Academia of Information Computing Research
Vol. 3, No. 1, May 2022, pp. 1-9
ISSN: 2716-6465

A Review on Search-Based Mutation Testing

Nor Ashila Abdul Rahman[1], Rohayanti Hassan[1], Johanna Ahmad[1], Noor Hidayah Zakaria[1], Sim Hiew Moi[1], Nor Azizah Sa'adon[1]
[1]Faculty of Computing, Universiti Teknologi Malaysia, Skudai, Johor, Malaysia
ashilarahman@gmail.com, rohayanti@utm.my, johanna@utm.my, noorhidayah.z@utm.my, hiewmoi@utm.my, azizahsaadon@utm.my

***Abstract*— Big Data is a larger and more complex collection of datasets that exceeds the processing. In order to improve the productivity of non-testable Big Data, machine learning is able to determine various types of high volume, velocity and variety of data that need to be processed. Search-based mutation testing works by formulating the test data generation/optimization and mutant optimization problems as search problems and by applying meta-heuristic techniques to solve them. This paper aims to present the researches carried out in mutation testing particularly in search-based approaches. 205 papers were reviewed and analyzed from 2014-2018. This paper later on proceeds to elaborate on SBMT functions, First and Higher Order Mutant as well as multi-objective optimization.**

***Keywords*—Mutation Testing, Search-Based, First Order Mutant, Higher Order Mutant, Multi-objective**

I. INTRODUCTION

Big Data has been seen as a crucial challenge in software testing process because the testing of big data will involve a reliable analytical software in order to increase the quality assessment of a software program or application. By definition, Big Data is a larger and more complex collection of datasets that exceeds the processing and analytic capacity of conventional software testing techniques [3]. The essential aspect in the process of software testing is to reduce the cost and enhance the benefits [1]. Under such circumstances, analyzing of Big Data takes longer time since it require more verification of its data processing rather than testing the individual features of the software product. In order to improve the productivity of non-testable Big Data, machine learning is able to determine various types of high volume, velocity and variety of data that need to be processed [4].

As a type of software testing, mutation testing works when certain statement in the source code are changed and check of the test case are able to find the errors. Mark [8] claimed that mutation testing was one of effective way to test program in order to determine the quality of the test cases because each test case should be sufficient enough to identify mutant code. The main objective of mutation testing is to support generation of test case set by introducing faults in mutants that are not present in the original program in order to improve reliability of the program [6]. Mutation testing can be described as a fault-based testing [19] technique which involve process of detecting hidden defects which might be impossible to identify using the conventional testing techniques. Figure 1 demonstrate the process of mutation testing. Mutation is derive from mutant operators which is produces by making changes in the original source code by applying syntactic changes. Test cases are then used to determine changes, if different output are produce from the original program and mutant program. Otherwise, another test case execution are require, if same output are produce from the original program and mutant program, and mutant is kept alive. Apart from that, Mark [18] classified that there are two types of mutants which are: (a) First Order Mutant (FOM) and (b) Higher Order Mutant (HOM).

TABLE I. EXAMPLE OF MUTANT PROGRAM

Original Program (P)	**Mutant (P_1) FOM$_1$**	**Mutant (P_2) FOM$_2$**	**Mutant (P_3) HOM (FOM$_1$ + FOM$_2$)**

read (a);
read (b);
sum=a+b;
if(sum <100)
sum = sum * 2;
print(sum)
	read (a);
	read (b);
	sum=a+b;
	if(sum <100)
	sum = sum + 2;
	print(sum)
		read (a);
		read (b);
		sum=a-b;
		if(sum <100)
		sum = sum + 2;
		print(sum)
			read (a);
			read (b);
			sum=a-b;
			if(sum <100)
			sum = sum -+2;
			print(sum)

Search based have been extensively used in software testing process as well as in several optimization problems. Search based mutation testing (SBMT) usually make use of a meta-heuristic technique [20] for the automation or partial automation a testing

task. A meta-heuristic can be used to generate test data using specific fitness function in order to search good solution within a search space. In SBMT, using meta-heuristic shall be able to encounter search problem such as test data generation, test data optimization and mutant optimization problems [1]. Additionally, according to Rodolfo [6] point out that testing process should be fully automated because during testing process, software tester are require to: (i) specify the test data to satisfy all code coverage, (ii) prioritize the test set and (iii) identify an equivalent mutants and unnecessary requirements and equivalent mutants. Therefore, the concepts of meta-heuristic search techniques were applicable to be implemented since high-level framework works used in heuristic were able to encounter computational cost problems involving combinations of results.

This paper aims to present the research carried out in mutation testing particularly in search-based approaches. Seeing that systematic literature review (SLR) has been seen as an essential step in the research process in order to summarize existing evidences concerning a technology and to identify gaps in current researches [3], this approach will be used as a guideline to help achieve the objectives of this study.

The remainder of this paper is structured as follows: Section 2 provides the details of research method for this SLR, highlighting the research questions and the used strategy. Section 3 analyses the studies in mutation testing followed by Section 4 that will present the result and discussion. Finally Section 5 describes the conclusion of this paper.

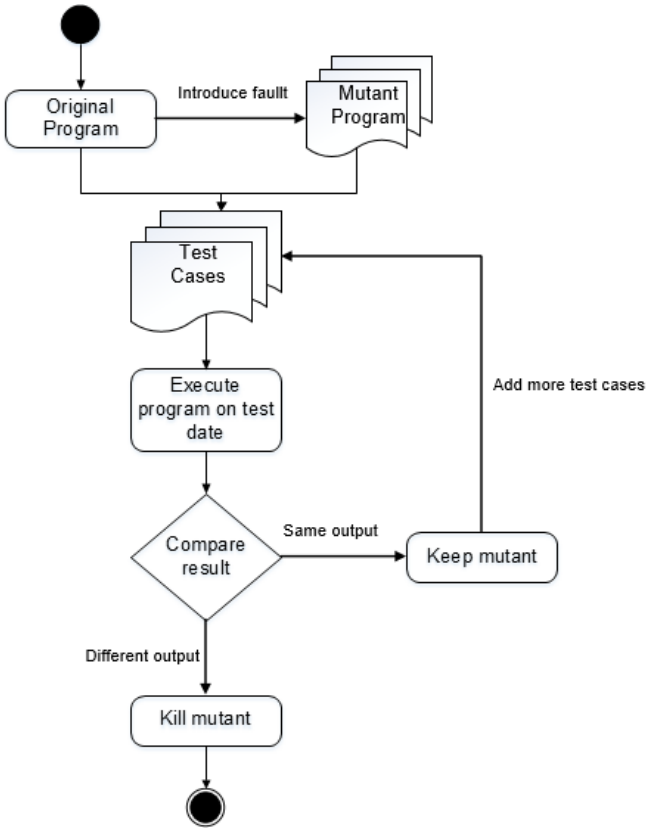

Fig. 1. Mutation testing process

II. RESEARCH METHOD

A systematic literature review is a process of identifying, evaluating and interpreting all accessible studies on a particular area or research question. Figure 2 demonstrates the overview of systematic review process used to carry out this study following Barbara's [7] guidelines. The objective is to provide an analysis of gaps in current researches with respect to suggested areas for future researches.

A. Defining Scope and Objectives

This paper aims to identify different types of studies that have been carried out in the area of SBMT. The automation of the software testing activity by Meta-heuristics has been applied extensively, thus this SLR focuses particularly in the point of view of SBMT classification and process.

This study reviews the publications in the field of SBMT from 2014 up until December 2018. This SLR primarily aims to review the studies in the field of SBMT up to 2014. However, a few relevant studies from 2015, 2016 and 2017 are also cited.

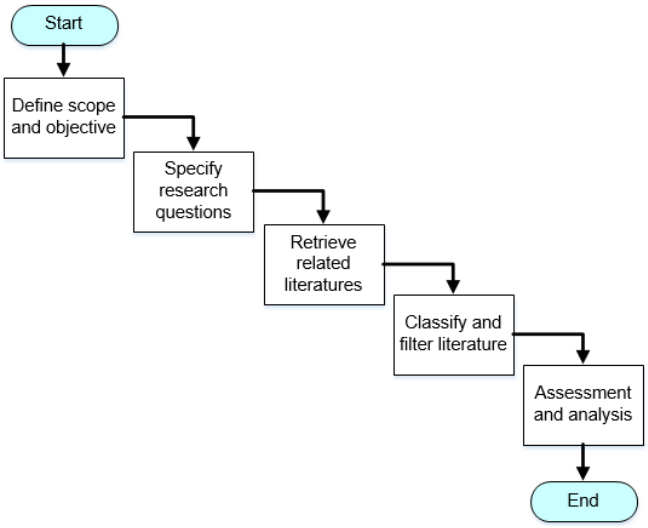

Fig. 2. Overview of SLR process

B. *Research Question*

The aim of this study is to find out how search-based approaches have been explored particularly in mutation testing. The following research questions (RQs) have been raised:

RQ1*: What is Search-Based Mutation Testing (SBMT) and what is its purpose?*

To get a better understanding of search-based mutation, we should first determine the correct definition of SBMT and the circumstances in which it is used. This leads to RQ1.1:

RQ1.1: *What is First Order Mutant (FOM) and Higher Order Mutant (HOM)?*

The first aspect in SBMT needs to be discovered in order to implement the concept of subsuming of HOM. In RQ1.2 we investigate the capability of FOM and HOM.

RQ1.2: *What are the capabilities of FOM and HOM?*

According to Rodolfo [6], the concept of a subsuming HOM is that it is more difficult to be killed in comparison with FOM that composes it.

RQ2: *How SBMT work on multi-objective test data generation?*

There is a broad range of specific testing activities in which mutation testing can be used such as test data generation and fault localization.

C. *Retrieve Related Literatures*

This study begins with searching queries and collecting papers regarding keywords “mutation testing” or “search-based mutation testing” in their titles, abstract and keywords. Four selected repositories are explored in order to retrieve the related publications tabulated in Table 1. A few duplicate journal copies found in more than one repository during the search are take-out manually.

TABLE II. DATABASES USED IN SEARCH

Database	Source
ACM	http://dl.acm.org

Science Direct	http://www.sciencedirect.com
IEEE Xplore	http://ieeexplore.ieee.org
SpringerLink	http://link.springer.com

D. Classified and Filtered Literatures

Inclusion (IC) and exclusion (EC) criteria are determined to verify the significance and usefulness of this study. The following inclusion and exclusion criteria are defined for the refinement of the result:

a) IC-1: Studies that must apply SBMT to enhance mutation testing.
b) EC-1: Studies that optimize mutation testing without using SBMT.
c) EC-2: Studies that apply SBMT technique to different testing criteria.
d) EC-3: Studies that present neither mutation testing, nor SBMT.

The search strings are constructed based on the keywords of the RQ such as mutation testing and search-based mutation testing. Figure 3 indicates the search string used to filter publications related to the RQ. The search strategy is carried out on title, abstracts and keywords in order to ensure the maximum retrieval of significant papers. In addition, this study explicitly focuses the fields of searching in computer science publications therefore papers which are not relevant to field of study are eliminated straightaway throughout the search.

Search String

("mutation test" **OR** "mutation testing" **OR** "search based" **OR** "search-based" **OR** "search based mutation testing" **OR** "search-based mutation")
AND ("first order mutation" **OR** "high order mutation" **OR** "multi-objectives" **OR** "mutation analysis" **OR** "fault based testing" **OR** "fault-based mutation testing")

Fig. 3. Search string

E. Assessment and Analysis

After classifying and filtering repositories, 205 papers are returned using the search string. Table 2 presents the number of papers by respective repository. Meanwhile, Figure 4 depicts the number of papers after exclusion from the repositories

TABLE III. NUMBER OF RETURNED PAPERS FROM REPOSITORIES

Database	Number of Returned Papers
ACM	94
Science Direct	31
IEEE Xplore	61
SpringerLink	19

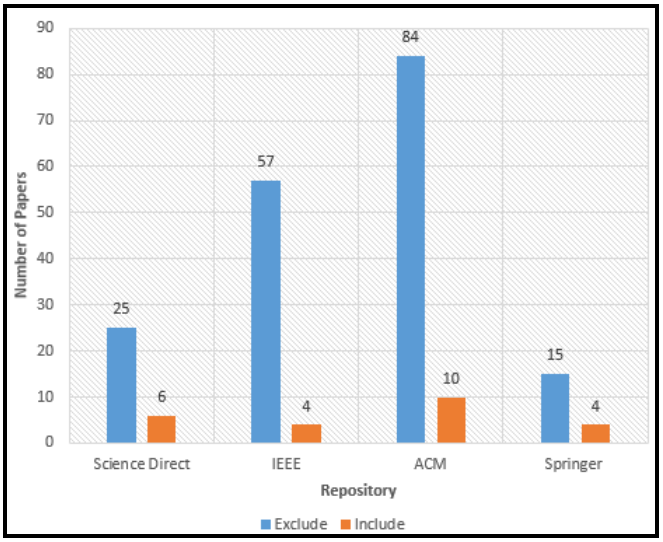

Fig. 4. Number of included and excluded papers

During the analysis, three points of interest particularly related to our RQ are derived. Figure 5 represents the number of papers that have been grouped according to the RQ. The points of interest are: (a) the roles of SBMT; (b) the concept/process of FOM and HOM; (c) the classification of multi-objective test data generation in SBMT. Through reading procedure, the corresponding details in each paper are identified and highlighted in order to answer the RQ for each point.

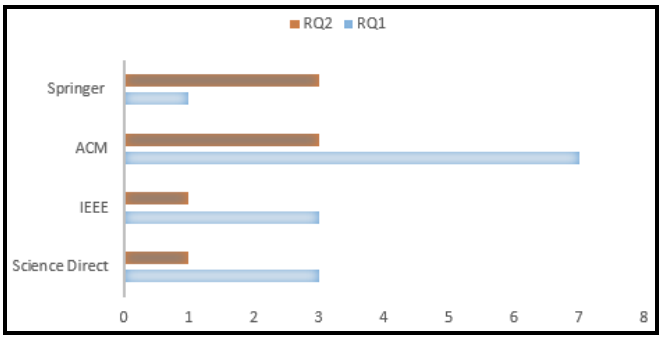

Fig. 5. Number of papers according to RQ

(a) The roles of SBMT

The first point concerns the roles of SBMT in software testing. SBMT consists of two feature combination in order to solve optimization problem. Mainly the description of features is particularly read to identify and classify each role in SBMT. The challenges or limitations in performing SBMT are also identified and listed out.

(b) The concept/process of FOM and HOM

The second point of interest focuses on the types of mutant which are First Order Mutant (FOM) and Higher Order Mutant (HOM). The papers are classified by reading the description to gain an overview of the 'subsuming' concept in HOM. The advantages of implementing HOM are also determined.

(c) The classification of multi-objective test data generation in SBMT

In multi-objective test data generation, the subjects of interest are the factors in solving optimization problems. There are a number of multi-objective algorithms that successfully apply with each algorithm adopting different evolutions and strategies.

III. Result and Discussion

This section provides answers to the RQs of the study. After reviewing and analyzing 24 papers in SBMT, the objective of finding out how search-based approaches can be explored particularly in mutation testing has been achieved.

A. RQ1 – Search-Based Mutation Testing

In general, Search-Based Mutation Testing (SBMT) was used to solve optimization problems by combining together meta-heuristic techniques with mutation analysis. In order to minimize cost in mutation testing, meta-heuristic techniques were significantly used to: (a) optimize test case generation; (b) test case prioritization; (c) select and minimize mutant or mutant optimization. Nishtha [8] stated that meta-heuristic optimization algorithm and its relative fitness function [9], [10] are determined according to the problems in SBMT. Apart from that, Nishtha [8] also mentioned the encountered challenges or limitations to applied SBMT that include: (a) the difficulty in selecting suitable mutation testing tool; (b) the time-consuming program and situation whereby mutation testing tool interferes with test data; (c) the condition whereby freeware mutation tool usually shows the mutation score [2] including line number of mutants that are not killed without further analysis and (d) complication in identifying equivalent mutants that act as false positives during mutation score estimation.

B. RQ1.1 – First Order Mutant (FOM) and Higher Order Mutant (HOM)

The progress of improving the test suites' quality had achieved a significant level through the use of mutation testing. There are two types of mutants that could be classified: (a) First Order Mutants (FOMs) and (b) Higher Order Mutants (HOMs). FOMs were generated by executing mutation operator [12] only once whereas HOMs were generated by executing mutation operator more than once. Elmahdi [11] in their study described FOM as "a single syntactic change to the source code" which is usually easily killed. On the other hand, HOM represents "a multiple syntactic change to the source code" which normally leads to complex faults.

C. RQ1.2 – Capabilities of FOM and HOM

Since HOMs are generated by combining different FOMs, a large number of HOMs will be produced of which they will usually be killed by any test suites that kill all the FOMs during program under test [13]. Jackson and Silvia [15] mentioned that HOMs could contribute to the determination of equivalent mutants and improve testing productivity seeing that HOMs can classify more realistic faults and are difficult to kill. However, this kind of test will cause more challenges and is more expensive due to the derivation of large space of mutants.

Elmahdi [11] presented the notion of Subtle HOMs that constitute FOMs interacting by distinguishing each other to produce new faulty behaviors which cannot be individually generated using all FOMs program under test. HOMs can be classified into two types – coupled and subsuming. Subsuming HOMs can be defined as "harder to kill than their constituent FOMs". For instance, if there are two different mutation operators executed to one mutant, subsuming HOMs will be the more difficult mutants to kill in comparison with the FOMs that compose them.

Yue and Mark [14] presented the concept of strongly subsuming HOM. According to them, a subsuming HOM is only killed by "a subset of the intersection of test cases" that kills each FOM from which it is generated. Figure 6 illustrates the classification of HOMs which represents the domains of all HOMs. The coloured areas illustrate the test sets with all test cases that kill HOMs. In addition, the advantages of HOMs have also been listed out respectively: (a) the increment of subtleness; (b) the less effort needed and (c) the decrease in the number of equivalent mutants.

D. RQ2 – Multi-objective Test Data Generation

In software testing, it was difficult to achieve a set of tests which is successfully able to detect bugs or defects in the respective program under test. As mentioned earlier, HOMs are more complex and difficult to be killed other than being able to generate other faults as well. Multi-objective optimization is one of the options to deal with such problems. Multi-objective optimization can be determined from the optimization problems that are caused by many factors [16], [17]. According to Jackson and Silvia [16], the optimization problems are usually caused by different objective functions and are related to various metrics which typically result in conflicts. Due to this, multi-objective aims to "find a set of solution representing a trade-off among the objectives". In order to overcome multi-objective problems, a number of Genetic Algorithms (GAs) was recognized to be applied. Rui and Silvia [17] presented three most representative GAs to deal with multi-objective optimization which include:

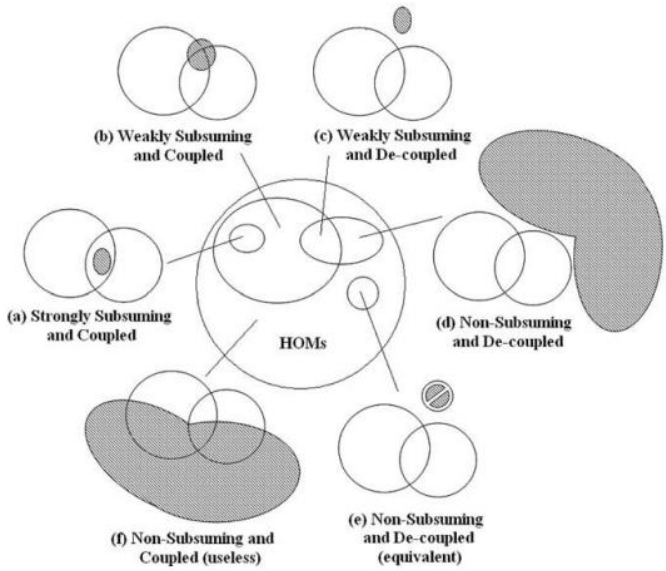

Fig. 6. Classification of HOM, Yue and Mark [14]

(a) Non-dominated sorting genetic algorithm (NSGA-II)
NSGA-II is a Multi-objective Evolutionary Algorithm (MOEA) which is based on GA that sorts the individuals from parent and offspring populations, considering the non-dominance and creating several fronts for each generation.

(b) Strength Pareto evolutionary algorithm (SPEA2)
For each generation, SPEA2 uses an external achieve that stores non-dominated solutions. Selection operators are used and a strength value is calculated for each solution in each generation.

(c) Indicator-based evolutionary algorithm (IBEA)
IBEA is a multi-objective algorithm which is based on indicators. For each solution, a weight is assigned accordingly to the quality indicator to favor the user optimization objectives.

IV. CONCLUSION

This paper has presented the results from the SLR on SBMT. Based on the literature search, there were 24 papers on SBMT, with 14 papers altogether related to RQ1 and eight papers used to detail out RQ2. This study also highlights several multi-objective algorithms to overcome multi-objective optimization problems.

All in all, there are still a lot of researches that can be carried out in the area of SBMT. Such include improving approaches for testing data generation, complexity analysis and decreasing cost in SBMT.

ACKNOWLEDGMENT

This work was supported/funded by Universiti Teknologi Malaysia under UTM Fundamental Research Grant (UTMFR): Q.J130000.3851.21H94.

REFERENCES

[1] Nishtha, J., Bharti, S., and Shweta, R., "Systematic Literature Review on Search Based Mutation Testing", eInformatica Software Engineering Journal, Vol. 11, No.1, 2017, pp. 59-76.

[2] Bharti, K., and Susheela, H., "A Survey and Analysis of Mutation Testing", International Journal of Engineering Trends and Technology, Vol 21, No. 10, 2015, pp. 179-182.

[3] Naveen, G., Sanjay, S., and Surender, J., "Challenges and Techniques for Testing of Big Data", Procedia Computer Science, Vol 85, 2016, pp. 940-948.

[4] Nan, L., Anthony, E., Yun, G., and Jeff, O., "A Scalable Big Data Test Framework", in 8th International Conference on Software Testing, Verification and Validation (ICST). IEEE, 2015.

[5] Qianqian, Z., Annibale, P., and Andy, Z., "A Systematic Literature Review of How Mutation Testing Supports Test Activities", Journal of Software Testing, Verification and Reliability, pp.

[6] Rodolfo, A., S., Simone, R.S., and Paula, S.L.S, "A Systematic Review on Search Based Mutation Testing", Information and Software Technology, No. 81, 2017, pp. 19-35.

[7] Barbara, K., "Procedures for Performing Systematic Reviews", Keele University, Keele,University, Keele, Staffs, UK, Joint Technical Report TR/SE-0401, 2004. http://csnotes.upm.edu.my/kelasmaya/pgkm20910.nsf/0/715071a8011d4c2f482577a700386d3a

[8] Nishatha, J., Shweta, R., and Bharti, S., "State of Art in the Field of Search-Based Mutation Testing", in 4th International Conference on Reliability, Infocom Technologies and Optimization (ICRITO), IEEE, 2015, pp. 1-6.

[9] Annibale, P., Fitsum, M.K., and Poalo, T., "A large scale empirical comparison of state-of-the-art search-based test case generators", Information and Software Technology, Vol. 104, 2018, pp. 236 – 256.

[10] Pedro, D.P., and Inmaculada, M.B., "Search-based mutant selection for efficient test suite improvement: Evaluation and results", Information and Software Technology, Vol. 104, 2018, pp. 130-143.

[11] Elmahdi, O., Sudipto, G., and Darrell, W., "Subtle higher order mutants", Information and Software Technology, Vol. 83, 2017, pp. 3-18.

[12] Vinita, C., Vineet, C., Hema, K., and Vikas, S., "Comprehensive set of Mutation Operators for the determination of adequacy of test set", International Conference on Advances in Computer Engineering and Applications (ICACEA), IEEE, 2015, pp. 1019-1023.

[13] Elmahdi, O., Sudipto, G., and Darrell, W., "HOMAJ: A tool for higher Order mutation testing in AspectJ and Java", 7th International Conference on Software Testing, Verification and Validation Workshops, IEEE, 2014, pp. 165-170.

[14] Yue, J., and Mark, H., "Higher order mutation testing", Information and Software Technology, Vol. 51, 2009, pp. 1369-1393.

[15] Jackson, A.P.L, and Silva, R.V., "Search-based higher order mutation testing: A mapping study", 18th Proceedings of the III Brazilian Symposium on Systematic and Automated Software Testing, ACM, 2018, pp. 87-96.

[16] Jackson, A.P.L., and Silvia, R.V., "A multi-objective optimization approach for selection of second oder mutant generation strategies", in Proceedings of the 2nd Brazilian Symposium on Systematic and Automated Software Testing, ACM, 2017, pp.

[17] Rui, A.M.F, and Silvia, R.V., "A multi-objective test data generation approach for mutation testing of feature models", Journal of Software Engineering Research and Development, 2016, Vol. 4, pp. 1-29.

[18] Mark, H., Yue, J., William, B.L. (2010). A Manifesto for Higher Order Mutation Testing. 3rd International Conference on Software Testing, Verification and Validation Workshops, pp. 80-89.

[19] Larry, J.M. (1990). A Theory of Fault-Based Testing, IEEE Transactions on Software Engineering, Vol. 16, No. 8, pp. 844-857.

[20] Phil, M.M. (2004). Search-based Software Test Data Generation: A Survey, Journal of Software Testing, Verification and Reliability, ACM, Vol. 14, No. 2, pp. 105-156.