International Journal of Marketing and Technology
(ISSN: 2249-1058)

CONTENTS

Chief Patron

Patron

Editorial Board

Editorial Advisory Board

Mr. P. PRABHU
Assistant Professor, Alagappa University, KARAIKUDI

Mr. MANISH KUMAR
Assistant Professor, DBIT, Deptt. Of MBA, DEHRADUN

Mrs. BABITA VERMA
Assistant Professor, Bhilai Institute Of Technology, DURG

Ms. MONIKA BHATNAGAR
Assistant Professor, Technocrat Institute of Technology, BHOPAL

Ms. SUPRIYA RAHEJA
Assistant Professor, CSE Department of ITM University, GURGAON

Title

CLUSTER BASED MUTATION TESTING USING HOMOGENEOUS AND HETEROGENEOUS N-MUTANTS

Author(s)

Mr. Ajay Jangra
CSE Department
U.I.E.T. Kurukshetra University,
Kurukshetra, INDIA

Ms. Jasleen kaur
CSE Department
U.I.E.T. Kurukshetra University,
Kurukshetra, INDIA

ABSTRACT:

Mutation testing is to check the quality of test cases by changing code. The proposal examines mutation testing based on clusters.Clusters contains homogeneous and heterogeneous mutants. It consists two mutagens in it and by one time only one mutagen (mutant) can be selected for performing mutation testing. Mutants can be homogeneous type or it can be heterogeneous mutants. Selection of number of mutants depends upon number of clusters working for the code. This paper represents cluster based testing using homogeneous and heterogeneous mutants. This is divided into three sections. Section 1, Section 2 and Section 3 described about homogeneous, heterogeneous and hybrid (homogeneous/ heterogeneous) mutants. For mutation generation, a tester can generate mutants according to number of clusters. Number of cluster is equivalent to number of mutants. After modifying original code, collaboration of all mutants performed which shows by end a tester can perform mutation on huge number of mutants. This paper explores the ideas of collaboration of cluster covers large number of mutants.

Keywords: Mutant, Mutation Operator, Equivalent Mutant, Test cases, Clusters.

INTRODUCTION:

Mutation testing (fault based testing) checks the quality of test cases by feeding wrong data into original code. Changes in original code makes mutant in program called as mutated program. It works on single mutant for checking the efficiency and effectiveness of test cases. Mutation testing is white box testing tests the internal structure of the code to detect all the mistakes in real code. Figure 1 represents the complete scenarios of the mutation process [1, 2].

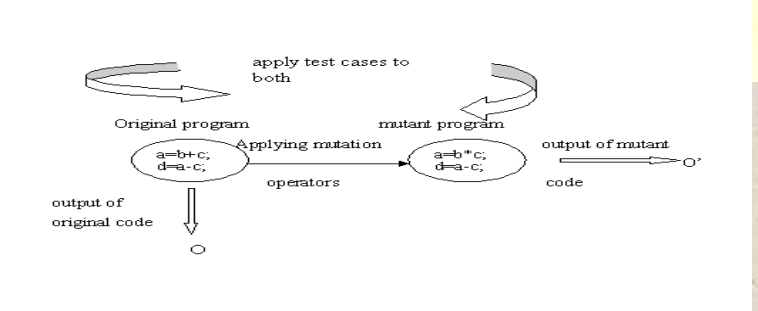

Figure: 1-Mutation process

A correct program can be shown in many wrong ways so mutation testing work under this hypotheses by making correct program to incorrect forms by using fault seeding concept. Fault seeding is done by using different kinds of mutation operators. Mutation operators are the basic and simple rules in mutation testing for the transformation of the original code in many other various mirror images with little changes in them. Mutation operators change one operand to another operand or may delete the entire statement [3, 4, 5].

PREVIOUS WORK:

- Work done on mutation testing is like some syntactic changes in the procedural and object oriented languages, types of mutation operators were introduced and different tools were used to show the mutation testing [6, 8].
- Two types of mutation testing are described in paper "Weak mutation testing and completeness of test sets" that one is strong mutation testing creates the mutant for the whole program and another one is weak mutation testing creates the mutant in the single statement. If the output shows the same result as the original program then that test case is not either effective or efficient [4,7].
- Different classification of mutation operators has defined i.e. method level mutation operators and class level mutation operators where the method level deals with arithmetic syntactic changes and class level deals with the combination of non object oriented programming and object oriented programming language in which the program contains polymorphism, inheritance, templates and exceptional handling [11,12, 13].

PROBLEM DOMAIN:

- **Expensive** due to generation large number of mutant[9]
- **Equivalent mutant's** execution gives the same result and it's very difficult to detect. It also shows the disadvantage in terms of cost and time [10].
- Development of mutagen for different code increases **effort** and **time** [5].

PROPOSED MODEL:

Testing approach called fault based testing is presented for overcoming the drawbacks of other testing approaches. However it seems to be expensive due to generation of single mutagen (mutant) version copies for the original code. Creation of different mutated codes for checking original one is time consuming and needs effort for testing. Mutation testing provides the clean way to achieve the goal. Clean way means effort can put in the early stages so that the less effort or time will be at last

Proposed model develops "**Cbm-nm (Cluster based mutation testing using homogeneous and heterogeneous n-mutant"** can be considered as already built in for testing. It means clusters are fixed and number of clusters depends upon the tester and the code for performing testing. Theoretical way of defining *Cluster based model* is presented here. Testing implementation depends upon the cluster. So a cluster plays an important role for coverage of n-mutants.

User can choose only a mutant at a time from cluster. So if number of cluster is four then user can pick one from each and can do coverage of 4 mutants at a time. Each cluster consists two mutants or double mutants in them. That's why the proposed model work can work on n-mutants.

Double mutants can be created by two ways such as:-

1. Homogeneous mutants: Homogeneous mutants are the same type of operators mistakes such as (+, +), (-,-), (*,*), (/, /), (%,%).
2. Heterogeneous mutants: Heterogeneous mutants are the different type of mutation operators such as (+, -), (+,-), (+,*), (+, /), (+, %).

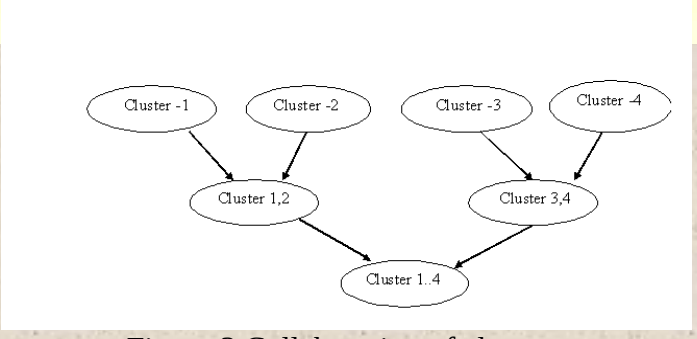

Figure-3 Collaboration of clusters

Figure 3 shows after generating mutants, cluster formation makes pairing with another cluster. By collaborating one cluster to another tester can pick two mutants at a time and the pairing three clusters tester can pick three mutants and so on. Therefore number of mutants depends upon number of clusters. So by the end, there is integration of all clusters in one cluster which can cover n-mutants in single execution which shows high coverage of mutants.

There are 3 sections for performing mutation testing.

Section 1: Homogeneous mutant's coverage with values a=9, b=8, c=7, d=9.
Section 2: Heterogeneous mutant's coverage with values a=9, b=8, c=7, d=9.
Section 3: Hybrid mutant's coverage (Homogeneous/ heterogeneous mutants).

```
main()
{
 int a,b,c,d;
printf("\nenter  the values a,b,c,d");
scanf("%d  %d  %d  %d",  &a,  &b,  &c,
&d);
if( a>b  &&  a>c && d<a)
{
 return a;
}
else if(b>c  &&  d<b)
{
return b; }
else if( d<c) {
return c; } else {return d; } }
```

Figure 3 Original code to be tested

Figure-4 generates homogeneous, heterogeneous and hybrid cluster for mutation. Cluster 1 & cluster 2 is for section 1, Cluster 3 & cluster 4 is for section 2 and Cluster 5 & cluster 6 is for section 3.

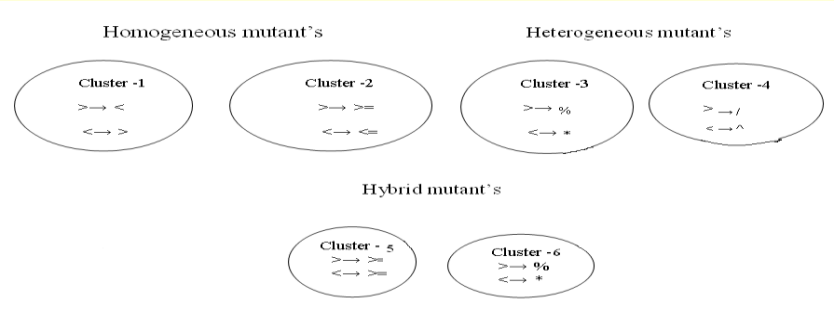

Figure 4- formation of clusters

How mutation testing do coverage with these clusters is shown below:-

Section 1: Select m1 from c1 and m2 from c2 and do checking of dual mutants in single execution. It gives different result b=8, which shows high coverage and develop highly efficient test case.

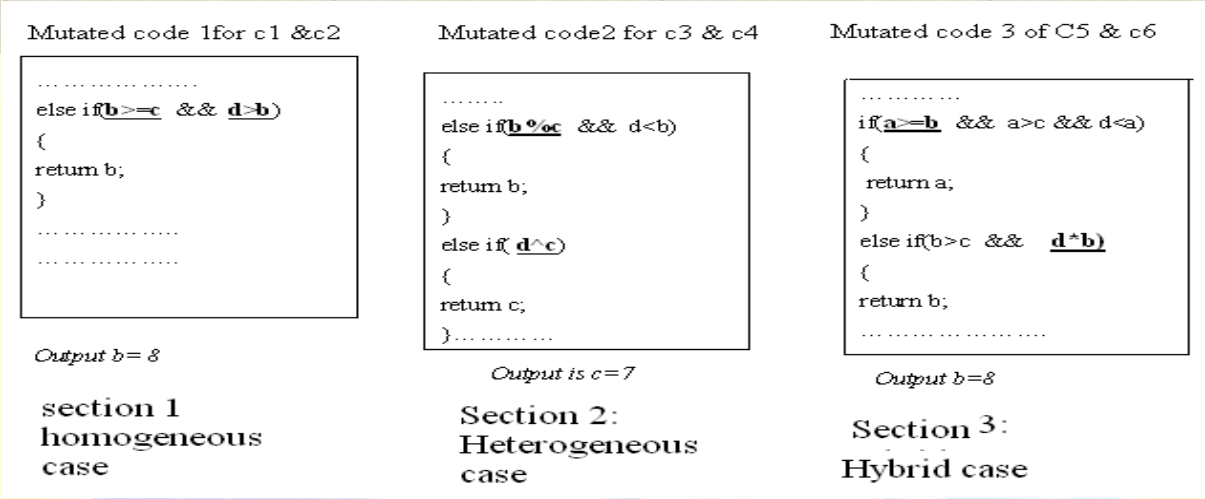

Figure 5- checking on dual mutants

Section 2: Select m1 from c3 and m2 from c4 and do checking of dual mutants in single execution. It gives different result c=8 reveal high coverage and develop highly efficient test case.

Section 3: Select m1 from c5 and m2 from c6 and do checking of dual mutants in single execution. It gives b=8 different output from original. Hence develop highly efficient test case.

RESULTS:

Formation of different clusters within a single code and combining them later shows higher coverage of mutants. So heterogeneous and homogeneous concept shows only the huge coverage in them for performing mutation testing in which clusters are important to define.

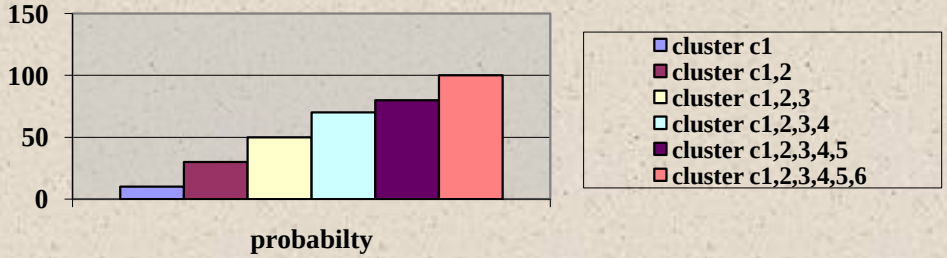

Figure 6 Probability determination

Figure 6: explores that combination of all mutants shows the high probability of coverage of mutants which can generate large mutants and can generate highly efficient and effective test case for testing.

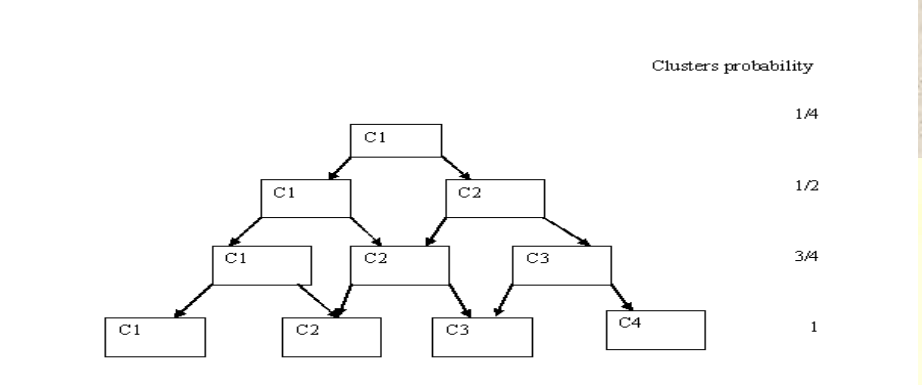

Figure 7: Hierarchal representations of clusters

Graph 7 depicts that as the cluster moves downwards by attaching with each other then number of mutants also increases which means that last collaboration shows the higher large number of mutants.

Number of clusters α number of mutants

Probability of clusters mutants = number of mutants chosen/ total number of cluster

Maximum probability can be 1 which means c1+c2+c3+c4 =1.

Maximum probability of coverage can be considered as 1.

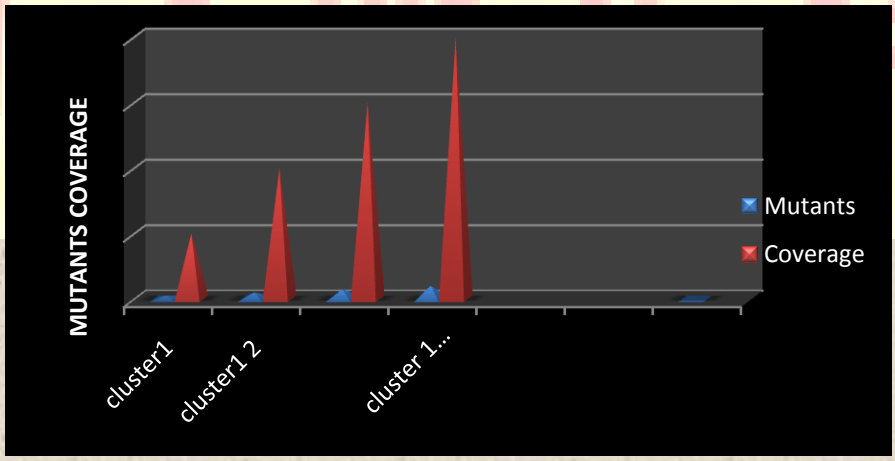

Figure 8: Mutants coverage against clusters

So mutant's execution can be done in single execution at last cluster combination which shows in figure 8.

Figure 9 shows benefits of proposal work. It increases the effectiveness and efficiency of the test cases and decreases the time, cost and effort. This can be considered as future scope of the dissertation. By making automatic framework for selecting one mutant from each pre-defined clusters and executing those meta mutants by single execution which never generate the bad test case or failed test case.

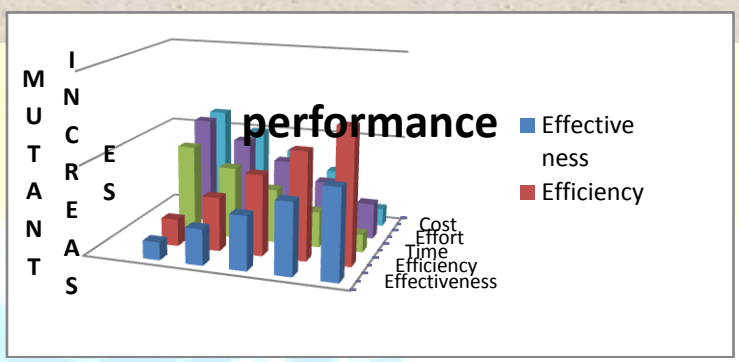

Figure 9: Performance

MAJOR ACHIEVEMENTS:

The proposal reveals the following achievements:-

- Quality of test cases increases.
- Effort gets reduced.
- Budget and time also decreases because the n-mutant worked efficiently.

APPLICATION AREAS:

Mutation testing can be useful in software engineering as well as in medical field.

- To check the quality of test cases, mutation testing can be applied to software's.
- Automatic generation of effective and efficient test cases.
- It can be used to check the circuit system as well by changing the behavior of the circuit, by removing the functionality of the circuit or by inserting nee functions in to the circuit.

- It can also be used in medical purpose like for DNA checking. Each gene of the human being contains chromosomes which can be test by applying mutation into those chromosomes in a different manner.

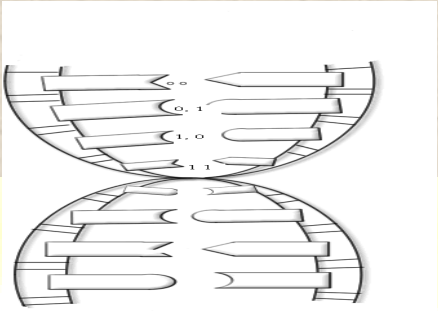

Figure-10: DNA structure testing

By applying mutation testing in the form of (0,0), (0,1), (1,0),(1,1). There can be probability of 2^n cases for each cluster. Suppose the single DNA part contains the four parts has probability of 2^n where n=4 that is 16 cases.

CONCLUSION:

Mutation testing analysis with homogeneous and heterogeneous mutants describes checking of n-mutants. By getting through all the necessary details of the mutation testing, result shows great benefit in the problem area of cost and time. Till today the authors described about single mutant and this paper reveals homogeneous and heterogeneous mutants testing concept. So instead of creating single mutagen, n-mutants can be generated with predefined clusters which can reduce time and cost instead of that it shows the great benefit in the area of mutation testing.

REFERENCES:

- WILLIAM E. HOWDEN "Weak Mutation Testing and Completeness Of Test Sets" IEEE 1985.
- A. Jefferson Offutt "Investigations of the Software testing Coupling Effect" Clemson University.
- T. A. Budd. *Mutation analysis of program test data*. PhD thesis, Yale University, New Haven CT , USA, 1980.

- T.H.Tse, T.Y. Chen, X.Feng "On the completeness of Test Cases for Atomic Arithmetic Expressions" IEEE 2000.
- Huo Yan Chen, Su Hu "Two new kinds of class level mutants for object oriented programs" IEEE 2006.
- Yue Jia, Mark Harman "An Analysis and Survey of the Development of Mutation Testing" King's College London, Technical Report.
- Hiralal Agarwal, Richard A. DeMillo, Bob Hathaway, William Hsu, E.w. Krauser, R.J. Martin, Aditya P. Mathur, Eugene Spaord "Design of mutant operators for C programming language" 2006.
- Laurie Williams "Mutation testing" Research group software engineering in NCSU 2007.
- M. R. Woodward "Mutation testing—its origin and evolution" Department of Computer Science, Chadwick Building, University of Liverpool, PO Box 147, Liverpool L69 3BX, UK.
- John A. Clark and Haitao Dan and Robert M. Hierons "Semantic Mutation Testing"
- Proceedings of the 5th International Workshop on Mutation Analysis (MUTATION'10)Paris, France, 6 April 2010.
- Mrs. R. Jeevarathinam, Dr.Antony Selvadoss Thanamani "Test Case Generation using Mutation Operators and Fault Classification" 2010.
- L. Zhang, S.-S. Hou, J.-J Hu, T. Xie and H. mei "Is Operator-Based Mutant Selection Superior to Random Mutant Selection?" in Proc. ICSE, 2010.
- Ajay Jangra' Jasleen Kaur "Operator Based Efficient Mutant Generation Methods for Procedural and Object-Oriented Languages" IJCST vol2 issue2, 2011.