



A REVIEW ON SOFTWARE TESTING TECHNIQUES

Swati Gupta

Assistant Professor, Department of Computer Science

Amity University Haryana, Gurgaon, India

Abstract: Software testing is a fundamental step in SDLC, which is being applied, on the data sets to reduce errors, thereby enhancing the software quality. They are broadly divided into two categories: black box testing & white box testing.

This paper describes the different categories of white box, black box and grey box testing. This paper also introduces the concept of new approach that is being followed in software testing process.

Keywords: Software Testing, SDLC, Black Box testing, White box testing.

I. INTRODUCTION

Software testing is a process or a series of steps which are being designed to verify computer code, so that the code behaves in exactly the same way it has been expected[4].

Another more appropriate definition is this: [3] Testing is the process of executing a program with the intent of finding errors. The concept of testing is as old as coding and is change along with time.

It can also be stated as process of validating and verifying that a software program or application or product:

- Meets the business and technical requirements
- Works as being expected
- Can be implemented with the same characteristic.

Performance of testing depends upon the process and associated stakeholders of the project. Most of the times, following professionals are involved in software testing:

- Project Manager
- Software Tester
- Software Developer
- End User

It is important to know when to start & stop testing in

On Set OF Testing: In simple saying Software Testing can be start with the first phase of SDLC i.e. Requirement Gathering and can be performed till the last phase i.e. Deployment phase. But in actual it depends upon the type of model, being selected by the developer. For example if we are using Waterfall Model then testing will be done in testing phase only, and if one is using incremental/Evolutionary Model then testing will be performed at each increment and so on.

When to End Testing Process: Software Testing is a Never Ending Process. Even after Satisfactorily completion of testing phase, we can't say that software is error free,because the Input Domain to the system is very large and it is not possible to test each & every input.

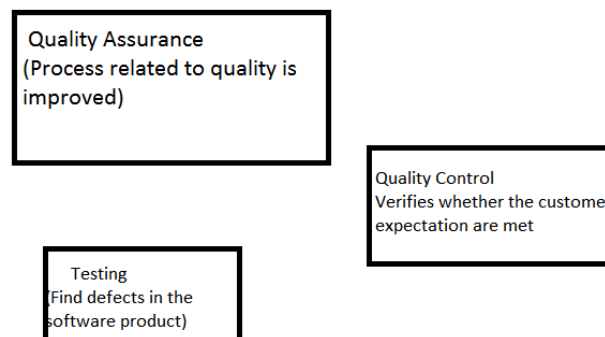


Figure 1. Software Testing Concepts

Some of the important terminology that are used frequently in Software testing are as follows:

- Fault - An incorrect requirement (functional/nonfunctional) causes program to behave in unanticipated manner.

- Error – Any mistake committed by developer at the time of development.
- Failure – Symptom of error.

Thus, whenever there is any fault in the program, Program is still able to run, But Performance be degraded. Whenever there is any Failure, Program is not able to run.

- Test Case – A test case is a set of conditions or variables under which the tester will determine whether a system under test satisfies requirement or work correctly. It can be denoted as triple [I,S,O] as shown below.

Table 1: Test Case Triplets

I	S	O
(Input to the System)	(State of the system)	(Output of the System)

- Test Suite – is a collection of test cases that are intended to be used to test a software program to show that it has some specified set of behaviors.

II. TESTING TECHNIQUES

Software testing techniques are broadly divided into two types:

A. White Box Testing

B. Black Box Testing

A. White Box Testing :

White box testing techniques [5] are used to check the internal structure of the system by examining the code. In this testing the tester test the entire Branches, Segments, Loops, and Conditions of the program. Testers who are performing the test must have complete knowledge of the code.

White-Box Testing is further divided into two categories: fault based or coverage based.

a. Fault-Based[2] : Fault-Based as the name suggest refers to detect certain types of faults. The example of fault based testing is Mutation Testing where errors or bugs are inserted by the programmer or testers by their own and then they try to detect the induced bugs in the program.

b. Coverage-Based Testing

Coverage Based Testing as the name implies refers to cover the different elements of a program. Examples are Statement Coverage, Path Coverage.

Statement Coverage[7] means that all the statement of the program are executed at least once. Let us consider the following example

1. While(p!=q)
2. { If(p<=q) then
3. q=q-p;
4. else p=p-q;
5. }
6. Return p; }

In this example if we choose the following vales for the test case {(p=2,q=2),(p=3,q=2),(p=2,q=3)}, then all the statement of the program will be executed at least once.

Path Coverage[7,8] means all basis paths in the program are executed at least once. In this technique we draw a CFG (Control Flow Graph) which is used to describe the sequence of instructions. Consider the following example

1. If (x > y)
2. z = 4
3. else z = 5
4. z= z + z

We will create the CFG in order to determine the sequence as shown in the diagram

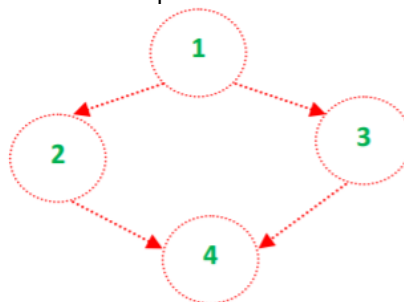


Figure 2: CFG for the Example

B. Black-Box Testing :

In Black box testing[6] the user does not intend to look into the source code. It is primarily used to check the functional requirements of the system.

Black-Box Testing can be further divided into two parts:

- a. Equivalence Class Partitioning
- b. Boundary Value Analysis

a) Equivalence Class Partitioning

It is a software testing technique that divides the input test data of the application under test into each partition at least once of equivalent data from which test cases can be derived.

The advantage of this approach is it reduces the time required for performing testing of a software due to less number of test cases.

Let us consider the following example: If we are testing an input box accepting numbers from 1 to 1000 then there is no use in writing thousand test cases for all 1000 valid input numbers plus other test cases for invalid data.

We will use equivalence partitioning method, the test cases can be divided into three sets of input data known as classes. Each test case is a representative of respective class. So in above example we can divide our test cases into three equivalence classes of some valid and invalid inputs.

Test cases for input box accepting numbers between 1 and 1000 using the Equivalence Partitioning are as follows:

- 1) One input data class with all valid inputs. Pick a single value from range 1 to 1000 as a valid test case. If we select other values between 1 and 1000 then result is going to be same. So one test case for valid input data is sufficient.
- 2) Input data class with all values below lower limit. I.e. any value below 1, as a invalid input data test case.
- 3) Input data with any value greater than 1000 to represent third invalid input class.

So using equivalence partitioning we have categorized all possible test cases into three classes

b) Boundary value analysis:

It's widely recognized that input values at the extreme ends of input domain cause more errors in system. More application errors occur at the boundaries of input domain. 'Boundary value analysis' testing technique is used to identify errors at boundaries rather than finding those exist in center of input domain.

Boundary value analysis is a next part of Equivalence partitioning for designing test cases where test cases are selected at the edges of the equivalence classes.

Test cases for input box accepting numbers between 1 and 1000 using Boundary value analysis:

- 1) Test cases with test data exactly as the input boundaries of input domain i.e. values 1 and 1000 in this case.
- 2) Test data with values just below the extreme edges of input domains i.e. values 0 and 999.
- 3) Test data with values just above the extreme edges of input domain i.e. values 2 and 1001.

Boundary value analysis is often called as a part of stress and negative testing.

III. NEW APPROACHES IN SOFTWARE TESTING: OBJECT ORIENTED PROGRAMS TESTING

Earlier when object oriented programming came into existence, it was believed that object oriented testing will definitely reduce the cost & effort in testing. This thinking was based on the various new programming features provided by object oriented programming including Encapsulation, Polymorphism, Data Abstraction, Inheritance etc. But very soon, it came to know that object oriented testing is taking more time, cost & effort as compare to testing of procedural programs. This is because the new features provide additional complications & various new types of bugs in the program which requires additional test cases to perform the testing. In this paper two different type of testing schemes performed on object oriented programs are being discussed

a) **Gray-Box Testing:**[3,1] It is done from the outside of the system. It is basically a combination of white-box and black-box testing which can be applied in real time systems.

Following are some subtypes of gray-box testing.

- State-Model-Testing: It tests each method of an object, transition & transition paths at each state of an object.
- Class-Diagram Testing: It tests all the derived classes of the base class.
- Sequence-Diagram Testing: It tests all the methods occurring in sequence diagram.

b) **Integration Testing:** It includes two main types as following:

- Thread-Based Testing: In this all the classes of a single Use Case are integrated together and then testing is performed. The process is continuously repeated until all the classes of all Use Cases have been considered.
- Use-Based Testing: It performs the testing on the classes that either need the services from other classes or does not need any services.

IV. CONCUSION

Testing is an integral part of SDLC, which is being carried out to detect faults in a system .In this paper detailed study on different types of testing are being carried out. The white box & black box testing are the two main types of testing that are being carried out in order to enhance the quality of the software product. Their testing methodoly & relevant example are being taken to understand white box & black box testing. We also conclude that 100% testing cannot be possible because domain of possible input is too large to test. Further we have discussed the object oriented programs testing in detail covering its two types namely: Gray Box testing & Integration testing.

REFERENCES

- [1] C.Easteal and G.Davis, Software Engineering Analysis and Design, Tata McGraw Hill.
- [2] Richard Fairley ,Software Engineeering Concepts ,Tata Mcgraw Hill.
- [3] Ian Sommeriele, "Software Engineering" , Addison Wesley.
- [4] Pressman, Software Engineering –A Practitioner’s Approach.
- [5] Pankaj Jalote , An Integrated Approach to Software engineering, Narosa Publication.
- [6] T.H.Shivkumar,"Software Testing Techniques"Volume 2,Issye 10,ISSN:2277 128X
- [7] Jovanovi,ć, Irena,"Software Testing Methods and Techniques"Page No-30-41
- [8] SoftwareTestingOverview,available:http://www.tutorialspoint.com/software_testing/software_testing_quick_guide.h