Software Testing Techniques

Sahil Munjal , Sahil Bhardwaj , Sachin Malik
Student, Dronacharya College of Engineering, Khentawas, Farrukhnagar, Gurgaon

***Abstract*—Testing a software is a complex issue. This Paper attempts to provide details on the field software Testing Techniques. Software Testing provides a way to find the bugs and errors in a software by analysing a software item to detect the differences between existing and required conditions. It also means to find the various errors that occur in the software because of which the software is not able to give the correct output. There are various software testing tools & techniques to evaluate the system. The major issue within software testing field is to detect bugs. There are various techniques available for testing software.**

Index Terms- Software Testing, Testing Techniques.

I. INTRODUCTION

Software Testing is one of the important topics in field of Software development. It is a very complex activity deserving a first-class role in software development. Software Testing is nothing but error detection. Thus, whenever to design and implement the computer based system or product one should keep in mind the testability. Testing is a process of evaluating the system or its components to find out the differences between existing conditions and the required conditions.

A. Who Does Testing

Testing can be done by everyone who involved in the development of software. Most of the time, following professionals is involved in software testing:

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

It is not possible to test the software at any time during
Software Development Life Cycle. So following describes when to start & stop the testing.

B. When to Start Testing

In simple saying Software Testing can be started from 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, the software developer, is using. For example if one is using Waterfall Model then testing performance 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.

C. When to Stop Testing

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.

Terminology-
Fault - An incorrect requirement (functional/non-functional) causes program to perform in unanticipated manner.
Error – Any mistake committed by developer at the time of development.
Failure – Symptom of error.

Fault → Error → Failure

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

II. TESTING MYTHS

Following are some common myths about software testing.

Myth: Testing is too expensive.
Reality: There is always saying that we should pay less for testing and more for maintenance. But in actual If there will be no proper testing then it may result in improper design of the software which will be expensive.

Myth: Missed defects are due to Testers.
Reality: It is not correct saying. The Main reason behind this is requirement changing. As requirements are keep on changing every time and it is also possible that test strategy result in bugs missed by testing team.

Myth: Complete Testing is Possible.
Reality: Due to the fact that during the lifetime of a software product a user never uses all the possible scenarios of the product execution as a result of which a software tester similarly exempt these, not so used, scenarios from testing.

Myth: Testing is Time Consuming.
Reality: It is very wrong to say that testing is time consuming. However, diagnosing and fixing the error during testing is time consuming but, It is a part of Software Development or we can say it is a Productive activity. If Testers will not identify the errors during testing phase then it will take more time to identify the errors.

Myth: Can only Testing Team perform the Testing.
Reality: No, Along with the testing team, also the end-user and customers are in better state of testing a software product as they are ultimate consumer of the product.

Myth: Quality of Product is responsibility of Testers.
Reality: No, in actual testers are just responsible for identify the errors, and then it will depend upon the development team that whether they want to fix that error or not, If they release the product as it is then the blame of errors comes on the testers.

Myth: Testing is done only after product is fully developed.
Reality: There is no Doubt that testing depends on the source code But it can also be done in previous phases like in requirement phase reviewing requirement and developing test cases.

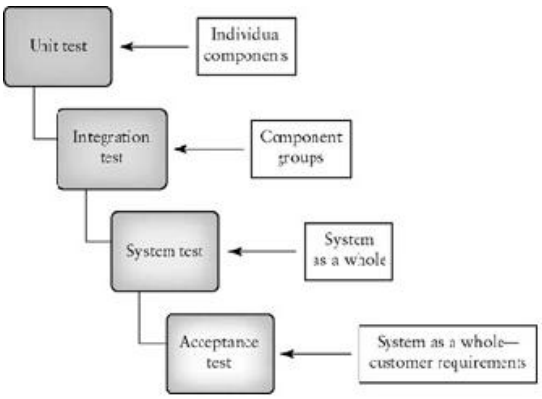
Myth: After testing Product is fully bug free.
Reality: This is very common myth that management team and end-user believe in. But in actual even after satisfactorily completion of testing we can't say that product is 100% bug free. The main reason behind this is the requirement changing time to time.

III. TESTING LEVELS

Whenever software is tested it has to go through three stages:

a) Unit Testing
b) Integration Testing
c) System Testing
d) Acceptance Testing

as shown in the following figure.



Testing Levels

A. Unit Testing

This is the first level in which each and every module of
software is tested separately. In this output of one module can be the input to the other. But unfortunately if output is wrong then another module to which we give the input will also collapse.

Thus it's better to test each and ever y module separately so that there will be less chance of collapse.

B. Integration Testing

After Unit Testing there comes Integration Testing in which all modules tested during unit testing are integrated together and then performing testing on them. It provides testing again on all that modules so that if any error remains that will be removes. It is of three types.

Top-Down Testing:

In this first of all main modules called Top module is tested only after that rest of the modules are tested.

Bottom-Up Testing:

In This firstly bottom up module is tested after that all the modules are tested.

Mixed-Approach
This is the best approach because it combines the features of Top-Down and Bottom-Up Approach. In this all the modules firstly present then testing is performed.

C. System Testing:
After all the bugs, errors have been removed it's the turn of system testing to occur. As shown in the figure it includes three sub-categories.

a Testing:
This type of system testing is performed by the developers of the software. So Development team is basically responsible for testing.

ß Testing:
This testing is performed by friendly set of customers .It means developers release their product from their side and then end user test the product, but it is not the final delivery of the product.

Acceptance Testing:
This testing we can say the combination of both a and ß
Testing in which customers test whether to accept the delivered product or not.

IV. TESTING TECHNIQUES

Here, I am discussing two testing techniques.
A. White Box Testing
B. Black Box Testing

A. White Box Testing :
Used to check the internal structure of the system.
Performed to test all the Branches, Segments, Loops, and
Conditions of the program. Testers who perform testing should have through knowledge of the system code and they should also know the aim [purpose] of the system for which it is developed. White-Box Testing can be either fault based or coverage based.

a. Fault-Based :
Fault-Based as the name implies refers to detect certain types of faults. Main Example of fault based testing is Mutation Testing where errors or bugs are inserted by the programmer or testers their self and then to verify whether the test cases are able to detect those errors or not.

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

Statement Coverage means there is no other way to check whether the errors exist in the statement unless that statement is executed at least once. So, in order to check a statement, it is necessary to execute it. Following Example will give an idea.

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

So, for this program, by choosing the test case {(x=2,y=2),(x=3,y=2),(x=2,y=3)}, then all the statement of the program will be executed at least once.

Path Coverage means all basis paths in the program are executed at least once. For this CFG (Control Flow Graph) is used that describes the sequences in which different instructions of the program are executed. Following Example will give an idea.

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

B. Black-Box Testing :
Used to check the functional requirements. Performed to check all the inputs and outputs of these
requirements.
Includes interaction between Input, Requirement, Events and output.
Black-Box Testing can be either:
a. Equivalence Class Partitioning
b. Boundary Value Analysis

McCabe's Cyclomatic Complexity Metric:
It defines an upper bound on the no. of basis path in program. Here I will discuss three ways to compute it. Let us take an example.

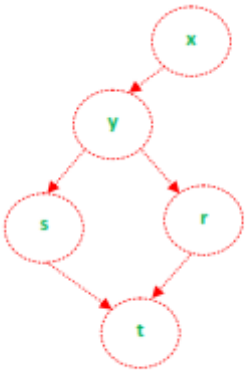


Figure McCabe's Metric

Method 1:

No. of Predicate + 1

i.e. x and y are predicates, So

2 + 1 = 3

Method 2:

E – N + 2

Where E is Edge and N is Node, So

6 – 5 + 2 = 3

Method 3:

No. of enclosed area + 1

i.e. 2 + 1 = 3

V. NEW APPROACH: OBJECT ORIENTED PROGRAMS TESTING

Earlier when object oriented programming was developed, it was believed that object oriented testing will definitely reduce the cost & effort. 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. Thus, here I am going to discuss two types of testing schemes performed on object oriented programs.

A. Gray-Box Testing:

It is done from the outside of the system. Actually we can say that it is the 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. This process is going on 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.

VI. CONCLUSION

Software Testing is and will forever be a fundamental
activity of Software Engineering. We will never find a test approach that is guaranteed to deliver a "perfect" product, whichever is the effort we employ.

Software Testing is a trial-and-error methodology. Software Testing can never be satisfactorily completed because of the input domain from customer.

Testing costs can be reduced by using different test
automation tools.

Testing helps to detect the errors in system but does
not prove that system is error free.

Testing Object Oriented Programs provides new features but including additional complications as well.

Object Oriented Testing Techniques takes more time
as compare to testing of Procedural Programs.

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,c, Irena,"Software Testing Methods and
Techniques"Page No-30-41

[8]Software Testing Overview , available: http://www.tutorialspoint.com/software_testing/software_testing_quick_guide.htm