Differences between Black Box Testing vs White Box Testing

[Software Testing](https://www.geeksforgeeks.org/software-testing-basics/) can be majorly classified into two categories:

1. [**Black Box Testing**](https://www.geeksforgeeks.org/software-engineering-black-box-testing/) is a software testing method in which the internal structure/ design/ implementation of the item being tested is not known to the tester
2. [**White Box Testing**](https://www.geeksforgeeks.org/software-engineering-white-box-testing/) is a software testing method in which the internal structure/ design/ implementation of the item being tested is known to the tester.

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

| **BLACK BOX TESTING** | **WHITE BOX TESTING** |
| --- | --- |
| 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 r the code or the program of the software. |
| It is mostly done by software testers. | It is mostly done by software developers. |
| No knowledge of implementation is needed. | Knowledge of implementation is required. |
| It can be referred as outer or external software testing. | It is the inner or the internal software testing. |
| It is functional test of the software. | It is structural test of the software. |
| This testing can be initiated on the basis of requirement specifications document. | This type of testing of software is started after detail design document. |
| No knowledge of programming is required. | It is mandatory to have knowledge of programming. |
| It is the behavior testing of the software. | It is the logic testing of the software. |
| It is applicable to the higher levels of testing of software. | It is generally applicable to the lower levels of software testing. |
| It is also called closed testing. | It is also called as clear box testing. |
| It is least time consuming. | It is most time consuming. |
| It is not suitable or preferred for algorithm testing. | It is suitable for algorithm testing. |
| Can be done by trial and error ways and methods. | Data domains along with inner or internal boundaries can be better tested. |
| **Example:** search something on google by using keywords | **Example:** by input to check and verify loops |
| **Types of Black Box Testing:**   * A. Functional Testing * B. Non-functional testing * C. Regression Testing | **Tyeps of White Box Testing:**   * A. Path Testing * B. Loop Testing * C. Condition testing |

# Differences between Verification and Validation

Prerequisite – [Verification and Validation](https://www.geeksforgeeks.org/software-engineering-verification-and-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.  
Verification means **Are we building the product right?**

**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.  
Validation means **Are we building the right product?**

The difference between Verification and Validation is as follow:

| **VERIFICATION** | **VALIDATION** |
| --- | --- |
| It includes checking documents, design, codes and programs. | It includes testing and validating the actual product. |
| Verification is the static testing. | Validation is the dynamic testing. |
| It does not include the execution of the code. | It includes the execution of the code. |
| Methods used in verification are reviews, walkthroughs, inspections and desk-checking. | Methods used in validation are Black Box Testing, White Box Testing and non-functional testing. |
| It checks whether the software conforms to specifications or not. | It checks whether the software meets the requirements and expectations of a customer or not. |
| It can find the bugs in the early stage of the development. | It can only find the bugs that could not be found by the verification process. |
| The goal of verification is application and software architecture and specification. | The goal of validation is an actual product. |
| Quality assurance team does verification. | Validation is executed on software code with the help of testing team. |
| It comes before validation. | It comes after verification. |

## What is Verification?

**Definition** : The process of evaluating software to determine whether the products of a given development phase satisfy the conditions imposed at the start of that phase.

Verification is a static practice of verifying documents, design, code and program. It includes all the activities associated with producing high quality software: inspection, design analysis and specification analysis. It is a relatively objective process.

Verification will help to determine whether the software is of high quality, but it will not ensure that the system is useful. Verification is concerned with whether the system is well-engineered and error-free.

**Methods of Verification :**[**Static Testing**](http://toolsqa.com/software-testing/static-testing/)

* Walkthrough
* Inspection
* Review

## What is Validation?

**Definition**: The process of evaluating software during or at the end of the development process to determine whether it satisfies specified requirements.

Validation is the process of evaluating the final product to check whether the software meets the customer expectations and requirements. It is a dynamic mechanism of validating and testing the actual product.

**Methods of Validation :**[**Dynamic Testing**](http://toolsqa.com/software-testing/dynamic-testing/)

* Testing
* End Users

## Difference between Verification and Validation

The distinction between the two terms is largely to do with the role of specifications.

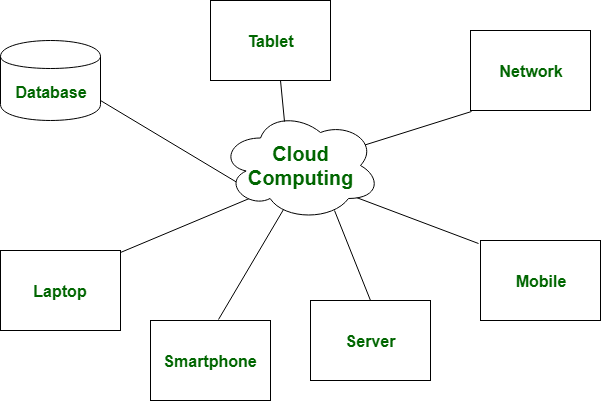
**Validation** is the process of checking whether the specification captures the customer’s needs. “Did I build what I said I would?”

**Verification** is the process of checking that the software meets the specification.  “Did I build what I need?”

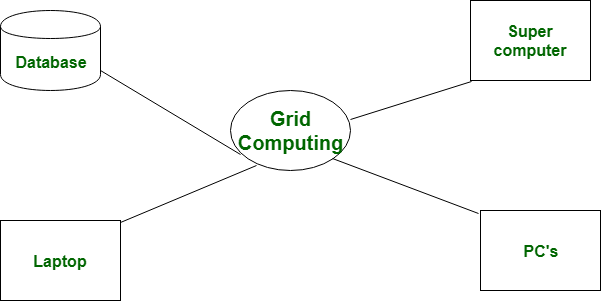
|  |  |
| --- | --- |
| **Verification** | **Validation** |
| 1. **Verification** is a static practice of verifying documents, design, code and program. | 1. **Validation** is a dynamic mechanism of validating and testing the actual product. |
| 2. It does not involve executing the code. | 2. It always involves executing the code. |
| 3. It is human based checking of documents and files. | 3. It is computer based execution of program. |
| 4. **Verification** uses methods like inspections, reviews, walkthroughs, and Desk-checking etc. | 4. **Validation** uses methods like black box (functional)  testing, gray box testing, and white box (structural) testing etc. |
| 5. **Verification**is to check whether the software conforms to specifications. | 5. **Validation** is to check whether software meets the customer expectations and requirements. |
| 6. It can catch errors that validation cannot catch. It is low level exercise. | 6. It can catch errors that verification cannot catch. It is High Level Exercise. |
| 7. Target is requirements specification, application and software architecture, high level, complete design, and database design etc. | 7. Target is actual product-a unit, a module, a bent of integrated modules, and effective final product. |
| 8. **Verification** is done by QA team to ensure that the software is as per the specifications in the SRS document. | 8. **Validation** is carried out with the involvement of testing team. |
| 9. It generally comes first-done before validation. | 9. It generally follows after **verification**. |

# Difference between Cloud Computing and Grid Computing

**Cloud Computing:**  
Cloud Computing is a Client-server computing architecture. In cloud computing, resources are used in centralized pattern and cloud computing is a high accessible service. It is a pay and use business means, in cloud computing, the users pay for the use



**Grid Computing:**  
Grid Computing is a Distributed computing architecture. In grid computing, resources are used in collaborative pattern and also in grid computing, the users do not pay for use.



Let’s see the difference between cloud and grid computing which are given below:

|  |  |  |
| --- | --- | --- |
| **S.NO** | **CLOUD COMPUTING** | **GRID COMPUTING** |
| 1. | Cloud computing is a Client-server computing architecture. | While it is a Distributed computing architecture. |
| 2. | Cloud computing is a centralized executive. | While grid computing is a decentralized executive. |
| 3. | In cloud computing, resources are used in centralized pattern. | While in grid computing, resources are used in collaborative pattern. |
| 4. | It is more flexible than grid computing. | While it is less flexible than cloud computing. |
| 5. | In cloud computing, the users pay for the use. | While in grid computing, the users do not pay for use. |
| 6. | Cloud computing is a high accessible service. | While grid computing is a low accessible service. |

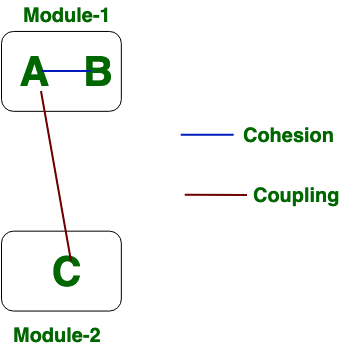
Difference Functional-oriented Approach v/s Object-oriented Design  
  
1.FOD: The basic abstractions, which are given to the user, are real world functions.  
OOD: The basic abstractions are not the real world functions but are the data abstraction where the real world entities are represented.  
  
2.FOD: Functions are grouped together by which a higher level function is obtained.an eg of this technique isSA/SD.  
OOD: Functions are grouped together on the basis of the data they operate since the classes are associated with their methods.  
  
3.FOD: In this appproach the state information is often represented in a centralized shared memory.  
OOD: In this approach the state information is not represented in a centralized memory but is implemented or distributed among the objects of the system.  
  
4.FOD approach is mainly used for computation sensitive application,  
OOD: whereas OOD approach is mainly used for evolving system which mimicks a business process or business case.  
  
5. In FOD - we decompose in function/procedure level  
OOD: - we decompose in class level  
  
6. FOD: TOp down Approach  
OOD: Bottom up approach  
  
7. FOD: It views system as Black Box that performs high level function and later decompose it detailed function so to be maaped to modules.  
OOD: Object-oriented design is the discipline of defining the objects and their interactions to solve a problem that was identified and documented during object-oriented analysis.  
  
8. FOD: Begins by considering the use case diagrms and Scenarios.  
OOD: Begins by identifiying objects and classes.

Function oriented design is dividing a bigger problem set  
to small functional units and then  
structure/organize/sequence these functional units to  
design the solution.  
OOD is identifying objects (entities) involved in the  
system and designing solution based on their relationships  
and interactions.  
FOD approach is mainly used for computation sensitive  
application, whereas OOD approach is mainly used for  
evolving system which mimicks a business process or  
business case.

# Software Engineering | Differences between Coupling and Cohesion

Prerequisite – [Coupling and Cohesion](https://www.geeksforgeeks.org/software-engineering-coupling-and-cohesion/)  
**Cohesion:**  
Cohesion is the indication of the relationship within module. It is concept of intra-module. Cohesion has many types but usually highly cohesion is good for software.

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



Now we will see the difference between Cohesion and Coupling. the differences between cohesion and coupling are given below:

|  |  |
| --- | --- |
| **COHESION** | **COUPLING** |
| Cohesion is the concept of intra module. | Coupling is the concept of inter module. |
| Cohesion represents the relationship within module. | Coupling represents the relationships between modules. |
| Increasing in cohesion is good for software. | Increasing in coupling is avoided for software. |
| Cohesion represents the functional strength of modules. | Coupling represents the independence among modules. |
| Highly cohesive gives the best software. | Where as loosely coupling gives the best software. |
| In cohesion, module focuses on the single thing. | In coupling, modules are connected to the other modules. |