Sana Khan et al. International Journal of Institutional & Industrial Research ISSN: 2456-1274, Vol. 1, Issue 1, Jan-April 2016, pp.6-9

An Analysis and Survey of Equivalent Mutant Problem

Sana khan

Department of Computer Science, Muhammad Ali Jinnah University, Islamabad, Pakistan

***Abstract*--Equivalent mutant problem is the most decisive problem in mutation testing and from decades efforts are doing for its betterment. Equivalent mutants are artificial seeding defects in program to certify mutation testing. Various techniques are anticipated that are efficient for finding equivalent mutant. In this survey different approaches are deliberated to analyze the performance of different approaches.Different techniques are satisfied by different parameters.**

***Keywords*— Equivalent Mutant, Mutation Testing, Mutant, higher order Mutation.**

1. Introduction

Testing is the crucial process to certify software quality. The precision and quality of a software product depends upon that how comprehensively it is tested. A software testing technique which are most significant in testing era known as Mutation testing which is proposed by Hamlet and DeMillo[1]. It is known as fault-based technique and has proved to be a valuable testing technique. Mutation testing is based upon inserting Faults Data or artificial defects in the original program through mutant operator to identify that testing is defining this fault or not. In mutation testing if at least one test case is failed then the mutant is said to be detected or killed and those which are not detected or killed are known as alivemutant. [1] Outmoded Syntactical Mutation Operators may be deletion of a statement, Boolean expressions, arithmetic operators or variables accumulation.

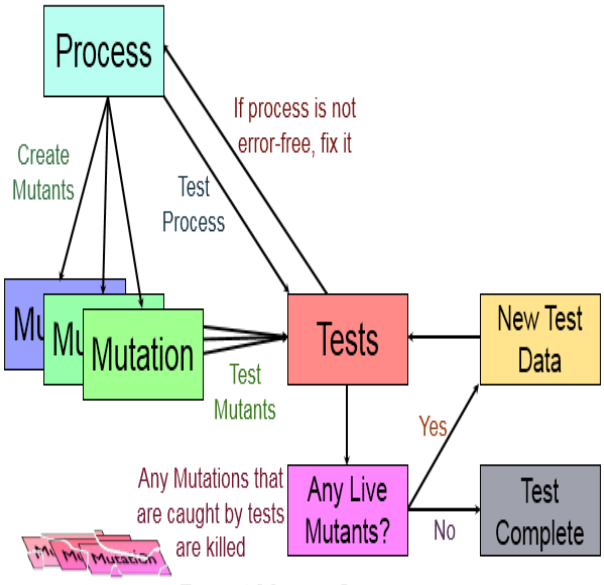

Figure.1 Mutation Process

Effectiveness of test suite defined by Mutation testing that how much mutants are killed by test suite. Mutation is unit based level technique. Mutation operators are used to produce mutants in program.
The mutation process described in following steps as shown in Figure.1 [11]

1.1 Equivalent Mutant Problem:

Alive mutants that cannot be killed are known as Equivalent Mutants. Mutation testing can never be get 100% without distinguishing all equivalent mutants. Finding equivalent mutants by hand is time consuming and make mutation testing highly cost able that's why tester will not have full assurance about defects in program and test data. The main advantage of equivalent mutant problem over general equivalence problem is that we do not examine equivalence of arbitrary pair of programs because mutants are syntactically like original program so we can develop techniques to find equivalent mutants by using this fact [2].

This survey paper introduces equivalent mutant problem. In sec. II background of equivalent mutant is described. In sec.III the existing techniques algorithm detail of equivalent mutants are described. In sec.IV the evaluation criteria between techniques are presented. In sec. V analysis is described. In sec.VI the conclusion is presented.

2. Background

This section describesequivalent mutant problem, all generated mutants should be killed as its a requirement of testing criteria in mutation testing. Application of mutation is slowed down by equivalent mutants. During generation of test cases and executing phase, it kills the mutants, the computational resources are surplus. For undecidable nature of mutants manual analysis required. Detection of mutants by hand is time consuming and its difficult to improve efficiency of program. Recent

studies show that manually detection takes approximately 15minutes to complete task and estimated equivalent mutantsrange lies between 10% to 40% [3]. To rheostat this problem, there is need to develop mechanism and algorithms which work efficiently. Compiler optimization techniques and Analysis of Equivalence Class Mutant Circuit Distributions are oldest one, butthe existing techniques works and focuses on the detection of equivalent mutant more precisely including Data flow patterns, Mutant Evaluation by Static Semantic Interpretation, through constraint systems, code similarity, through feasible path problem and many more which are working on the detection of equivalent mutants. Some existing techniques are described in sec.3.

3. Existing approaches

There are numerous existing techniques to detect equivalent mutant problem which is deliberated here with brief summary.

3.1 Bayesian Learning-Based Equivalent Detection Technique:

This technique support the tester to find live mutants. This technique used Brute-Force algorithm to estimate stronger group of mutants. Its experimental description consist of four steps to provide guidelines to help the tasks of determining equivalent and non-equivalent mutants.

1) Program Selection: In this 5-UNIX program used. They are simple but we can find Bayesian Learning in other program domains too.

2) Tool selection:Proteum was used to support mutation Analysis. This tool help in unit level testing and implement mutant in C programs.

3) Test set generation: 500 test cases was formed for 5-UNIX programs.

4) Results and data analysis: A test session was created for programs using mutant operators. Mutant operator were applied on 5-UNIX programs, 15 generate no mutant and get no statistical information and then used Proteum to enable and disable test cases. By adding more test cases the variation was check in the number of equivalent and non-equivalent mutant. Test cases generate 19 subsets to collect information about live mutants produced by specified operator to be equivalent or non-equivalent then Bayes theorem was applied to check the probability of mutant being equivalent. Execute 19 test cases by Bayes theorem and after 20 probability of live mutant being equivalent increases. So concluded that by increasing the execution of test cases higher the assurance that live mutant is equivalent [4].

3.2 EqMutDetect – Tool for Equivalent Mutant Detection in Embedded Systems:

EqMutDetect is a tool which is developed in Java and used for detection and deletion of equivalent mutants. Tool itself comprises of Generation of mutants, Detection of mutants, Mutation test case generation. Several implementation done by EqMutDetect. Tool distinguish test cases that kill the mutants. If there was no test case then this tool identified an equivalent mutant. Mutation score enhanced by adding test cases in test suite. This tool used in embedded system software where data types are in loops, conditions and integers and give effective output. [5]

3.3 Higher Order Mutation:

Three programs TCAS,TRIANGLE,TAXLEVELCALC were selected for reducing equivalent mutant and programs are executed in C language.2 programs were downloaded from SIR and the third one is written by self for producing equivalent mutants. MiLu tool is used to generate mutants. MiLu generate first order mutant (FOMS) of all these programs. Fixed sixe mutant sampling were used for FOMS. By using C, compiler executed FOMs manually to check that FOM is equivalent or not. This was done by considering that which test case kill the FOM. MiLu also generate second order and random order mutants. From a pool of 400, 200 for second order and 200 for random mutants test cases was selected. Same pattern repeated for second order and random order to check that FOM is equivalent or not. TCAS program test cases were downloaded from software and for TRIANGLE and TAXLEVELCALC done manually, where it's difficult to kill mutants the code were check manually that test cases are capable to kill the mutants or mutant is equivalent. The result was confirmed by different categories of mutants with the help of pie charts and bar charts. HOM testing proves that it truly capable of reducing the number of equivalent mutants [6].

3.4 Code Similarity:

In code similarity mirrored mutants was introduced. In which the mutants are similar to code fragments of the program. The purpose is to improve the opposed effects of the equivalent mutant problem. In this two mirrored mutant m1 and m2 and similar code fragments cf1 and cf2 was discussed and characterized as intra-method or inter-method. It takes 3 test subject of triangle.Information deduced by all three and drawn result. Then rate calculation of kill mutants and live mutants, empirical study and empirical evaluation was done by using tool CCFinderX clone detection tool and mujava mutation testing framework. By doing

experimental evaluation it is concluded that mirrored mutants and equivalence has same behavior. The cost of equivalents can be reduced in the existence of mirrored mutants.50% of equivalent mutants can be classified automatically. [3]

3.5 Constraint Systems:

In this technique the first algorithm explained to eliminate equivalent mutant and then the algorithm defined for mutation mark of test suite. The tool used in this technique was MuJava for mutant computation and MINION constraint solver for test cases. Algorithm explained which take input as original program and give output as constraint system. Three Algorithms 1)Algorithm Transform to CSP 2) Algorithm Eliminate Equivalent Mutants 3) Algorithm Generate Test Cases were described. It is defined through experimentation that original program which was close to the mutant take as consideration. Through this technique number of equivalent mutant reduced and generate distinguishing test cases. By adding more test cases in test suite it showed that mutation score of new test suite increases and assure that mutant is not equivalent. [7]

3.6 Data Flow Patterns:

Patterns are able to identify equivalent mutant.Mutants are identified through specific paths in which mutant is equivalent to original program. Nine patterns were used in this technique and all are capable to identify equivalent mutants. Use-Def, SameLine-UD, DifferentBB-UD, Def-Def (DD), Use-Ret problematic patterns were implemented for detection purpose. Specific conditions between Variable definition and Uses were introduced for each pattern path.This is the latest technique and give tremendous result [8].

4. Evaluation criteria

4.1 Cost:

Mutation process is highly cost able process. Main disadvantage of this testing is that Mutation testing is based on creating mutants, executing mutants and calculation of mutation score which make high cost **[**9]. Cost played highly role in Mutation. Mutant Schema are used to reduce execution cost of and technique is known as program schema technique [11].Higher Order Mutation is expensive technique among all. Techniques such that Mutant sampling, selective mutation, separate compilation, schema based mutation are developed to reduce the cost probability and enhance the detection criteria of mutants. [1]

More common techniques are formed for cost aspects i.e. Randomly Selected Mutation, Constrained Mutation and Selective Mutation [4].

4.2 Performance:

Performance depend on the detection of mutants from original programs.Performance measures,more the execution of mutations higher the chance of being non-equivalent. Data Flow Pattern contain best performance among all. Higher Order Mutation proved that it is truly capable of reducing mutants and enhance performance criteria. Compiler optimization technique is lower in performance parameter.

4.3 Tool Strength:

Different automated tools MuJava, Proteum, EqMutDetect, and MilLuare used to increase the strength of algorithms in mutation testing to find mutants. They affect the performance parameter.

4.4 Time:

Time is required to identify that mutant is equivalent or non-equivalent.Second order mutation testing reduced the testing time as compared to higher order mutation. Higher order testing take much time for execution process. Empirical study shows that one single mutation take 14 minutes and 28 seconds approximately [10]

4.5 Operator Efficiency:

An evaluation are held to check compatibility of mutation operators to find equivalent mutants. According to operators 140 manually classified mutations were taken. Different Operators produces different number of mutants (col.2) [10]

The result is summarized in Table.1

TABLE I. Mutation Operators

Mutation operator	Number of mutants	Non-equivalent mutants	Equivalent mutants
Replace numerical constant	78	34 (44%)	44 (56%)
Negate jump condition	12	10 (83%)	2 (17%)
Replace arithmetic operator	7	3 (43%)	4 (57%)
Omit method call	43	30 (70%)	13 (30%)

5. Analysis

In this survey, various techniques of equivalent mutant have been discussed.

Based on survey, equivalent mutant techniques are compared through different parameters like performance, effectiveness, and efficiency and more.Different issues like recognition of equivalent mutants are increasing the computational cost as described in different techniques. Newly techniques reduces the time complexity butdiffer

in cost. Compiler optimization is old technique which predict 10% of equivalent mutant. Randomly Selected Mutation, Constrained Mutation, Selective Mutation used to reduce equivalent mutant but they don't give perfect knowledge .There are still need to improve algorithms. Then Bayesian learning and another learning Algorithm Brute-Force MAP are used to calculate percentages of equivalent mutant which give high efficiency. Constraint system also used to detect mutants but empirical studies showed that it is applicable to find 50% of mutants [4]. EqMutDetect tool give effective result in embedded system to check the quality of test suite which are used during testing. There are many tools which are used for detection purpose and have different performances according to algorithms like *Mu Java*, Mothra, *MILU*, AOIS, Proteumand many more. HOM is created by applying mutants more than once; number of mutants is reduced by using this technique. Data flow pattern is the latest technique which satisfies efficiency, performance parameters. It is satisfactory technique and literature study revealed that it helps to detect 70% of equivalent mutants [8].

6. Conclusion

Mutation testing is based upon different steps in which creation, execution and identification of mutants involved.This is a costly technique but highly preferable for detection purpose. When all equivalent mutants are removed then mutation score is significantly enhanced. Reduction of equivalent mutants shows a significant role in the efficiency of test suites. In this paper different approaches are highly preferable for detection purpose but there is more need to enhance performance and efficiency of algorithms using different tools. Emergent techniques day by day increasing value of mutation testing butin future all parameters should be standardize in perfect technique.

References

[1]. Adamopoulos, Konstantinos, Mark Harman, and Robert M. Hierons. "How to overcome the equivalent mutant problem and achieve tailored selective mutation using co-evolution." *Genetic and Evolutionary Computation–GECCO 2004*.Springer Berlin Heidelberg, 2004.

[2]. Offutt, A. Jefferson, and Jie Pan. "Detecting equivalent mutants and the feasible path problem." *Computer Assurance, 1996.COMPASS'96, Systems Integrity.Software Safety.Process Security.Proceedings of the Eleventh Annual Conference on*.IEEE, 1996.

[3]. Kintis, Marinos, and NicosMalevris. "Identifying more equivalent mutants via code similarity." *Software Engineering Conference (APSEC, 2013 20th Asia-Pacific*.Vol. 1.IEEE, 2013.

[4]. Maldonado, Jose Carlos, Marcio Eduardo Delamaro, And Roseli Aparecida Francelin Romero. "Bayesian-Learning Based Guidelines To Determine Equivalent Mutants." *Machine Learning Applications in Software Engineering* 16.6 (2005): 150.

[5]. Nica, Simona, and Franz Wotawa. "EqMutDetect—a tool for equivalent mutant detection in embedded systems." *Intelligent Solutions in Embedded Systems (WISES), 2012 Proceedings of the Tenth Workshop on*. IEEE, 2012.

[6]. Akinde, AderonkeOlusola. "Using higher order mutation for reducing equivalent mutants in mutationtesting." *Asian Journal of Computer Science & Information Technology* 2.3 (2013)

[7]. Nica, Simona, MihaiNica, and Franz Wotawa. "Detecting equivalent mutants by means of constraint systems." *VALID 2011, the Third International Conference on Advances in System Testing and Validation Lifecycle*. 2011.

[8]. Kintis, Marinos, and NicosMalevris. "Using Data Flow Patterns for Equivalent Mutant Detection." *Software Testing, Verification and Validation Workshops (ICSTW), 2014 IEEE Seventh International Conference on*.IEEE, 2014.

[9]. Mateo, Pedro Reales, and Macario Polo Usaola. "Mutant execution cost reduction: Through music (mutant schema improved with extra code)."*Software Testing, Verification and Validation (ICST), 2012 IEEE Fifth International Conference on*.IEEE, 2012.

[10]. [10] Schuler, David, and Andreas Zeller. "Covering and uncovering equivalent mutants." *Software Testing, Verification and Reliability* 23.5 (2013): 353-374.

