**Introduction:-**

Testing is the process of executing a program with the aim of finding 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. 

**Principles of Testing:-**

(i) All the test should meet the customer requirements   
(ii) To make our software testing should be performed by a third party   
(iii) Exhaustive testing is not possible. As we need the optimal amount of testing based on the risk assessment of the application.   
(iv) All the test to be conducted should be planned before implementing it   
(v) It follows the Pareto rule(80/20 rule) which states that 80% of errors come from 20% of program components.   
(vi) Start testing with small parts and extend it to large parts.

**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.   
Example: 

a) In a program we are checking if loop, method or

function is working fine

b) Misunderstood or incorrect, arithmetic precedence.

c) Incorrect initialization

**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.

Integration testing is of four types: (i) Top-down (ii) Bottom-up (iii) Sandwich (iv) Big-Bang   
Example 

(a) Black Box testing:- It is used for validation.

In this we ignore internal working mechanism and

focuse on **what is the output?**.

(b) White Box testing:- It is used for verification.

In this we focus on internal mechanism i.e.

**how the output is achieved?**

**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.   
Example 

In school record suppose we have module staff, students

and finance combining these modules and checking if on

integration these module works fine is regression testing

**4. Smoke Testing**

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

If project has 2 modules so before going to module

make sure that module 1 works properly

**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.   
Example: 

When software testing is performed internally within

the organization

**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-timeenvironment   
Example: 

When software testing is performed for the limited

number of people

**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   
Example: 

This include functional as well as non functional

testing

**8. Stress Testing**

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

(a) Test cases that require maximum memory or other

resources are executed

(b) Test cases that may cause thrashing in a virtual

operating system

(c) Test cases that may cause excessive disk requirement

**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.  
Example: 

Checking number of processor cycles.

**10. Object-Oriented Testing**

This testing is a combination of various testing techniques that help to verify and validate object-oriented software. This testing is done in the following manner:

* Testing of Requirements,
* Design and Analysis of Testing,
* Testing of Code,
* Integration testing,
* System testing,
* User Testing.

We use this OOT, for discussing test plans and for executing the projects.

## What is Debugging?

**Definition:** The important technique to find and remove the number of[errors](https://www.elprocus.com/what-are-errors-in-measurement-types-of-errors-with-calculation/) or bugs or defects in a program is called Debugging. It is a multistep process in software development. It involves identifying the bug, finding the source of the bug and correcting the problem to make the program error-free. In software development, the developer can locate the code error in the program and remove it using this process.  Hence, it plays a vital role in the entire software development lifecycle.

**Debugging Techniques**

To perform the debugging process easily and efficiently, it is necessary to follow some techniques. The most commonly used debugging strategies are,

* Debugging by brute force
* Induction strategy
* Deduction strategy
* Backtracking strategy and
* Debugging by brute force is the most commonly used technique. This is done by taking memory dumps of the program which contains a large amount of information with intermediate values and analyzing them, but analyzing the information and finding the bugs leads to a waste of time and effort.
* Induction strategy includes the Location of relevant data, the Organization of data, the Devising hypothesis (provides possible causes of errors), and the Proving hypothesis.
* Deduction strategy includes Identification of possible causes of bugs or hypothesis Elimination of possible causes using the information Refining of the hypothesis( analyzing one-by-one)
* The backtracking strategy is used to locate errors in small programs. When an error occurs, the program is traced one step backward during the evaluation of values to find the cause of bug or error.

# **Coupling and Cohesion**

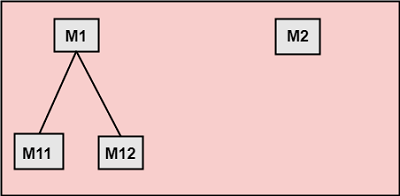
## **Module Coupling**

In software engineering, the coupling is the degree of interdependence between software modules. Two modules that are tightly coupled are strongly dependent on each other. However, two modules that are loosely coupled are not dependent on each other. **Uncoupled modules** have no interdependence at all within them.

**1. No Direct Coupling:** There is no direct coupling between M1 and M2.

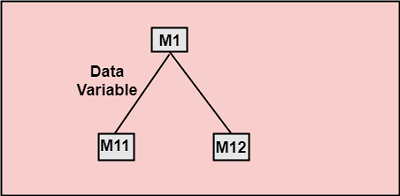
42.179Difference between JDK, JRE, and JV**Next**

**Stay**



In this case, modules are subordinates to different modules. Therefore, no direct coupling.

**2. Data Coupling:** When data of one module is passed to another module, this is called data coupling.



**3. Stamp Coupling:** Two modules are stamp coupled if they communicate using composite data items such as structure, objects, etc. When the module passes non-global data structure or entire structure to another module, they are said to be stamp coupled. For example, passing structure variable in C or object in C++ language to a module.

**4. Control Coupling:** Control Coupling exists among two modules if data from one module is used to direct the structure of instruction execution in another.

**5. External Coupling:** External Coupling arises when two modules share an externally imposed data format, communication protocols, or device interface. This is related to communication to external tools and devices.

**6. Common Coupling:** Two modules are common coupled if they share information through some global data items.

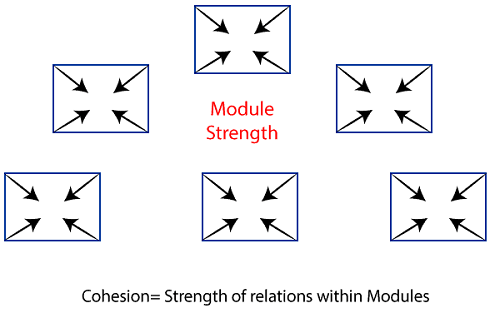


**7. Content Coupling:** Content Coupling exists among two modules if they share code, e.g., a branch from one module into another module.

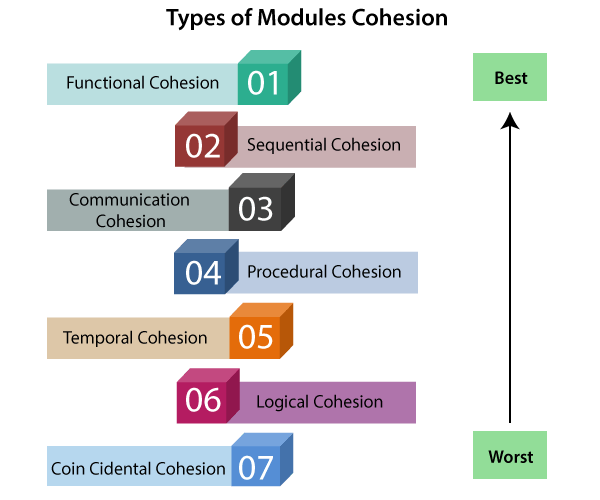
## **Module Cohesion**

In computer programming, cohesion defines to the degree to which the elements of a module belong together. Thus, cohesion measures the strength of relationships between pieces of functionality within a given module. For example, in highly cohesive systems, functionality is strongly related.

Cohesion is an **ordinal** type of measurement and is generally described as "high cohesion" or "low cohesion."



### Types of Modules Cohesion



1. **Functional Cohesion:** Functional Cohesion is said to exist if the different elements of a module, cooperate to achieve a single function.
2. **Sequential Cohesion:** A module is said to possess sequential cohesion if the element of a module form the components of the sequence, where the output from one component of the sequence is input to the next.
3. **Communicational Cohesion:** A module is said to have communicational cohesion, if all tasks of the module refer to or update the same data structure, e.g., the set of functions defined on an array or a stack.
4. **Procedural Cohesion:** A module is said to be procedural cohesion if the set of purpose of the module are all parts of a procedure in which particular sequence of steps has to be carried out for achieving a goal, e.g., the algorithm for decoding a message.
5. **Temporal Cohesion:** When a module includes functions that are associated by the fact that all the methods must be executed in the same time, the module is said to exhibit temporal cohesion.
6. **Logical Cohesion:** A module is said to be logically cohesive if all the elements of the module perform a similar operation. For example Error handling, data input and data output, etc.
7. **Coincidental Cohesion:** A module is said to have coincidental cohesion if it performs a set of tasks that are associated with each other very loosely, if at all