Better Code by Making Bugs: SWOT Analysis of Mutation Testing

Dr. Mamta Madan [1], Dr.Kavita [2] , Radhika Thapar [3]

[1]*(Professor computer science, VIPS, Delhi, Supervisor Jagannath university , Jaipur)*

[2]*(Co-Supervisor Jagannath university, Jaipur)*

[3] *(Radhika Thapar Research Scholar, Department of Computer Science, Jagannath university, Jaipur)*

Email: [1]mamta.vips@gmail.com, [2]kavita@jagannathuniversity.org

[3]radhika.thapar@rdias.ac.in

ABSTRACT: *In today's world, quality is an important factor for customer satisfaction. Quality assurance and testing are one of the important dimensions. But how can we test? If one does not find a bug, then does that really mean that there aren't any? Have any one wondered, what happens to the guards who are actually guarding us, who is the one who guard the guard's? From this analogy ,question arises how to deliver higher quality software by increasing the efficiency and effectiveness of our testing processes. Solution is that we need extensive type of testing to evaluate the quality of test cases how fault resistant are they? And how much code has been covered by them. Mutation testing is the possible solution which is basically based on this. Mutation testing purposely makes fluctuations in a program's code, then re-run a suite of valid unit tests against the mutated program. This paper presents brief description about mutation testing, its process along with the SWOT analysis of mutation testing; it is evaluated on the bases of its strength, weakness, opportunities and threats. This analysis may help in highlighting and addressing issues in order to adopt mutation testing as an efficient testing technique or not and help in answering the questions like: Whether mutation testing is a worth technique for improving our tests? Do mutation testing really tests the test? What can be possible deficiencies found if one uses this technique? Is there anything stored for research community to identify the issues and opportunities in mutation testing.*

Keywords -Testing, Mutants, SWOT, Equivalent mutants, Mutation

I. INTRODUCTION

Testing is one of the important part of any software development process. It is also a very vast field consisting of many techniques with their own pros and cons. Mutation testing is a very powerful tool to detect testing inadequacies or to check coverage on testing software. Software testers have known this method for many years. However, not many of them are using it for various reasons. There are several reasons that holds back software industry from using this testing. However, this testing has its own share of advantages and disadvantages. It has a potential to be a very cost effective form of testing. Mutation testing is a fault-based testing technique to assess and improve the quality of a test suite. [1]. Mutation testing provides a repeatable process for measuring the effectiveness of test cases and identifying disparities in the test set [2].

II. OVERVIEW OF MUTATION TESTING

The goal of mutation testing is to assess the quality of tests and use these assessments to help construct more adequate test and thus produce a suite of valid tests which can be used on real programs [3].

Mutation testing can be used to detect bugs in earlier stages of development where fixing bugs cost much less . Three important check points in testing process which needs focus are: test data generation, test execution, test checking .There is always a need for improving the quality of test methods. We are never at a position to say that a particular method is capable of determining all sorts of errors, bugs, failures. In mutation testing, mutations are created by so called mutation operators that mimic typical program errors. Mutation testing works in conjunction with the conventional testing techniques.

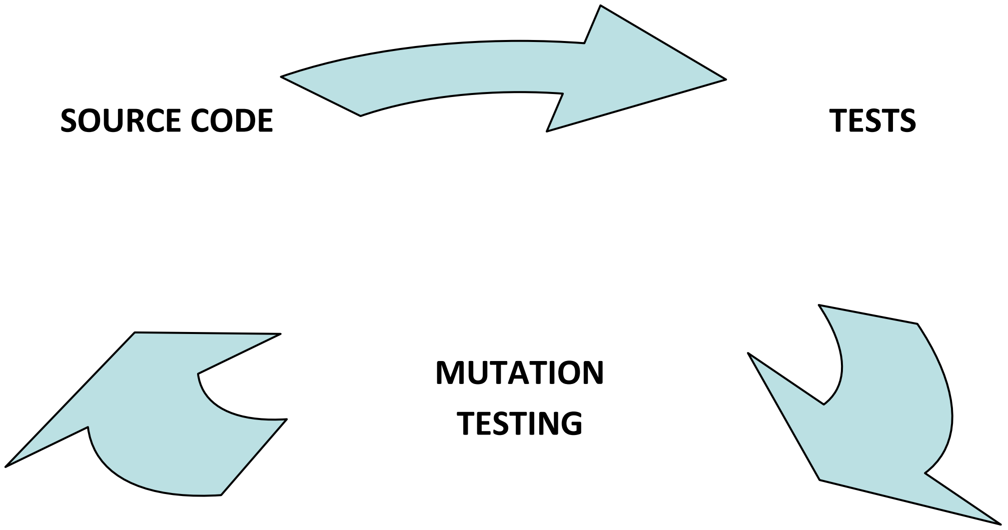


Fig. 1: Mutation Testing Cycle

Mutation testing can empower software testing process with trust and fidelity. Which will eventually help the development team to create a software application that can pass all other test procedures? Mutation testing is a method of software testing which involves modifying program's source code in small ways. A test suite which does not detect and reject the mutated code is considered defective. In mutation testing, intentionally some changes or errors are introduced in the program and then it is checked that which test cases are able to find out these errors. The goal is to make sure that during testing, each mutant (change) produces an output different from the output of the original program. Mutants are introduced when the programs are started by the mutant operators. Each mutation produces a mutant program, produced by a mutation operator [4]. Therefore, this testing can be used to check the efficiency of our test suits.

Various possible changes that can be done in a program are:

a) Replacing an arithmetic operator with some other arithmetic operator
b) Change an array reference with other array reference
c) Change the label for a go to statement
d) Replace a variable by some special value
e) The resulting program with changed statements or mutant statements is called mutant.

III. PROCESS OF CREATING MUTANT

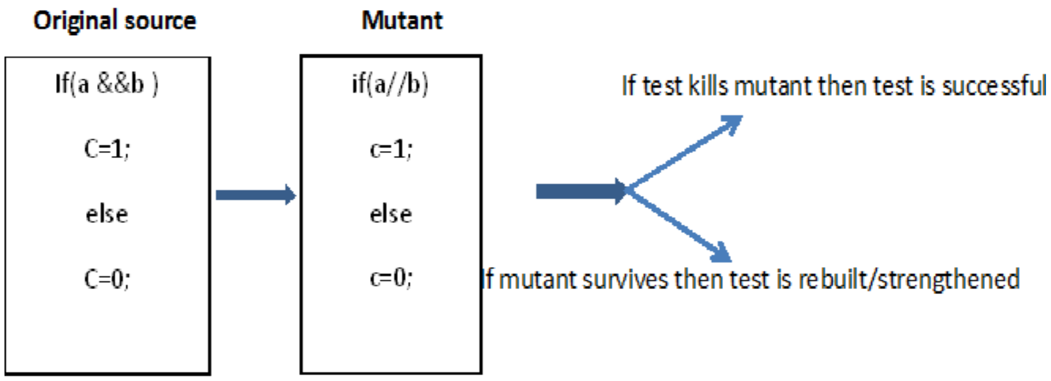
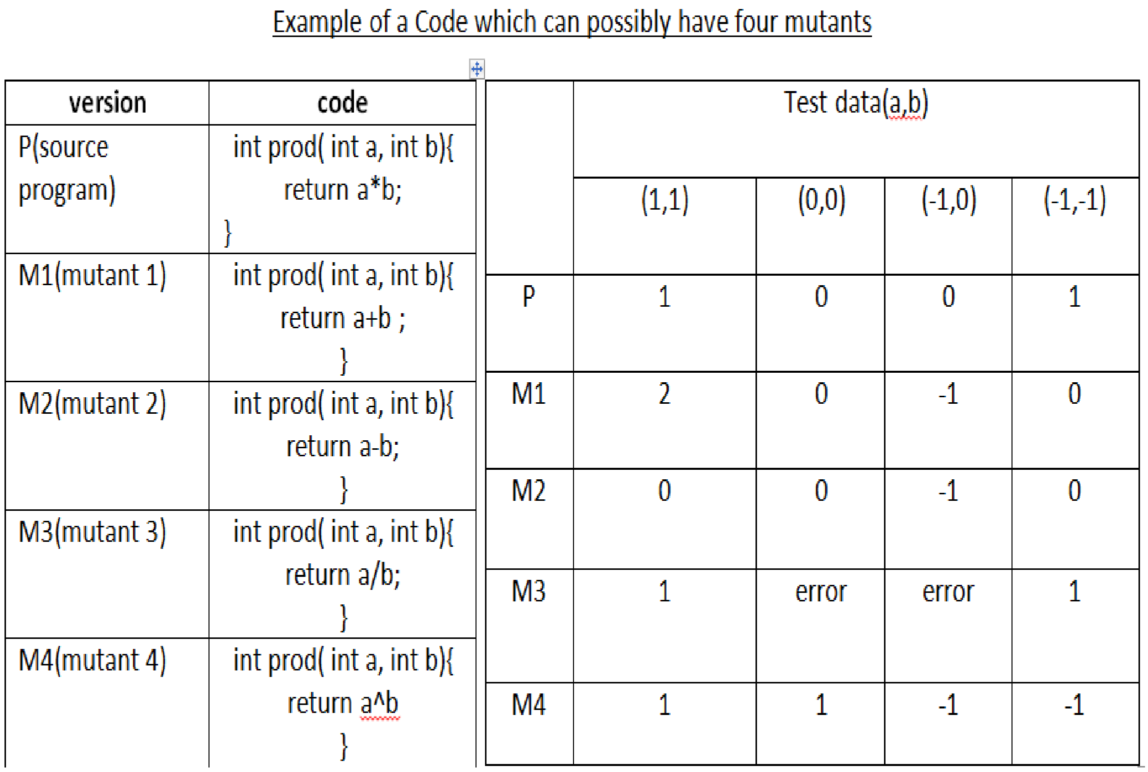

Original source
If(a &&b)
C=1;
else
C=0;
Mutant
if(a//b)
c=1;
else
c=0;
If test kills mutant then test is successful
If mutant survives then test is rebuilt/strengthened

Fig. 2: An Example to Exhibit Mutation Testing Process



Example of a Code which can possibly have four mutants

version	code

P(source program)	int prod(int a, int b){
	return a*b;
	}

M1(mutant 1)	int prod(int a, int b){
	return a+b ;
	}

M2(mutant 2)	int prod(int a, int b){
	return a-b;
	}

M3(mutant 3)	int prod(int a, int b){
	return a/b;
	}

M4(mutant 4)	int prod(int a, int b){
	return a^b
	}

	Test data(a,b)			
	(1,1)	(0,0)	(-1,0)	(-1,-1)
P	1	0	0	1
M1	2	0	-1	0
M2	0	0	-1	0
M3	1	error	error	1
M4	1	1	-1	-1

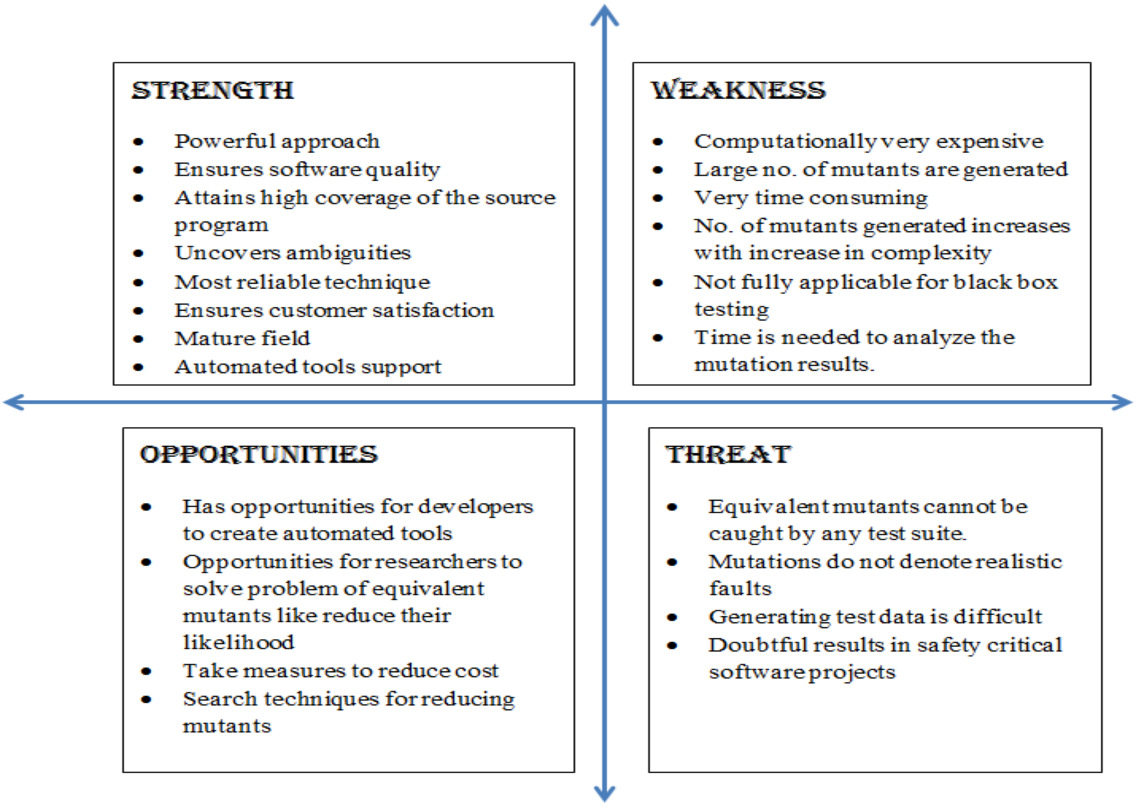
Fig. 3: Possible Mutants of Source Code

IV. WHAT IS SWOT ANALYSIS?

The SWOT analysis is a generally a framework for analyzing the factors that influence the company's competitive position in the market with an eye to the future. Every company is confronted with a variety of internal and external forces which, on the one hand can comprise potential stimulants, or on the other hand can compromise potential limitations as regards the performances of the company or the objectives the company wishes to achieve[5].However, this technique can also be usefully applied outside of the pure business domain. Keeping in mind its usefulness, this analysis is applied on mutation testing technique. It is expected that this planned examination of the factors relevant to the current and future status of mutation testing technique will throw insight in to the key issues and concerns that are important for understanding and advancing this vital application area.

V. SWOT ANALYSIS OF MUTATION TESTING TECHNIQUE

Mostly, a SWOT analysis is used to uncover the best possible match between the internal strengths and weaknesses of a given entity and the environmental trends (opportunities and threats) that the entity must face in the marketplace. Strength can be viewed as a resource, a unique approach, or capacity that allows an entity to achieve its defined goals. A weakness is a limitation, fault, or defect in the entity that impedes progress toward defined goals .An opportunity pertains to internal or external forces in the entity's operating environment, such as a trend that increases demand for what the entity can provide or allows the entity to provide it more effectively. A threat can be any unfavourable situation in the entity's environment that impedes its strategy by presenting a barrier or constraint that limits achievement of goals [6]. SWOT analysis of mutation testing techniques is summarised in Fig 4.



STRENGTH

- Powerful approach
- Ensures software quality
- Attains high coverage of the source program
- Uncovers ambiguities
- Most reliable technique
- Ensures customer satisfaction
- Mature field
- Automated tools support

WEAKNESS

- Computationally very expensive
- Large no. of mutants are generated
- Very time consuming
- No. of mutants generated increases with increase in complexity
- Not fully applicable for black box testing
- Time is needed to analyze the mutation results.

OPPORTUNITIES

- Has opportunities for developers to create automated tools
- Opportunities for researchers to solve problem of equivalent mutants like reduce their likelihood
- Take measures to reduce cost
- Search techniques for reducing mutants

THREAT

- Equivalent mutants cannot be caught by any test suite.
- Mutations do not denote realistic faults
- Generating test data is difficult
- Doubtful results in safety critical software projects

Fig. 4: SWOT Analysis of Mutation Testing

VI. STRENGTHS OF MUTATION TESTING

Following are the strengths of Mutation Testing:

- It is a powerful approach to attain high coverage of the source program.
- Mutation testing brings a good level of error detection to the software developer.
- This method uncovers ambiguities in the source code, and has the capacity to detect all the faults in the program.
- Customers are benefited from this testing by getting most reliable and stable system.
- Mutation is an excellent testing technique from the point of view of research it is mature and from the point of view of industry it is user friendly as Various Mutation testing tools exists for the software industry [7]. Like MOTHRA,PROTEUM,JUDY,MILU,NESTER,MUJAVA, JUMBLE ,MUCLIPS etc.

VII. WEAKNESS OF MUTATION TESTING

- There is possibility of having a large number of mutants which will be generated in the mutant generation phase of mutation testing. There can be faults in many possible places. By inserting one semantic fault ,there will be at least one mutant.
- Since mutation testing is time consuming, it's fair to say that this testing cannot be done without an automation tool. These tools will apply a set of mutation operators in order to achieve some syntactic changes.
- As a program's complexity increases so does the number of mutants generated which will increase the execution costs .Each mutation will have the same number of test cases than that of the original program. So, a large number of mutant programs may need to be tested against the original test suite. Approx. no. of mutants generated= No.of data reference * data objects
- For example, assume that we have a program under test with 150 mutants and 200 test cases, it requires (1+150)*200 = 30200 executions with their corresponding results.

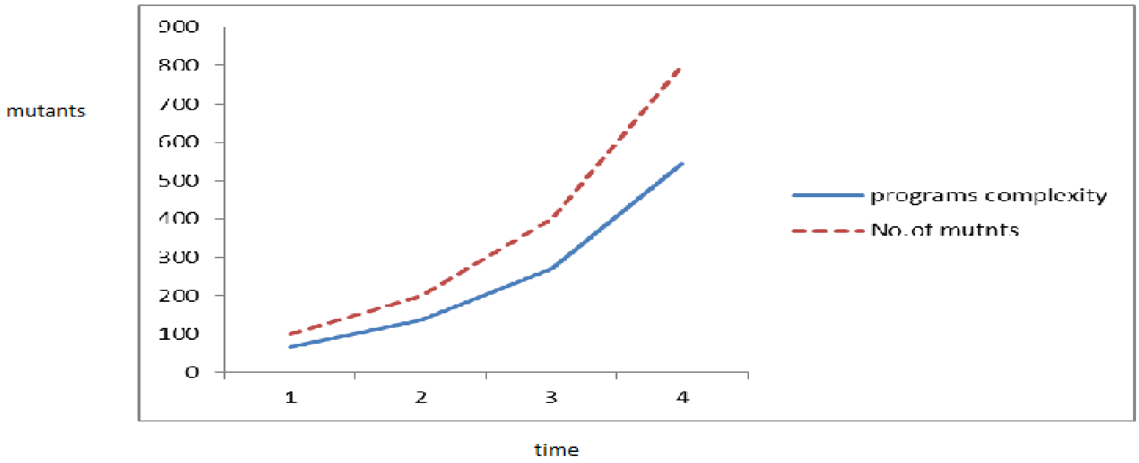


Fig. 5: Relation Between No. of Mutants Produced and Program Complexity

- As this method involves changes in source code, it is not at all applicable for black box testing.
- Developer will take time to analyse the mutation results [8].
- Computational expense i.e. Time and effort is required to perform mutation testing; and automation [8].

VIII. Opportunities

- Mutation testing is growing at a rapid rate , great deal of work can be done to extend mutation testing to new languages, paradigms and new domains of application can be searched.
- In future mutants are used for security policies to find the weak positions in security features. Efficiency of the program can also be increased by calculating the mutant operators. Mutants effectiveness can also be categorized into highly effective, effective and low effective [9].
- The development and improvement of mutation testing tools will help solve these problems and will help lead to more industry adoption of mutation testing[10]
- Very promising and powerful criteria
- Mutation is not only a white box technique but it can also be applied in black box, component, web services, models etc.
- New open source and industrial automated tools can be created, which possibly can be adopted by industrial world.
- Mutation testing is the most versatile method of software testing.
- Research can be done in developing techniques for reducing mutants.
- Customers are benefited by getting most reliable and stable system.
- High automation and effective techniques are required for evaluating the quality of the test data.
- Approaches can be worked on which seek to avoid equivalent mutants, their initial creation and reduce their likelihood.
- Reduction of cost without over compromising the quality
- Techniques can be developed in order to reduce the effect of equivalent mutants.
- It inspires additional test cases you might not have otherwise considered.
- Genetic Algorithms in evolution of mutants and test cases offers new possibilities in addressing some of the main problems of mutation testing[11]

IX. Threat

- What we were not prepared for, though, was the problem of equivalent mutants— mutations that leave the program's overall semantics unchanged, and therefore cannot be caught by any test suite [12].
- Equivalent mutants a problem many mutation operators can produce equivalent mutants have the same behavior as original program. Equivalent mutants are semantically equivalent to the original program; they

produce the same output as the original program for every possible input. Generation & execution of tests on equivalent Mutants are waste of computational time

- Realism is a threat in mutation testing. Mutations are generated by single and simple syntactic changes; hence they do not denote realistic faults. , it is not sure that we have found a large proportion of real faults present even if we have killed all the killable mutants
- Threats to validity
- Mutation testing becomes expensive for large applications, if any tools are not used.
- Generating test data is challenge as identifying a set of test data that maximizes the number of killed mutants is difficult.
- Very challenging in safety critical software projects.[13]

X. CONCLUSION

Mutation testing can be used to detect bugs in earlier stages of development where bugs cost much less to fix. The opinion that appears (to us) from the above SWOT analysis suggests that the technique of mutation testing is very powerful in order to achieve quality software. Weaknesses exist, , but do not threaten the capability of the field in light of recent and expected opportunities it is predicted that mutation testing technique will continue to gradually grow and gain acceptance as a mainstream tool.

REFERENCES

[1] SHABNAM MIRSHOKRAIE, ALI MESBAH AND KARTHIK PATTABIRAMAN, GUIDED MUTATION TESTING FOR JAVASCRIPT WEB APPLICATIONS, IN IEEE TRANSACTIONS ON SOFTWARE ENGIEERING, DOI:10.1109/TSE.2014.2371458, vol. 41 (5), pp. 429-444, Nov. 2014.

[2] R.A. DeMillo, R.J. Lipton, and F.G. Sayward, Hints on Test DataSelection: Help for the Practical Programmer, *Computer, vol. 11,no. 4, pp. 34-41, Apr. 1978.*

[3] Madhuri Sharma Center for Development of Advanced Computing, Noida ,Neha Bajpai Center for Development of Advanced Computing, Noida Automatic "Generation and Execution of Mutants" International Journal of Computer Applications (0975 – 8887) *Volume 44– No.3, April 2012*

[4] Pawan Kumar Chaurasia Assistant Professor Department of Information Technology, Babasaheb Bhimrao Ambedkar University, (A Central University) Lucknow (U.P), India, 226025 pkc.gkp@gmail.com *"MUTATION TESTING: A REVIEW"*

[5] G. Houben, K. Lenie, K. Vanhoof Department of Applied Economics, Limburg UniÕersity, UniÕersitaire Campus, 3590 Diepenbeek, Belgium *"A knowledge-based SWOT-analysis system as an instrument for strategic planning in small and medium sized enterprises" Decision Support Systems 26 1999 125–135 www.elsevier.comrlocaterorms*

[6] Albert "Skip" Rizzo Integrated Media Systems Center University of Southern California Los Angeles, California arizzo@usc.edu Gerard Jounghyun Kim Virtual Reality Laboratory Department of Computer Science and Engineering Pohang University of Science and Technology (POSTECH) Pohang, South Korea "A SWOT Analysis of the Field of Virtual Reality Rehabilitation and Therapy", Presence, *Vol. 14, No. 2, April 2005, 119–146 © 2005 by the Massachusetts Institute of Technology*

[7] A Study and Review on the Development of Mutation Testing Tools for Java and Aspect-J Programs Pradeep Kumar Singh Department of Computer Science Engineering, Amity University, Uttar Pradesh, India Email: pradeep_84cs@yahoo.com Om Prakash Sangwan School of ICT, Gautam Buddha University, Greater Noida, India Email: sangwan_op@yahoo.co.in Arun Sharma Indira Gandhi Delhi Technical University for Women, New Delhi, India Email: arunsharma2303@gmail.com, I.J. Modern Education and Computer Science, 2014, 11, 1-10 Published Online November 2014 in MECS (http://www.mecs-press.org/) DOI: 10.5815/ijmecs.2014.11.01

[8] Peter Stephen May May 2007 a thesis submitted to The University of Kent in the subject of computer science for the degree of doctor of philosophy. *"TEST DATA GENERATION: TWO EVOLUTIONARY APPROACHES TO MUTATION TESTING"*

[9] Pawan Kumar Chaurasia Assistant Professor Department of Information Technology, Babasaheb Bhimrao Ambedkar University, (A Central University) Lucknow (U.P), India, 226025 pkc.gkp@gmail.com" MUTATION TESTING: A REVIEW" Volume 5, No. 2, February 2014 *Journal of Global Research in Computer Science REVIEW ARTICLE Available Online at www.jgrcs.info*

[10] Bernhard J. M. Grun¨ , David Schuler ✁ Andreas Zeller Saarland University, Saarbrucken, Germany ¨ gruen, schuler, zeller@st.cs.uni-saarland.de *"The Impact of Equivalent Mutants"*

[11] Konstantinos Adamopoulos , Mark Harman, Robert M. Hierons" How to Overcome the Equivalent Mutant Problem and Achieve Tailored Selective Mutation Using Co-evolution, *Volume 3103* of the series Lecture Notes in Computer Science pp 1338-1349

[12] Johnathan Snyder, Department of Computer Science, University of Alabama Adviser: Jeff Gray, Department of Computer Science, University of Alabama "Analysis of Mutation Testing Tools"

[13] D. Daniels, R. Myers, and A. Hilton, "White Box Software Development," Proc. *11th Safety-Critical Systems Symp., Feb. 2003*