IJCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper Vol.-6, Issue-10 Oct 2018 E-ISSN: 2347-2693

Regression Test Suite Management using Data Clustering Technique

Fayaz Ahmad Khan

Dept. of Computer Science and Applications, Barkatullah University Bhopal (M.P)

*Corresponding Author: kfayaz1012@gmail.com

Available online at: www.ijcseonline.org

Accepted: 25/Oct/2018, Published: 31/Oct/2018

Abstract- To test the modified code, we employ regression testing procedures with an aim to provide assurance that modified code behaves correctly and those modifications have not adversely affected the existing behavior or functionality of the code. Retest-all regression testing is the basic approach in which all the test cases in the initial test suite are re-executed to validate the changes. But re-running all the test cases from an existing test suite in order to test the code that is undergone minor change may be expensive as it requires an unacceptable amount of time and resources to perform it. An important problem found during regression testing is how to select a subset of test cases from an existing test suite in order to retest the modified code. Therefore, in this study we propose an efficient test suite management technique that utilizes data clustering approach for regression testing in order to effectively partition an initially random and large test suite to re-test the modified section of the code that has been modified within resource and time constraints.

Keywords- Software testing, Regression testing, Test Case Selection, Data Clustering, K-Means.

I. INTRODUCTION

Regression testing is an important activity performed on modified code to provide assurance that the modified code works correctly and that modifications have not affected the previous functionality of the code. Regression testing is expensive software maintenance activity [1]: as it requires to re-run all the test cases in a existing test suite in order to re-test the code that is modified to provide confidence that modified code behaves correctly. For software maintenance, developers usually create an initial test suite and re-use it for regression testing. The traditional technique for conducting regression testing is *retest-all* in which every test case in the test suite is re-run to re-test the modified code. This approach is expensive as it requires unacceptable amount of time and resources to re-run all the existing test cases in the initial test suite. An alternative strategy would be to select and re-run only a subset of the test cases from the initial test suite to test the modified code. So most of the research on regression testing [2] focuses on (1) how to select a subset of test cases from an existing test suite for regression testing (the regression testing selection problem), and (2) how to identify the segment of the code that undergoes modification and should be re-tested for quality and functionality assurance. In [2], the regression testing is defined as follows: "Let P be an original version of a program and $P^{/}$ be its modified version. Also, let T be the initial test suite developed for testing P. The role of regression testing selection technique is finding a subset of test cases $T^{/}$ of T in order to execute $P^{/}$".

The use of regression testing selection technique according in [1] reduces the cost of software testing compared to retest-all approach by running only a subset of the test cases from the initial test suite. The other useful technique is random selection approach which randomly selects a subset from the initial test suite and uses the subset for carrying out regression testing. But the randomly selected sample or subset may or may not be effective in terms of the modified code. The regression testing selection approaches try to overcome the draw-backs present in the existing techniques (like retest-all and random) and make regression testing feasible and economical by running only a subset of few effective test cases to retest the modified code. A substantial amount of research results are reported on regression test selection and minimization techniques in literature, but according to studies [3] very less number of software industries have actually deployed the support for automation during regression testing. The most commonly used approaches for the selection of regression test cases are either based on manual code analysis or based on expert judgment, but both results in unnecessarily high regression testing cost.

A number of regression test selection techniques have been proposed to target different programming paradigms such as procedural programming [4,5,6], object-oriented programming [7,8,9,10], component-based programming [11, 12, 13] and web applications [14, 15, 16]. A great number of approaches have also been proposed using different techniques including data flow analysis based [17, 18], slicing based [19, 20], firewall based [21, 22], control flow based [23, 24] and differencing based [25, 26]. There are also certain studies on regression test selection techniques which have been reviewed by the authors in [27, 28, 29, 30]. In [29], a set of metrics are proposed for the

evaluation of regression test selection techniques. In [27, 28], an experimental investigations are performed on the effectiveness of some regression test selection techniques targeting procedural programming. Based on the empirical results, it was found difficult to select any technique as a solution because the studies were performed on different kind of programs and also in different environment conditions. Also, the authors in [30] observe that it is very difficult to design a new generic and superior regression test suite management technique that will be applicable to a wide range of application programs. Therefore regression test suite management is still an open problem where further work could be done in order to improve the existing techniques or propose new techniques that will reduce the effort and cost of the software maintenance activity either by reducing the number of test cases or by selecting a subset of test cases efficiently from an initial test suite. This study is an effort in this direction as all the existing studies [5, 9, 11, 13, 15, 25, 29] try to select a subset of test cases for regression after software undergoes change, but with the proposed study, it is possible to select a subset before software undergoes change and as well as after the change when the software is regression tested. Thus, with the proposed approach, we can run the entire test suite or the selected test cases from each portioned segment or only a single partition or cluster for carrying-out testing activity. The flow chart of the proposed approach is shown in figure. 1.

The entire paper is organized as follows, Section I highlights the introduction and related work done. In Section II, the discussion on different types of regression testing types are provided. In Section III a brief overview about data clustering is given and in Section IV, the steps of K-Means Algorithm are mentioned. In Section V describes the implementation of the proposed approach on a sample program and results of the study. And finally, Section VI describes the application of the proposed study on a large study and Section VII provides the conclusion and future scope of the study.

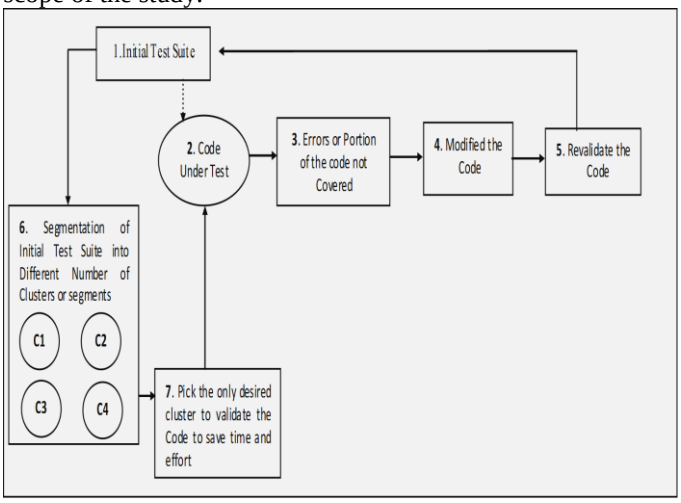

Figure. 1: The Flow Chart of the Proposed Approach

II. CATEGORIES OF REGRESSION TESTING TECHNIQUES

According to the study reported in [31], three techniques are commonly used for test suite management in order to reduce the cost of regression testing. They are

(1) Regression test selection techniques.
(2) Regression test suite minimization techniques.
(3) Test case prioritization techniques.

Regression test selection techniques [32, 33] attempt to reduce the cost of regression testing by selecting only a subset of test cases from the initial test suite according to the original and modified program. Test suite minimization techniques [34, 35] attempt to minimize the cost of regression testing by selecting a subset of test cases that provide the same coverage as provided by the existing test suite according to the specified coverage metric. Test suite minimization techniques form an effective minimized subset by permanently discarding the test cases which are found redundant and obsolete in the initial test suite according to the requirement coverage criteria defined. Test case prioritization techniques [36, 37] are also very useful as they help a tester to order the test cases by assigns the priority to each test case so that those test cases with higher priority can be executed before than those with lower value according to the need [36].

III. DATA CLUSTERING

Data clustering is a method used for the segmentation of data objects into groups (or clusters) in such a way that objects in one group are similar to one another in comparisons to the other objects present in other groups (clusters). According to [38], cluster analysis is used for two important purposes (1) understanding and (2) utility. Cluster analysis for understanding means applying clustering analysis for finding the meaningful groups from the data objects that share common properties [38]. Clustering plays an important role in analyzing, describing, and utilizing the hidden information present in the data. Clustering for utility attempts to abstract the prototypes or the representative objects from the clusters that serve as the basis for the number of data processing techniques like summarization, compression, and nearest neighbor finding [38]. The other important application domains of cluster analysis include machine learning, business intelligence, information retrieval, and pattern recognition.

IV. K-MEANS ALGORITHM

The K-Means algorithm is one of the simple, oldest, and widely used clustering algorithms [39, 40]. This algorithm is ranked second among top-10 data mining algorithms [41]. K- Mean's algorithm belongs to the class of hard or partitional based algorithms that attempts to find K non-overlapping clusters from the specified data. K is an external specified value that indicates the number of clusters that

needs to be formed after clustering the data. The basic steps of K-Means algorithm for finding k clusters are as follows:

I. Select 'k' as the number of number of initial centriods.
II. Calculate the distance of all the objects from the centriods and assign the objects accordingly to the closest centriods.
III. Recalculate the centriods of each cluster.
IV. Continue steps II and III until the centriods of each cluster do not change.

V. IMPLEMENTATION OF THE PROPOSED APPROACH

The graphical representation of the proposed approach is shown in figure. 1. We have implemented the proposed approach for the regression testing of the sample program shown in fig. 2. To test the sample program we have generated a suite of random test cases using a free online available tool known as generatedata.com [42]. The genratedata.com tool utilizes the random search technique for the generation of test cases and is commonly used for database and software testing. But the tool has certain limitations as it generated the test cases based on random approach that is (1) it generates a lot of redundant test cases and (2) cannot generate some combinations of effective test cases that are vital in order to cover certain components of the code and hence results into less percentage of the code coverage. Therefore, in order to remove these inconsistencies, we have seeded few effective combinations of test cases to the test suite generated with the tool using the study reported in [43]. In our previous studies [44, 45,46], we have also proposed some studies to handle the test suite size and randomness issues with respect to test suite minimization perspectives. But, here in this study the goal is to select a subset of test cases from the entire test suite with respect to modifications done to the code and to test the modified portion of the code without discarding the other test cases. The suite of test cases generated for testing the sample program is shown in figure. 3.

```
public class Traingleclass {
public  static void main(String [] args){
   Scanner S =new Scanner(System.in);
   int A,B,C;
System.out.println("ENTER THREE SIDES OF TRIANGLE");
A= S.nextInt();
B= S.nextInt();
C= S.nextInt();
S.close();
if(A>0 && B>0 && C>0){

1       if(A >= (B+C) || C >= (B+A) || B >= (A+C) )
           System.out.println("Not a triangle");

        else if (A==B && B==C)
2          System.out.println("Equilateral");

        else if(A!=B && B!=C && C!=A)
3          System.out.println("Scalene");

        else if((A==B) || (C==A))
4          System.out.println("Isosceles");
        }
        else{
5               System.out.println("Not Valid inputs");
        }
            }
```

Figure. 2. Program under Testing

Test Case ID	Side A	Side B	Side C	Expected Output
t1	0	1	1	Invalid
t2	10	10	10	Equilateral
t3	5	5	3	Isosceles
t4	8	7	4	Scalene
t5	9	6	8	Scalene
t6	9	9	9	Equilateral
t7	0	0	0	Invalid
t8	3	1	1	Invalid
t9	6	7	6	Isosceles
t10	6	5	3	Scalene
t11	-1	-2	-3	Invalid
t12	4	3	3	Isosceles
t13	-4	-4	8	Invalid
t14	5	10	7	Scalene
t15	6	-3	3	Invalid
t16	12	12	12	Equilateral
t17	2	2	0	Invalid
t18	2	0	2	Invalid

Figure. 3. The Initial Test Suite

For unit testing the sample code, we have used JUnit testing framework. JUnit is an open source as well as automatic unit testing framework for java code. And for code coverage measurement, EclEmma tool is used which is also an open source java code coverage tool for Eclipse. After executing all the test cases, the test cases except (T12) in the initial test suite execute all the statements of the code and hence achieve the 91.5% instruction, 88.5% line, 50% method, and 100% type code coverage as shown in figure. 4, but could not achieve an acceptable branch and complexity coverage due to some missing functionality or an error in the code. The code when executed with the test case $\mathbf{t_{12}}$ (**4, 3, 3, Isosceles**) fails because there is some missing functionality or the condition in the code as shown in **figure 2**. Therefore, it is important to rectify the error by adding the statements **(C==B)** to the existing code as shown in **figure.** 5 and then regression test it in order to handle all the combinations of test cases. Here, in order to regression test the code either we need to re-run all the test cases to re-test all the segments of the code or we need to select a subset of test cases to test the segment of code that is modified.

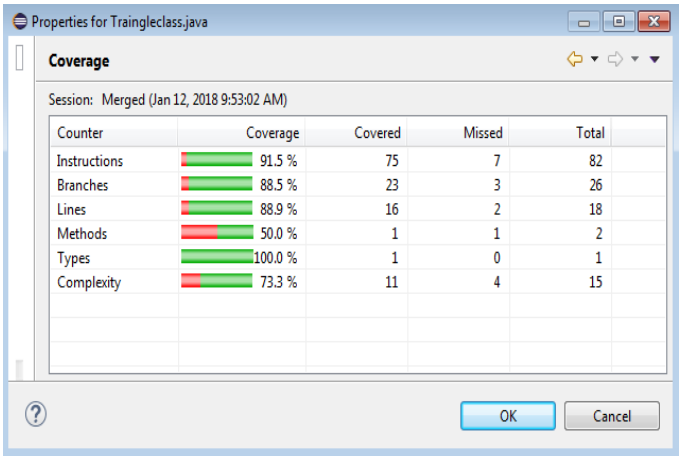

Figure. 4. Coverage of different structural components achieved by the initial test suite


```
public class Traingleclass {
public  static void main(String [] args){
   Scanner S =new Scanner(System.in);
   int A,B,C;
System.out.println("ENTER THREE SIDES OF TRIANGLE");
A= S.nextInt();
B= S.nextInt();
C= S.nextInt();
S.close();
if(A>0 && B>0 && C>0){

1       if(A >= (B+C) || C >= (B+A) || B >= (A+C) )
           System.out.println("Not a triangle");

        else if (A==B && B==C)
2          System.out.println("Equilateral");

        else if(A!=B && B!=C && C!=A)
3          System.out.println("Scalene");

        else if((A==B) || (C==A) || (C==B))
4         System.out.println("Isosceles");
         }
        else{
5             System.out.println("Not Valid inputs");
        }
            }
```

Figure. 5. The modified Program.

Therefore regression testing and particularly regression test case selection techniques are most commonly used after the software undergoes change in order to make assurance that the bugs or errors previously encountered have been fixed and the existing functionality of the code has not been altered. Hence, for quality assurance, we need to re-run all the test cases in order validate the changes happened to the code. But running all the test cases for validation when the changes are minor will consume extra effort, time and resources. The better alternative would be to use only few test cases that are related to the change not the whole test suite. For this reason, we have applied K-means algorithm for segmenting the test suite into different profiles or clusters so that when the need arises it become possible to use any particular segment or cluster of test cases for re-testing the segment of the code that has been modified not the whole test suite. Hence, after implementation of the proposed approach, we have segmented the test suite into four clusters C1, C2, C3, and C3 using K-means algorithm in WEKA, an open source machine learning and data mining toolkit. The segmented test suite is shown in figure. 6.

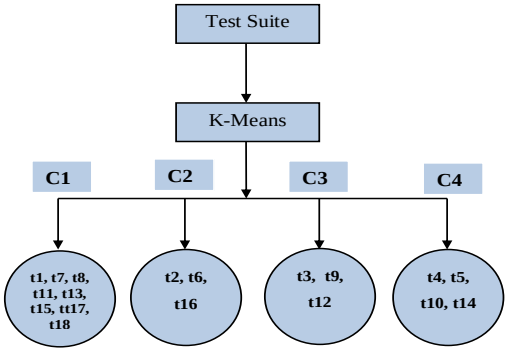

Figure. 6. Segmented Test Suite

After segmentation, each cluster contains the test cases which execute the same number as well as same type of structural components of the code. The structural components of the code executed by the clustered test suite is shown in Table 1.Thus with the proposed approach it become possible that if any component or statement of the code is modified or changed than only that particular cluster of test cases needs to be re-run to save the time and validate the changes not all the test cases present in other clusters. Here, in this case only the test cases in cluster 3 are needed to be re-run for validation. Also with the implementation of the proposed approach, it was observed that the redundancy that exists in the test suite is reduced by partitioning it into different segments or clusters based on the similarities in the test cases. The code coverage after re-running the test cases from the cluster 3 is shown in figure. 7. Therefore, a considerable amount of time, effort and resources can be saved during regression testing by the application of the proposed approach.

Table 1. Different Sections of the Code Covered by Each Cluster

Different Segments or Clusters formed using the Proposed Approach	**Area of the code shown in figure 5 that is Covered by each Segment or Cluster**
C1	Turquoise and Bright Green portion
C2	Pink portion
C3	Yellow portion
C4	Gray portion

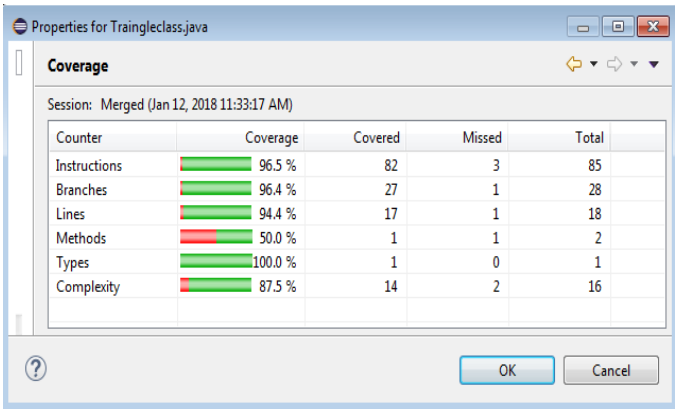

Figure. 7. The code coverage of the test suite after maintenance

VI. APPLICATION OF THE PROPOSED APPROACH ON DIFFERENT SET OF PROGRAMS

For the determination of applicability of the proposed approach, we use four well-known programs (Modified Versions) with different characteristics as shown in table 2.

Table 2: Sample Programs and their characteristics.

S. No.	Program Name	No. of Instructions	No. of Lines	No. of Branches	Complexity	Test Suite Size
P1	Roots of Quadratic Eq.	54	16	6	7	50
P2	Largest of Three Numbers	64	17	18	11	50
P3	Number is Prime or Not	45	17	8	6	50
P4	Number of Digits in a Given Number	42	14	4	4	50

For an effective management of the test suites (initially random and un-minimized) for an initial testing as well as regression testing, we have segmented each of them into an appropriate number of segments or clusters. The number of partitions or clusters (or the value of k) plays an important role in successful implementation of the K-Means algorithm. Therefore, to divide each test suite of the sample programs shown in table 2, the value of k for each test suite is chosen based on the number of requirements that are needed to be satisfied by each test suite. For example, the ideal value of k for segmenting the test suite of program P1 would be 4 because upon its execution it should classify the roots into equal, or imaginary, or unequal or invalid as shown in fig.8:

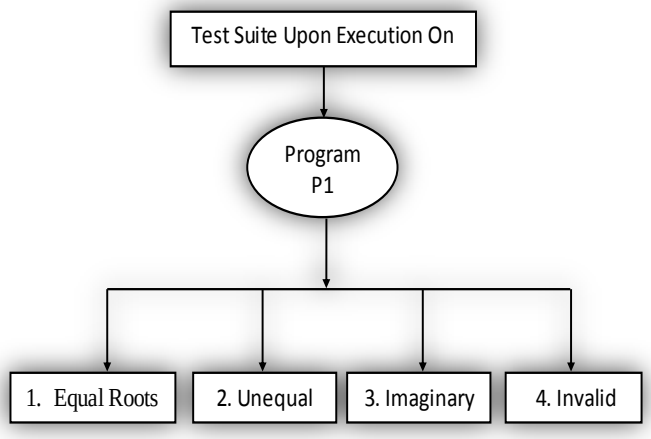

Figure.9: Requirements of the Program P1.

Therefore based on these facts, we have segmented each test suites of P1, P2, P3, and P4 into a different number of partitions or clusters as shown in figure 9, 10, 11, 12 for carrying out an effective regression testing activity.

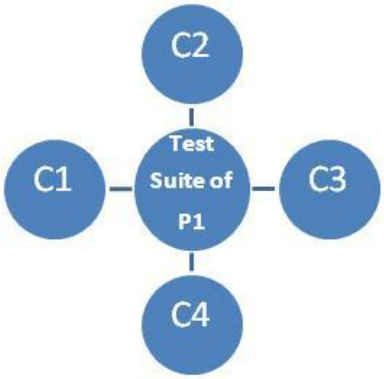

Figure. 9: Clusters formed after Clustering

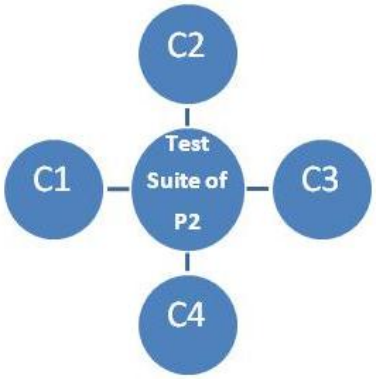

Figure. 10: Clusters formed after Clustering

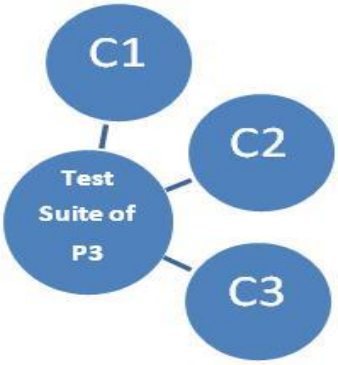

Figure. 11: Clusters formed after Clustering

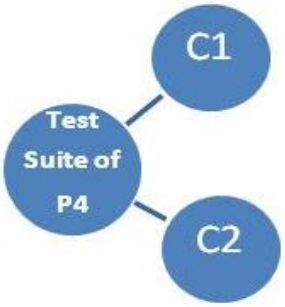

Figure. 12: Clusters formed after Clustering

Thus with the proposed approach all the test suites that are initially random and un-minimized are now minimized and partitioned into different segments for effective utilization in regression testing. The other benefits of the proposed approach are that it provides the tester a choice in selecting an appropriate cluster or portion of the test suite (a single test case from each cluster) for execution during an initial as well as in regression testing process. Therefore for regression testing the sample programs shown in table 2, an appropriate cluster or clusters of test cases from each partitioned test suite can be selected as shown in fig.9, fig. 10, fig.11 and fig.12. Also, with the help of the proposed approach, the time, effort and cost case be reduced as it become possible to selectively run few test cases instead of re-running the entire test suite for regression testing.

VII. CONCLUSION AND FUTURE SCOPE

Regression testing is an important activity used to validate the modified software in order to gain assurance that no errors are introduced into the previously tested code. But regression testing is an expensive activity as it involves repeatedly running the entire test suite whenever the code changes. To reduce some of this expense, researchers have proposed various regression test selection methods that attempt to reduce the cost by selecting and running only a subset of test cases from an existing test suite. With the same aim we have also proposed a regression test cases selection approach in which K-means clustering algorithm is implemented on the test suite in order to partition the test suite into a different clusters or profiles according the requirement they satisfy. Later on if any modification is done to the code we may be able to select only the particular group or cluster of test cases for retesting the code that is modified. The experimental results proved that the proposed approach is very effective and also a useful technique that will reduce the cost of regression testing by running only a subset or cluster of test cases for retesting the modified code. The future work in the direction would be to propose some approaches so that test cases in each cluster are automatically selected and executed for retesting the code that is changed as the selection and execution of test cases are currently done manually.

REFERENCES

[1] H. Leung, L. White," *Insights into regression testing"*, In Proceedings of the Conference on Software Maintenance, pages 60–69. 1989

[2] G. Rothermel., M. Harrold., "*Selecting tests and identifying test coverage requirements for modified software"*, In Proceedings of the International Symposium on Software Testing and Analysis, pages 169–184. 1994

[3] M. Grindal, J. Offutt, J. Mellin, "*On the testing maturity of software producing organizations"*. In TAIC-PART '06: Proceedings of the Testing: Academic & Industrial Conference on Practice and Research Techniques, pages 171–180. 2006

[4] J. Guan, J. Offutt, .P. Ammann , "*An industrial case study of structural testing applied to safety critical embedded software"*, In Proceedings of the 2006 ACM/IEEE international symposium on Empirical software engineering, pages 272–277. 2006

[5] T. Ball, "*On the limit of control flow analysis for regression test selection"*, In ISSTA '98: Proceedings of the 1998 ACM SIGSOFT international symposium on Software testing and analysis, pages 134–142. 1998

[6] R. Gupta, M. Harrold, and M. Soffa, "*Program slicing-based regression testing techniques"*. Journal of Software Testing, Verification, and Reliability, 6(2):83–112. 1996

[7] D. Binkley, "*Semantics guided regression test cost reduction"*, IEEE Transactions on Software Engineering, 23(8):498–516. 1997

[8] L. Briand, Y. Labiche, and S. He, "*Automating regression test selection based on UML designs"*. Information and Software Technology, 51(1):16–30. 2009

[9] M. Harrold., J. Jones,et,al;, "*Regression test selection for Java software"*, In Proceedings of the 16th ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages and Applications, pages 312–326. 2001

[10] A. Orso, N. Shi, and M. Harrold, "*Scaling regression testing to large software systems"*, In Proceedings of the 12th ACM SIGSOFT Twelfth International Symposium on Foundations of Software Engineering, pages 241–251. 2004

[11] C. Mao, Y Lu, and J. Zhang, "*Regression testing for component-based software via built-in test design*", In Proceedings of the 2007 ACM symposium on applied computing, pages 1416–1421. 2007

[12] J. Zheng, B. Robinson, L. Williams, K Smiley, "*Applying regression test selection for COTS based applications*". In ICSE '06: Proceedings of the 28th international conference on Software engineering, pages 512–522. 2006

[13] J. Gao D. Gopinathan., Q. Mai, "*A systematic regression testing method and tool for software components*". In Proceedings of the 30th Annual International Computer Software and Applications Conference (COMPSAC'06), pages 455–466. 2006.

[14] M. Ruth, S. Tu, "A *safe regression test selection technique for web services*". In Proceedings of the Second International Conference on Internet and Web Applications and Services, IEEE Computer Society. 2007

[15] A. Tarhini., H. Fouchal., N. Mansour., "*Regression testing web services-based applications*". In AICCSA '06 Proceedings of the IEEE International Conference on Computer Systems and Applications, pages 163–170. 2006

[16] L. Feng, M. Ruth, S. Tu ," *Applying safe regression test selection techniques to Java web services*". In International Conference on Next Generation Web Services Practices,. NWeSP 2006., pages 133–142. 2006

[17] M .Harrold. , M .Soffa, "*mInter-procedural data flow testing*". In Proceedings of the ACM SIGSOFT '89 third symposium on Software testing, analysis, and verification, pages 158–167. 1989

[18] A Taha, S. Thebaut, and S. Liu, "*An approach to software fault localization and revalidation based on incremental data flow analysis*". In Proceedings of the 13th Annual International Computer Software and Applications Conference, pages 527–534. 1989

[19] D. Binkle, "*Semantics guided regression test cost reduction"*. IEEE Transactions on Software Engineering,23(8):498–516. 1997

[20] S Bates, S Horwitz, "*Incremental program testing using program dependence graphs*". In Conference Record of 20th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, pages 384–396. 1993

[21] H. Leung, L. White, "A *study of integration testing and software regression at the integration level*". In Proceedings of the Conference on Software Maintenance, pages 290–300. 1990

[22] H Leung, L.White, "*A firewall concept for both control-flow and data-flow in regression integration testing*". In Proceedings of the Conference on Software Maintenance, pages 262–270. 1992

[23] G Rothermel, M Harrold, "*A safe, efficient regression test selection technique*". *ACM Transactions on Software Engineering and Methodology*, 6(2):173–210. 1997

[24] J. Laski, W. Szermer, "Identification *of program modifications and its applications in software maintenance*". In *Proceedings of the Conference on Software Maintenance*, pages 282–290. 1992

[25] F.Vokolos, P. Frankl. "*Empirical evaluation of the textual differencing regression testing Technique*". In ICSM '98: Proceedings of the International Conference on Software Maintenance, pages 44–53. 1998

[26] F .Vokolos, P. Frankl, Pythia: "*A regression test selection tool based on textual differencing*". In Proceedings of the 3rd International Conference on Reliability, Quality & Safety of Software-Intensive Systems (ENCRESS' 97), pages 3–21. 1997

[27] G. Baradhi , N. Mansour , "*A comparative study of five regression testing algorithms*". In Proceedings of Australian Software Engineering Conference, Sydney, pages 174–182. 1997

[28] J Bible, G. Rothermel, D. Rosenblum, "*A comparative study of coarse- and fine-grained safe regression test-selection techniques*". *ACM Transactions on Software Engineering and Methodology*, 10(2):149–183. 2001.

[29] E Engström., P Runeson., and M Skoglund, "*A systematic review on regression test selection techniques*". *Information and Software Technology*, 52(1):14–30. 2010.

[30] E Engström., M Skoglund., and P Runeson. "*Empirical evaluations of regression test selection techniques: a systematic review"*. In *Proceedings of the Second ACM-IEEE international symposium on Empirical software engineering and measurement*, pages 22–31, 2008.

[31] G Rothermel, H Roland. Untch, Chu Chengyun, and M. J. Harrold. " *Prioritizing test cases for regression testing"*. IEEE Transactions on Software Engineering, 27(10):929-948. 2001

[32] H Do and G Rothermel, " *An empirical study of regression testing techniques incorporating context and lifetime factors and improved cost-benefit models*". In Proceedings of the 14th ACM SIGSOFT International Symposium on Foundations of Software Engineering. 2, 22.

[33] L. G Todd, M. J Harrold., J.M Kim., A Porter, and G Rothermel, "*An empirical study of regression test selection techniques*". ACM Transactions on Software Engineering and Methodology, 10:188-197. 2001

[34] M. J. Harrold, R Gupta, and M. L Sofa, "*A methodology for controlling the size of a test suite*". ACM Transactions on Software Engineering and Methodology (TOSEM), 2(3):270-285,. 1993

[35] H. Hwa-You and A. Orso (2009). Mints: "*A general framework and tool for supporting test-suite minimization"*, In Proceedings of the IEEE 31st International Conference on Software Engineering (ICSE'09), pages 419-429.

[36] L. Chen, Z. Wang, L. Xu, Hongmin, and Xu Baowen, "*Test case prioritization for web service regression testing*". In Proceedings of the 5th IEEE International Symposium on Service Oriented System Engineering (SOSE'10), pages 173-178. 2010.

[37] S. Elbaum, A.G .Malishevsky, and G. Rothermel, "*Test case prioritization: a family of empirical studies*". IEEE Transactions on Software Engineering, 28(2):159-182. 2002

[38] P. N Tan, , M. Steinbach, V. Kumar," *Introduction to Data Mining"*, Addison-Wesley, Reading .2005

[39] S Lloyd. "*Least squares quantization in pcm"*. IEEE Trans. Info. Theory 28(2), 129–137. 1982.

[40] A Jain., R Dubes. "*Algorithms for Clustering Data*". Prentice Hall, Englewood Cliffs. 1988

[41] X. Wu,. V. Kumar, J.R Quinlan, , J Ghosh, Q. Yang, et,al, "*Top 10 algorithms in data mining"*, Knowl. Inf. Syst. 14(1), 1–37. 2008.

[42] Genratedata.com test data generation tool, htpp://www.generatedata.com.

[43] A. K. Gupta., F. A. Khan, "*An Efficient Test Data Generation Approach For Unit Testing"*, IOSR (JCE), Volume 18, Issue 4, Ver. V, PP 97-107. 2016.

[44] F. A. Khan., A.K Gupta, D.J Bora, "*Profiling of Test Cases with Clustering Methodology*". International Journal of Computer Applications, Vol.106 (14), pp. 32-37. 2014.

[45] F. A. Khan ,A.K Gupta, D.J Bora. "*An Efficient Heuristic Based Test Suite Minimization Approach*", Indian Journal of Science and Technology, ISSN (Print): 0974-6846, ISSN (Online) : 0974-5645, Volume 10(29), pp. 1-8. 2017.

[46] F.A. Khan, A.K. Gupta, D.J. Bora, "*An Efficient Technique to Test Suite Minimization using Hierarchical Clustering Approach*", International Journal of Emerging Science and Engineering (IJESE) ISSN: 2319–6378, Volume-3 Issue-11, 2015.

