Journal of Computer Science 10 (1): 138-142, 2014
ISSN: 1549-3636
© 2014 Science Publications
doi:10.3844/jcssp.2014.138.142 Published Online 10 (1) 2014 (http://www.thescipub.com/jcs.toc)

A NOVEL APPROACH FOR TEST SUITE PRIORITIZATION

[1]Thangavel Prem Jacob and [2]Thavasi Anandam Ravi

[1]Department of Computer Science and Engineering, Sathyabama University, Chennai, India
[2]Srinivasa Institute of Engineering and Technology, Chennai, India

Received 2013-09-10; Revised 2013-10-05; Accepted 2013-11-12

ABSTRACT

Software testing is an expensive, time consuming, important activity that controls the quality of the software and important part of the software development and the maintenance. In testing the time is spent mainly for generating test cases and to test them. Whenever the software product gets modified, a group of the test cases has to be re-executed and the new output has to be compared with old one for avoiding the unwanted changes. If there is a match then the modifications that are made in the software will not affect other parts of the software. It is not practically possible to re execute all the test case in the program if any change has occurred. This problem of selection of those test case in regression testing can be re-solved by prioritizing the test case. This technique will reduce the testing effort. Different techniques were proposed in the past decades and still require further improvement. Here we propose a clustering based prioritization of the test case. The results achieved shows that prioritizing the test case has enhanced effectiveness of the test case.

Keywords: Regression Testing, Test Case, Genetic Algorithm, Test Suite

1. INTRODUCTION

Software developers frequently save the available test suites in such a way if there is any modifications they can able to reuse them. Software testing has a vast area of applications that ranges from the subroutines to some large application system which may have huge number of statements. Many of the software development organization have believed that the software which is independently developed and its operation will provide better testing and improved security (Arafeen and Do, 2013). The main objective of testing is to prevent the bugs. By designing the suitable test cases will improve the software quality.

The test case designing will be the challenging task. The testing process that uses some procedures that are predefined and the outcomes are predictable. But can't judge whether the program that is under test passes the specified test will be unpredictable. The testing process has to be well planned, scheduled, designed and has to be prioritized (El-Koka *et al.*, 2013). The testing process will show that the faults present in the code and design. The testing process will prove the program failure. The test automation will be achieved only in the design and execution. The main objective of prioritization will be minimizing the test suites. The prioritization of the test case will schedule the test case in such a way that it maximizes the objective function.

Since the testers schedule the test cases in such a way that it achieves the maximum code coverage in the possible faster rate. These minimization techniques will lower the cost since it reduces the test suite. There are different prioritization categories are constructed for those test cases, in order to remove the bugs, test cases has to be executed earlier before releasing the final software product. The test cases can be executed if there is enough time (Karnavel and Santhoshkumar, 2013). Before the current product release, test cases will not be given importance. Only after the current software is released it will be tested. The test case will not be given importance since the impact of the test case will be negligible.

Corresponding Author: Thangavel Prem Jacob, Department of Computer Science and Engineering, Sathyabama University, Chennai, India

This priority scheme will ensure the test cases of low priority will not create any problem for the software. Some customers may demand all the features in the software product has to be presented and tested at the initial version of the software product. The coverage criteria have to be met in the earlier stage of the testing process (Di Nardo *et al*., 2013).

2. PRIORITIZATION OF THE TEST CASE

Prioritization will schedule the test suites in accordance with some criteria. If the test suites are arranged in specific order then they meet the objective rather they would not meet if grouped in any other order (Jacob and Ravi, 2013a).

By prioritization of the test case we can able to address some different variety of the objectives like fault detection rate can be increased by the testers, faults with higher risks can be detected earlier (Polo *et al*., 2013). The regression errors possibility can be enhanced to a faster rate and it can make the system reliable.

2.1. Prioritizing the Fault Detection Rate

Different prioritization techniques can be applied for the test suites. The test case can be prioritized by the failure rates or we can prioritize the test case by increasing the cost-per-coverage in the requirement features (Shaccour *et al*., 2013). In this approach the possibility of revealing the faults have to be increased at an earlier stage. Early feedback provides information of the quality goals that are not met (Jacob and Ravi, 2013b). This process also allows the debuggers to begin their allotted work at the earliest stage.

3. PRIORITIZATION BASED ON CLUSTERING

3.1. The Motivation Process

The redundancy will make the pair wise comparisons much robust but it will be expensive and discourages it for applying it to prioritize the test case. Total comparisons that are required for comparing pair wise will be O (n2) number of comparisons. The humans can able to make a maximum of 100 comparisons above this level significant growth of inconsistency will reduce the effectiveness.

If there are less than 100 comparisons then those test suites may contain no more than 15 test cases. The scalability issue will be challenging task in the scenario of the real world. For instance if there are 1000 number of test cases that has to be prioritized then the total pair wise comparisons required will be 499,640. But for this human tester can provide the reliable responses to this huge comparison.

This approach uses k-means prioritization based on the cluster will reduce the total number of the comparisons and will be much effective. Here clusters of the test cases will be prioritized by using the techniques like prioritization based on clustering.

3.2. Criteria Based on K-Means

By using the two methods in clustering like data that are arranged as an individual group or in a group of hierarchy. In the group cluster analysis the data objects are grouped into clusters so that the objects that belongs to same cluster will be similar, whereas objects that belong to some different clusters will be dissimilar. These clustering techniques are categorized as partitional and hierarchical modes.

3.3. Partitional Mode

Given the set of database of objects the algorithm for partitional clustering will construct the partition of data and each cluster will optimize the clustering criteria so that minimization of the squared distance sum from mean that are within the clusters. The partitional clustering will be more complex since it can enumerates all possible groupings, it tries for finding the global optimum. The partitions number will be huge even for a tiny number of objects. For this problem the common solution will start at the initial partition, proceed, random with its own refinement.

If the partitional algorithm is well practiced will run for different set in the initial points and will investigate whether each solution will lead to those similar final partition. These algorithms will try for improving some certain criterion. First the similarity values are computed and the results are ordered and selects the one which optimizes these criteria. So the majority will be considered like greedy algorithms.

3.4. Hierarchial Mode

These algorithms will create the hierarchal decompositions with those objects, which will be either bottom-up (anglomarative) or top-down (divisive). The agglomerative algorithm that starts with treating each object as a separate cluster and the groups are merged successfully in accordance with distance measure. This clustering will stop when all the objects are gathered in a single group.

This method follows the greedy bottom-up like merging whereas the diverse algorithm start with the single group of the entire object and the groups are split into smaller groups, until each of the objects that falls into any one of the clusters. This approach will divide each of the data objects into disjoint groups in each step and it follows the similar pattern until the entire objects fall to a unique cluster. It will be similar to the divide-and-conquer algorithm.

3.5. The Clustering k-Means Method

This method will produce clusters that are from the set of the objects based on the objective function squared-error:

$$E = \sum_{i=1}^{k} \sum p \in C_i \left| p - m_i \right|^2$$

c_i	-	Clusters
p	-	Point of the cluster
m_i	-	Mean of the cluster ci

The mean of the cluster is represented as a vector for each of the attribute, mean values for the data in the cluster, the input parameter will be the total number of the clusters k. As the output of the algorithm will return the means or centers for each cluster c_i. The distance is usually measured in Euclidean distance. Proximity index and the optimization criteria have no restrictions which can be represented according to user preference or the application.

Algorithm

Step 1: The initial centres are selected as k objects
Step 2: Assign the data objects in the centre
Step 3: The centre of each cluster are recalculated
Step 4: Repeat the steps 2,3, unless the data object distribution in the clusters is not changed

The algorithm will be relatively scalable.

4. THE EXPERIMENT

We have to analyze whether the clustering technique can facilitate the test case prioritization for the test suites. Whether the prioritization of the test case improve rate of fault detected from those test suites.

4.1. Prioritization Measure Based on Clustering and Efficacy

Once we apply the prioritization technique based on clustering to the problem of quadratic equation. The co-efficients are read and the roots are determined.

Procedure:

```
1.    Initialize the co-efficients
2.    Calculate the roots
3.    If roots less than zero
4.    Imaginary roots
5.    If roots equals zero
6.    Calculate Root1
      Assign root 2 equals root1
7.    If root greater than zero
8.    Calculate Root1, Root2
9.    End
```

4.2. The Cyclomatic Complexity Measure

Flow graph is drawn as in **Fig. 1** for the procedure to find roots. The flow graph is used to evaluate the cyclomatic complexity.

Cyclomatic complexity can be found in three ways:

$$V(G) = P+1$$

Cyclomatic complexity which equals the predicate nodes plus one, where P represents the number of predicate nodes:

$$V(G) = 3+1 = 4$$

Predicate nodes are the nodes that contain the conditions whereas in the flow graph there are three predicate nodes A, B, D.

Cyclomatic Complexity can be calculated by the number of regions. Regions are the area that is surrounded by the nodes and the edges:

$$V(G) = \text{No of regions} + 1$$
$$V(G) = 3+1 = 4$$

Cyclomatic Complexity can be calculated as the number of edges minus the number of nodes plus 2:

$$V(G) = n-e+2$$
$$V(G) = 11-9+2 = 4$$

Hence the Cyclomatic complexity of the problem is determined as 4.

Independent paths for the given problem is:

P1: 1 2 3 8
P2:1 2 4 5 8
P3: 1 2 4 6 7 8
P4: 1 2 4 6 1

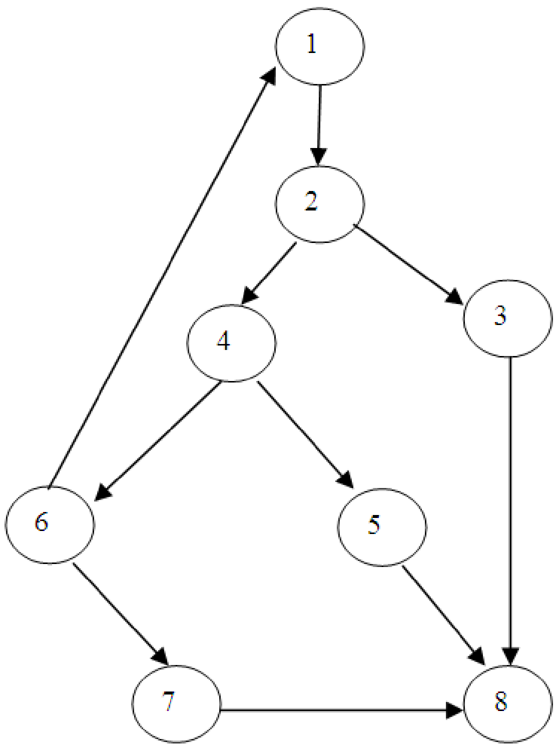

Fig. 1. The flow graph for quadratic equation problem

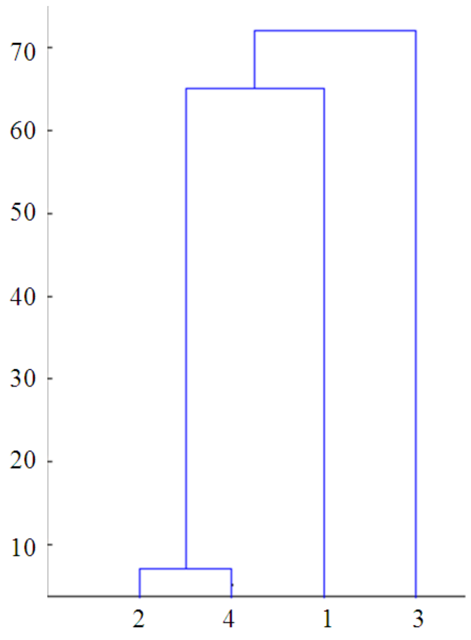

Fig. 2. Dendrogram for the test case

5. RESULTS

By applying the k-means clustering approach for the quadratic problem, we use the independent paths. There are four independent paths in this path based testing. Two clusters are initially used as k-value, we calculate the two clusters that have the combinations:

C1: P1, P2, P3
C2:P3

Clusters are prioritized in accordance with dendrogram method. The order in which the test cases will be executed is as follows, P2, P4, P1, P3, the path2 will get the highest priority and the sequence will be followed.

We use an APFD method for calculating the effective for calculating the effectiveness of this method by applying the formula:

APFD = 1-((TF1 + TF2+ TF3+……..+TFM)/nm) + 1/2n

When the dendrogram method that are obtained without the clustering method for the prioritization the value of APFD will be 0.5 but when it is applied for prioritization, value will be 0.625 as in **Fig. 2**.

6. CONCLUSION

The prioritization of the test case will schedule the test case in a specific order which increases the effectiveness to meet the performance goals. The APFD will increase the fault detection rate in the testing life cycle and which improves the software quality. We have to use many techniques that address this issue. As we know developing the test suite will be an expensive process and the test suite cannot be run entirely since it consumes huge time and resources. These issues are well addressed successfully in this method.

7. REFERENCES

Arafeen, M.J. and H. Do, 2013. Test case prioritization using requirements-based clustering. Proceedings of the IEEE 6th International Conference on Software Testing, Verification and Validation, (ICST) Mar. 18-22, IEEE Xplore Press, Luembourg, pp: 312-321. DOI: 10.1109/ICST.2013.12

Di Nardo, D., N. Alshahwan, L. Briand and Y. Labiche, 2013. Coverage-based test case prioritisation: An industrial case study. Proceedings of the IEEE 6th International Conference on Software Testing, Verification and Validation (ICST), Mar. 18-22, IEEE Xplore Press, Luembourg, pp: 302-311. DOI: 10.1109/ICST.2013.27

El-Koka, A., K.H. Cha and D.K. Kang, 2013. Regularization parameter tuning optimization approach in logistic regression. Proceedings of the 15th International Conference on Advanced Communication Technology (ICACT), Jan. 27-30, IEEE Xplore Press, PyeongChang, pp: 13-18.

Jacob, T.P. and T. Ravi, 2013a. Regression Testing: Tabu search technique for code coverage. Indian J. Comput. Sci. Eng., 4: 208-215.

Jacob, T.P. and T. Ravi, 2013b. Detecting of software source code defects using test case prioritization rules. Proceedings of the 2nd International Conference on Latest Computational Technologies, Jun. 17-18, London, UK., pp: 29-32.

Karnavel, K. and J. Santhoshkumar, 2013. Automated software testing for application maintenance by using Bee Colony Optimization algorithms (BCO). Proceedings of the International Conference on Information Communication and Embedded Systems (ICICES), Feb. 21-22, IEEE Xplore Press, Chennai, pp: 327-330. DOI: 10.1109/ICICES.2013.6508211

Polo, M., P. Reales, M. Piattini and C. Ebert, 2013. Test automation. IEEE Software, 30: 84-89. DOI: 10.1109/MS.2013.15.

Shaccour, E., F. Zaraket and W. Masri, 2013. Coverage specification for test case intent preservation in regression suites. Proceedings of the IEEE 6th International Conference on Software Testing, Verification and Validation Workshops, (ICSTW), Mar. 18-22, IEEE Xplore Press, Luxembourg, pp: 392-395. DOI: 10.1109/ICSTW.2013.50