# Mutation Testing Vs. Regression Testing

Mahesh Kumar Tiwari[1], Shalini Lamba [2]

[1, 2] *Assistant Professor, Computer Science Department, National P.G. College, Lucknow, U.P.*

***Abstract*:- Testing is the process of finding as many errors as possible before software is delivered to customer. Since, there are various testing techniques available to establish quality, performance and reliability of software but Mutation Testing and Regression Testing is focused in this paper. Mutation testing involves manipulating program slightly and testing it with intention to find effectiveness of test suite selected. Regression testing intends to find bugs in software, if software is modified after delivery either due to result of fixes or due to new or enhanced functionality. The use of regression testing is to check that enhancements have not affected previous functionality as well as working correctly.**

***Keywords*: - Mutation Testing, Regression Testing**

## I. INTRODUCTION

Testing can be defined as verifying and validating something (any software in terms of software testing). These two activities play major role in software testing process. Before software is delivered to customer, it has to be sure that all requirements are met. The process of checking whether the software is fulfilling all the customer requirements is termed as **software verification**. It basically checks that all expectations are properly met or not. While the process of checking that is software is satisfying all user requirements or not is termed as **software validation.** The aim of validating software is to remove all the bugs form software. **Example:** Consider a small program of Calculator. In this calculator, checking if it has options of addition, deletion, multiplication and division is regarded as it's verification while checking whether it is producing correct result for every operation (including worst case scenario) is termed as its validation.

Testing applies to real world scenarios also. Suppose that a customer want to purchase TV of any particular brand (say XYZ), so prior to billing he would like to verify TV's brand by seeing label on it and then customer would see its performance i.e. customer will validate it. Thus, testing is an essential part before purchasing any product. Testing gives an idea about efficiency and quality of product under test and at the same time minimizes the future risks.

Software testing is very important activity of Software Development Life Cycle. It is very vast and time consuming activity which consists of one third to one half of total development process. It basically consists of two techniques- Functional testing or black box testing and Structural testing or white box testing. As the name suggests, in functional testing the focus is on **what is the output produced not how it is produced** while in structural testing, focus is on both i.e. **what is the output produced and how it is produced.** Therefore, in structural testing, source code is reviewed thoroughly and test cases are derived from it. Functional testing includes Boundary Value Analysis (BVA), Equivalence Class Testing, Decision Based Table Testing, Cause Effect Graphing Technique etc. while Structural testing includes Control Flow Testing, Data Flow Testing, Slice Based Testing, Mutation Testing etc[1].

## II. MUTATION TESTING

Mutation testing involves changing the program and testing it. In this, a copy of program is created and one or more changes are introduced to this copy. The changed program is termed as **mutant of actual program** and process of changing program is known as **mutation.** This changed program is tested and result produced is compared with expected output. The purpose of mutation testing is to find effectiveness of test suite selected. While creating the mutants of program, following precautions should be considered-

- Mutants should be syntactically correct,
- They should compile correctly,
- Changes should be small so that objective of program remains unchanged,
- Each mutant should differ from original program by one and only one mutation.

Mutants can be created by changing access modifier, static modifier, argument order, operator, any numeric value etc[2].
The mutants of programs are executed and their result is tested against original program. If test cases are able to find changes made i.e. if actual output and expected output is different, then it is regarded as **Killed Mutant.** Otherwise if actual output and expected output is same, then it is termed as **Alive Mutant[3].**

Mutation testing is based on the assumption that a program will be well tested if all simple faults are detected and removed[4].

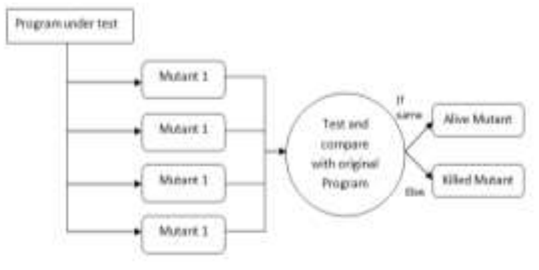

Figure 1 Process of Mutation Testing

After mutation testing is done mutation score is calculated as-

**Mutation Score = Number of Mutants Killed / Total Number of Mutants**

Mutation score determines the accuracy and sensitivity of program towards changes. Mutation score is always lies between 0 and 1. A higher value of mutation score indicates the effectiveness of test suite.

Mutation was originally proposed in 1971 but being very expensive, it lost its importance but now, again it is being widely used for languages like Java and XML.

*Mutation Testing Example*

To understand the concept of Mutation testing consider a program that calculates the income tax based on following criteria-

| | |
|---|---|
| Income <= 40000 | No Tax |
| Income >40000 and <=60000 | tax=10% of income exceeding 40000 |
| Income>60000 and <=150000 | tax=2000+20% of income exceeding 60000 |
| Income >150000 | tax=20000+30% of income exceeding 150000 |

The program for above code will be-

```
1)  import java.io.*;
2)  public class Income_Tax
3)  {
4)    public static void main(String agrs[])throws
    IOException
5)      {
6)          int income;
7)          double tax;
8)              BufferedReader br=new BufferedReader(new
              InputStreamReader(System.in));
9)          System.out.print("Income\t: ");
10)         income=Integer.parseInt(br.readLine());
11)         if(income<=40000)
12)             System.out.println("Tax\t: No Tax");
13)         else if(income<=60000)
14)         {
15)             tax=10.0/100*(income-40000);
16)             System.out.println("Tax\t: "+tax);
17)         }
18)         else if(income<=150000)
19)         {
20)             tax=2000+20.0/100*(income-60000);
21)             System.out.println("Tax\t: "+tax);
22)         }
23)         else
24)         {
25)             tax=20000+30.0/100*(income-150000);
26)             System.out.println("Tax\t: "+tax);
27)         }
28)     }
29) }
```

Let the test suite selected is

| ID | Income | Tax (Expected Output) |
|---|---|---|
| 1 | 37000 | No Tax |
| 2 | 57000 | 1700.0 |
| 3 | 100000 | 10000.0 |
| 4 | 200000 | 35000.0 |

Let the mutants of program are-

| Mutation No. | Line No. | Original Line | Modified Line |
|---|---|---|---|
| M1 | 11 | if (income<=40000) | if(income==40000) |
| M2 | 13 | else if (income<=60000) | else if(income==40000 \|\|income<=60000) |
| M3 | 20 | tax=2000+20.0/100 *(income-60000); | tax=20.0/100* (income-60000)-2000; |
| M4 | 25 | tax=20000+30.0/100 *(income-150000); | tax=20000+30.0/100* (income-15000); |

Test cases for Mutant M1

| ID | Income | Tax (Expected Output) | Tax (Original Output) |
|---|---|---|---|
| 1 | 37000 | No Tax | -300.0 |
| 2 | 57000 | 1700.0 | 1700.0 |
| 3 | 100000 | 10000.0 | 10000.0 |
| 4 | 200000 | 35000.0 | 35000.0 |

Test cases for Mutant M2

| ID | Income | Tax (Expected Output) | Tax (Original Output) |
|---|---|---|---|
| 1 | 37000 | No Tax | No Tax |
| 2 | 57000 | 1700.0 | 1700.0 |
| 3 | 100000 | 10000.0 | 10000.0 |
| 4 | 200000 | 35000.0 | 35000.0 |

Test cases for Mutant M3

| ID | Income | Tax (Expected Output) | Tax (Original Output) |
|---|---|---|---|
| 1 | 37000 | No Tax | No Tax |
| 2 | 57000 | 1700.0 | 1700.0 |
| 3 | 100000 | 10000.0 | 6000.0 |
| 4 | 200000 | 35000.0 | 35000.0 |

Test cases for Mutant M4

| ID | Income | Tax (Expected Output) | Tax (Original Output) |
|---|---|---|---|
| 1 | 37000 | No Tax | No Tax |
| 2 | 57000 | 1700.0 | 1700.0 |
| 3 | 100000 | 10000.0 | 6000.0 |
| 4 | 200000 | 35000.0 | 75500.0 |

As we can see that Mutation M2 is producing same expected and desired output hence it will be alive while Mutation M1, M3, M4 will be killed. Hence,

Mutation score=3/4

=0.75

Higher the mutation score, better the effectiveness of test suite.

## III. REGRESSION TESTING

Once software is tested, it is delivered to customer. Since testing shows presence of errors not absence of errors, therefore there may be possibility that any error may arise after delivery of software. In this case software will be modified. The process of testing the modified portion of program and portion likely to be affected by modification is termed as **Regression Testing.** Modification in program or software is not always carried out due to errors. It may also be carried out if customer wants to add certain functionality to existing software.

Regression testing is part of **system maintenance activity**, carried out to ensure that modified portion has not introduced errors in previously tested software and is working properly. Thus, regression testing is carried out when any changes are made to software after delivery.

For regression testing, already available test cases are used. There are three possibilities of selecting test cases-

1) Select all test cases, which is very simple but at same time very time consuming,
2) Select random test cases, which is less time consuming but there is possibility that it may escape test cases related to modified portion of code,
3) Select test cases that traverse the modified portion of code.

The idea of regression testing is to save time and money. As there is no logic in testing completely tested program again.

*Regression Testing Example*

To understand the concept more clearly, consider the previous program of computing income tax. Suppose that now customer wants to add clause that if income is above 200000 then surcharge of 2% on tax is calculated as per previous rates. So, the programmer will modify the program according to additional requirements and the modified program will be as-

```
1)  import java.io.*;
2)  public class Income_Tax
3)  {
4)  public static void main(String agrs[])throws IOException
5)      {
6)          int income;
7)          double tax;
8)  BufferedReader br=new
        BufferedReader(new
        InputStreamReader(System.in));
9)          System.out.print("Income\t: ");
10)         income=Integer.parseInt(br.readLine());
11)         if(income<=40000)
12)            System.out.println("Tax\t: No Tax");
13)         else if(income<=60000)
14)         {
15)            tax=10.0/100*(income-40000);
16)            System.out.println("Tax\t: "+tax);
17)         }
18)         else if(income<=150000)
19)         {
20)            tax=2000+20.0/100*(income-60000);
21)            System.out.println("Tax\t: "+tax);
22)         }
23)         else if(income>150000&&income<=200000)
24)         {
25)            tax=20000+30.0/100*(income-150000);
26)            System.out.println("Tax\t: "+tax);
27)         }
28)         else
29)         {
30)            tax=1.02*(20000+30.0/100*(income-150000));
31)            System.out.println("Tax\t: "+tax);
32)         }
33)     }
34) }
```

As one can see that, previous code is modified from line 23. So test cases will be chosen such that it covers all statements from line 23, to ensure that the modified code is correct and properly implemented. Therefore test cases for this will be as follows-

| S. No. | Income | Tax (Expected Output) | Tax (Original Output) |
|---|---|---|---|
| 1 | 160000 | 23000.0 | 23000.0 |
| 2 | 200000 | 35000.0 | 35000.0 |
| 3 | 200001 | 35700.306 | 35700.306 |
| 4 | 250000 | 51000.0 | 51000.0 |

Thus, modified program is also correct; also it has not disturbed working of previous program. Hence regression testing is successful.

## IV. CONCLUSION

Testing is an essential phase of SDLC. Out of various testing techniques available mutation testing and regression technique is highlighted in this paper. Mutation testing and Regression testing are two completely different testing techniques but often resembled same by beginners due to modification associated with it. In mutation testing focus is on finding effectiveness of test suite selected, while in regression testing effectiveness of modified application is tested. Since in both process code is modified, but in regression testing modification is requested by client and is system maintenance activity while in mutation testing modification is part of testing and is fault based testing technique as faults are forcefully introduced by modification. Thus, purpose of both testing is to provide a bug free software but the techniques used for same are totally non identical.

## REFERENCES

[1]. Megha Jhamb, Abhishek Singhal, And AbhayBansal, "A Survey on Different Approaches for Efficient Mutation Testing" International Journal of Scientific and Research Publications, Volume 3, Issue 4, April 2013.
[2]. Lingming Zhang, Darko Marinov, Lu Zhang, Sarfraz Khurshid "Regression Mutation Testing", ISSTA-2012.
[3]. Y. Jia and M. Harman. "An analysis and survey of the development of mutation testing", *IEEE TSE*, 37(5):649–678, 2011.
[4]. A Jefferson Offutt, Kanupriya Tewary, "Experiments with Data flow and Mutation Testing" , Feburary 1994.