A Test Case Selection Framework and Technique: Weighted Average Scoring Method

Rafaqat Kazmi[1], Dayang N. A. Jawawi[1], Radziah Mohamad[1], Imran Ghani[2] and Muhammad Younas[1]
[1]*Faculty of Computing, Universiti Teknologi Malaysia, 81310 Johor Bahru, Johor, Malaysia.*
[2]*School of Information Technology, Monash University Malaysia.*
rafaqutkazmi@gmail.com

Abstract—Test case selection techniques identify and eliminate the modification revealing test cases and try to reduce the test suite size for optimization of regression testing. The objective of this experiment is to assess the effectiveness of weighted average scoring method of test case selection against single objective test case selection techniques. The multi-objective test case selection with the weighted average scoring framework and technique are proposed in this study to select the test cases. This method is trying to solve conflicting test case selection objectives with six selection scenarios. The method used test data of cost, coverage, fault detection ability and code change information, convert them into the weighted average score as scalar function and presented this score to 100-index slabs related to low to high scores, then select the test cases. The results for these selection scenarios are computed and evaluated using size reduction of the test suite, inclusiveness, and precision. The results showed that all scenarios performed acceptable level within conditions applied from 17% minimum to 86% maximum in size reduction metrics. The inclusiveness showed 17% to 88% and 33% to 85% for precision metric.

Index Terms—**Regression Testing; Test Case Selection; Test Suite Effectiveness.**

I. INTRODUCTION

Regression testing is very expensive and repetitive activity and utilized whenever a program is changed, modified or updated. The regression testing is to build the confidence that changes do not harm the program. The complete testing of a software is not possible especially in the case of regression testing, the only possible way out is an adequate testing with certain objectives to fulfils. The JB Goodenough [1], put the question, What is the criterion for adequate software testing? The test case selection methods are purposed in 1977 [2] for maintenance of modified software. Test case selection is used to select a subset of test cases already available for previously executed test cases [3]. Test cases are input to the testing process and act as execution conditions with expected outputs [4]. The set of more than one test case (test suite) for testing software under test (SUT) must grow with the evolution of software. To re-execute, all these test cases are not wise and consume considerable costs [5]. In order to execute the most related tests regarding modification made into code, test case selection is carried out.[6].

The prioritization methods of regression testing try to order the test suites in such a way that they should expose the faults as early as possible. The main difference between selection methods with prioritization, the first one primary focus on code changes and modifications while the later primary focus on fault detection as early as possible. Furthermore, prioritization methods did not eliminate the test cases from original test suites, only change their order of execution, while selection methods remove redundant test cases from the test suites. The reduction methods for regression testing primary focus on minimization of the size of the test suite with the intent to reduce the cost of regression testing. The objectives of reduction and selection are same except selection methods also care for modifications in the code. There are many different objectives and possible direct or indirect benefits with regression test case selection. The goals observed in literature for regression test case selection are continuous integration, N release development, continuous development and continuous quality enhancements [7, 8]. Regression Test Selection (RTS) techniques ultimately contributes directly and in some cases indirectly to overall quality of software product, maintenance activities [9], reliability of software product [10], transition of software system from old systems [11], deployment activities of product, software upgrades [12], training of applications and staff and version control systems. But RTS techniques work with code/requirement information, risk management, software production and metrics to evaluate the results of testing process

This study is extension of the previous research [13] provided the justification for test case selection parameters cost, coverage, fault detection ability and code change information as single objective for RTS The second study which is part of the same research provides the justification for cost trade-offs [14], discuss the different cost measures and their trade-off with respect to each other.The main objective of this controlled experiment is to select the relevant subset of test cases from original test suites and minimize the size of test suite keeping cost, coverage, fault detection ability and code change based effectiveness in control. The second objective is to establish a continuous test case selection process which includes cost, coverage, fault detection ability and change information as the separate measure as well as accumulative measure for test case selection measure. The third objective is to embed the tester experience as part of test case selection process which provides the flexibility of choice in selection parameters and may increase the the effectiveness.

The rest of the study is organized as the Section 2 contains the Related Work, Section 3 contains the Proposed Framework, Section 4 contains the Test Case Selection Technique based on the proposed framework, Section 5 contains experimental setup, Section 6 contain the results, Section 7 contains discussion and Section 8 contains conclusion and future work.

II. Related Work

The experimental studies which investigate regression test case selection techniques use single objective (cost, coverage, fault detection ability, code change information) or multiple adequacy measures like any combinations of cost, coverage, fault detection ability and code change information. The Leung and White proposed a cost model which includes executional, validation and analysis costs model. This model provides a relative effectiveness measure for regression testing [15]. A code analysis based RTS [16] proposed for software written in java and.NET environments, applied on intermediate code, try to reduce the executional cost but ignoring the cost to maintain the test suite, analysis cost and maintenance of code repository. A control flow graph based RTS [17] technique is designed to reduce the cost of regression testing. The focus is on changes appeared on the edges of the graph that were used to select the test cases for the current version of the software. This method is based on code instrumentations which limits its effectiveness. Furthermore, the study did not consider the graph size and algorithm runtime for test selection in executional cost.

The code coverage based RTS technique [18] compares four prioritization techniques, one test case selection, one reduction and one hybrid method. These all techniques are using five coverage types. The study reports that updated coverage information was more effective as compared to simple coverage measures. A graph based RTS technique [19] used coverage information to reduce the overall cost of test case selection. The study provides a system ReTest to perform regression test case selection, by identifying finer granularity levels of coverage to select the dangerous edges in control flow graph and then based on these edges, selection of test cases is accomplished.

A code change based RTS [20] try to compute updated coverage without re-executing the tests on SUT. The techniques try to reduce the executional cost by using selective code instrumentation to assess the code changes into the SUT. The selection of tests is based on the comparison of out-dated coverage with the re-computed coverage information but its accuracy is based on change types and location inside the code of SUT.

The multi-objective test case selection technique [21] investigates branch coverage with an executional cost of the test suites for SUT. The objectives are carried out by applying Particle Swarm Optimization(PSO). A competitive analysis [22] using mutation scores to identify the bugs from SUT and also to select the test case from existing test suites. The study compared the results of five benchmarks based on mutation score. This study applied Genetic Classification(GC) to measure genetic effectiveness based on genetic operators with mutation operators. All these measures are incorporated into the Integrated Coevolutionary Genetic algorithm. The comparison was based on three scenarios, test case selection, mutation score and reduction of the execution cost. The study ignores the complexity and the size of the test suites and the size of SUT. The genetic effectiveness with mutation score is used to measure the effectiveness of the selection technique, but there was no distinction between equivalent and non-equivalent mutants. Similarly, there was no discussion on mutation operators used for analysis, because there are so many mutation operators in use with different performance measures.

The change identification RTS [23] using difference engine to assess and analyse the results from the old and current version of the SUT. The difference engine identifies the correlations between code and test cases and recommends test cases which show strong relationship between code entities and test suites based on modifications made to the SUT. The evaluation metrics like precision, recall, and efficiency is used to justify the results.

The understanding of these selection methods is narrow. The main reasons for this were these studies normally established a base or original version and then execute the same method on modified version. Then after executing selection technique, they simply compare the results of base application with modified version of SUT. The two-major limitation with these selection techniques are, these studies model RTS as the one-time process instead of continuous activity. These studies ignored resource constraints like time and cost of the regression testing. The historical test data is also ignored in these classic selection and prioritization techniques. The historical data is important because of the repetitive nature of the problem. Due to ignoring historical data, these techniques are memoryless. These techniques solely based on the current information of test cases and sacrificing the benefits of test history.

In this controlled experiment, we try to embed history in terms of tester experience and test case executional data from previous runs. The proposed framework and test case selection technique also embed aggregated impact of effectiveness measures by weighted average score which also makes this technique flexible and more useful with local as well as generalized testing requirements.

III. Proposed Framework for Test Case Selection Based on Weighted Average Score

In this section, we propose a framework for regression test case selection based on weighted average sum [24] of measurable aspects of test case selection. These aspects are code coverage, the cost of testing, fault detection ability of the test suites and code change information. These measures are primary contributors of the effectiveness of test case selection process. The abstract view of proposed framework is shown in Figure 1. The framework consists of four layers contributing equally to fulfills the purpose of test case selection. This process starts with the first execution of test suites on subject programs and collect data for code coverage, fault detection ability and execution costs of the test cases. The code change information is collected from the second version of the SUT, for which this selection process is carried out with respect to previous version. The second input to this process is the selection criterion, based on testers experience and requirement for tests for next iteration of regression testing.

These criterions for current settings are cost-based test case selection, coverage-based test case selection, fault detection ability-based test case selection, code change based test case selection, balanced scoring test case selection (uses all four parameters with equal weight) and customized (varying weighting factors for four parameters) scoring test case selection. The third step is to measure the weighted factors for each test case, executed on previous version and criterion inputted by the tester for next version. The detailed systematic process is elaborated in algorithm presented in next section After computing these weighted values for each test case, the weighted average sum for each test case is computed based on the selection criterion.

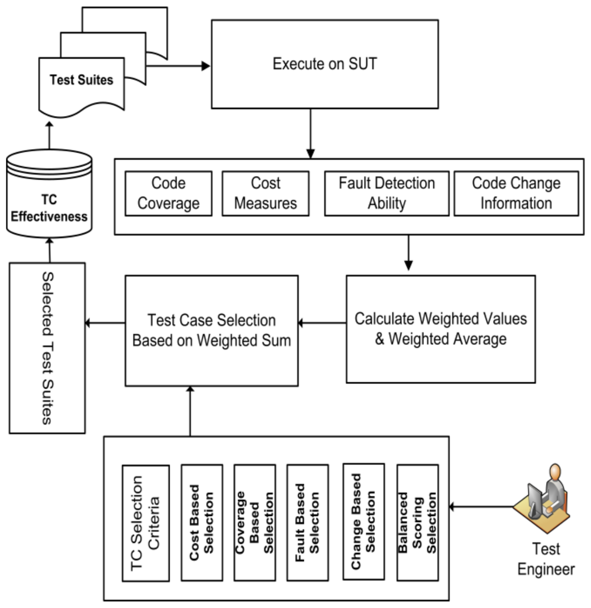


Figure 1: Proposed framework for regression test case selection

Then based on this weighted average sum, a selection index is defined to select or skip the test cases for next version. Here we use 100-index scale to present the strength or weakness of the score. This indexing mechanism is adapted from a feature analysis method [25], which use the index of 0 to 1 to show features score for selection, but here we use the 100-index to make it more meaningful for test case selection. After selection of these test cases, the data is stored for next analysis and selection for the next iteration and so on.

IV. Test Case Selection on Weighted Average Score(TCSWAS) Technique

In this section, we elaborate the proposed TCSWAS technique to select the test case from previously executed test cases. The proposed method aims to: a) Decrease the overall cost of the test suite execution, b) decrease the size of test suite size from previous test suite, c) Include the weighted average score (cost, coverage, fault detection, code changes) for test case selection, d) Improve the overall effectiveness of test suites. The TCSWAS process is based on the framework proposed in Figure 1. The test case selection is carried out in two steps. First, the values for each effectiveness measure (cost, coverage, fault detection, code changes) is computed and then these computations are converted into their weighted average by using metrics weighted average sum [26]. In second step, these measures are presented on 100-index scale, to compare them on an equal level for better understanding. The weighting average score and presentation scale are shown in the Figure 2. The scaling of heterogeneous data also used in regression testing to compare and present data [27] to find relationships between cost, coverage and effectiveness of the test suites.

The multi-objective optimization problems are converted into single objective by using scalar function [24]. Different weights are used with different testing requirements. An experiment was conducted to select features from multiple features with the different weights, reported that optimized results are found with 0.4 on the scale of of 1, it is converted to 25 on the scale of 100 for weighted average scoring method [25] as shown in Figure 2. There are total six different selection criterions used to select the test cases. These criterions are 1) cost based selection, 2) coverage based selection, 3) fault based selection, 4) change based selection, 5) balanced scoring selection and 6) customized scoring selection. The complete scoring scheme is shown in the Figure 2. The intent is to use the weighted average for each effectiveness measure and include the impact of each measure to a final score. Then this accumulative score is presented on the scale of 100 for the test case selection. The 100-index has five slabs. The range from 0 to 20 presented as very low, 21 to 39 as low, 40 to 59 as the medium, 60 to 74 as high and 75 to 100 considered as the very high score. The similar scheme is used to find the relationship between these effectiveness measures (cost, coverage, fault detection ability) in study [27]. An experiment is conducted to assess the co-relationship between effectiveness measures which rate this relationships from 0 to 100 points slabs using kendall Tau [28]. Here in this experiment, select the test cases with index greater than 39, considered medium to very high index test cases.

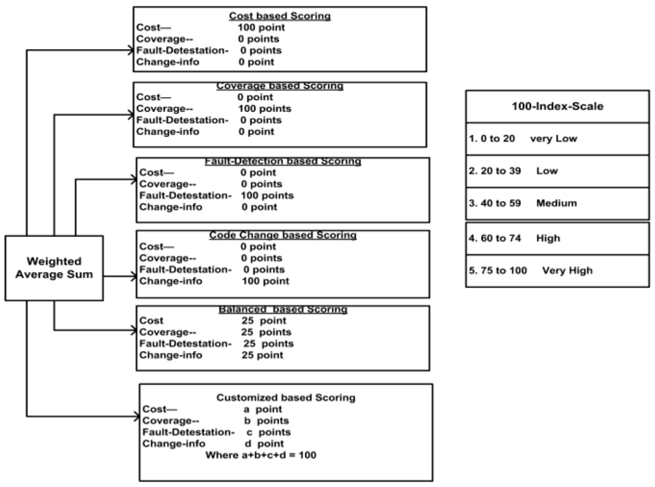


Figure 2: The weighted scoring scheme and indexing slabs

V. Experimental Setup

Three datasets are selected to apply the proposed test case selection technique based on the test case selection framework. These three datasets are listed in Table 1 with their respective characteristics. Two of them Triangles [29] and TreeDataStructure [29] are academic datasets with test suites written in java. The third dataset dataset JodaTime [30], an open source library to replace java date and time library.

Table 1
The Characteristics of System Under Test

Number	Data Set	Version	LOC	Test Case
1	JodaTime	2	280464	279
2	TreeDataStructure	2	2200	22
3	Triangles	2	116	12

The Figure 3, provides the experimental process for regression test case selection with weighted average sum and 100-index scale. This experimental process based on the framework proposed in section II and consists of five layers with a dedicated role in the selection process. In the first layer in the experimental process, the subject program is prepared with test suites, the environment for program execution is

Eclipse [31] and Junit framework is selected for test suite creation and execution. The second layer is tool support, for collecting relevant information for proposed technique .For current environment, the size of system under testing(SUT) and the coverage information(statement coverage) are measured by EclEmma [32],test suite size and test results are collected through Junit, the faulty versions of SUT are produced and analysed by PIT [33] and the code changes between different versions of the same software are measured by JDiff [34].In the third layer, relevant data for coverage, size of test suite, mutation score and code change information is received from second layer and prepared the data in 100-index format for further processing. The fourth layer takes the data for each measure from third layer and assign the average sum. The fifth layer of this process gets the user requirement about test case selection scenario and compute the weighting factor for each measurement collected from the previous layer. Then based on test case selection criteria and computed the weighted score of test cases, the test cases score is converted again into the 100-index scale and then test cases are selected from the original test suites.

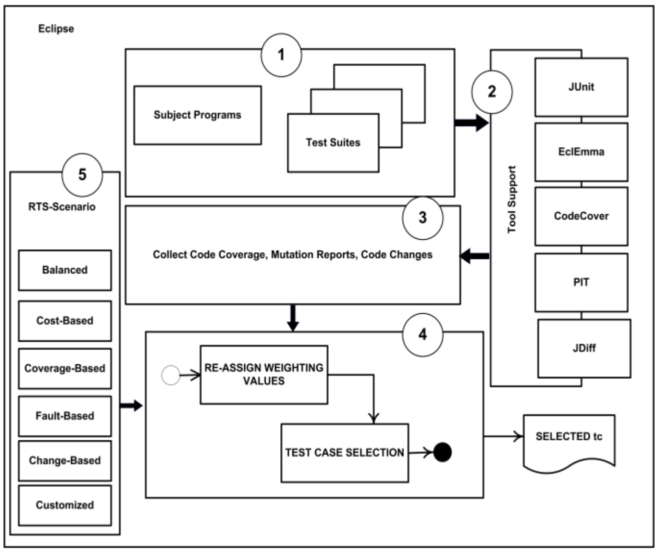


Figure 3: The Procedure for Experimental Setup

The algorithm for test case selection using TCSRAW technique is presented in Figure 4. The algorithm takes three inputs a program P, a modified program P′ and a test suite T. The algorithm also takes test engineer choice to choose the selection scenario and return a test suite T′.

The metrics used for data collection, analysis and presentation of results are listed in the Table 2. The coverage is measured in terms of statement coverage for a single test case. The cost is taken as the execution cost of each test case, divided by total execution cost of the test suite, and then multiplied by hundred. Fault detection ability is measured in terms of mutation score. The code changes, we mean here the number of statements modified, deleted, or added in a unit under test. The change ratio is calculated by the number of statements changed, divided by lines of code of unit under test and then multiplied by hundred. The precision and inclusiveness is calculated to assess the test case selection technique, how much modification revealing test cases are selected is called inclusiveness and how much non-modification test cases are not included is called precision.

ALGORITHM: Selecting test cases

```
1.  TCSRAW(P, P′,T)
2.  T = {t1, t2, t3,…..Tn}
3.  Cov = Collect statemnt coverage of T on P for each t ε T.
4.  Cost = Collect executional time of each t εT on P.
5.  FaultDetection = Collect MutationScore for each t ε T on P.
6.  ChangeInfo = Compare P , P′, collect statement change info.
7.  Convert each Cov, Cost, FaultDetection, ChangeInfo on 100-
    index.
8.  SelectionCriterion = Cost OR Cov. OR FaultDetection OR
    ChangeInfo OR BalancedScoring OR CustomScoring
9.  Assign weights such that
    9.1  w1 + w2 + w3 + w4 = 100
    9.2  IF SelectionCriterion = Cost
         9.2.1 SelectionIndex = (100 * Cost + 0 * Cov, + 0 *
         FaultDetection + 0 * ChangInfo)
    9.3  IF SelectionCriterion = Cov
         9.3.1 SelectionIndex = (0 * Cost + 100 * Cov, + 0 *
         FaultDetection + 0 * ChangInfo)
    9.4  IF SelectionCriterion = FaultDetection
         9.4.1 SelectionIndex = (0 * Cost + 0 * Cov, + 100 *
         FaultDetection + 0 * ChangInfo)
    9.5  IF SelectionCriterion = ChangeInfo
         9.5.1  SelectionIndex = (0 * Cost + 0 * Cov, + 0 *
                FaultDetection + 100 * ChangInfo)
    9.6  IF SelectionCriterion = BalancedScore
         9.6.1 (SelectionIndex = 25 * Cost + 25 * Cov, + 25 *
         FaultDetection + 25 * ChangInfo)
    9.7  IF SelectionCriterion = CustomScoring
         9.7.1(SelectionIndex =w1 * Cost + w2* Cov, + w3*
         FaultDetection + w4 * ChangInfo)
10. Move test Ti from T to T′ for all those tests
      SelectionIndex > 39
11. Return T′
```

Figure 4: Algorithm for test case selection

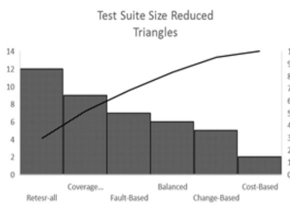
Table 2
Metrics used for Analysis and Evaluation of TCSWAS Technique

Measure	Description	Formula
Coverage	The coverage is the number of lines of code executed by a test suite/test case divided by total testable number of lines of unit under test.	Coverage =(LOC covered by TC)/(Total LOC of unit under test)*100
Cost	The cost is measured as time taken by a test case divided by total time take by the test suite, multiplied by hundred.	Cost=(Execution time taken by TC)/(Total Execution time of test suite)*100
Fault Detection	The mutants killed by a test case/test suite divided by a total number of mutants generated, multiplied by hundred.	Mutation Score=(Killed mutants)/(Total Mutants)*100
Change Ratio	The total number of lines changed, divided by a total number of lines covered by test case/test suite, multiplied by hundred.	Change Ratio Score=(LOC Changed)/(Total LOC for unit under test)*100
Inclusiveness	Let a test suite T contains n tests which are modification revealing for a program P and M selects "m" modification revealing for P′.	Inclusiveness = 100(m/n) iff n≠ 0
Precision	Let a test suite T contains n tests which are non-modification revealing for a program P and M rejects m non-modification revealing for P′.	Precision = 100(m/n) iff n≠ 0

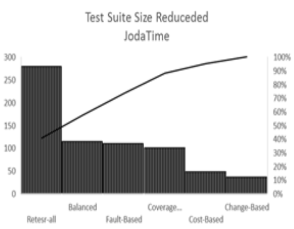
VI. Results

The objective of all test case selection techniques is to select a subset of test cases from the existing test suite and the subset is as minimum as possible without compromising the overall effectiveness of the testing process. The first objective of TCSWAS is also to select a sub set of test cases from existing test suites. The study was executed on three datasets and the results for test suites selection with respect to *retest-all* are shown in Figure 5. There were total five scenarios created based on tester requirements and then compiled and computed the weighted average score and 100-index scale.

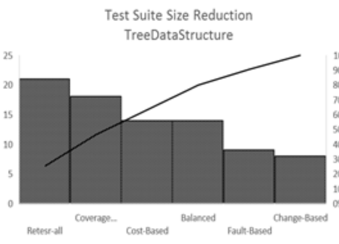
The customized scenario is same as balanced scoring scenario, the results for customized scenario are not included to keep the study precise. The 100-indexing is used to present the data of all effectiveness measures on a similar scale and easy to understand and help in selection process as well as makes simple assessments in verification and validation process.



(a)



(b)



(c)

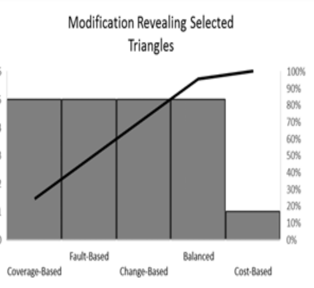
Figure 5: The Reduction in Test Suite Size Graph

The graph in Figure 5, representing the reduction in the test suite size in terms of number of the test case with respect to *retest-all* and the reduction in test suite in terms of percentage to total test suite size. The Y-axis showing the total reduction in test suite size in number on left and test suite reduction on right side. Similarly, each scenario, cost-based selection, coverage based selection, fault detection based selection, balanced selection is represented on X-axis, in order to compare them with *retest-all*, as well as with each other. The line intersecting the graph is frequency distribution curve which indicating the normality and skewness of data collected. For current experiment, the details of data normality and skewness are not discussed in detail to keep the analysis simple.

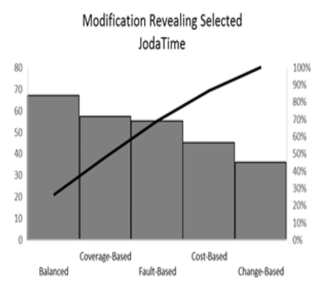
The results of dataset Triangles in Figure 5-a show that cost based selection criterion selects 17% of the test cases, coverage based selection criterion selects 75% of test cases, fault detection ability based criterion selects 58% test cases, the balanced weighted score selection criterion selects 50% test cases and change information based selection criterion selects 42% test cases as compared to original test suite. This is observed that in all scenarios, the overall test suite reduction is good without compromising the overall effectiveness of the testing process within the conditions of current environment.

In the second dataset under study, Figure 5-b shows that the cost based selection scenario selects 17%, coverage based selection scenario selects 36%, fault detection ability based scenario selects 39%, change based selection criterion selects 13% and balanced scenario which gives all measures equal weight selects 41% test cases as compared to original test suite. The third dataset TreeDataStructure in Figure 5-c shows that for cost-based selection, it selects 67% test cases, coverage based scenario selects 86% test cases, fault detection ability based scenario selects 43% test cases, change information based scenario selects 38% test cases and balanced scoring scenario selects 67% of the test cases as compared to original test suite. The results show that for all scenario, the test suite size reduction is good except for coverage criterion, which selects more than 70% test cases for two datasets. It is also observed that balanced scoring criteria perform optimum as compared to other scenarios under analysis, selects 41% to 67% of the test cases. The choice of the selection criterion depends on the testing requirement provided by the test engineer of the SUT.

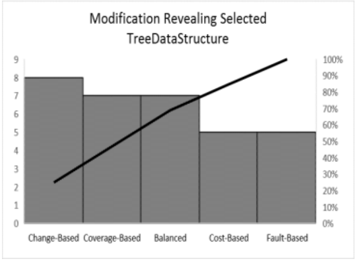
In order to assess and analyse the regression test case selection techniques, inclusiveness and precision are measured. The inclusiveness is the measure to assess that how much modification revealing test cases are selected by the selection technique. The comparison of TCSWAS for inclusiveness metric for the data sets under the study of all test case selection criterions are shown in the Figure 6. The X-axis in the graph below is showing the test case selection criterion which are cost based selection, coverage based selection, fault detection ability based selection, change information based selection criterion and balanced score based selection criterion. The graph bars representing the modification revealing test cases selected in comparison with a total number of modification revealing test cases on the left side of Y-axis and in percentage to right side of Y-axis.



(a)



(b)

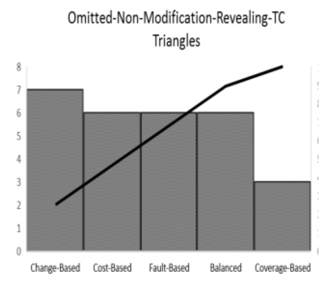


(c)

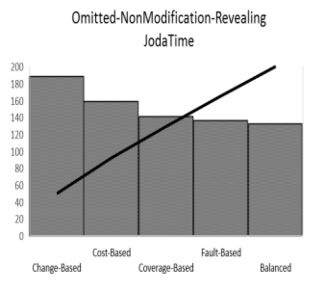
Figure 6: The inclusiveness evaluation graph for TCSWAS technique

In the Figure 6-a, showing the inclusiveness measure for dataset Triangles. It shows that the inclusiveness for all scenarios is 83% except for cost-based test case selection criterion which is 17%. The Figure 6-b, the inclusiveness values for JodaTime are presented. The balanced approach includes 72% of modification revealing test cases, coverage based selection criterion includes 61% of modification revealing test cases, fault detection ability selection criterion selects 59% of the modification revealing test cases, cost-based selection criterion selects 48% of modification revealing test cases and change information based selection criterion selects 39% of modification revealing test cases. The minimum inclusiveness for TCSWAS is 39% and the maximum is 72% which is the acceptable value for a test case selection technique providing four different measures and six selection criterions.

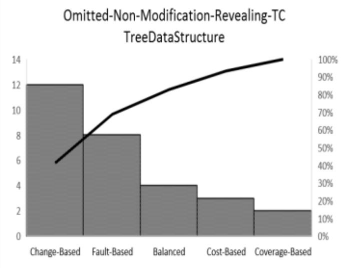
The second viewpoint to assess and analyse the test case selection technique is to assess, how many non-modifications revealing test cases are omitted by the test case selection technique, is called precision metric for regression test case selection techniques. The Figure 7 showing the results for precision measure for three datasets under study. The X-axis in the graph showing the test case selection criterions and Y-axis representing the total number of test case which is non-modification revealing on the left and percentage of non-modification test cases on the right.



(a)



(b)



(c)

Figure 7: The precision evaluation graph for TCSWAS technique

The Figure 6-a shows the precision values for dataset Triangles. The cost-based test case selection scenario, fault detection ability based scenario and balanced scoring scenario rejects the non-modification test cases during test case selection process which is 67%. The change information based selection criterion rejects 78% of non-modification test cases. The coverage based test case selection criterion rejects 33% of non-modification revealing test cases. In Figure 7-b, the change based test case selection criterion rejects the maximum number of non-modification test cases which is 97%, the coverage based selection criterion rejects 76%, fault detection ability selection scenario rejects 73%, balanced scoring selection criterion rejects 71% and cost based selection criterion rejects 35% of non-modification revealing test cases. The change based test case selection scenario performs better as compared to other scenarios, the reason is obvious that this scenario based solely on the change information detected during the testing process so its ability to include modification revealing test cases and rejection of non-modification based test cases is better as compared to other scenarios.

VII. Discussion

This controlled experimental study proposed a framework and test case selection technique on a weighted average score and 100-indexing method to select the test cases from already executed test suites. The main concern is to investigate the mutual impact of cost, coverage, fault detection ability and code change information on the selection criterions already available from previously executed test suite. The cost, coverage and fault detection has many types of dependencies and relationships on each other. The code changes are the primary concern of all regression test case selection criterion, in this study, authors try to combine all these effectiveness measures in a single test case selection criterions.

The third important parameter for this technique is to give the flexibility to the test engineers to select effectiveness measures by choosing the test case selection scenario from cost-based test selection, coverage based selection, fault detection ability based selection, code change based selection, balanced scoring selection criterion or customized test selection criterion.

The cost-based selection only based on cost measures of an individual test case and ignore other measures. The coverage based scenario only selects the test cases with coverage satisfied test cases. The fault detection ability chooses test cases with fault detection scores. The change information selection scenario only focuses on change information and selects the test cases related to code changes between current and previous versions of system under testing. The balanced scoring scenario gives equal weights to all measures which are cost, coverage, fault detection ability and code changes. The customized scenario gives the independence to test engineer to rate all these effectiveness measures between 0 to 100, but their rating sum must equal to 100. This mechanism provides a broad range of possibilities to include local testing requirements.

In this study, three datasets are used to evaluate the proposed framework and TCSWAS regression test case selection technique. The inclusiveness and precision metrics are used to assess the ability of selection of modification revealing and rejection of non-modification revealing test cases for the proposed technique. The evaluation results show that each scenario is performed at the acceptable level with different conditions. The cost-based selection criterions reduced the maximum possible number of test cases in cost based selection criterions with all datasets. But the one observation for small suite sizes was that cost values were so small and sometimes very near to zero. Therefore, cost alone is not a good test case selection criterions because the cost values are insignificant in some situations. Code coverage based selection criterion always returns more than 50% test cases selected. It seems good indicator in-order to test a maximum number of lines of code, but it also includes non-modification revealing test cases and high-cost values. The second interesting observation was that fault detection ability in terms of mutation analysis return similar trends and data with coverage based analysis but again costs are bit high. Therefore, the coverage alone not return effective test suites. The authors recommend that coverage measure used with code change based metrics for selection criterions of regression testing. The fault detection ability is used as performance measure so far in regression testing, but in this study authors try to use this measure as selection criterion as well. The results are satisfactory with fault detection ability. But one observation was that tools used to collect and analyse fault data are not well matured yet and they also put some extra cost and analysis overhead to the regression test technique.

The change information based selection is the primary concern of all selection criterion for all test case selection techniques. In this experiment change information improves the balanced scoring technique considerably. The balanced scoring criterion equally weights all the effectiveness measures and always return satisfactory results for selection. The inclusiveness and precision also show balanced scoring scenario performs reasonably acceptable and show the combined behaviour of cost, coverage, fault detection ability and code change information. The code change information is used as the proxy for a balanced scoring scenario, the possible reason was coverage and fault detection ability return similar data and cost behaves skewed in small size datasets, but code change information refined the results of previous measures and returns reasonable acceptable reduced test suites, which also fulfils the testing requirements.

VIII. Conclusion and Future Work

This study proposes a regression test case selection framework and a test case selection technique TCSWAS based on weighted average sum to fulfils the following objectives. Decrease the number of test case without compromising effectiveness with respect to an original test suite. Increase the overall effectiveness of RTS process in terms cost, coverage, fault detection ability and code changes. Establish a continuous selection process which uses previous test data and tester experience. There were three datasets to evaluate the framework and proposed technique. The results show that all scenarios used performs reasonably acceptable manner in terms of inclusiveness and precision measures with their environment and testing requirements. The cost behaves differently for different test suite sizes, while coverage and fault detection ability perform in similar fashion returning almost equally reduced test suites. The code change information behaves as the proxy to validate and refine the reduced test suite by cost, coverage and fault detection ability. Therefore, it is concluded that combined measures of cost, coverage, fault detection ability and change information are good predictors of effectiveness of reduced test suites by selection technique as compared to use them separately. The test engineers experience can also further improve the effectiveness, in this study, test case selection scenarios provide the flexibility to choose relevant measures to select the test cases but results conclude that balanced scoring produces more effective results for selection.

The future work for this experiment is to embed more change based granularity levels, right now to keep the process simple, only statement changes are used as change information. But method changes, class changes and change impact analysis also need to investigate in future. The code coverage may also need to add stronger coverage types like condition coverage and modified condition coverage. Similarly, mutation analysis was a good start but this study may also be evaluated by the real fault with industrial project.

Acknowledgement.

We are thankful to Ministry of Science, Technologi and Innovation (MOSTI) Malaysia for supporting this research under Vot: 4S113. We are also thankful to Universiti Teknologi Malaysia (UTM) for providing the research facilities.

References

[1] J. B. Goodenough and S. L. Gerhart, “Toward a theory of test data selection,” *IEEE Transactions on Software Engineering*, vol. SE-1, no. 2, pp. 156-173, 1975.

[2] K. F. Fischer, “A test case selection method for the validation of software maintenance modifications,” in *Proceedings of COMPSAC*, 1977, pp. 421-426.

[3] B. Beizer, *Black-Box Testing: Techniques for Functional Testing of Software and Systems*. John Wiley & Sons, Inc., 1995.

[4] M. Zavvar and F. Ramezani, “Measuring of software maintainability using adaptive fuzzy neural network,” *International Journal of Modern Education & Computer Science*, vol. 10, pp. 27-32, 2015.

[5] M. A. Askarunisa, M. L. Shanmugapriya, and D. N. Ramaraj, “Cost and coverage metrics for measuring the effectiveness of test case prioritization techniques,” *INFOCOMP Journal of Computer Science*, vol. 9, no. 1, pp. 43-52, 2010.

[6] S. Elbaum, P. Kallakuri, A. Malishevsky, G. Rothermel, and S. Kanduri, “Understanding the effects of changes on the cost-effectiveness of regression testing techniques,” *Software Testing, Verification and Reliability*, vol. 13, no. 2, pp. 65-83, 2003.

[7] A. Orso, N. Shi, and M. J. Harrold, “Scaling regression testing to large software systems,” in *ACM SIGSOFT Software Engineering Notes*, 2004, pp. 241-251.

[8] W. E. Lewis, *Software Testing and Continuous Quality Improvement*. CRC press, 2016.

[9] W. E. Wong, J. R. Horgan, S. London, and H. A. Bellcore, “A study of effective regression testing in practice,” in *ISSRE '97 Proceedings of the Eighth International Symposium on Software Reliability Engineering*, 1997, pp. 264-274.

[10] J. D. Musa, “Operational profiles in software-reliability engineering,” *IEEE Software*, vol. 10, no. 2, pp. 14-32, 1993.

[11] C. Larman and V. R. Basili, “Iterative and incremental developments. a brief history,” *Computer*, vol. 36, no. 6, pp. 47-56, 2003.

[12] A. Bertolino, “Software testing research: Achievements, challenges, dreams,” in *Future of Software Engineering, 2007 (FOSE '07)*, 2007, pp. 85-103.

[13] R. Kazmi, D. N. Jawawi, R. Mohamad, and I. Ghani, “Effective regression test case selection: a systematic literature review,” ACM Computing Surveys (CSUR), vol. 50, no.2, pp. 1-29, 2017.

[14] R. Kazmi, I. Ghani, R. Mohamad, M. Tariq, I. S. Bajwa, and S. R. Jeong, “Trade-off between automated and manual testing: a production possibility curve cost model,” *Int. J. Advance Soft Compu. Appl*, vol. 8, no.1, pp. 12-27, 2016.

[15] H. K. Leung and L. White, “A cost model to compare regression test strategies,” in *Proceedings Conference on Software Maintenance*, 1991, pp. 201-208.

[16] T. Koju, S. Takada, and N. Doi, “Regression test selection based on intermediate code for virtual machines,” in *Proceedings of International Conference on Software Maintenance (ICSM 2003)*, 2003, pp. 420-429.

[17] G. Rothermel, M. J. Harrold, and J. Dedhia, “Regression test selection for C++ software,” Software Testing Verification and Reliability, vol. 10, pp. 77-109, 2000.

[18] D. D. Nardo, N. Alshahwan, L. Briand, and Y. Labiche, “Coverage-based regression test case selection, minimization and prioritization: A case study on an industrial system,” *Software Testing, Verification and Reliability*, vol. 25, no. 4, pp. 371-396, 2015.

[19] M. J. Harrold, J. A. Jones, T. Li, D. Liang, A. Orso, M. Pennings, S. Sinha, S. A. Spoon, and A. Gujarathi, “Regression test selection for Java software,” in *OOPSLA '01 Proceedings of the 16th ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications* 2001, pp. 312-326.

[20] P. K. Chittimalli and M. J. Harrold, “Recomputing coverage information to assist regression testing,” *IEEE Transactions on Software Engineering* , vol. 35, no.4, pp. 452-469, 2009.

[21] L. S. de Souza, R. B. Prudêncio, and F. d. A. Barros, “A Hybrid binary multi-objective particle swarm optimization with local search for test case selection,” in *Intelligent Systems (BRACIS), 2014 Brazilian Conference*, 2014, pp. 414-419.

[22] A. A. L. de Oliveira, C. G. Camilo-Junior, and A. M. Vincenzi, "A coevolutionary algorithm to automatic test case selection and mutant in mutation testing," in *2013 IEEE Congress on Evolutionary Computation (CEC)*, 2013, pp. 829-836.

[23] E. D. Ekelund and E. Engström, "Efficient regression testing based on test history: An industrial evaluation," in *2015 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, 2015, pp. 449-457.

[24] A. Konak, D. W. Coit, and A. E. Smith, "Multi-objective optimization using genetic algorithms: A tutorial," *Reliability Engineering & System Safety*, vol. 91, no. 9, pp. 992-1007, 2006.

[25] S. Wang, S. Ali, and A. Gotlieb, "Minimizing test suites in software product lines using weight-based genetic algorithms," in *Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation*, 2013, pp. 1493-1500.

[26] T. Hesterberg, "Weighted average importance sampling and defensive mixture distributions," *Technometrics*, vol. 37, no. 2, pp. 185-194, 1995.

[27] L. Inozemtseva and R. Holmes, "Coverage is not strongly correlated with test suite effectiveness," in *Proceedings of the 36th International Conference on Software Engineering*, 2014, pp. 435-445.

[28] P. K. Sen, "Estimates of the regression coefficient based on Kendall's Tau," *Journal of the American Statistical Association*, vol. 63, no. 324, pp. 1379-1389, 1968.

[29] Nayuki, *Project Nayuki*. University of Toronto, 2017. Available at https://www.nayuki.io/

[30] *JodaTime*. Joda.org., 2017. Available at http://www.joda.org/joda-time/

[31] *Java Editor*. The Eclipse Foundation. Available at https://eclipse.org/

[32] M. R. Hoffmann, "Java Code Coverage," 2017.

[33] PIT. (2016). Available at http://pitest.org/

[34] M. B. Doar, *JDiff-An HTML Report of API Differences*. LGPL. Available at http://javadiff.sourceforge.net/