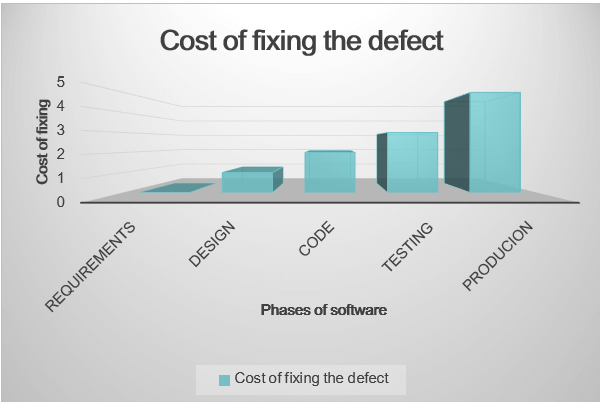
**Objective Of Testing**

**Software Testing has following objectives:**

* + Preventing defects  by verifying requirements are implemented completely and correctly.
  + Testing early in the life cycle by reviewing requirement documents.



* + - Finding defects. Eg.[Integration Testings](https://tutorialshut.com/test-levels/integration-testing/), [System Testing](https://tutorialshut.com/test-levels/system-testing/) ,[Regression Testing](https://tutorialshut.com/smoke-testing-sanity-testing/" \l "regression-testing) to detect defects at all levels.
    - To make software predictable in nature. Eg.: software is behaving as per requirements
    - To reduce cost of overall software development process. Eg.: Testing the software relatively early in the life cycle reduces the cost of development
    - Retention of customers by gaining confidence about the level of quality. Eg. [Acceptance Testing](https://tutorialshut.com/test-levels/acceptance-testing/)
    - Providing information for decision-making. Eg. providing reports to help take decisions based on all metrics, Information on risk of releasing for decision making

To ensure software is delivered without defects which could affect customers reputation and everyday operations can be achieved by Testing.

# Unit Testing – Software Testing

Unit testing is a type of software testing that focuses on individual units or components of a software system. The purpose of unit testing is to validate that each unit of the software works as intended and meets the requirements. Unit testing is typically performed by developers, and it is performed early in the development process before the code is integrated and tested as a whole system.

Unit tests are automated and are run each time the code is changed to ensure that new code does not break existing functionality. Unit tests are designed to validate the smallest possible unit of code, such as a function or a method, and test it in isolation from the rest of the system. This allows developers to quickly identify and fix any issues early in the development process, improving the overall quality of the software and reducing the time required for later testing.

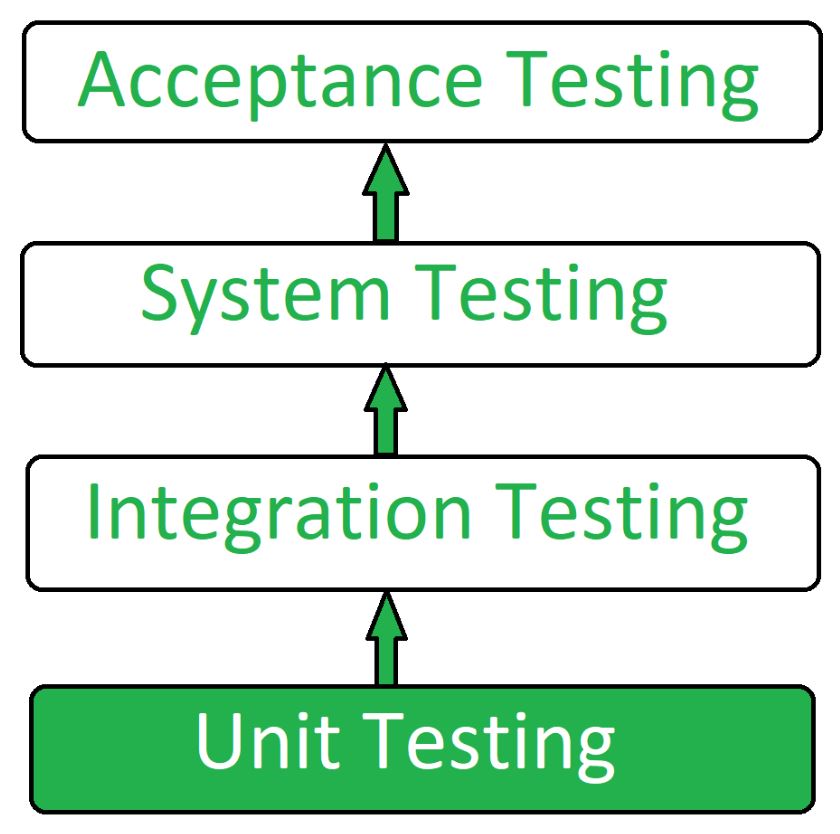
Prerequisite – [Types of Software Testing](https://www.geeksforgeeks.org/types-software-testing/)

**Unit Testing** is a software testing technique using which individual units of software i.e. group of computer program modules, usage procedures, and operating procedures are tested to determine whether they are suitable for use or not. It is a testing method using which every independent module is tested to determine if there is an issue by the developer himself. It is correlated with the functional correctness of the independent modules. Unit Testing is defined as a type of software testing where individual components of a software are tested. Unit Testing of the software product is carried out during the development of an application. An individual component may be either an individual function or a procedure. Unit Testing is typically performed by the developer. In SDLC or V Model, Unit testing is the first level of testing done before integration testing. Unit testing is a type of testing technique that is usually performed by developers. Although due to the reluctance of developers to test, quality assurance engineers also do unit testing.

### Objective of Unit Testing:

The objective of Unit Testing is:

1. To isolate a section of code.
2. To verify the correctness of the code.
3. To test every function and procedure.
4. To fix bugs early in the development cycle and to save costs.
5. To help the developers understand the code base and enable them to make changes quickly.
6. To help with code reuse.



### Types of Unit Testing:

There are 2 types of Unit Testing: **Manual**, and **Automated**.

### Workflow of Unit Testing:https://media.geeksforgeeks.org/wp-content/uploads/20190418130430/Capture884444.jpg****Unit Testing Techniques:****

There are 3 types of Unit Testing Techniques. They are

1. **Black Box Testing:**This testing technique is used in covering the unit tests for input, user interface, and output parts.
2. **White Box Testing:**This technique is used in testing the functional behavior of the system by giving the input and checking the functionality output including the internal design structure and code of the modules.
3. **Gray Box Testing:**This technique is used in executing the relevant test cases, test methods, and test functions, and analyzing the code performance for the modules.

### Unit Testing Tools:

Here are some commonly used Unit Testing tools:

1. Jtest
2. Junit
3. NUnit
4. EMMA
5. PHPUnit

### Advantages of Unit Testing:

1. Unit Testing allows developers to learn what functionality is provided by a unit and how to use it to gain a basic understanding of the unit API.
2. Unit testing allows the programmer to refine code and make sure the module works properly.
3. Unit testing enables testing parts of the project without waiting for others to be completed.
4. Early Detection of Issues: Unit testing allows developers to detect and fix issues early in the development process before they become larger and more difficult to fix.
5. Improved Code Quality: Unit testing helps to ensure that each unit of code works as intended and meets the requirements, improving the overall quality of the software.
6. Increased Confidence: Unit testing provides developers with confidence in their code, as they can validate that each unit of the software is functioning as expected.
7. Faster Development: Unit testing enables developers to work faster and more efficiently, as they can validate changes to the code without having to wait for the full system to be tested.
8. Better Documentation: Unit testing provides clear and concise documentation of the code and its behavior, making it easier for other developers to understand and maintain the software.
9. Facilitation of Refactoring: Unit testing enables developers to safely make changes to the code, as they can validate that their changes do not break existing functionality.
10. Reduced Time and Cost: Unit testing can reduce the time and cost required for later testing, as it helps to identify and fix issues early in the development process.

### Disadvantages of Unit Testing:

1. The process is time-consuming for writing the unit test cases.
2. Unit Testing will not cover all the errors in the module because there is a chance of having errors in the modules while doing integration testing.
3. Unit Testing is not efficient for checking the errors in the UI(User Interface) part of the module.
4. It requires more time for maintenance when the source code is changed frequently.
5. It cannot cover the non-functional testing parameters such as scalability, the performance of the system, etc.
6. Time and Effort: Unit testing requires a significant investment of time and effort to create and maintain the test cases, especially for complex systems.
7. Dependence on Developers: The success of unit testing depends on the developers, who must write clear, concise, and comprehensive test cases to validate the code.
8. Difficulty in Testing Complex Units: Unit testing can be challenging when dealing with complex units, as it can be difficult to isolate and test individual units in isolation from the rest of the system.
9. Difficulty in Testing Interactions: Unit testing may not be sufficient for testing interactions between units, as it only focuses on individual units.
10. Difficulty in Testing User Interfaces: Unit testing may not be suitable for testing user interfaces, as it typically focuses on the functionality of individual units.
11. Over-reliance on Automation: Over-reliance on automated unit tests can lead to a false sense of security, as automated tests may not uncover all possible issues or bugs.
12. Maintenance Overhead: Unit testing requires ongoing maintenance and updates, as the code and test cases must be kept up-to-date with changes to the software.

# Integration Testing – Software Engineering

**Integration testing** is the process of testing the interface between two software units or modules. It focuses on determining the correctness of the interface. The purpose of integration testing is to expose faults in the interaction between integrated units. Once all the modules have been unit-tested, integration testing is performed.

Integration testing is a software testing technique that focuses on verifying the interactions and data exchange between different components or modules of a software application. The goal of integration testing is to identify any problems or bugs that arise when different components are combined and interact with each other. Integration testing is typically performed after unit testing and before system testing. It helps to identify and resolve integration issues early in the development cycle, reducing the risk of more severe and costly problems later on.

Integration testing can be done by picking module by module. This can be done so that there should be a proper sequence to be followed. And also if you don’t want to miss out on any integration scenarios then you have to follow the proper sequence. Exposing the defects is the major focus of the integration testing and the time of interaction between the integrated units.  
**Integration test approaches –** There are four types of integration testing approaches. Those approaches are the following:

**1. Big-Bang Integration Testing –** It is the simplest integration testing approach, where all the modules are combined and the functionality is verified after the completion of individual module testing. In simple words, all the modules of the system are simply put together and tested. This approach is practicable only for very small systems. If an error is found during the integration testing, it is very difficult to localize the error as the error may potentially belong to any of the modules being integrated. So, debugging errors reported during Big Bang integration testing is very expensive to fix.

Big-bang integration testing is a software testing approach in which all components or modules of a software application are combined and tested at once. This approach is typically used when the software components have a low degree of interdependence or when there are constraints in the development environment that prevent testing individual components. The goal of big-bang integration testing is to verify the overall functionality of the system and to identify any integration problems that arise when the components are combined. While big-bang integration testing can be useful in some situations, it can also be a high-risk approach, as the complexity of the system and the number of interactions between components can make it difficult to identify and diagnose problems.

**Advantages:**

1. It is convenient for small systems.
2. Simple and straightforward approach.
3. Can be completed quickly.
4. Does not require a lot of planning or coordination.
5. May be suitable for small systems or projects with a low degree of interdependence between components.

**Disadvantages:**

1. There will be quite a lot of delay because you would have to wait for all the modules to be integrated.
2. High-risk critical modules are not isolated and tested on priority since all modules are tested at once.
3. Not Good for long projects.
4. High risk of integration problems that are difficult to identify and diagnose.
5. This can result in long and complex debugging and troubleshooting efforts.
6. This can lead to system downtime and increased development costs.
7. May not provide enough visibility into the interactions and data exchange between components.
8. This can result in a lack of confidence in the system’s stability and reliability.
9. This can lead to decreased efficiency and productivity.
10. This may result in a lack of confidence in the development team.
11. This can lead to system failure and decreased user satisfaction.

**2. Bottom-Up Integration Testing –** In bottom-up testing, each module at lower levels are tested with higher modules until all modules are tested. The primary purpose of this integration testing is that each subsystem tests the interfaces among various modules making up the subsystem. This integration testing uses test drivers to drive and pass appropriate data to the lower-level modules.

**Advantages:**

* In bottom-up testing, no stubs are required.
* A principal advantage of this integration testing is that several disjoint subsystems can be tested simultaneously.
* It is easy to create the test conditions.
* Best for applications that uses bottom up design approach.
* It is Easy to observe the test results.

**Disadvantages:**

* Driver modules must be produced.
* In this testing, the complexity that occurs when the system is made up of a large number of small subsystems.
* As Far modules have been created, there is no working model can be represented.

**3. Top-Down Integration Testing –** Top-down integration testing technique is used in order to simulate the behaviour of the lower-level modules that are not yet integrated. In this integration testing, testing takes place from top to bottom. First, high-level modules are tested and then low-level modules and finally integrating the low-level modules to a high level to ensure the system is working as intended.

**Advantages:**

* Separately debugged module.
* Few or no drivers needed.
* It is more stable and accurate at the aggregate level.
* Easier isolation of interface errors.
* In this, design defects can be found in the early stages.

**Disadvantages:**

* Needs many Stubs.
* Modules at lower level are tested inadequately.
* It is difficult to observe the test output.
* It is difficult to stub design.

**4. Mixed Integration Testing –** A mixed integration testing is also called sandwiched integration testing. A mixed integration testing follows a combination of top down and bottom-up testing approaches. In top-down approach, testing can start only after the top-level module have been coded and unit tested. In bottom-up approach, testing can start only after the bottom level modules are ready. This sandwich or mixed approach overcomes this shortcoming of the top-down and bottom-up approaches. It is also called the hybrid integration testing. also, stubs and drivers are used  in mixed integration testing.

**Advantages:**

* Mixed approach is useful for very large projects having several sub projects.
* This Sandwich approach overcomes this shortcoming of the top-down and bottom-up approaches.
* Parallel test can be performed in top and bottom layer tests.

**Disadvantages:**

* For mixed integration testing, it requires very high cost because one part has a Top-down approach while another part has a bottom-up approach.
* This integration testing cannot be used for smaller systems with huge interdependence between different modules.

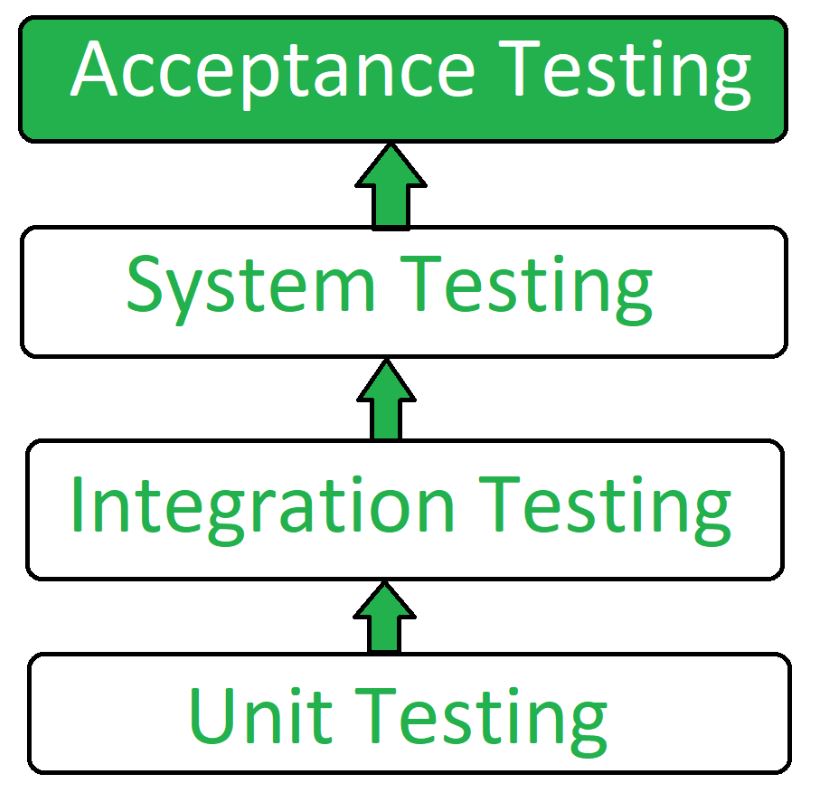
### Applications:

1. **Identify the components:** Identify the individual components of your application that need to be integrated. This could include the frontend, backend, database, and any third-party services.
2. **Create a test plan:** Develop a test plan that outlines the scenarios and test cases that need to be executed to validate the integration points between the different components. This could include testing data flow, communication protocols, and error handling.
3. **Set up test environment:** Set up a test environment that mirrors the production environment as closely as possible. This will help ensure that the results of your integration tests are accurate and reliable.
4. **Execute the tests:** Execute the tests outlined in your test plan, starting with the most critical and complex scenarios. Be sure to log any defects or issues that you encounter during testing.
5. **Analyze the results:** Analyze the results of your integration tests to identify any defects or issues that need to be addressed. This may involve working with developers to fix bugs or make changes to the application architecture.
6. **Repeat testing:**Once defects have been fixed, repeat the integration testing process to ensure that the changes have been successful and that the application still works as expected.

# Acceptance Testing | Software Testing

Acceptance Testing is a method of software testing where a system is tested for acceptability. The major aim of this test is to evaluate the compliance of the system with the business requirements and assess whether it is acceptable for delivery or not. **Standard Definition of Acceptance Testing:**

*It is a formal testing according to user needs, requirements and business processes conducted to determine whether a system satisfies the acceptance criteria or not and to enable the users, customers or other authorized entities to determine whether to accept the system or not.*

Acceptance Testing is the last phase of software testing performed after System Testing and before making the system available for actual use.   
   
**Types of Acceptance Testing:**

1. **User Acceptance Testing (UAT):** User acceptance testing is used to determine whether the product is working for the user correctly. Specific requirements which are quite often used by the customers are primarily picked for the testing purpose. This is also termed as *End-User* Testing.
2. **Business Acceptance Testing (BAT):** BAT is used to determine whether the product meets the business goals and purposes or not. BAT mainly focuses on business profits which are quite challenging due to the changing market conditions and new technologies so the current implementation may have to being changed which results in extra budgets.
3. **Contract Acceptance Testing (CAT):** CAT is a contract that specifies that once the product goes live, within a predetermined period, the acceptance test must be performed and it should pass all the acceptance use cases. Here is a contract termed a Service Level Agreement (SLA), which includes the terms where the payment will be made only if the Product services are in-line with all the requirements, which means the contract is fulfilled. Sometimes, this contract happens before the product goes live. There should be a well-defined contract in terms of the period of testing, areas of testing, conditions on issues encountered at later stages, payments, etc.
4. **Regulations Acceptance Testing (RAT):** RAT is used to determine whether the product violates the rules and regulations that are defined by the government of the country where it is being released. This may be unintentional but will impact negatively on the business. Generally, the product or application that is to be released in the market, has to go under RAT, as different countries or regions have different rules and regulations defined by its governing bodies. If any rules and regulations are violated for any country then that country or the specific region then the product will not be released in that country or region. If the product is released even though there is a violation then only the vendors of the product will be directly responsible.
5. **Operational Acceptance Testing (OAT):** OAT is used to determine the operational readiness of the product and is non-functional testing. It mainly includes testing of recovery, compatibility, maintainability, reliability, etc. OAT assures the stability of the product before it is released to production.
6. **Alpha Testing:** Alpha testing is used to determine the product in the development testing environment by a specialized testers team usually called alpha testers.
7. **Beta Testing:** Beta testing is used to assess the product by exposing it to the real end-users, usually called beta testers in their environment. Feedback is collected from the users and the defects are fixed. Also, this helps in enhancing the product to give a rich user experience.

**Use of Acceptance Testing:**

* To find the defects missed during the functional testing phase.
* How well the product is developed.
* A product is what actually the customers need.
* Feedback help in improving the product performance and user experience.
* Minimize or eliminate the issues arising from the production.

**Advantages of Acceptance Testing :**

* This testing helps the project team to know the further requirements from the users directly as it involves the users for testing.
* Automated test execution.
* It brings confidence and satisfaction to the clients as they are directly involved in the testing process.
* It is easier for the user to describe their requirement.
* It covers only the Black-Box testing process and hence the entire functionality of the product will be tested.

**Disadvantages of Acceptance Testing :**

* Users should have basic knowledge about the product or application.
* Sometimes, users don’t want to participate in the testing process.
* The feedback for the testing takes long time as it involves many users and the opinions may differ from one user to another user.
* Development team is not participated in this testing process.

# Regression Testing – Software Engineering

**Regression Testing**

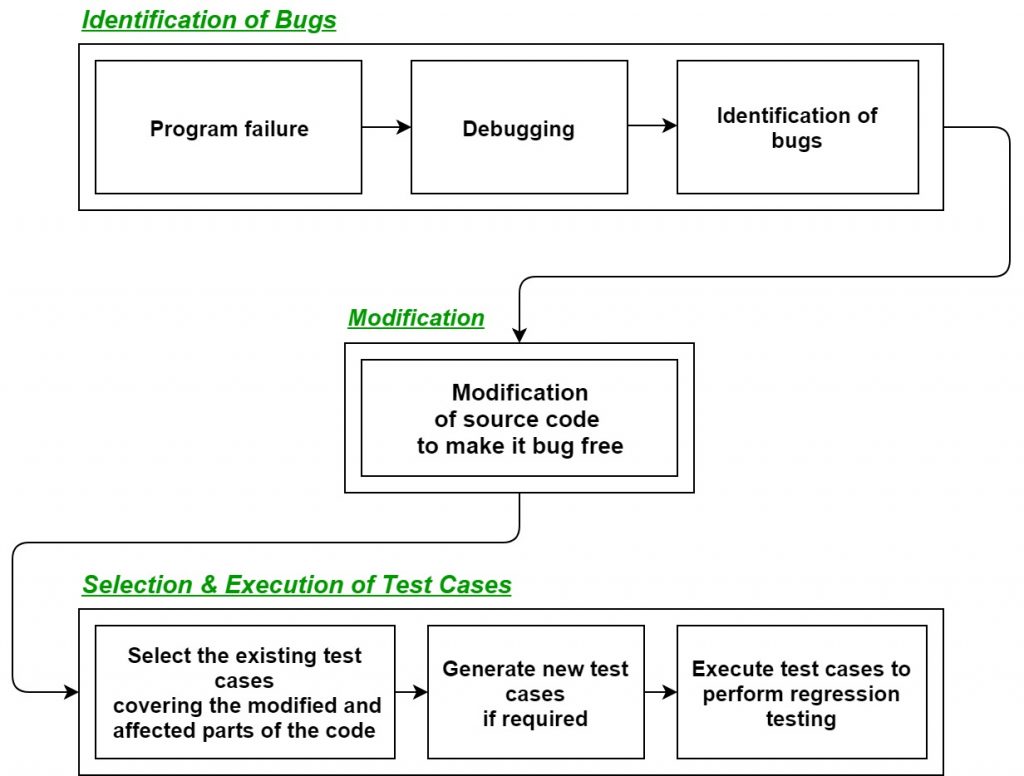
is the process of testing the modified parts of the code and the parts that might get affected due to the modifications to ensure that no new errors have been introduced in the software after the modifications have been made. Regression means the return of something and in the software field, it refers to the return of a bug.

**When to do regression testing?**

* When a new functionality is added to the system and the code has been modified to absorb and integrate that functionality with the existing code.
* When some defect has been identified in the software and the code is debugged to fix it.
* When the code is modified to optimize its working.

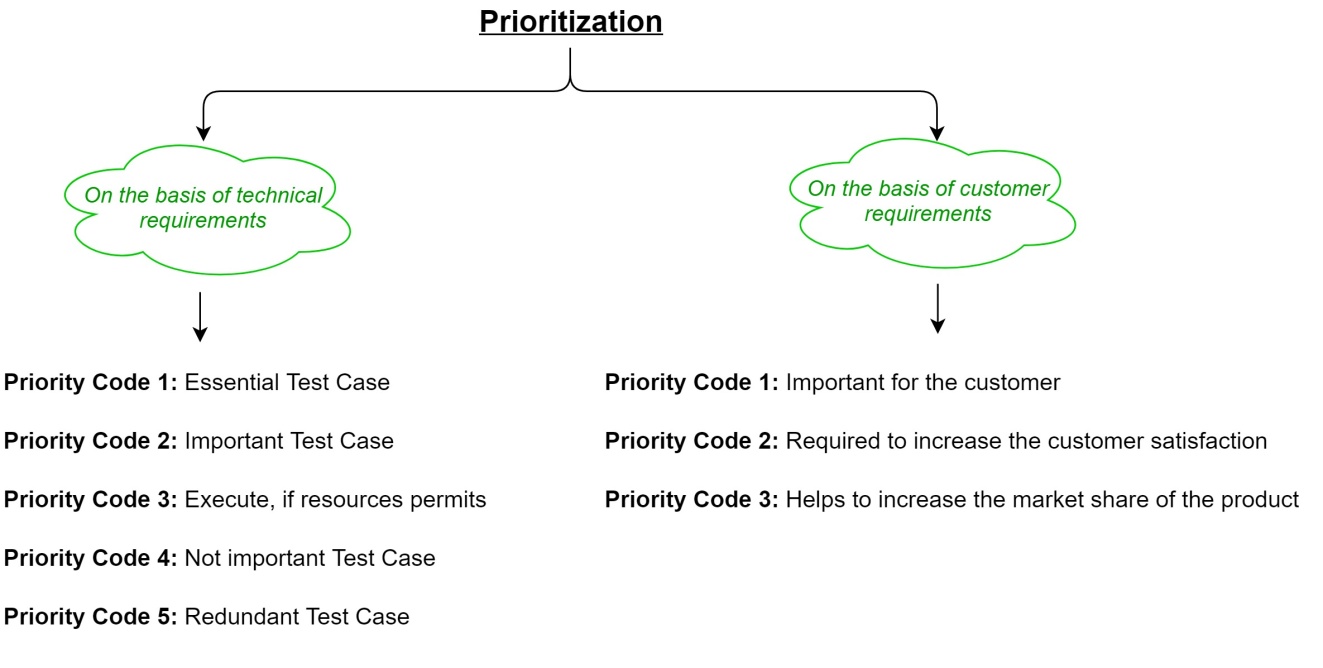
**Process of Regression testing:**

Firstly, whenever we make some changes to the source code for any reason like adding new functionality, optimization, etc. then our program when executed fails in the previously designed test suite for obvious reasons. After the failure, the source code is debugged in order to identify the bugs in the program. After identification of the bugs in the source code, appropriate modifications are made. Then appropriate test cases are selected from the already existing test suite which covers all the modified and affected parts of the source code. We can add new test cases if required. In the end, regression testing is performed using the selected test cases.



**Techniques for the selection of Test cases for Regression Testing:**

* **Select all test cases:**In this technique, all the test cases are selected from the already existing test suite. It is the simplest and safest technique but not much efficient.
* **Select test cases randomly:**In this technique, test cases are selected randomly from the existing test-suite, but it is only useful if all the test cases are equally good in their fault detection capability which is very rare. Hence, it is not used in most of the cases.
* **Select modification traversing test cases:**In this technique, only those test cases are selected which covers and tests the modified portions of the source code the parts which are affected by these modifications.
* **Select higher priority test cases:**In this technique, priority codes are assigned to each test case of the test suite based upon their bug detection capability, customer requirements, etc. After assigning the priority codes, test cases with the highest priorities are selected for the process of regression testing. The test case with the highest priority has the highest rank. For example, test case with priority code 2 is less important than test case with priority code 1.



* **Tools for regression testing:**

In regression testing, we generally select the test cases from the existing test suite itself and hence, we need not compute their expected output, and it can be easily automated due to this reason. Automating the process of regression testing will be very much effective and time saving. Most commonly used tools for regression testing are:

* Selenium
* WATIR (Web Application Testing In Ruby)
* QTP (Quick Test Professional)
* RFT (Rational Functional Tester)
* Winrunner
* Silktest

**Advantages of Regression Testing:**

* It ensures that no new bugs has been introduced after adding new functionalities to the system.
* As most of the test cases used in Regression Testing are selected from the existing test suite, and we already know their expected outputs. Hence, it can be easily automated by the automated tools.
* It helps to maintain the quality of the source code.

**Disadvantages of Regression Testing:**

* It can be time and resource consuming if automated tools are not used.
* It is required even after very small changes in the code.

**What Is Functional Testing?**

[Functional testing](https://www.bairesdev.com/software-development/functional-testing-services/) ensures a software application meets functional specifications based on project requirements. It’s conducted by testers who perform each software function, including the following items:

* **Login:** Observes the login process to see if users are able to get into the application with no problems
* **Primary Functions:** Makes sure the application does what it’s supposed to do
* **User Interface:** Ensures the screens are easy and intuitive to navigate and that each button or link takes the user to an expected location
* **Errors:** Determines what conditions must occur to produce an error and whether appropriate error messages appear
* **Payment:** Examines the stability of the payment system, to ensure users are able to make payments seamlessly

Functional testing can be done manually or automatically and incorporates several different types, including:

* **Unit Testing:** Ensures each component (or “unit”) of the application works as expected
* **Smoke Testing:** Determines if each function operates as it should
* **Sanity Testing:** Checks to be sure minor modifications have fixed the issues they were meant to resolve
* **Regression Testing:**  Makes certain updates don’t cause additional problems
* **System Testing:** A high-level test that shows whether the system meets technical, functional, and business requirements
* **User Acceptance Testing:** Typically performed last in a series of tests, it ensures the application can handle “real-world” scenarios

**What Is Performance Testing?**

The aim of [performance testing](https://www.bairesdev.com/blog/software-load-performance-testing-services/) is to eliminate any impediments to smooth performance. Whereas functional testing looks at the features of the application, performance testing checks the software as a system to make sure it can handle the expected amount of traffic, respond quickly to user requests, and operate at appropriate speeds. The system is tested under a variety of conditions to ensure it can handle different scenarios.

This type of testing is just as important as functional testing because it can have just as big an impact on companies and their customers. For example, a web application that doesn’t load quickly is one customers will quickly abandon. A customer relationship management (CRM) program that crashes when too many people are using it is one that employees will avoid.

Firms that offer performance testing services provide several specific types of testing:

* **Load Testing:** Checks to see how big a traffic load the system can handle
* **Endurance Testing:** Examines the load over a period of time to ensure ongoing stability
* **Volume Testing:** Determines the efficiency of the application when loaded with a large volume of data
* **Scalability Testing:** Makes sure the software can deal with increasing numbers of users or data
* **Stress Testing:** Determines how well the hardware resources, such as CPU, memory, and disk space, can support the software

**1. Top Down Integration Testing :** Top Down Integration testing which is also known as Incremental integration testing. In this Top Down approach the higher level modules are tested first after higher level modules the lower level modules are tested. Then these modules undergo for integration accordingly. Here the higher level modules refers to main module and lower level modules refers to submodules. This approach uses Stubs which are mainly used to simulate the submodule, if the invoked submodule is not developed this Stub works as a momentary replacement. 

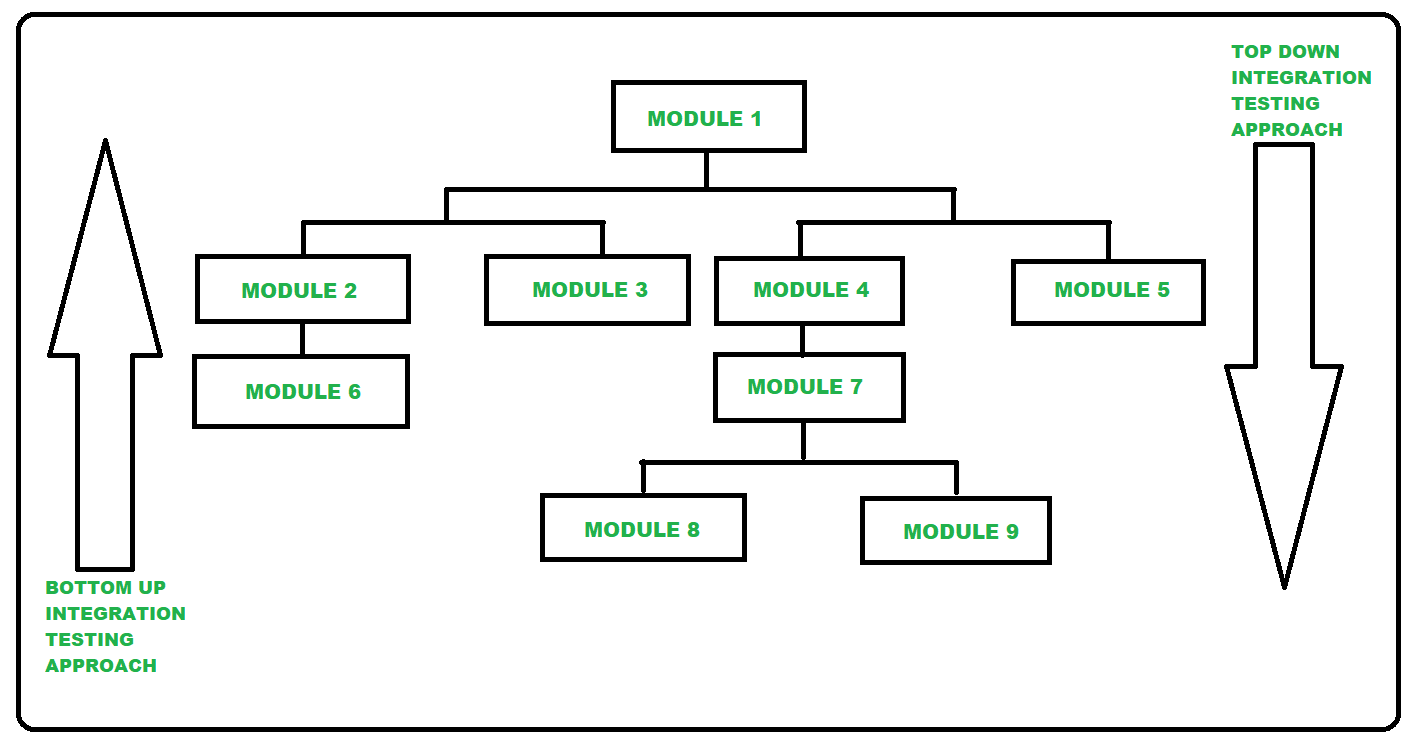
**Features of Top Down Integration Testing :**

* **System-centric approach:** Testing focuses on the system as a whole, starting from top-level components.
* **Early testing of higher-level functionality:** Testing higher-level functionality is prioritized to identify major issues early.
* **High-level test cases used:**Test cases are designed to cover the functionality and interactions of higher-level modules.
* **Mock objects/stubs used:** Placeholder objects or stubs simulate lower-level components during testing.
* **Testing becomes progressively detailed:** More specific tests are performed as lower-level components are integrated.
* **Efficient approach:**Testing higher-level functionality first allows for a quick evaluation of system behavior.
* **Risk-based testing:**Critical and risky areas of the system are tested early to mitigate potential risks.

**2. Bottom Up Integration Testing :** Bottom Up Integration testing is another approach of Integration testing. In this Bottom Up approach the lower level modules are tested first after lower level modules the higher level modules are tested. Then these modules undergo for integration accordingly. Here the lower level modules refers to submodules and higher level modules refers to main modules. This approach uses test drivers which are mainly used to initiate and pass the required data to the sub modules means from higher level module to lower level module if required. The below **figure** represents the Top Down and Bottom up Integration testing approach.

**Features of Bottom Up Integration Testing :**

* **Component-Centric:** Bottom-up integration testing focuses on testing individual components first before integrating them into larger subsystems.
* **Early Detection:**This approach enables early detection of defects in individual components, which can be corrected before integration.
* **Independent Testing:** Each component is tested independently, ensuring that it functions correctly in isolation.
* **Incremental Testing:** Testing proceeds incrementally from lower-level to higher-level components.
* **Parallel Development:** This approach enables parallel development of components by different teams.
* **Efficient Testing:** Bottom-up integration testing can be an efficient approach.
* **Risk-Based Testing:** Risk-based testing can be applied in bottom-up integration testing to prioritize the testing of high-risk components.

   
**Difference between Top Down Integration Testing and Bottom Up Integration Testing :**

| **S.No.** | **TOP DOWN INTEGRATION TESTING** | **BOTTOM UP INTEGRATION TESTING** |
| --- | --- | --- |
| **01.** | **Top Down Integration testing is one of the approach of Integration testing in which integration testing takes place from top to bottom means system integration begins with top level modules.** | **Bottom Up Integration testing is one of the approach of Integration testing in which integration testing takes place from bottom to top means system integration begins with lowest level modules.** |
| **02.** | **In this testing the higher level modules are tested first then the lower level modules are tested and then the modules are integrated accordingly.** | **In this testing the lower level modules are tested first then the higher level modules are tested and then the modules are integrated accordingly.** |
| **03.** | **In this testing stubs are used for simulate the submodule if the invoked submodule is not developed means Stub works as a momentary replacement.** | **In this testing drivers are used for simulate the main module if the main module is not developed means Driver works as a momentary replacement.** |
| **04.** | **Top Down Integration testing approach is beneficial if the significant defect occurs toward the top of the program.** | **Bottom Up Integration testing approach is beneficial if the crucial flaws encounters towards the bottom of the program.** |
| **05.** | **In Top Down Integration testing approach the main module is designed at first then the submodules/subroutines are called from it.** | **In Bottom Up Integration testing approach different modules are created first then these modules are integrated with the main function.** |
| **06.** | **It is implemented on Structure/procedure-oriented programming languages.** | **It is implemented on Object-oriented programming languages.** |
| **07.** | **The complexity of this testing is simple.** | **The complexity of this testing is complex and highly data intensive.** |
| **08.** | **It works on big to small components.** | **It works on small to big components.** |
| **09.** | **In this approach Stub modules must be produced.** | **In this approach, Driver modules must be produced.** |
| **10.** | **In terms of cost, Top Down testing is more expensive because it requires the complete system for testing.** | **Bottom Up testing is less expensive as compared to Top Down because it allows early identification and resolution of the model issues.** |

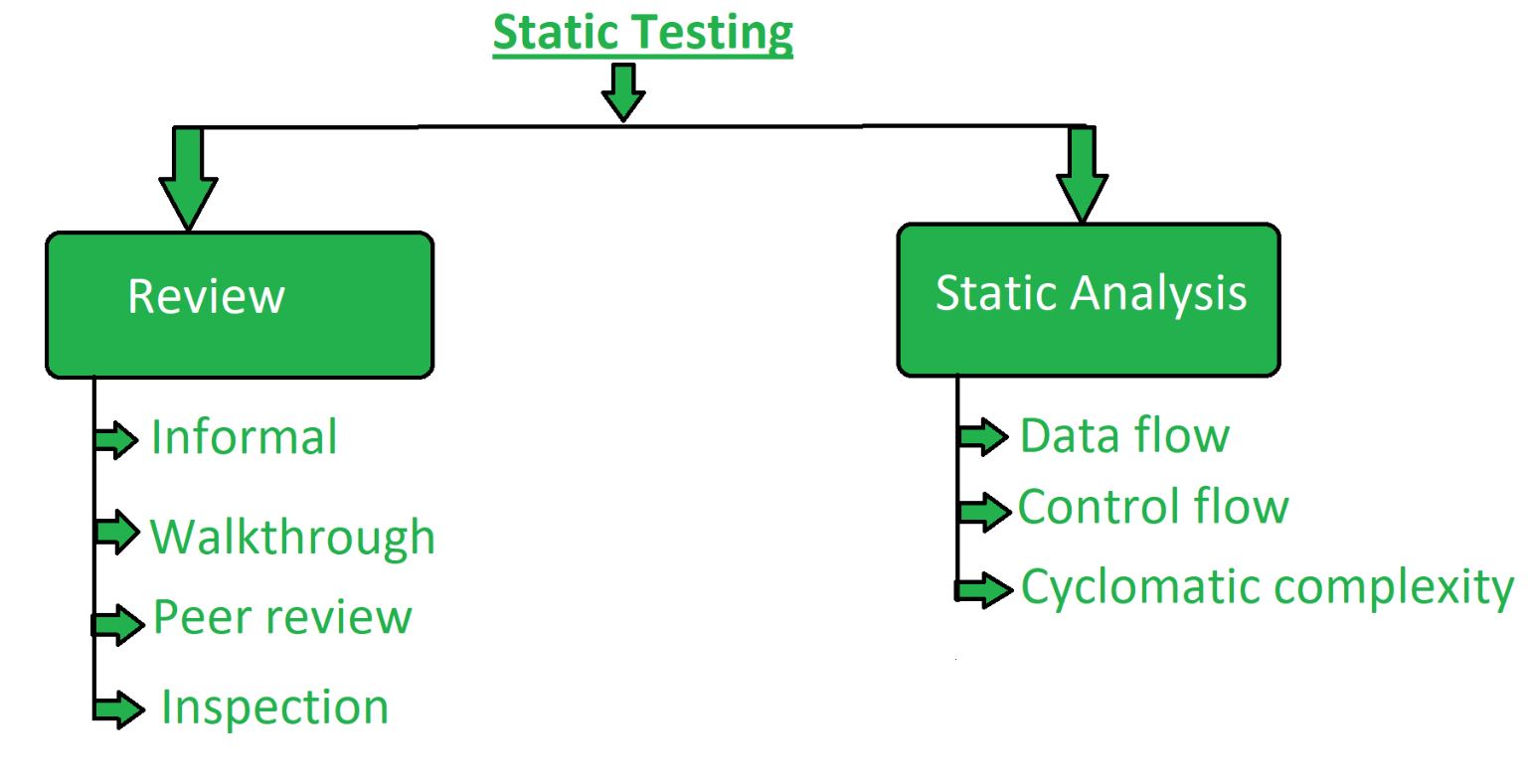
Alpha Testing is a type of software testing performed to identify bugs before releasing the product to real users or to the public. Alpha Testing is one of the user acceptance tests.

Beta Testing is performed by real users of the software application in a real environment. Beta testing is one type of User Acceptance Testing.

**Difference between Alpha and Beta Testing:**

The difference between Alpha and Beta Testing is as follows:

| **Alpha Testing** | **Beta Testing** |
| --- | --- |
| [Alpha testing](https://www.geeksforgeeks.org/alpha-testing-software-testing/) involves both the white box and black box testing. | [Beta testing](https://www.geeksforgeeks.org/beta-testing-software-testing/) commonly uses black-box testing. |
| Alpha testing is performed by testers who are usually internal employees of the organization. | Beta testing is performed by clients who are not part of the organization. |
| Alpha testing is performed at the developer’s site. | Beta testing is performed at the end-user of the product. |
| Reliability and security testing are not checked in alpha testing. | Reliability, security and robustness are checked during beta testing. |
| Alpha testing ensures the quality of the product before forwarding to beta testing. | Beta testing also concentrates on the quality of the product but collects users input on the product and ensures that the product is ready for real time users. |
| Alpha testing requires a testing environment or a lab. | Beta testing doesn’t require a testing environment or lab. |
| Alpha testing may require a long execution cycle. | Beta testing requires only a few weeks of execution. |
| Developers can immediately address the critical issues or fixes in alpha testing. | Most of the issues or feedback collected from the beta testing will be implemented in future versions of the product. |
| Multiple test cycles are organized in alpha testing. | Only one or two test cycles are there in beta testing. |

**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. Whereas in Dynamic Testing checks, the code is executed to detect the defects. 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 cannot 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. **Review:** In static testing review is a process or technique that is performed to find the potential defects in the design of the software. It is process to detect and remove errors and defects in the different supporting documents like software requirements specifications. People examine the documents and sorted out errors, redundancies and ambiguities. Review is of four types:

* **Informal:** In informal review the creator of the documents put the contents in front of audience and everyone gives their opinion and thus defects are identified in the early stage.
* **Walkthrough:** It is basically performed by experienced person or expert to check the defects so that there might not be problem further in the development or testing phase.
* **Peer review:** Peer review means checking documents of one-another to detect and fix the defects. It is basically done in a team of colleagues.
* **Inspection:** Inspection is basically the verification of document the higher authority like the verification of software requirement specifications (SRS).

2. **Static Analysis:** Static Analysis includes the evaluation of the code quality that is written by developers. Different tools are used to do the analysis of the code and comparison of the same with the standard. It also helps in following identification of following defects:

**(a)** Unused variables

**(b)** Dead code

**(c)** Infinite loops

**(d)** Variable with undefined value

**(e)** Wrong syntax

Static Analysis is of three types:

* **Data Flow:** Data flow is related to the stream processing.
* **Control Flow:** Control flow is basically how the statements or instructions are executed.
* **Cyclomatic Complexity:** Cyclomatic complexity defines the number of independent paths in the control flow graph made from the code or flowchart so that minimum number of test cases can be designed for each independent path.