primarily concerned with reducing the gap between problem and software implementation through the use of models that describe complex systems at multiple levels of abstraction and from a variety of perspectives. A model is an abstraction of some aspect of an existing or planned system. Models are created to serve particular purposes, for example, to present a human-understandable description of some aspect of a system or to present information in a form that can be mechanically analyzed.

**some of the factors to be considered during System Modeling.**

* [agent based modeling](https://en.wikipedia.org/wiki/Agent_based_modeling) - An **agent-based model** (**ABM**) is a [computational model](https://en.wikipedia.org/wiki/Computational_models) for [simulating](https://en.wikipedia.org/wiki/Computer_simulation) the actions and interactions of [autonomous agents](https://en.wikipedia.org/wiki/Autonomous_agents) (both individual or collective entities such as organizations or groups) in order to understand the behavior of a system and what governs its outcomes.
* [data modeling](https://en.wikipedia.org/wiki/Data_modeling) - Data modeling is a [process](https://en.wikipedia.org/wiki/Software_development_process) used to define and analyze data [requirements](https://en.wikipedia.org/wiki/Requirement) needed to support the [business processes](https://en.wikipedia.org/wiki/Business_process) within the scope of corresponding information systems in organizations.
* [mathematical modeling](https://en.wikipedia.org/wiki/Mathematical_modeling) - A **mathematical model** is a description of a [system](https://en.wikipedia.org/wiki/System) using [mathematical](https://en.wikipedia.org/wiki/Mathematics) concepts and [language](https://en.wikipedia.org/wiki/Language_of_mathematics). The process of developing a mathematical model is termed **mathematical modeling**. Mathematical models are used in the [natural sciences](https://en.wikipedia.org/wiki/Natural_science)

**16. Define Verification &Validation.**

**Verification** is the process of checking that a software achieves its goal without any bugs. It is the process to ensure whether the product that is developed is right or not. It verifies whether the developed product fulfills the requirements that we have. Verification is static testing.

**Validation** is the process of checking whether the software product is up to the mark or in other words product has high level requirements. It is the process of checking the validation of product i.e. it checks what we are developing is the right product. it is validation of actual and expected product. Validation is the dynamic testing.