UNIT II

USING BLACK BOX APPROACHES TO TEST CASE DESIGN

The black box is a powerful technique to check the application under test from the user’s perspective. Black box testing is used to test the system against external factors responsible for software failures. This testing approach focuses on the input that goes into the software, and the output that is produced. The testing team does not cover the inside details such as code, server logic, and development method.

Black box testing is based on the requirements and checks the system to validate against predefined requirements.

Various parameters checked in black box testing are:

Accurate actions performed by users

System’s interaction with the inputs

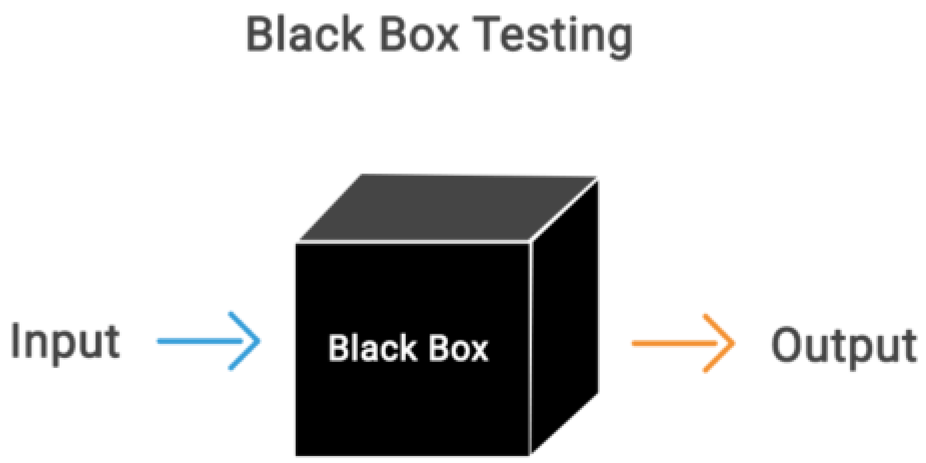
The response time of the system

Use of data structures Issues in the user interface

Usability issues

Performance issues

Abrupt application failure, unable to start or finish



Types of Black Box Testing

There are many different types of Black Box Testing, some of them are given below:

Functional testing – This is a type of black box testing which is related to the functional requirements of a system; Functional testing is concerned only with the functional requirements of a system and covers how well the system executes its functions.

Non-functional testing – This black box testing type is not related to testing of specific functionality, Non functional testing is concerned with the non-functional requirements and is designed specifically to evaluate the readiness of a system according to the various criteria which are not covered by functional testing.

Regression testing – Regression Testing is performed after code fixes, upgrades or any other system maintenance to check the new changes has not affected any existing functionality.

When we do Black Box testing?

Unlike traditional white box testing, black box testing is beneficial for testing software usability.

The overall functionality of the system under test

- Black box testing gives you a broader picture of the software.
- This testing approach sees an application from a user’s perspective.
- To test the software as a whole system rather than different modules.

Various approaches to black-box testing

There are a set of approaches for black-box testing.

Manual UI Testing: In this approach, a tester checks the system as a user. Check and verify the user data, error messages.

Automated UI Testing: In this approach, user interaction with the system is recorded to find errors and glitches. Testers can set record demand as per schedule.

Documentation Testing: In this approach, a tester purely checks the input and output of the software. Testers consider what system should perform rather than how. It is a manual approach to testing.

What are Black Box testing techniques?

There are various test case design techniques applied for black-box testing:

- Boundary Value Analysis
- Equivalence partitioning
- State Transition Testing
- Decision Table Testing
- Graph-Based Testing
- Error Guessing Technique

1- Boundary Value Analysis

It is the widely used black-box testing, which is also the basis for equivalence testing. Boundary value analysis tests the software with test cases with extreme values of test data. BVA is used to identify the flaws or errors that arise due to the limits of input data.

For example: Taking inputs for a test case data for an age section should accept a valid data of anything between 1-100. According to BVP analysis, the software will be tested against four test data as -1, 1, 100, and 101 to check the system's response using the boundary values.

2- Equivalence partitioning

This test case designing techniques checks the input and output by dividing the input into equivalent classes. The data must be tested at least once to ensure maximum test coverage of data. It is the exhaustive form of testing, which also reduces the redundancy of inputs.

For example: Taking inputs for a test case data for the example mentioned above will have three classes from which one data will be tested.

Valid class: 1 to 100 (any number), Invalid class: -1 (checking the lowest of lowest), Invalid class: 101(highest of highest).

[Also Read: What is Boundary Value Analysis and Equivalence Partitioning?]

3- State Transition Testing

This testing technique uses the inputs, outputs, and the state of the system during the testing phase. It checks the software against the sequence of transitions or events among the test data.

Based on the type of software that is tested, it checks for the behavioral changes of a system in a particular state or another state while maintaining the same inputs.

For example, A login page will let you input username and password until three attempts. Each incorrect password will be sent the user to the login page. After the third attempt, the user will be sent to an error page. This state transition method considers the various states of the system and the inputs to pass only the right sequence of the testing.

4- Decision Table Testing

This approach creates test cases based on various possibilities. It considers multiple test cases in a decision table format where each condition is checked and fulfilled, to pass the test and provide accurate output. It is preferred in case of various input combinations and multiple possibilities.

For example, A food delivery application will check various payment modes as input to place the order — decision making based on the table.

Case1: If the end-user has a card, then the system will not check for cash or coupon and will take action to place the order.

Case2: If the end-user has a coupon will not be checked for a card or cash and action will be taken.

Case3: if the end-user has cash, the action will be taken.

Case4: If the end-user doesn’t have anything, then action will not be taken.

5- Graph-Based Testing:

It is similar to a decision-based test case design approach where the relationship between links and input cases are considered.

6- Error Guessing Technique:

This method of designing test cases is about guessing the output and input to fix any errors that might be present in the system. It depends on the skills and judgment of the tester.

Comparison testing

This method uses the two different versions of the same software to compare and validate the results.

How to do Black Box testing?

When you get the basic understanding of black-box testing then the next question which comes up in mind is: How to perform the Black box testing? Below you can check the steps to perform this testing:

- The first step to black-box testing is to understand the requirement specifications of the application under test. An accurate and precise SRS document should be there.
- The next step is to evaluate the set of valid inputs and test scenarios to test the software. The goal is to save time and get good test coverage.
- Prepare the test cases to cover a maximum range of inputs.
- The test cases are run in the system to generate output, which is validated with the expected outcome to mark pass or fail.
- The failed steps are marked and sent to the development team to fix them.
- Retest the system using various testing techniques to verify its recurring nature or to pass it.

The black box testing can be easily used to check and validate the entire software development life cycle. It can be used at various stages such as unit, integration, acceptance, system, and regression to evaluate the product.

What are the benefits of Black Box testing?

- The tester doesn't need any technical knowledge to test the system. It is essential to understand the user's perspective.
- Testing is performed after development, and both the activities are independent of each other.
- It works for a more extensive coverage which is usually missed out by testers as they fail to see the bigger picture of the software.

Test cases can be generated before development and right after specification.

Black box testing methodology is close to agile.

RANDOM TESTING IN SOFTWARE TESTING

Random testing is software testing in which the system is tested with the help of generating random and independent inputs and test cases. Random testing is also named monkey testing. It is a black box assessment outline technique in which the tests are being chosen randomly and the results are being compared by some software identification to check whether the output is correct or incorrect.

Some important points about Random Testing:

1. Random testing was first examined by Melvin Breuer in the year 1971.
2. This testing was initially assessed by Pratima and Vishwani Agrawal in the year 1975 to check the successful output of the software.
3. For random testing, there is also a book that contains some formulas for the number of tests that can be taken and the number of successful results and failure results.

Working Random Testing:

Step-1: Identify Input domain

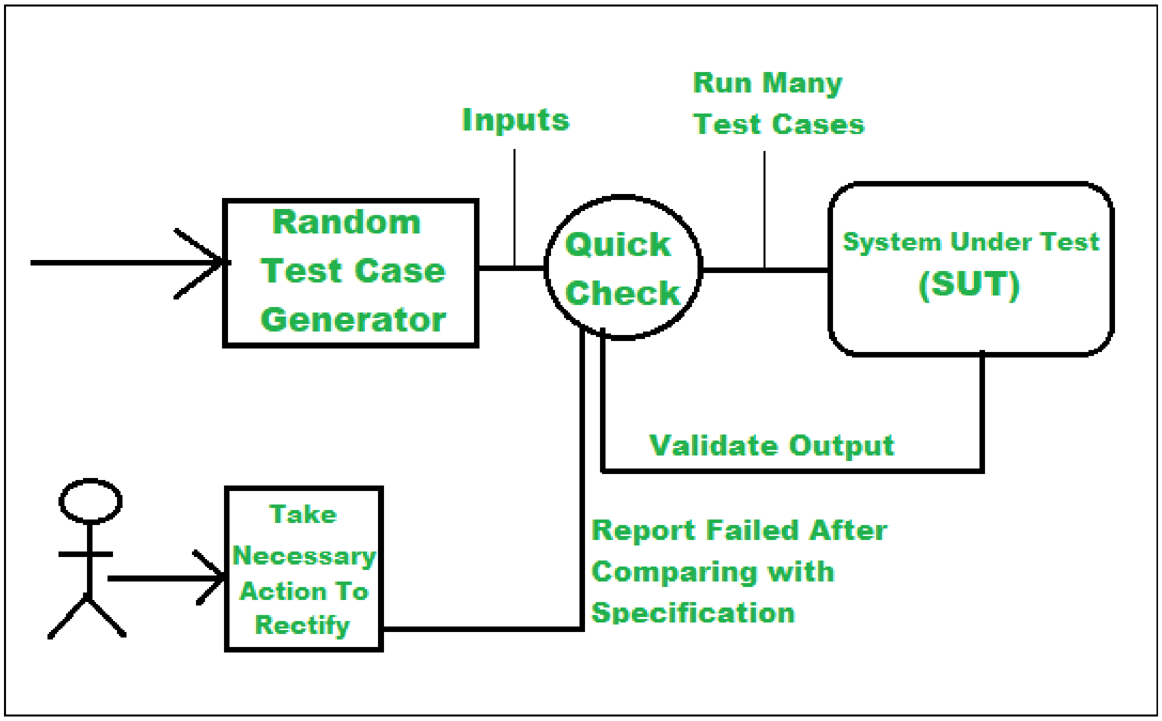
Step-2: Select test inputs independently/randomly from the input domain

Step-3: Test the system on these inputs and form a random test set

Step-4: Compare the result with system specification

Step-5: If the Report fails then take necessary action.

The below image represents the working of Random Testing more clearly.


Inputs
Run Many Test Cases
Random Test Case Generator
Quick Check
System Under Test (SUT)
Validate Output
Take Necessary Action To Rectify
Report Failed After Comparing with Specification

Types of Random Testing

1. Random input sequence generation: It is also known as Random Number Generator (RNG) in which a random sequential number or symbols is being generated which cannot be assumed during the random selection.

2. Random sequence of data inputs: In this, all the data are selected randomly for the inputs which can be used during the testing.

3. Random data selection from an existing database: The record where all the data are available from that record only the data can be selected for testing afterward no additional data cannot be added which are not available in the record.

Characteristics of Random Testing:

1. Random testing is implemented when the bug in an application is not recognized.
2. It is used to check the system's execution and dependability.
3. It saves our time and does not need any extra effort.
4. Random testing is less costly, it doesn't need extra knowledge for testing the program.

Methods to Implement Random Testing:

To implement the random testing basically, four steps are applied:

1. The user input domain is analyzed.
2. After that, from that domain, the data of test inputs are chosen separately.
3. With the help of these test inputs, the test is executed successfully. These input tests conduct random sets of tests.
4. The outcomes are compared with the system identification. The outcome of the test becomes unsuccessful if any test input doesn't

match with the original one otherwise the outcomes are always successful.

Advantages of Random Testing

1. It is very cheap so that anyone can use this software.
2. It doesn’t need any special intelligence to access the program during the tests.
3. Errors can be traced very easily; it can easily detect the bug throughout the testing.
4. This software is lacking bias means it makes the groups evenly for the testing and it prefers not to repeatedly check the errors as there can be some changes in the codes throughout the testing process.

Disadvantages of Random Testing

1. This software only finds changes errors.
2. They are not practical. Some tests will be of no use for a longer time.
3. Most of the time is consumed by analyzing all the tests.
4. New tests cannot be formed if their data is not available during testing.

Tools used for Random Testing

1. **QuickCheck:** It is a famous test tool, which was introduced for Haskell which is available in many different languages. This tool generates

random orders for API calls that are related to the model and the properties of the system which can give successful results after every test.

2. **Randoop:** This tool provides an order of methods and constructor acknowledgment for the classes during the test and generates JUnit tests.
3. **Simulant:** It is a Clojure tool that runs according to the system's different specifications, the behavior of the model.
4. **Gram Test:** This random testing tool is based on grammar which is written in Java, It utilizes BNF notation for specifying the grammar which is used as input during testing.

What is Requirements based Testing?

Requirements-based testing is a testing approach in which test cases, conditions and data are derived from requirements. It includes functional tests and also non-functional attributes such as performance, reliability or usability.

Stages in Requirements based Testing:

Defining Test Completion Criteria - Testing is completed only when all the functional and non-functional testing is complete.
Design Test Cases - A Test case has five parameters namely the initial state or precondition, data setup, the inputs, expected outcomes and actual outcomes.

Execute Tests - Execute the test cases against the system under test and document the results.

Verify Test Results - Verify if the expected and actual results match each other.

Verify Test Coverage - Verify if the tests cover both functional and non-functional aspects of the requirement.

Track and Manage Defects - Any defects detected during the testing process goes through the defect life cycle and are tracked to resolution. Defect Statistics are maintained which will give us the overall status of the project.

Requirements Testing process:

Testing must be carried out in a timely manner.

Testing process should add value to the software life cycle, hence it needs to be effective.

Testing the system exhaustively is impossible hence the testing process needs to be efficient as well.

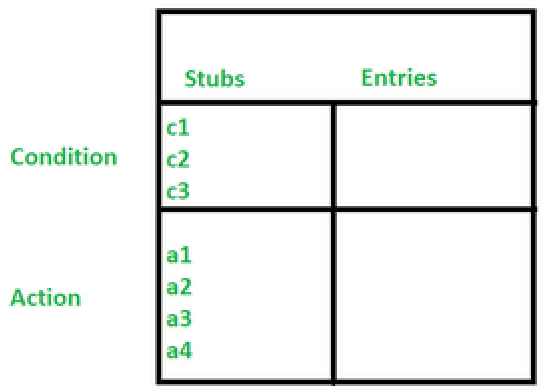
Testing must provide the overall status of the project, hence it should be manageable.

Decision Table Based Testing in Software Testing

Decision tables are used in various engineering fields to represent complex logical relationships. This testing is a very effective tool in testing the software and its requirements management. The output may be dependent on many input conditions and decision tables give a tabular view of various combinations of input conditions and these conditions are in the form of True(T) and False(F). Also, it provides a set of conditions and its corresponding actions required in the testing.

Parts of Decision Tables :

In **software testing**, the decision table has 4 parts which are divided into portions and are given below :



1. **Condition Stubs :** The conditions are listed in this first upper left part of the decision table that is used to determine a particular action or set of actions.
2. **Action Stubs :** All the possible actions are given in the first lower left portion (i.e, below condition stub) of the decision table.

3. **Condition Entries :** In the condition entry, the values are inputted in the upper right portion of the decision table. In the condition entries part of the table, there are multiple rows and columns which are known as Rule.
4. **Action Entries :** In the action entry, every entry has some associated action or set of actions in the lower right portion of the decision table and these values are called outputs.

Types of Decision Tables :

The decision tables are categorized into two types and these are given below:

1. **Limited Entry :** In the limited entry decision tables, the condition entries are restricted to binary values.
2. **Extended Entry :** In the extended entry decision table, the condition entries have more than two values. The decision tables use multiple conditions where a condition may have many possibilities instead of only ‘true’ and ‘false’ are known as extended entry decision tables.

Applicability of Decision Tables :

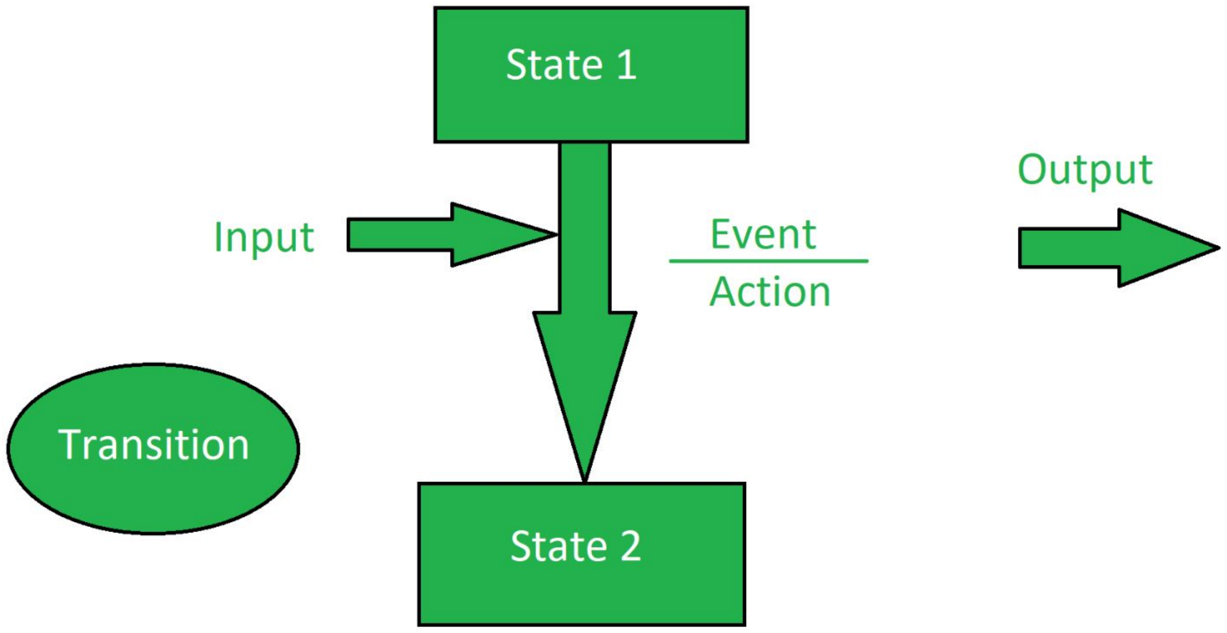
- The order of rule evaluation has no effect on the resulting action.
- The decision tables can be applied easily at the unit level only.
- Once a rule is satisfied and the action selected, n another rule needs to be examined.
- The restrictions do not eliminate many applications.

State Transition Testing

State Transition Testing is a type of software testing which is performed to check the change in the state of the application under varying input. The condition of input passed is changed and the change in state is observed.

State Transition Testing is basically a black box testing technique that is carried out to observe the behavior of the system or application for different input conditions passed in a sequence. In this type of testing, both positive and negative input values are provided and the behavior of the system is observed.

State Transition Testing is basically used where different system transitions are needed to be tested.



Objectives of State Transition Testing:

The objective of State Transition testing is:

- To test the behavior of the system under varying input.
- To test the dependency on the values in the past.
- To test the change in transition state of the application.
- To test the performance of the system.

Transition States:

- **Change Mode:**

  When this mode is activated then the display mode moves from TIME to DATE.
- **Reset:**

  When the display mode is TIME or DATE, then reset mode sets them to ALTER TIME or ALTER DATE respectively.
- **Time Set:**

  When this mode is activated, display mode changes from ALTER TIME to TIME.
- **Date Set:**

  When this mode is activated, display mode changes from ALTER DATE to DATE.

State Transition Diagram:

State Transition Diagram shows how the state of the system changes on certain inputs.

It has four main components:

1. States
2. Transition
3. Events
4. Actions

Advantages of State Transition Testing:

- State transition testing helps in understanding the behavior of the system.
- State transition testing gives the proper representation of the system behavior.
- State transition testing covers all the conditions.

Disadvantages of State Transition Testing:

- State transition testing can not be performed everywhere.

Cause Effect Graphing

Cause Effect Graphing based technique is a technique in which a graph is used to represent the situations of combinations of input conditions. The graph is then converted to a decision table to obtain the test cases. Cause-effect graphing technique is used because boundary value analysis and equivalence class partitioning methods do not consider the combinations of input conditions. But since there may be some critical behaviour to be tested when some combinations of input conditions are considered, that is why cause-effect graphing technique is used.

Steps used in deriving test cases using this technique are:

1. **Division of specification:**

   Since it is difficult to work with cause-effect graphs of large specifications as they are complex, the specifications are divided into small workable pieces and then converted into cause-effect graphs separately.
2. **Identification of cause and effects:**

   This involves identifying the causes(distinct input conditions) and effects(output conditions) in the specification.
3. **Transforming the specifications into a cause-effect graph:**

   The causes and effects are linked together using Boolean expressions to obtain a cause-effect graph. Constraints are also added between causes and effects if possible.
4. **Conversion into decision table:**

   The cause-effect graph is then converted into a limited entry decision table. If you're not aware of the concept of decision tables, check out this link.
5. **Deriving test cases:**

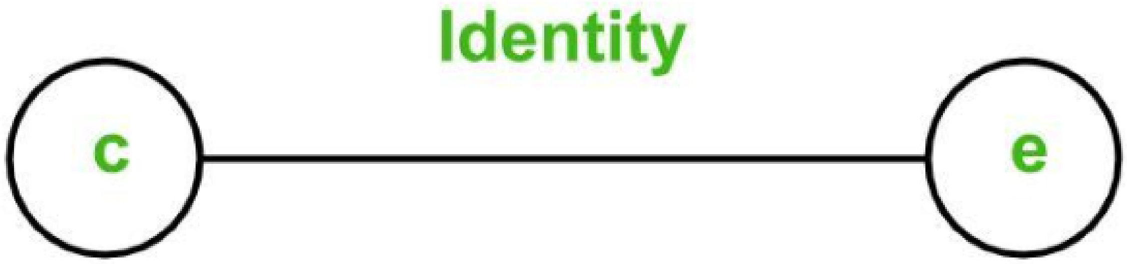
   Each column of the decision-table is converted into a test case.

Basic Notations used in Cause-effect graph:

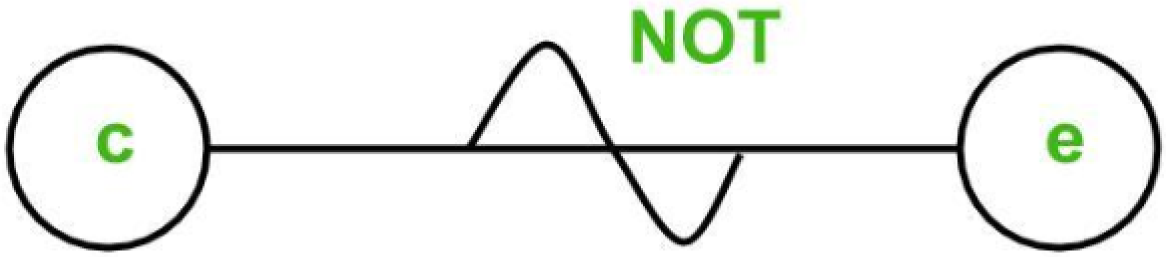
Here **c** represents **cause** and **e** represents **effect**.

The following notations are always **used between a cause and an effect**:

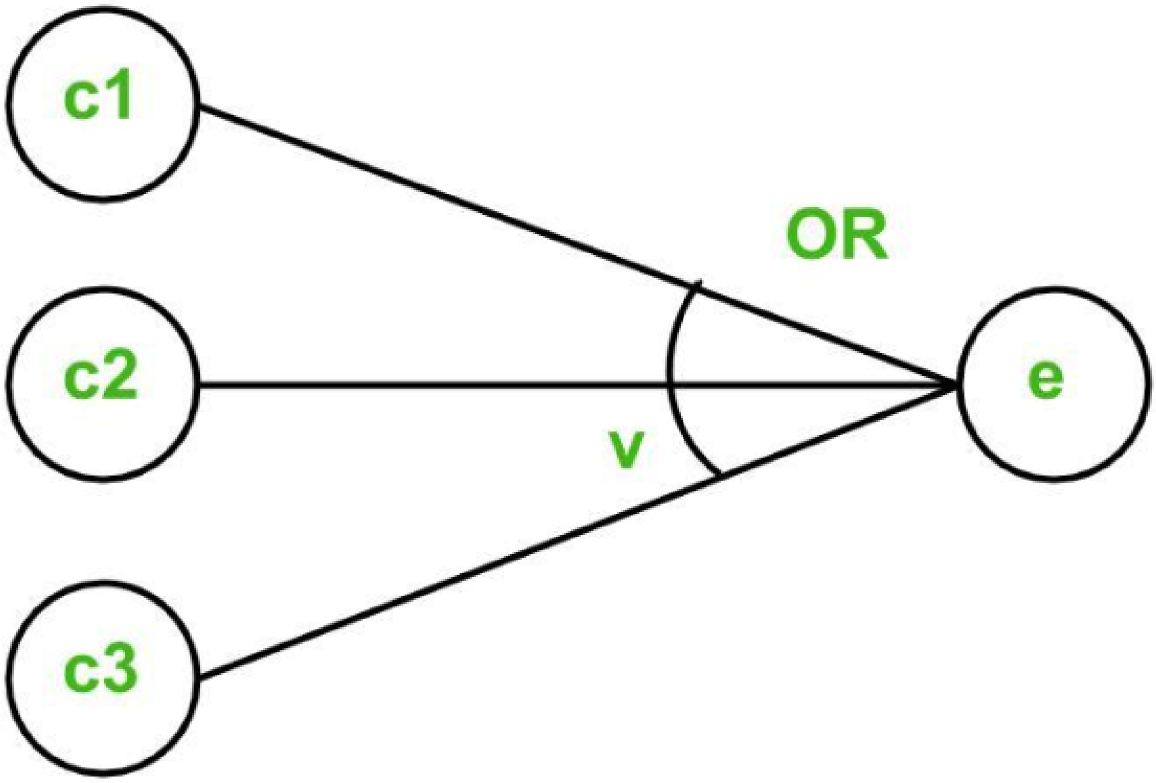
1. **Identity Function:** if c is 1, then e is 1. Else e is 0.



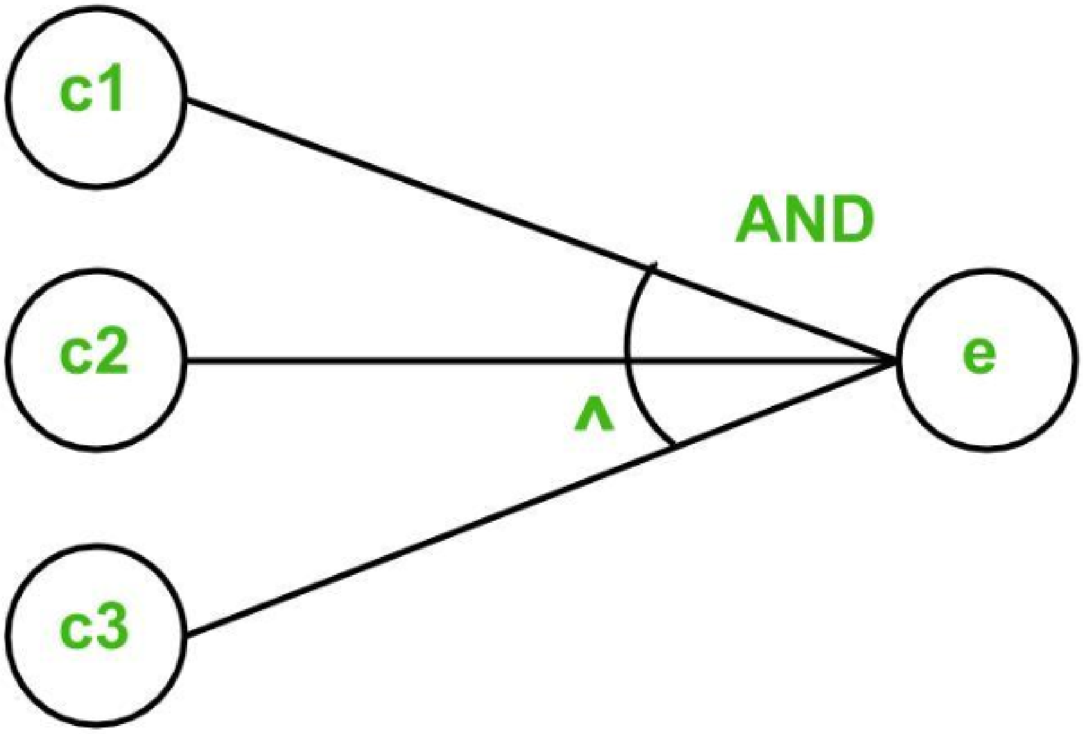
2. **NOT Function:** if c is 1, then e is 0. Else e is 1.



3. **OR Function:** if c1 or c2 or c3 is 1, then e is 1. Else e is 0.



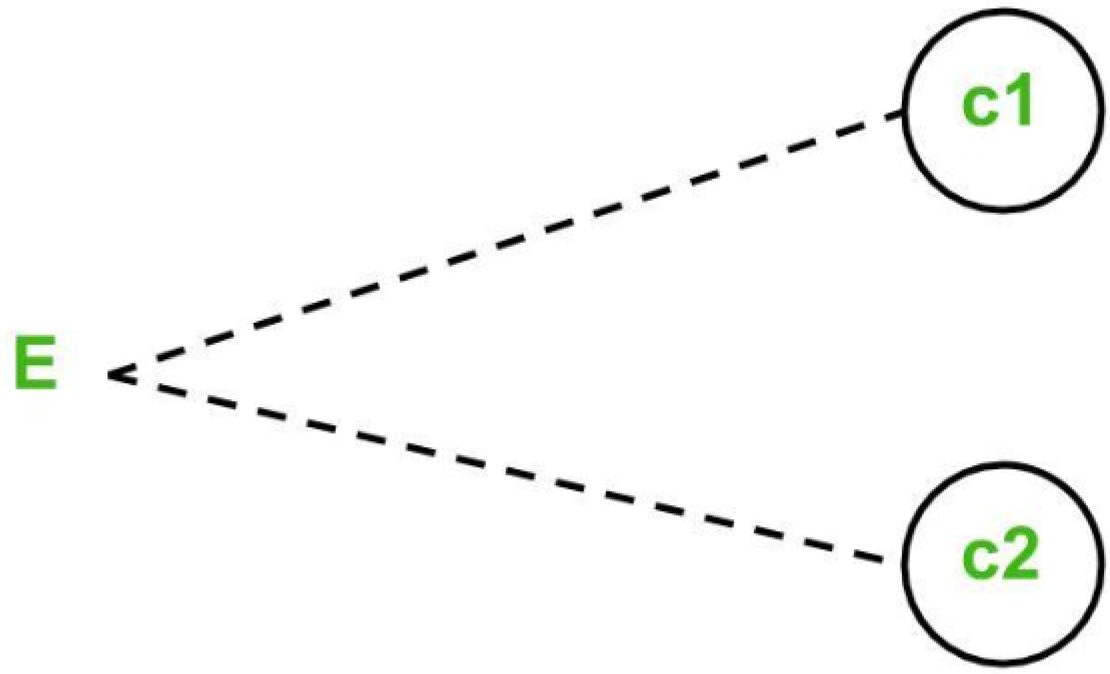
4. **AND Function:** if both c1 and c2 and c3 is 1, then e is 1. Else e is 0.



To represent some impossible combinations of causes or impossible combinations of effects, constraints are used. The following **constraints** are used in cause-effect graphs:

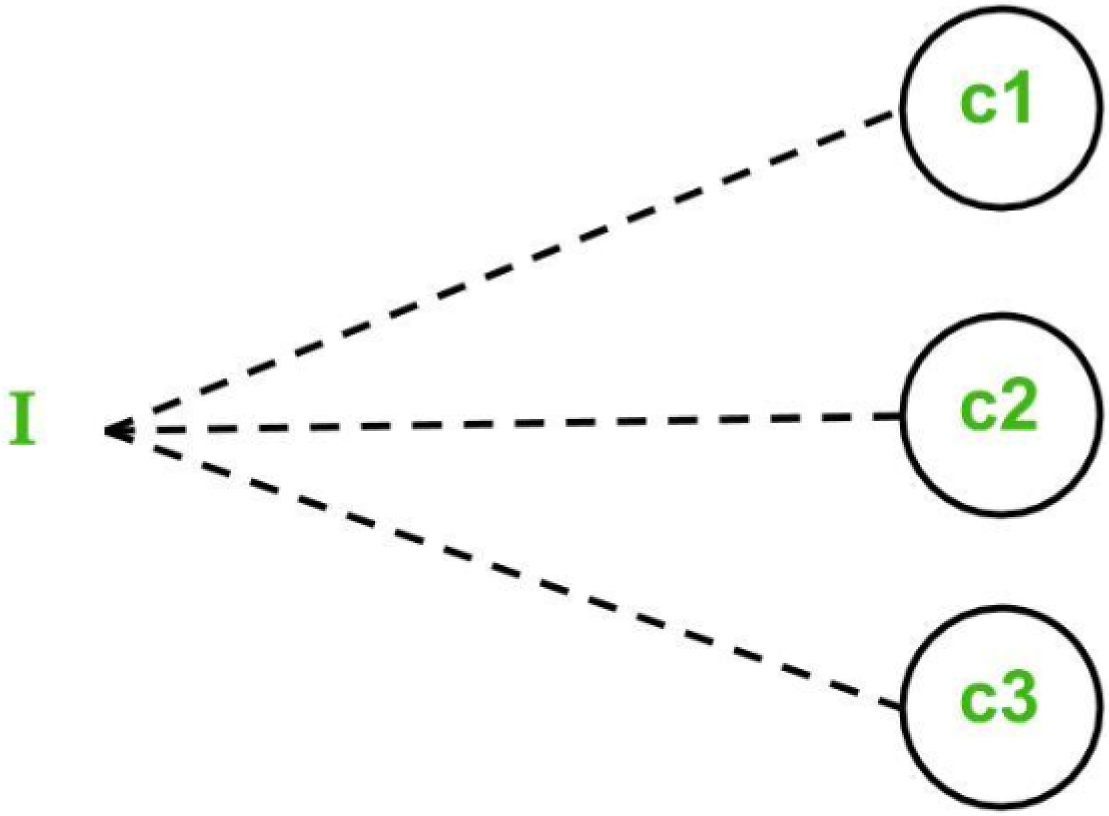
1. **Exclusive constraint** or **E-constraint:** This constraint exists between causes. It states that either c1 or c2 can be 1, i.e., c1 and c2 cannot be

1 simultaneously.



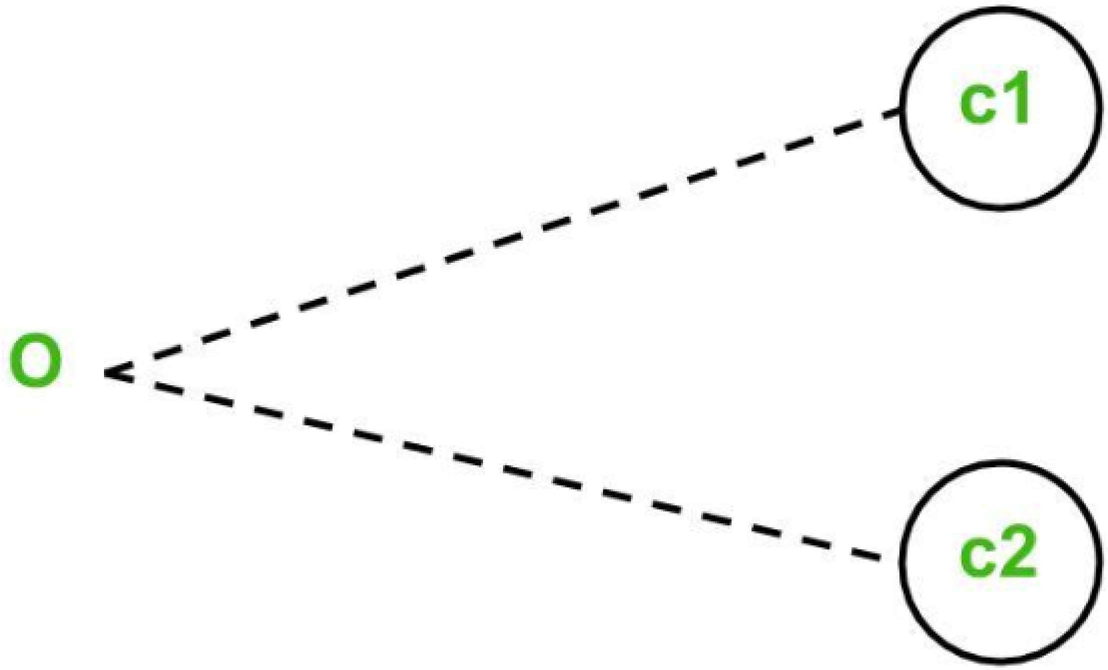
2. **Inclusive constraint** or **I-constraint:** This constraint exists between causes. It states that atleast one of c1, c2 and c3 must always be 1, i.e.,

c1, c2 and c3 cannot be 0 simultaneously.



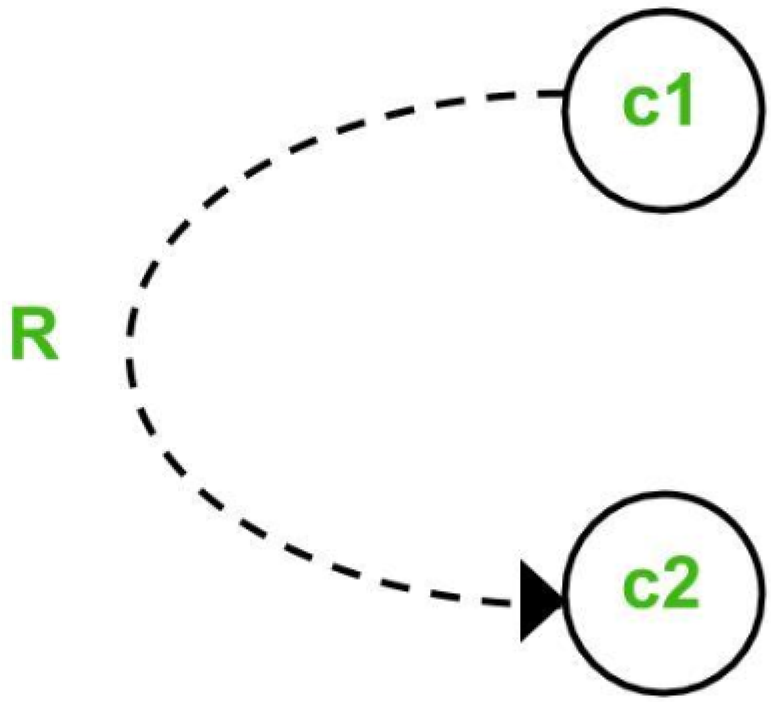
3. **One and Only One constraint** or **O-constraint:** This constraint exists between causes. It states that one and only one of c1 and c2 must be

1.

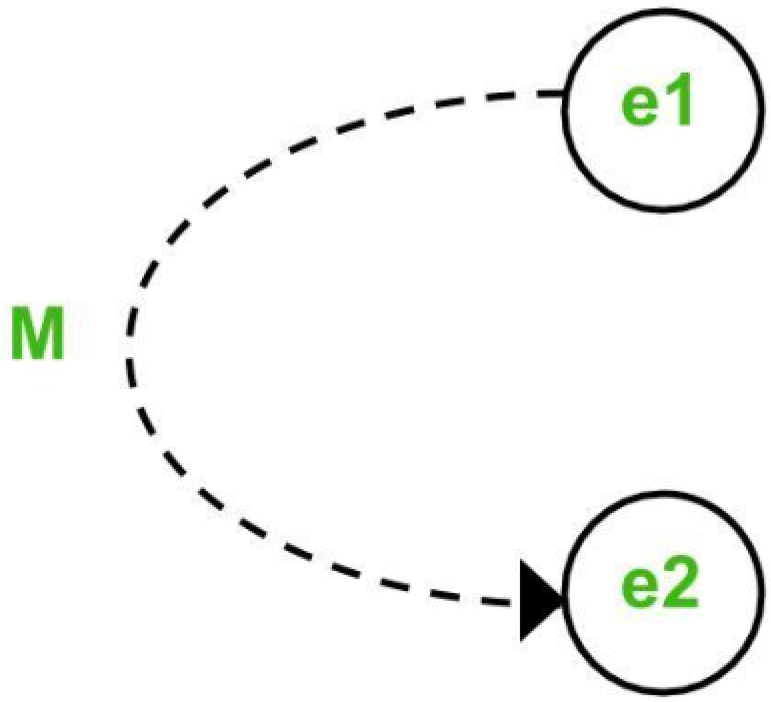


4. **Requires constraint** or **R-constraint:** This constraint exists between causes. It states that for c1 to be 1, c2 must be 1. It is impossible for c1

to be 1 and c2 to be 0.



5. **Mask constraint** or **M-constraint:** This constraint exists between effects. It states that if effect e1 is 1, the effect e2 is forced to be 0.



Error Guessing in Software Testing

Software application is a part of our daily life. May be in laptop or may be in our mobile phone, or it may be any digital device/interface our day starts with the use of various software applications and also ends with the use of various software applications. That's why software companies are also trying their best to develop good quality error free software applications to the users.

So when a company develops any software application software testing plays a major role in that. Testers not only test the product with a set of specified test cases they also test the software by coming out of the testing documents. There the term error guessing comes which is not specified in any testing instruction manual still it is performed. So in this article we will discuss about that error then error guessing, where and how it is performed. The benefits that we get by performing it. So let's start the topic.

Actually an error appears when there is any logical mistake in code by developer. And It's very hard for a developer to find an error in large system. To solve this problem Error guessing technique is used. Error guessing technique is a software technique where test engineer guesses and try to break the software code. Error Guessing technique is also applied to all of the other testing techniques to produce more effective and workable tests.

What is the use of Error Guessing ?

In software testing error guessing is a method in which experience and skill plays an important role. As here possible bugs and defects are guessed in the areas where formal testing would not work. That's why it is also called as experience based testing which has no specific method of testing. This is not a formal way of performing testing still it has importance as it sometimes solves many unresolved issues also.

Where or how to use it ?

Error guessing in software testing approach which is a sort of black box testing technique and also error guessing is best used as a part of the conditions where other black box testing techniques are performed, for instance, boundary value analysis and equivalence split are not prepared to cover all of the condition which are slanted to error in the application.

Advantages and Disadvantages of Error Guessing Technique :

Advantages :

- It is effective when used with other testing approaches.
- It is helpful to solve some complex and problematic gareas of application.
- It figures out errors which may not be identified through other formal ttesting techniques.
- It helps in reducing testing times.

Disadvantages :

- Only capable and skilled tests can perform.
- Dependent on testers experience and skills.
- Fails in providing guarantee the quality standard of the application.
- Not an efficient way of error detection as compared to effort.
- Drawbacks of Error Guessing technique:
- Not sure that the software has reached the expected quality.
- Never provide full coverage of an application.

Factors used in error guessing :

1. Lessons learned from past releases.

2. Experience of testers.
3. Historical learning.
4. Test execution report.
5. Earlier defects.
6. Production tickets.
7. Normal testing rules.
8. Application UI.
9. Previous test results.

Error Guessing is one of the popular techniques of testing, even if it is not an accurate approach of performing testing still it makes the testing work simple and saves a lots of time. But when it is combined with other testing techniques we get better results. In this testing, it is essential to have skilled and experienced testers.

Compatibility Testing in Software Engineering

Compatibility testing :

Compatibility testing is software testing which comes under the non functional testing category, and it is performed on an application to check its compatibility (running capability) on different platform/environments. This testing is done only when the application becomes stable. Means simply this compatibility test aims to check the developed software application functionality on various software, hardware platforms, network and browser etc. This compatibility testing is very important in product production and implementation point of view as it is performed to avoid future issues regarding compatibility.

Types of Compatibility Testing :

Several examples of compatibility testing are given below.

1. Software :

- Testing the compatibility of an application with an Operating System like Linux, Mac, Windows
- Testing compatibility on Database like Oracle SQL server, MongoDB server.
- Testing compatibility on different devices like in mobile phones, computers.

Types based on Version Testing :

There are two types of compatibility testing based on version testing

1. **Forward compatibility testing :** When the behavior and compatibility of a software or hardware is checked with its newer version then it is called as forward compatibility testing.
2. **Backward compatibility testing :** When the behavior and compatibility of a software or hardware is checked with its older version then it is called as backward compatibility testing.

2. Hardware :

Checking compatibility with a particular size of

- RAM
- ROM
- Hard Disk
- Memory Cards
- Processor
- Graphics Card

3. Smartphones :

Checking compatibility with different mobile platforms like android, iOS etc.

4.Network :

Checking compatibility with different :

- Bandwidth
- Operating speed
- Capacity

Along with this there are other types of compatibility testing are also performed such as browser compatibility to check software compatibility with different browsers like Google Chrome, Internet Explorer etc. device compatibility, version of the software and others.

So for now we have known the uses of compatibility in different fields. Now the question rises is HOW TO PERFORM A COMPATIBILITY TEST?

How to perform Compatibility testing ?

Testing the application in a same environment but having different versions. For example, to test compatibility of Facebook application in your android mobile. First check for the compatibility with Android 9.0 and then with Android 10.0 for the same version of Facebook App.

Testing the application in a same versions but having different environment. For example, to test compatibility of Facebook application in your android mobile. First check for the compatibility with a Facebook application of lower version with a Android 10.0(or your choice) and then with a Facebook application of higher version with a same version of Android.

Why compatibility testing is important ?

1. It ensures complete customer satisfaction.
2. It provides service across multiple platforms.
3. Identifying bugs during development process.

Compatibility testing defects :

1. Variety of user interface.
2. Changes with respect to font size.
3. Alignment issues.
4. Issues related to existence of broken frames.

5. Issues related to overlapping of content.

Levels of Software Testing

Software Testing is an activity performed to identify errors so that errors can be removed to obtain a product with greater quality. To assure and maintain the quality of software and to represents the ultimate review of specification, design, and coding, Software testing is required. There are different levels of testing :

1. Unit Testing **:**

   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 **:**

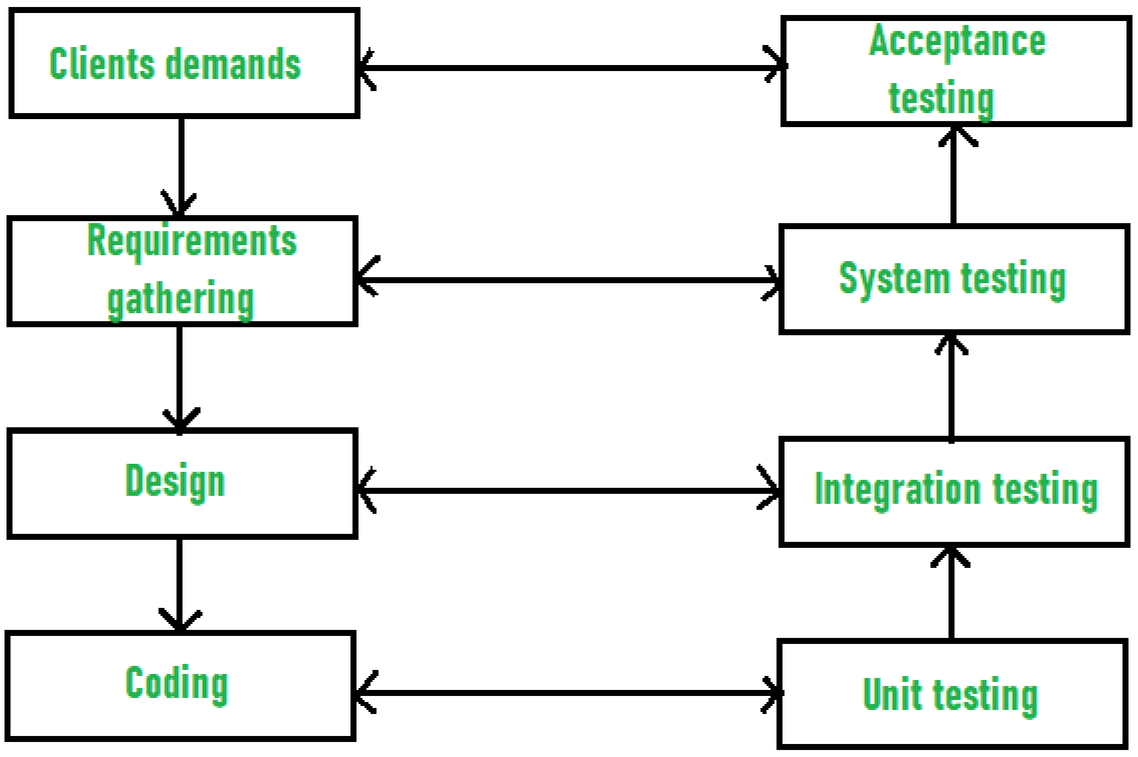
   In this testing, two or more modules which are unit tested are integrated to test i.e. technique interacting components and are then verified if these integrated modules work as per the expectation or not and interface errors are also detected.
3. System Testing **:**

   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 **:**

   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 –



LEVELS OF TESTING

While performing the software testing, following Testing principles must be applied by every software engineer:

- The requirements of customers should be traceable and identified by all different tests.
- Planning of tests that how tests will be conducted should be done long before the beginning of the test.
- The Pareto principle can be applied to software testing- 80% of all errors identified during testing will likely be traceable to 20% of all program modules.
- Testing should begin “in the small” and progress toward testing “in the large”.
- Exhaustive testing which simply means to test all the possible combinations of data is not possible.
- Testing conducted should be most effective and for this purpose, an independent third party is required.