# TESTING

Testing is basically a process to detect errors in the software product. Before going into the details of testing techniques one should know what errors are. In day-to-day life we say whenever something goes wrong there is an error. This definition is quite vast. When we apply this concept to software products then we say whenever there is difference between what is expected out of software and what is being achieved, there is an error.

For the output of the system, if it differs from what was required, it is due to an error. This output can be some numeric or alphabetic value, some formatted report, or some specific behavior from the system. In case of an error there may be change in the format of out, some unexpected behavior from system, or some value different from the expected is obtained. These errors can due to wrong analysis, wrong design, or some fault on developer's part.

All these errors need to be discovered before the system is implemented at the customer's site. Because having a system that does not perform as desired be of no use. All the effort put in to build it goes waste. So testing is done. And it is equally important and crucial as any other stage of system development. For different types of errors there are different types of testing techniques. In the section that follows we'll try to understand those techniques.

## Objectives of testing

First of all the objective of the testing should be clear. We can define testing as a process of executing a program with the aim of finding errors.To perform testing, test cases are designed.A test case is a particular made up artificial situation upon which a program is exposed so as to find errors.So a good test case is one that finds undiscovered errors. If testing is done properly, it uncovers errors and after fixing those errors we have software that is being developed according to specifications.

## Test Information Flow

Testing is a complete process. For testing we need two types of inputs.First is software configuration. It includes software requirement specification, design specifications and source code of program.Second is test configuration. It is basically test plan and procedure.

Software configuration is required so that the testers know what is to be expected and tested whereas test configuration is testing plan that is, the way how the testing will be conducted on the system. It specifies the test cases and their expected value. It also specifies if any tools for testing are to be used. Test cases are required to know what specific situations need to be tested. When tests are evaluated, test results are compared with actual results and if there is some error, then debugging is done to correct the error. Testing is a way to know about quality and reliability. Error rate that is the occurrence of errors is evaluated. This data can be used to predict the occurrence of errors in future.

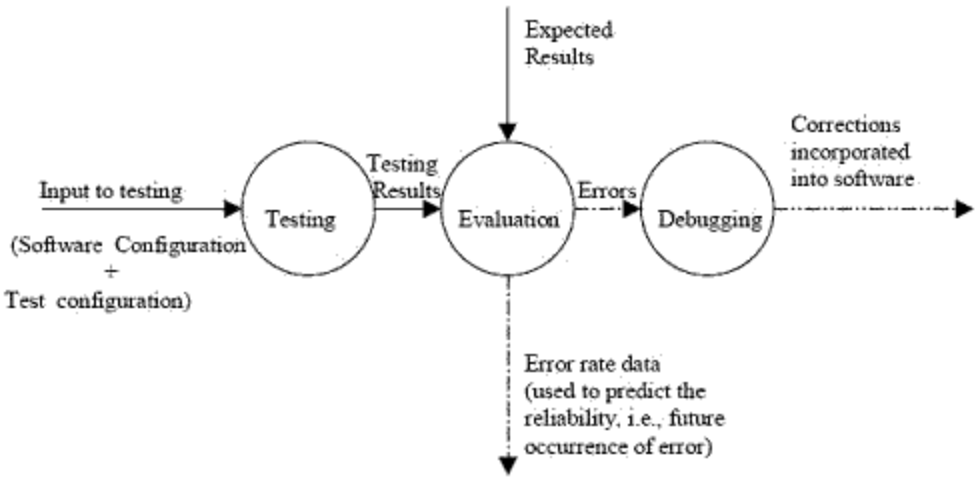



*Fig . Testing Process*

## Test Case design

We now know, test cases are integral part of testing. So we need to know more about test cases and how these test cases are designed. The most desired or obvious expectation from a test case is that it should be able to find most errors with the least amount of time and effort.

A software product can be tested in two ways. In the first approach only the overall functioning of the product is tested. Inputs are given and outputs are checked. This approach is called black box testing. It does not care about the internal functioning of the product.

The other approach is called white box testing. Here the internal functioning of the product is tested. Each procedure is tested for its accuracy. It is more intensive than black box testing. But for the overall product both these techniques are crucial. There should be sufficient number of tests in both categories to test the overall product.

## WHITE BOX TESTING

White box testing focuses on the internal functioning of the product. For this different procedures are tested. White box testing tests the following

- Loops of the procedure
- Decision points
- Execution paths

For performing white box testing, basic path testing technique is used. We will illustrate how to use this technique, in the following section.

## Basis Path Testing

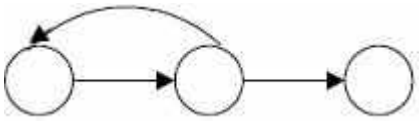
Basic path testing a white box testing technique .It was proposed by Tom McCabe.These tests guarantee to execute every statement in the program at least one time during testing. Basis set is the set of all the execution path of a procedure.

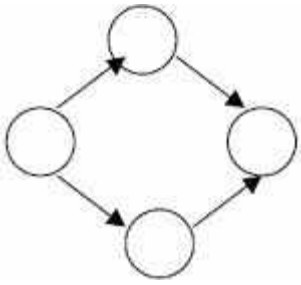
## Flow graph Notation

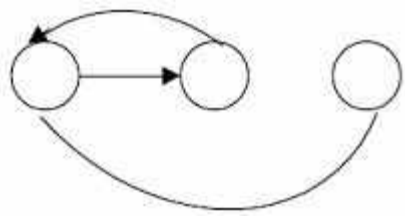
Before basic path procedure is discussed, it is important to know the simple notation used for the representation of control flow. This notation is known as flow graph. Flow graph depicts control flow and uses the following constructs.

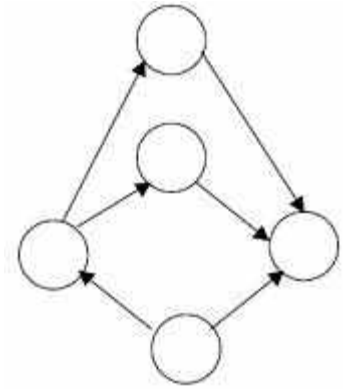
These individual constructs combine together to produce the flow graph for a particular procedure

Sequence - 

Until - 

If - 

While - 

Case - 

Basic terminology associated with the flow graph

**Node:** Each flow graph node represents one or more procedural statements. Each node that contains a condition is called a predicate node.

**Edge:** Edge is the connection between two nodes. The edges between nodes represent flow of control. An edge must terminate at a node, even if the node does not represent any useful procedural statements.

**Region:** A region in a flow graph is an area bounded by edges and nodes.

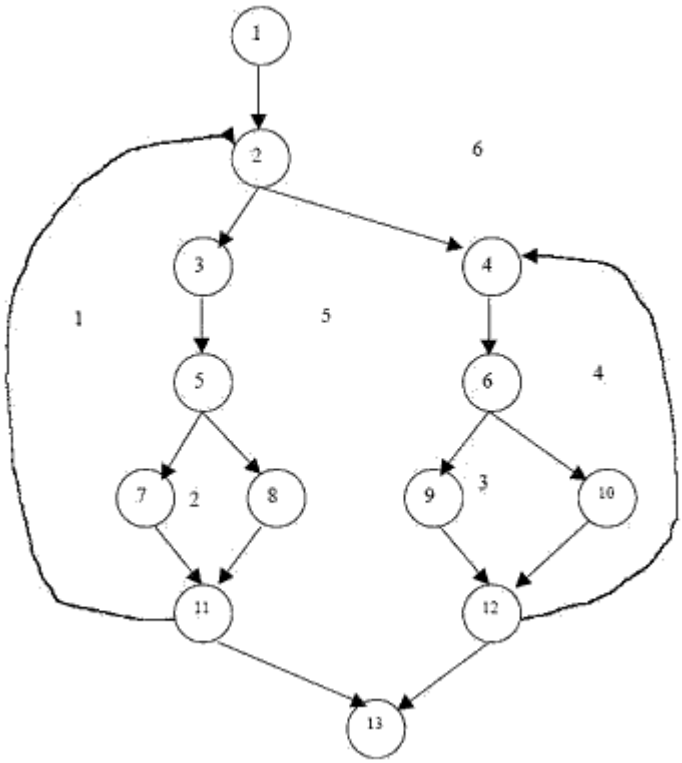
Cyclomatic complexity: Independent path is an execution flow from the start point to the end point. Since a procedure contains control statements, there are various execution paths depending upon decision taken on the control statement. So Cyclomatic complexity provides the number of such execution independent paths. Thus it provides a upper bound for number of tests that must be produced because for each independent path, a test should be conducted to see if it is actually reaching the end point of the procedure or not.

Cyclomatic Complexity

Cyclomatic Complexity for a flow graph is computed in one of three ways:

1. The numbers of regions of the flow graph correspond to the Cyclomatic complexity.
2. Cyclomatic complexity, V(G), for a flow graph G is defined as
   V(G) = E – N + 2
   where E is the number of flow graph edges and N is the number of flow graph nodes.
3. Cyclomatic complexity, V(G), for a graph flow G is also defined as
   V(G) = P + 1
   Where P is the number of predicate nodes contained in the flow graph G.

Example: Consider the following flow graph



Region, R= 6
Number of Nodes = 13
Number of edges = 17
Number of Predicate Nodes = 5

Cyclomatic Complexity, V( C) :

V( C ) = R = 6;
Or

V(C) = Predicate Nodes + 1
=5+1 =6
Or

V( C)= E-N+2
= 17-13+2

**Deriving Test Cases**

The main objective of basic path testing is to derive the test cases for the procedure under test. The process of deriving test cases is following

1. From the design or source code, derive a flow graph.
2. Determine the Cyclomatic complexity, V(G) of this flow graph using any of the formula discussed above.
   · Even without a flow graph, V(G) can be determined by counting the number of conditional statements in the code and adding one to it.
3. Prepare test cases that will force execution of each path in the basis set.
   · Each test case is executed and compared to the expected results.

**Graph Matrices**

Graph matrix is a two dimensional matrix that helps in determining the basic set. It has rows and columns each equal to number of nodes in flow graph. Entry corresponding to each node-node pair represents an edge in flow graph. Each edge is represented by some letter (as given in the flow chart) to distinguish it from other edges. Then each edge is provided with some link weight, 0 if there is no connection and 1 if there is connection.

For providing weights each letter is replaced by 1 indicating a connection. Now the graph matrix is called connection matrix.Each row with two entries represents a predicate node. Then for each row sum of the entries is obtained and 1 is subtracted from it. Now the value so obtained for each row is added and 1 is again added to get the cyclomatic complexity.

Once the internal working of the different procedure are tested, then the testing for the overall functionality of program structure is tested. For this black box testing techniques are used which are discussed in the following pages.

## BLACK BOX TESTING

Black box testing test the overall functional requirements of product. Input are supplied to product and outputs are verified. If the outputs obtained are same as the expected ones then the product meets the functional requirements. In this approach internal procedures are not considered. It is conducted at later stages of testing. Now we will look at black box testing technique.

Black box testing uncovers following types of errors.

1. Incorrect or missing functions
2. Interface errors
3. External database access
4. Performance errors
5. Initialization and termination errors.

The following techniques are employed during black box testing

### Equivalence Partitioning

In equivalence partitioning, a test case is designed so as to uncover a group or class of error. This limits the number of test cases that might need to be developed otherwise.

Here input domain is divided into classes or group of data. These classes are known as equivalence classes and the process of making equivalence classes is called equivalence partitioning.Equivalence classes represent a set of valid or invalid states for input condition.

An input condition can be a range, a specific value, a set of values, or a boolean value. Then depending upon type of input equivalence classes is defined. For defining equivalence classes the following guidelines should be used.

1. If an input condition specifies a range, one valid and two invalid equivalence classes are defined.
2. If an input condition requires a specific value, then one valid and two invalid equivalence classes are defined.
3. If an input condition specifies a member of a set, then one valid and one invalid equivalence class are defined.
4. If an input condition is Boolean, then one valid and one invalid equivalence class are defined.

For example, the range is say, 0 < count < Max1000. Then form a valid equivalence class with that range of values and two invalid equivalence classes, one with values less than the lower bound of range (i.e., count < 0) and other with values higher than the higher bound( count > 1000).

## Boundary Value Analysis

It has been observed that programs that work correctly for a set of values in an equivalence class fail on some special values.These values often lie on the boundary of the equivalence class. Test cases that have values on the boundaries of equivalence classes are therefore likely to be error producing so selecting such test cases for those boundaries is the aim of boundary value analysis.

In boundary value analysis,we choose input for a test case from an equivalence class, such that the input lies at the edge of the equivalence classes. Boundary values for each equivalence class, including the equivalence classes of the output, should be covered. Boundary value test cases are also called "extreme cases".

Hence, a boundary value test case is a set of input data that lies on the edge or boundary of a class of input data or that generates output that lies at the boundary of a class of output data.

In case of ranges, for boundary value analysis it is useful to select boundary elements of the range and an invalid value just beyond the two ends (for the two invalid equivalence classes. For example, if the range is 0.0 <= x <= 1.0, then the test cases are 0.0,1.0for valid inputs and –0.1 and 1.1 for invalid inputs.

For boundary value analysis, the following guidelines should be used:

For input ranges bounded by a and b, test cases should include values a and b and just above and just below a and b respectively.

If an input condition specifies a number of values, test cases should be developed to exercise the minimum and maximum numbers and values just above and below these limits.

If internal data structures have prescribed boundaries, a test case should be designed to exercise the data structure at its boundary.

Now we know how the testing for software product is done. But testing software is not an easy task since the size of software developed for the various systems is often too big. Testing needs a specific systematic procedure, which should guide the tester in performing different tests at correct time. This systematic procedure is testing strategies, which should be followed in order to test the system developed thoroughly. Performing testing without some testing strategy would be very cumbersome and difficult. Testing strategies are discussed the following pages of this chapter.

## Validation Testing

After the integration testing we have an assembled package that is free from modules and interfacing errors. At this stage a final series of software tests, validation testing begin.Validation succeeds when software functions in a manner that can be expected by the customer.

Major question here is what are expectations of customers. Expectations are defined in the software requirement specification identified during the analysis of the system. The specification contains a section titled “Validation Criteria” Information contained in that section forms the basis for a validation testing.

Software validation is achieved through a series of black-box tests that demonstrate conformity with requirements. There is a test plan that describes the classes of tests to be conducted, and a test procedure defines specific test cases that will be used in an attempt to uncover errors in the conformity with requirements.

After each validation test case has been conducted, one of two possible conditions exists:

The function or performance characteristics conform to specification and are accepted, or

A deviation from specification is uncovered and a deficiency list is created. Deviation or error discovered at this stage in a project can rarely be corrected prior to scheduled completion. It is often necessary to negotiate with the customer to establish a method for resolving deficiencies.

## Alpha and Beta testing

## What is Alpha Testing ?

Alpha testing is a type of acceptance testing; performed to identify all possible issues/bugs before releasing the product to everyday users or public. Alpha testing is carried out in a lab environment and usually the testers are internal employees of the organization. To put it as simple as possible, this kind of testing is called alpha only because it is done early on, near the end of the development of the software, and before beta testing.


## What is Beta Testing?

Beta Testing of a product is performed by "real users" of the software application in a "real environment" and can be considered as a form of external User Acceptance Testing.

Beta version of the software is released to a limited number of end-users of the product to obtain feedback on the product quality. Beta testing reduces product failure risks and provides increased quality of the product through customer validation.

It is the final test before shipping a product to the customers. Direct feedback from customers is a major advantage of Beta Testing. This testing helps to tests the product in real time environment.

Alpha Testing Vs Beta testing:

Following are the differences of Alpha and Beta Testing:

| Alpha Testing | Beta Testing |
|---|---|
| Alpha testing performed by Testers who are usually internal employees of the organization | Beta testing is performed by Clients or End Users who are not employees of the organization |
| Alpha Testing performed at developer's site | Beta testing is performed at client location or end user of the product |
| Reliability and Security Testing are not performed in-depth Alpha Testing | Reliability, Security, Robustness are checked during Beta Testing |
| Alpha testing involves both the white box and black box techniques | Beta Testing typically uses Black Box Testing |
| Alpha testing requires lab environment or testing environment | Beta testing doesn't require any lab environment or testing environment. Software is made available to the public and is said to be real time environment |
| Critical issues or fixes can be addressed by developers immediately in Alpha testing | Most of the issues or feedback is collected from Beta testing will be implemented in future versions of the product |

## Recovery Testing

Many computer-based systems must recover from faults and resume operation within a pre-specified time. In some cases, a system may be fault tolerant; that is, processing faults must not cause overall system function to cease. In other cases, a system failure must be corrected within a specified period or severe economic damage will occur.

Recovery testing is a system test that forces the software to fail in a variety of ways and verifies that recovery is properly performed. It the recovery is automated (performed by system itself), re-initialization mechanisms, data recovery, and restart are each evaluated for correctness. If the recovery requires human intervention, the mean time to repair is evaluated to determine whether it is within acceptable limits.

## Stress Testing

Stress testing (sometimes called torture testing) is a form of deliberately intense or thorough testing used to determine the stability of a given system or entity. It involves testing beyond

normal operational capacity, often to a breaking point, in order to observe the results. Reasons can include:

- to determine breaking points or safe usage limits
- to confirm mathematical model is accurate enough in predicting breaking points or safe usage limits
- to confirm intended specifications are being met
- to determine modes of failure (how exactly a system fails)
- to test stable operation of a part or system outside standard usage

Reliability engineers often test items under expected stress or even under accelerated stress in order to determine the operating life of the item or to determine modes of failure.

The term "stress" may have a more specific meaning in certain industries, such as material sciences, and therefore stress testing may sometimes have a technical meaning – one example is in fatigue testing for materials.

In hardware, stress testing, in general, should put computer hardware under exaggerated levels of stress in order to ensure stability when used in a normal environment. These can include extremes of workload, type of task, memory use, thermal load (heat), clock speed, or voltages. Memory and CPU are two components that are commonly stress tested in this way.

## Security Testing

Any computer-based system that manages sensitive information or causes actions that can harm or benefit individuals is a target for improper or illegal penetration.

Security testing attempts to verify that protection mechanism built into a system will protect it from unauthorized penetration. During security testing, the tester plays the role of the individual who desires to penetrate the system. The tester may attack the system with custom software designed to break down any defenses that have been constructed; may overwhelm the system, thereby denying service to others; may purposely cause system errors, hoping to find the key to system entry; and so on.

Given enough time and resources, good security testing will ultimately penetrate a system. The role of the system designer is to make penetration cost greater than the value of the information that will be obtained in order to deter potential threats.