ISSN: 2278-4632
Vol-10 Issue-6 No. 9 June 2020

Optimized Fuzzy Hierarchical Clustering based Automated Test case generation for Improving Regression Testing on Mobile Apps

R.Balamurugan

Assistant Professor Computer of Computer Science

Sri Ramakrishna Mission Vidyalaya

College of Arts and Science

Coimbatore

M.Ravichandran

Associate Professor of Computer Science

Sri Ramakrishna Mission Vidyalaya College of Arts and Science

Coimbatore

Abstract

In this modern era, usage of mobile applications is growing in a high rate, to fulfill the users need and to increase the reliability of the mobile applications, the software related to it has to be updated frequently. It is common to undergo alterations and improvements in mobile operating system to have a negative impact on mobile apps. This is due to the factor that testing such upgrade version has to be done very efficiently, so that regression testing is usually for software maintenance because, it may increase cost of maintenance in mobile applications. The main objective of this paper is to decrease the cost of regression testing, by developing an automated test case, that cluster the test cases into effective and non-effective groups. The vagueness in determining the effective test cases are tackled by developing fuzzy based hierarchical clustering which represents each test case in terms of degree of membership towards effective and non-effective. This proposed fuzzy hierarchical clustering focuses to discover only on effective test cases by eliminating the non-effected test cases for re-execution process. The observed outcome of the results showed the test case clustering using fuzzy can efficiently determine the can potential test cases which performs maximum coverage of tests with high accuracy while comparing the standard k-means clustering models.

Key words: test case, fuzzy hierarchical clustering, regression testing, k-means, effective. Software maintenance

ISSN: 2278-4632
Vol-10 Issue-6 No. 9 June 2020

Introduction

Decisive goal, while performing regression testing is to confirm that any new updates of applications or features do not present new bugs. The software testing done on traditional applications like desktop or web differs from mobile applications. The principal of testing is same, but the methodologies used for developing mobile applications and their scope involved are more complex than traditional applications. While working with web applications it holds the details of which browser it uses, version supported, etc. This is also complex while performing usability, functionality, performance and security testing.

While performing mobile testing it comprised of various combinations of mobile browsers, but they also using progressive web applications. While using different types of mobile OS like android and iOS it is really very difficult and challenging to perform testing process.

The speed and changes done in mobile application development are agile which leads to continuous testing and delivery process. In such continuous testing of mobile applications or operating systems regression defects are an effective way of determining the quality of the updated software. The regression testing helps to save both cost and time. An important activity in software maintenance is regression testing which is very expensive. The regression testing in mobile applications helps the testers to ensure whether there is any introduction of bugs while performing updates or changes in the existing software. This test also helps the testers to focus on only the portion of the module that has been affected during refactoring or maintenance activities. The software defects revealed during regression testing are chiefly examined by two sources, new bugs introduced while performing the modification or because of added code and existing bugs which are hidden or unrevealed in the previous testing stage.

Problem statement

The complexity of standard regression testing process depends on whether re-executing a portion or all test cases devised for the program or software under test. A major issue is the cost related with re-executing entire large test suite because cost of executing regression testing covers two third of software life cycle []. This necessitates to select a better regression tests which by selecting optimal test cases, searching for execution of test case order based on its cost effectiveness and finally minimizing the set of test cases to be involved to apply on a newly updated version.

Contribution

This paper devised a clustering model which categorizes the test cases into two different

ISSN: 2278-4632
Vol-10 Issue-6 No. 9 June 2020

groups. The goal is to discover and emphasis only on effected test cases and eliminating the needs for re-executing non-effected test cases in mobile applications. While clustering the test cases into two different groups, the test cases within the cluster are more similar to each other. Henceforth, the cluster that holds previously failing test cases, must be given higher priority of execution. Those higher priority test cases while applied to regression testing, it will reduce the cost substantially []. Additionally, to handle the vagueness of some of the test cases which lie in the border of both the clusters, that is when there is confusion in deciding whether the particular belongs to effective or ineffective, this proposed model introduced a connectivity-based clustering which includes the fuzziness in representing the test cases with the degree of membership towards effective and ineffective group.

The main contributions of this proposed model are as follows:

- Selecting an optimal set of test cases to reduce the cost of regression testing
- Developing fuzzy hierarchical clustering model with test cases profile information to successfully cluster the test cases based on the membership values.
- Examining and Assessing the performance of the proposed model with various factors such as source code coverage, quantity of faults and feature construction

Background

Regression Testing

There are different types of regression testing methods available they are, selection of test case, test case execution based on priority and minimization o f test suite. To prioritize the test case two basic ideas are used in common, greedy search-based prioritization and heuristic based search. In greedy search method, ordering test cases in a descending order, with the anticipation to detect newly introduced faults in earlier stages of regression testing. But in heuristic search, the minimum set of test cases which can able to determine existing hidden or newly injected faults are chosen. Based on the type of strategy involved in searching, the prevailing search approaches accomplish the adequacy criteria for regression testing. More accurately, the tester is unaware about when the goal of regression testing will be reached or enough testing is done. Henceforth, the testers re-execute the complete test suite for newly upgraded version of mobile software's.

ISSN: 2278-4632
Vol-10 Issue-6 No. 9 June 2020

While using the selection-based testing in regression, its goal is to discover a set of test suit from the previously released applications by focusing on the portion of code, which may induce faults. In selection process the test cases which has the ability to detect faults caused by the change or affected parts alone are considered for testing. It is also stated that depending on the type of techniques used the regression testing cost can be reduced considerably []. Minimization of test suit mainly focuses on discovering redundant test cases and alleviating their execution with the objective of reduced test suite size, which results in reducing time and effort involved to preform regression testing.

Clustering

A common technique in statistical and data mining techniques are modelling clustering methods which can be classified into four different models, such as centroid based clustering, connectivity-based clustering, density-based clustering and distribution-based clustering. In this paper hierarchical based clustering and centroid based clustering which fit for performing regression testing is explained.

K-means as Centroid-based Clustering

In this clustering model, the clusters are framed based on the central vector, this work used K-means clustering to categorize the test cases involved in mobile applications to perform regression testing. K-means is one of the popular and simplest centroid algorithms which is applied on a set of data points. In this paper the data points are test case, instead of using entire test cases, the prominent test cases which can cover maximum test path are discovered by this model. This model is best suited even of large datasets and thus it is applied successfully in various areas of real time applications.

With a set of test cases X = (x_1, x_2, ..., x_n), where each test case is a d dimensional real vector, the object of k means is to segment the n testcases into k sets depending on the similarity among the test cases by determining the sum of squares within the clusters. μ_j is the mean of the cluster X_m.

$$\sum_{m=1}^{n} \sum_{x_i \in X_m} \|x_i - \mu_m\|$$

The basic steps involved in k-means clustering are as follows:

- ✓ Initializing: Define K test case into a search space to denote then as centroid of each group
- ✓ Clustering: Assign each test case to the group that has the closest centroid
- ✓ Apprising: Each group centroid has to be updated

ISSN: 2278-4632
Vol-10 Issue-6 No. 9 June 2020

- ✓ Reiterating: Perform Re-clustering has to be done until the termination condition are met

Table 1. *Test Case Classification using k-Means Clustering*

Algorithm 1 *Test Case Classification using k-Means Clustering*

Input: *Test Cases (TC)*
Output: STC, IS*TC*
1: Initialize *STC*, *ISTC*
2: Calculate *mean(STC)*, *mean(ISTC)*
3: **while** *mean(STC)*, *mean(ISTC)* different **do**
4: **for** each *tci* in *TC* **do**
5: *DSTC*=Euclidean-Distance(*tci*,*mean(STC)*)
6: *DISTC*= Euclidean-Distance (*tci*, *mean(ISTC)*)
7: **if** (*DSTC*> *DISTC*)
8: *STC* = STC $\vee\{tci\}$
9: **else**
10: *ISTC* = ISTC $\vee\{tci\}$
11: **end if**
12: **end for**
13: Update *meanSTC*, *meanISTC*
14: **end while**

Hierarchical Clustering

Another way of clustering strategy is to construct a hierarchy of clusters, in general by adopting bottom up approach is termed as hierarchical clustering or agglomerative. In this clustering model it begins with its own cluster, and gradually start merging with pair of clusters as it moves up the hierarchy. The distance between two clusters are measured as follows

$$dIst\ (x,y) = \frac{1}{\|C\|.\|D\|}\sum_{x\in C}\sum_{y\in D} dist(x,y)$$

$$dist(x,y) = \sqrt{\sum_{k}(x_k - y_k)^2}$$

ISSN: 2278-4632
Vol-10 Issue-6 No. 9 June 2020

Where dist refers to euclidean distance.

Procedure for performing hierarchical clustering as follows:

- Assigning cluster: construct cluster for each instance, if there are m number of instances, m clusters are constructed, each cluster comprised of single instance. Similarity among clusters are defined by the distance among them.
- Merging clusters. Find the closest pair of clusters and merge them in to single, so that the quantity of clusters is reduced by one during each iteration.
- Repeating: repeat merging process until all the instances are clustered in to single cluster.

Proposed Methodology

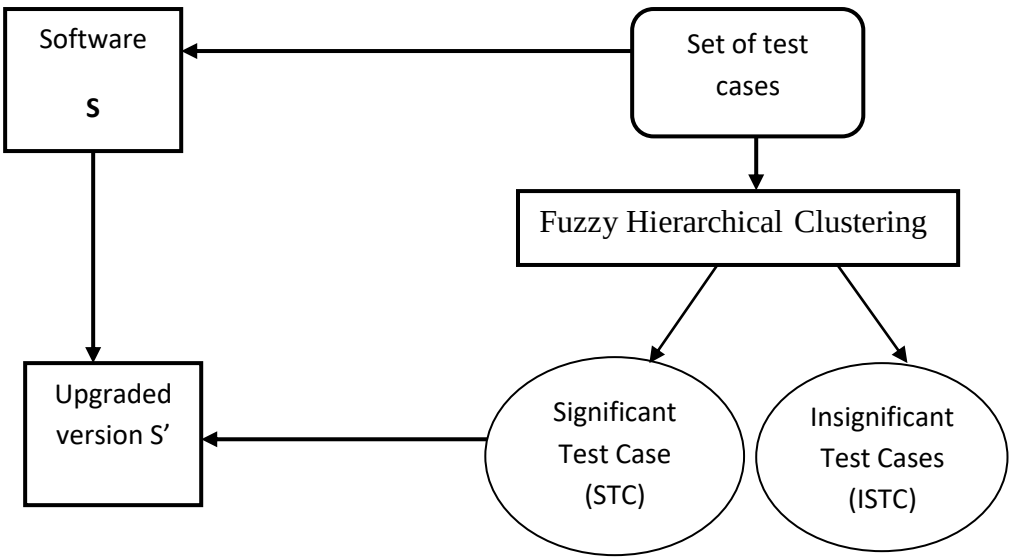


Fig. 1. Fuzzy Hierarchical Clustering based test case generation

The figure 1 illustrates the proposed model of Fuzzy hierarchical clustering to perform test case generation for regression testing. For a given two consecutive versions of software's by fixing the previous statements and altering or updating some executive statements. With a given set of test cases this work aims at classifying each test case into one of the two categories name the significant and insignificant test cases. The main objective of this research work is to discover significant test cases by applying fuzzy hierarchical clustering, which selects potential test cases and avoid insignificant test cases which results in

ISSN: 2278-4632
Vol-10 Issue-6 No. 9 June 2020

minimizing the number of test cases that it has to be executed for the newly released version of software.

- Given software under test which is represented as S
- S' is the updated version of S
- Tc = (tc_1, tc_2, $tc_{3,}$...tc_m) is a possible test suite for m test cases for s
- Stmt = {$stmt_i$} is the affected set of statements in S when upgraded to S' 1<= m<=loc where local denotes line of codes
- STC is a subset of T which consist of significant test cases
- ISTC is a subset of T which comprised of insignificant test cases

In this proposed work the fuzzy hierarchical clustering is involved in determining the significant test cases for the updated version of S' by using the previous testcases used for the older release S. The basic functionality of this clustering model is to cluster the testcases as significant and insignificant test cases based on the updated version of the software. Thus by eliminating the group of insignificant test cases the optimality of the test case generation becomes effective then the standard test strategies.

The Algorithm

The approach employs Euclidean distance metric along with code coverage information to measure the similarities/dissimilarities between two test cases. Code coverage is a measure used in software structure testing to determine the adequacy of testing. The code coverage based on program statements is the simplest form of this adequacy criterion, which aims at checking whether each executable statement in a given program has been exercised. In the proposed technique, using the binary numeric values 1 and 0 to represent covered/not covered statement; it is possible transform the representation of coverage of statements by each test case to a real value vector. If the Euclidean distance between the vector representations of two test cases is zero then the two test cases are seemingly similar. Similarly, if the Euclidean distance value obtained is some non-zero value, we may assume that the two test cases are different. It is important to note that the magnitude of the Euclidean distance may reflect the significance differences between two test cases and thus their code coverage. A large Euclidean distance indicates that the difference between two test cases is significant. Algorithm 1 and Algorithm 2 describe the procedure of the

ISSN: 2278-4632
Vol-10 Issue-6 No. 9 June 2020

proposed technique when *k*-means and hierarchical clustering algorithms are adopted, respectively.

ISSN: 2278-4632
Vol-10 Issue-6 No. 9 June 2020

Algorithm 2: Discovering Significant Test cases using Fuzzy Hierarchical clustering

Input: set of Test Cases TC

Output: *STC, ISTC*

1: *Clust*={}

2: **for** each *tci* in *S* **do**

Convert tci to ftci // conversion of crisp data to fuzzy data

$$\mathrm{MS(tci)}\ (x;\tau,\sigma,\lambda) = \begin{cases} 0 & x \le \tau \\ 2\left(\frac{x-\tau}{\lambda-\tau}\right)^2 & \tau \le x \le \lambda \\ 1-2\left(\frac{x-\tau}{\lambda-\tau}\right)^2 & \sigma \le x \le \lambda \\ 1 & x \ge \lambda \end{cases}$$

3: Construct a cluster cl_i for *tci*

4: *Clust=Clust* $\cup$ { cl_i }

5: **end for**

6: **while** more than two number of clusters **do**

7: Discover the closest pair of clusters using Fuzzy Euclidean Distance

8: Join the pair into one cluster

9: Remove the pair of clusters from *Clust*

10: Add the new cluster into *Clust*

11: **end while**

12: cl_i = the cluster which contains the previous failing test cases

13: *STC*=cl_i

14: *ISTC =TC/STC/*

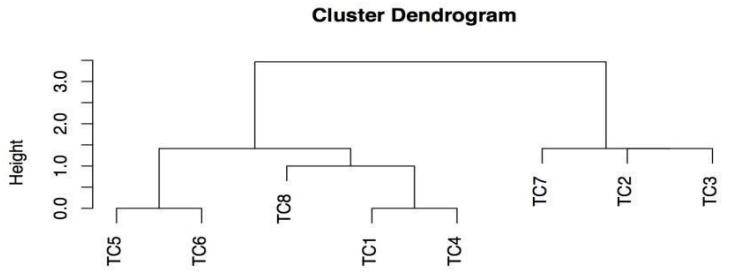


Fig. 2. Results of Fuzzy hierarchical clustering.

ISSN: 2278-4632
Vol-10 Issue-6 No. 9 June 2020

An Illustrative Example

In order to have a better insight of the proposed technique, we present an illustrative example. The code snippet given in Table 3 implements a class to compute sum of the abstract values of two values. Since the input numbers could be either double or integer, an override method is implemented which can take different input types. The code is composed of 13 lines with one class *AbsSum* along with two methods. A fault is injected on line 11. Eight test cases are devised for the purpose of testing the class and its functionality, and the coverage profile is shown in Table 1 where "–" means non-covered, and "√" indicates the underlying line is covered.

Table 3. An Illustrative Example

		TC1	TC2	TC3	TC4	TC5	TC6	TC7	TC8
	c	-3	-2.5	-2.5	-4	3	2	2.5	2
	d	-2	-1.5	2.5	-3	2	3	1.5	-2
1	*public class ABC{*	√	√	√	√	√	√	√	√
2	*public int Absdiff(int c, int d){*	√	–	–	√	√	√	–	√
3	*if(c<0)*	√	–	–	√	√	√	–	√
4	*c=-c;*	√	–	–	√	–	–	–	–
5	*if(d<0)*	√	–	–	√	√	√	–	√
6	*d=-d;*	√	–	–	√	–	–	–	√
7	*return c-d;}*	√	–	–	√	√	√	–	√
8	*public int Absdiff(double c, double d){*	–	√	√	–	–	–	√	–
9	*if(c<0)*	–	√	√	–	–	–	√	–
10	*c=-c;*	–	√	√	–	–	–	–	–
11	***if(d<-4)***	–	√	√	–	–	–	√	–
12	*d=-d;*	–	√	–	–	–	–	–	–
13	*return c+d;}}*	–	√	√	–	–	–	√	–
		P	F	P	P	P	P	P	P

Machine learning algorithms often involve building vectors, in which intermediate data are held for further processes. Moreover, the vectors represent a high-dimensional space and hold values for possible features that have been taken into account while performing the classification. We simply use binary value 0 or 1 for the purpose of building read value vectors. Table 3 shows the vectors representing each test case.

By calling Algorithms 1 and 2, we obtain the classification results. The *k*-means clustering divides

ISSN: 2278-4632
Vol-10 Issue-6 No. 9 June 2020

the test cases into two sets, {1, 4, 5, 6, 8} and {2, 3, 7}, labeled as 1 and 2. The previously failing test cases fall into cluster 2. Therefore, we label all the test cases in cluster 2 as effective test cases. When performing regression testing, the test cases in this cluster will be re-executed. Similarly, cluster 1 holds non-effective test cases. Fig. 2 depicts the results of hierarchical clustering. As shown in Figure 2,, the results of hierarchical clustering and *k*-means clustering are consistent.

Experimental Study

Subject Programs

Table 4 lists the subject programs used for the experimentation. We obtained these extensively used modest sized Java programs, including Nanoxml, Jtopas, Jmeter, XML-security, and ant from the Software Infrastructure Repository [17]. The first two programs are TSL (Test Specification Language) test suits, and the last three are based on Junit test framework. Nanoxml is an XML parser for Java. Jtopas is a small Java library for tokenizing and parsing texts. Jmeter is a Java desktop designed to load test functional behavior and measure performance. XML-security library includes a mature digital signature and encryption implementation. The ant program is a Java library and command line tool whose mission is to drive processes described in build files as targets and extension points dependent upon each other. Table 4 lists the number of versions for each program, the number of class files in the most recent version, the number lines of statements in the most recent version, and the number of test cases available for the most recent version.

Table 4. Subject Programs. LOC: Lines of Codes, NC: Number of Classes, NT: Number of Test Cases, NV: Number of

Program	Description	NV	NC	LOC	NT
NanoXML	XML parser	6	26	7,646	216
Jtopas	XML encryption	3	50	5,400	126
Jmeter	Text parser	6	389	43,400	78
XML-security	Load tester	4	143	21, 613	83
Ant	Text parser	9	627	85,400	877

Experimental Setup

To perform test case classification, we require two sets of information: coverage information and

ISSN: 2278-4632
Vol-10 Issue-6 No. 9 June 2020

initial clustering data (i.e., training data). The *TE* includes all the failing test cases in the previous version, and the *TN* contains just one case, the virtual test case that covers no statements. We obtained coverage information by running test cases on the instrumented subject programs. We instrumented each program by inserting **print** statement into each block to get the converge information. The coverage information obtained for the original program was then used to cluster current version's test cases.

Evaluation Metric

In the field of statistics and for the classification purposes, four key terms including true positives (*tp*), true negatives (*tn*), false positives (*fp*), and false negatives (*fn*) are usually computed for comparing the results and assessing the performance of the classifier utilized. The terms positive and negative refer to the classifier's prediction, also known as the expectation, and the terms true and false refer to whether that prediction corresponds to the external judgment, also known as the observation [18]. These terms and their associations are illustrated in Table 5 for classification of test cases.

Table 5. Test Cases Classification

Truly Effective	**True Non-effective**	
Predicted Effective	tp	fp
Predicted Non-Effective	fn	tn

Accordingly, three major measurement metrics, i.e. accuracy, precision and recall are usually used to assess how well a binary classification is performed. The accuracy of a measurement system is the degree of closeness of measurements of a quantity to that quantity's actual (true) value. It is the percentage of sum of all true positives and false negatives out of the sum of all the true positives, true negatives, false positives,
and false negatives [18].

$$accuracy = \frac{tp + tn}{tp + tn \;\; fp + fn}$$

ISSN: 2278-4632
Vol-10 Issue-6 No. 9 June 2020

The precision of a measurement system, also called as *reproducibility* or *repeatability*, is the degree to which the repeated measurements under unchanged conditions show the same results [18]. It is formulated as the fraction of the number of true positives to the sum of true positives and false positives.

$$precision = \frac{tp}{tp + fp}$$

ISSN: 2278-4632
Vol-10 Issue-6 No. 9 June 2020

The recall measurement in this context is also referred to as the true positive rate or *sensitivity*, is the ratio of true positives over the sum of true positives and false negatives or the percentage of flows in an

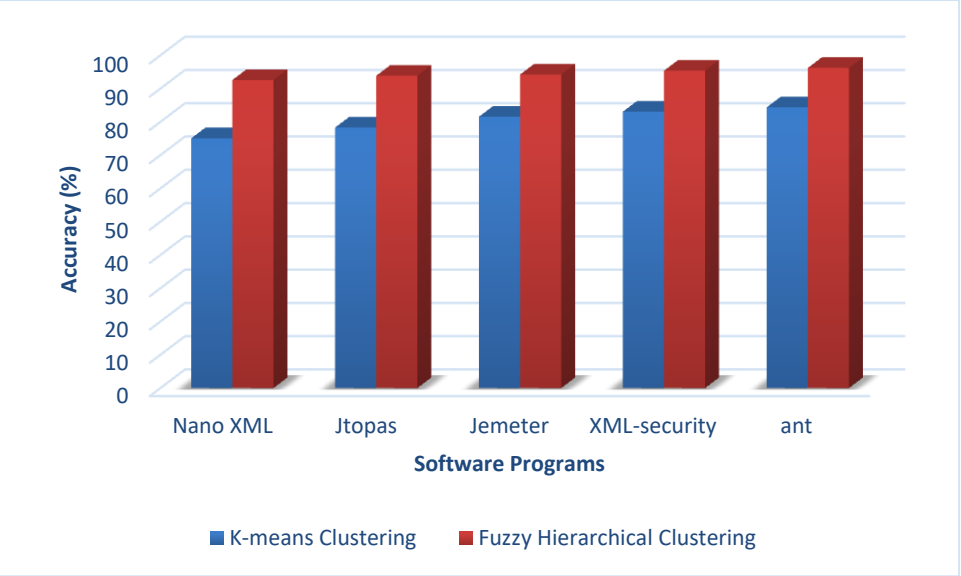
application class that are correctly identified [18].

$$recall = \frac{tp}{tp + fn}$$

Table : Performance of the Proposed Fuzzy Hierarchical Clustering with K-means Clustering

	K-means Clustering			**Fuzzy Hierarchical Clustering**		
	Accuracy	**Precision**	**Recall**	**Accuracy**	**Precision**	**Recall**
Nano XML	75.06	79.04	81.62	92.57	94.28	96.41
Jtopas	78.29	80.09	82.47	93.85	96.52	97.16
Jemeter	81.54	83.17	83.42	94.22	95.89	96.72
XMLSec	83.06	86.49	86.51	95.36	96.09	96.95
ant	84.37	84.92	85.74	96.21	96.87	97.57

ISSN: 2278-4632
Vol-10 Issue-6 No. 9 June 2020



Comparison based on Accuracy

For the five subjects' programs, the recall ratio is 100% demonstrating that we can effectively identify almost all of the actual effective test cases. In other words, the value 100% indicates that all test cases, which can expose a fault, were classified into the effective category by our approach. High accuracy indicates that for most cases, both the actual effective and actual non-effective test cases are classified properly and thus minimizing the cost of regression testing by not running non-effective test cases.

Conclusion

This work introduced a test case classification methodology based on fuzzy hierarchical clustering to enhance regression testing. Based on our empirical study concluded that the clustering-based test case classification can partition test cases with high recall ratio and considerable accuracy percentage. The paper also found out that the clustering-based approach performs better when first the block coverage criterion is utilized, second when single statement or pairwise feature are constructed, and third the performance deteriorated when the number of faults increases. It is also observed that for some subject programs failing test cases are always assigned into different clusters thus make it impossible to do binary clustering. One possible solution is to build more than one clusters and further experiments are needed

ISSN: 2278-4632
Vol-10 Issue-6 No. 9 June 2020

References

[1] Pressman, R. (2002). *Software Engineering: A Practitioner Approach*. McGraw-Hill, New York.

[2] Pang, Y., Xue, X., & Namin, A. S. (2013). Identifying effective test cases through k-means clustering for enhancing regression testing. *Proceedings of the 2013 12th International Conference on,Machine Learning and Applications* (pp. 78–83).

[3] Anderberg, M. R. (1973). Cluster analysis for applications. *DTIC Document*.

[4] J. MacQueen., *et al.* (1967). Some methods for classification and analysis of multivariate observations.

Proceedings of the fifth Berkeley Symposium on Mathematical Statistics and Probability.

[5] Sibson, R. (1973). Slink: An optimally efficient algorithm for the single-link cluster method. *The*

Computer Journal, *16(1)*, 30–34.

[6] Defays, D. (1977). An efficient algorithm for a complete link method. *The Computer Journal*, *20(4)*, 364–

366.

[7] Mirarab, S., & Tahvildari, L. (2008). An empirical study on Bayesian network-based approach for test case prioritization.

[8] Elbaum, S. G., Malishevsky, A. G., & Rothermel, G. (2002). Test case prioritization: A family of empirical

studies. *IEEE Trans. Software Eng.*, *28(2)*, 159–182.

[9] Rothermel, G., Untch, R. H., Chu, C., & Harrold, M. J. (2001). Prioritizing test cases for regression testing.

IEEE Trans. Software Eng., *27(10)*, 929–948.

[10] ASE 2009. *Proceedings of the 24th IEEE/ACM International Conference on Automated Software*

Engineering.

[11] Harrold, M. J., Rosenblum, D. S., Rothermel, G., & Weyuker, E. J. (2001). Empirical studies of a prediction model for regression test selection. *IEEE Trans. Software Eng., 27(3)*, 248–263.

[12] (2009). *Proceedings of the 25th IEEE International Conference on Software Maintenanc*.

[13] Rothermel, G., & Harrold, M. J. (1998). Empirical studies of a safe regression test selection technique.

IEEE Trans. Software Eng., 24(6), 401–419.

[14] Travassos, G. H., Maldonado, J. C., & Wohlin, C. (2006). *Proceedings of the International Symposium on*

Empirical Software Engineerin.

[15] Marre, M., & Bertolino, A. (2003). Using spanning sets for coverage testing. *IEEE Trans. Software Eng.*,

29(11), 974–984.

[16] Ernst, M. D., & Jensen, T. P. (2005). *Proceedingsofthe 2005 ACM Sigplan-Sigsoft Workshop on Program*

Analysis for Software Tools and Engineering.

[17] Do, H., Elbaum, S. G., & Rothermel, G. (2005). Supporting controlled experimentation with testing techniques: An infrastructure and its potential impact. *Empirical Software Engineering: An International Journal, 10(4)*, 405–435.

[18] Olson, D. L., & Delen, D. (2008). Advanced data mining techniques.

[19] Xue, X., Pang, Y. & Namin, A. S. (2014). Feature selections for effectively localizing faulty events in GUI applications. *Proceedings of the 2014 13th International Conference on Machine Learning and Applications*.

[20] Xue, X., Pang, Y. & Namin, A. S. (2014). Trimming test suites with coincidentally correct test cases for

enhancing fault localizations. *Proceedings of the 2014 IEEE 38th Annual. Computer Software and*

Applications Conference.

[21] Xue, X., & Namin, A. S. (2013). How significant is the effect of fault interactions on coverage-based fault localizations?. *Proceedings of the 2013 ACM/IEEE International Symposium on*

ISSN: 2278-4632
Vol-10 Issue-6 No. 9 June 2020
Empirical Software Engineering and Measurement.

