IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661, p- ISSN: 2278-8727Volume 16, Issue 1, Ver. VII (Feb. 2014), PP 75-80
www.iosrjournals.org

Cosine Similarity Based Clustering For Software Testing Using Prioritization

[1]R.Kanimozhi, [2]Prof.J.RBalakrishnan
PG student, Director & professor
Department of computer science & engineering Anand Institute of Higher Technology Kazhipattur, chennai

***Abstract**: prioritizing the test cases helps to increase the rate of fault detection. The difficulty of ensuring the dependability in the growth of the scale of software and software testing in distributed environment a sufficient software testing technique is not possible. It is often difficult to test a parallel and distributed system in the real world deployment. Hence in this paper, after generating test cases using functional requirements, dependency structure prioritization technique is used to prioritize the test cases based on the functional structure of dependency. A new technique namely cosine similarity based clustering approach is used to group the test cases based on the similarity values to form clusters. Each cluster is distributed in the distributed environment for parallel execution in order to reduce the computation time and to improve the rate of fault detection.*

***Index Terms:** Cosine similarity approach, dependency structure prioritization, software testing.*

I. Introduction

Software engineering is the systematic design and development of software products and maintenance of software. Software testing plays a major role in software engineering. Software testing is the process of validating and verifying the product to meet the requirements stated during the design and development. Software bugs will almost always exist in any software module with moderate size: not because programmers are careless or irresponsible, because the complexity of software is generally intractable and humans have only limited ability to manage complexity. It is also true that for any complex systems, design defects can never be completely ruled out.

Regression testing is process of retesting the modified software and ensures that new error does not introduce into the previously tested source code due to these modifications. regression testing is very expensive testing process .in order to decrease the cost of regression testing the software tester may prioritize the test case so that the test case which are more important are run earlier during regression testing process. In this context, prioritization techniques can take advantage of information collected about the previous execution of test cases to obtain test case orderings. For regression testing the clustering technique is used for test case prioritization. in this the test case having common properties and similar fault detection ability are group together within same group. Test case prioritization improves the cost effectiveness of regression testing. The technique is developed in order to run test cases of higher priority in order to minimize time, cost and effort during software testing process.

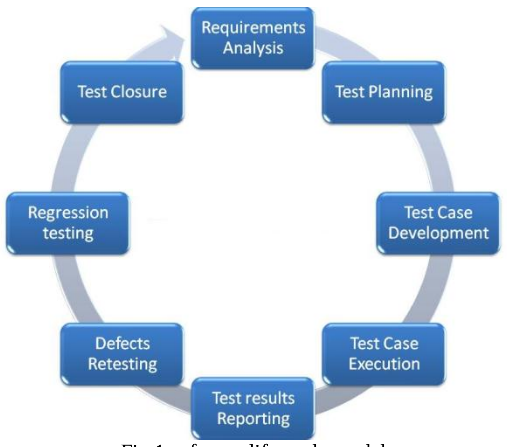

Fig 1 software life cycle model

Software testing involves test suites and test cases for finding fault where test suite is a collection of test cases that are grouped for test execution purposes. In order to generate test cases, find the sequence of interaction between the software testing scenarios for the software modules. Test case prioritization technique involves scheduling the test cases in an order that improves the performance so that the test cases with higher priority executed first which allow increasing the rate of fault detection.

Dependency structure prioritization technique is based on the functional dependencies between the test cases. Functional dependencies are the interactions and relationships between the system functionality which determines the run sequence. A Dependency Structure prioritization technique reduces the cost of testing and fault detection capabilities.

This prioritization technique classified into two types namely open and closed dependency structure. This technique is based on four algorithms namely DSP_volume, DSP_height, Weighted_DFS and Weighted_DFS_Visit.

Clustering is the method of analyzing and organizing data with similar characteristics are grouped together. Clustering is used in various fields including Data Mining, software engineering and Machine Learning. Much research has been done in the past to efficiently cluster data. Various algorithms and methods have been proposed for clustering. Hierarchical clustering, partitional clustering, nearest neighbor clustering, fuzzy clustering and defect clustering are some popular techniques used for clustering.

Defect Clustering in Software testing means that the majority of the defects are caused by a small number of modules, i.e. the distribution of defects are not across the application but rather centralized in limited sections of the application. A new technique namely cosine similarity based clustering is applied to cluster the test cases that are prioritized by using dependency structure. Cosine similarity is a vector based similarity measure between two vectors derived by using the Euclidean dot product formula.

Each clusters executed in distributed environment for parallel execution of test case clusters to measure the average fault rate. The results in the distributed environment are integrated to produce a single detection rate.

II. Literature Survey

1. A clustering approach to improving test case prioritization: an industrial case study – Ryan Carlson (2011) says that they have limited their attention to a financial subsystem of Dynamics Ax product since it is a case study. It affects the choice of the number of clusters and order of clusters because they are based on the number of feature areas in Dynamics; different cluster order will affect the results. The choice of use of lines of code at the class level was based on the availability of information extracted from project repository this will affect their results.
2. Clustering based on cosine similarity measure – satyasree (2012) in this paper, cosine similarity is applied in data mining concept. Selecting different dimensional space and frequency levels leads to different accuracy rate in the clustering results although it is proved more accurate than traditional methods but still accuracy must be measured.
3. Using dependency structure for prioritization of functional test suites- Shifa-e-Zehra Haidry and Tim Miller (2013) in this paper, they proposed a new test case prioritization technique that uses the dependency information from the test suites to prioritize. Dependency structure prioritization technique includes four algorithms for prioritizing. The open dependency proves to have lower execution cost and closed dependency achieved better fault rate detection than the traditional methods. Average rate of fault detection is used to calculate the percentage of fault rate but clustering approach is not considered to improve the fault rate further.
4. Enhanced distributed document clustering algorithm using different similarity measures - Neethi Narayanan, J.E.Judith, Dr.J.Jayakumari (2013) in this paper, a distributed environment is considered in which all peers form a ring structure and the information are stored in DHT. A local model is formed using EDK-means using similarity algorithm. All local models aggregated to form a global model using EPCP2P this improves clustering quality and accuracy. Even though Jaccard and Pearson coefficients show better results than cosine similarity in data mining but this is not possible in case of software testing.
5. Comparison of Jaccard, Dice, Cosine similarity coefficient to find best fitness value for web retrieved document using genetic algorithm - Vikas Thada, Dr Vivek Jaglan (2013) in this paper, they made a comparative analysis for finding out the most relevant document for the given set of keyword by using three similarity coefficients namely cosine, dice and jaccard this was performed by using genetic algorithm approach. They selected first 10 paged out of the google search result for their experiment based on that the best fitness value were obtained using the cosine similarity coefficients than the dice and jaccard.
6. prioritize code for testing to improve code coverage of complex software - J.Jenny Li (2005) Code prioritization for testing promises to achieve the maximum testing coverage with the least cost. This is an innovative method to provide hint on which part of code should be tested first. It helps to provide better code average. It includes two parts: extending the traditional dominator analysis method to include global impact of function/method calls, and relaxation of the “guaranteed” rule to “at least” to make the dominator analysis simpler. They had implemented both the original dominator analysis method and this relaxed method with global view, for calculating code priority for testing. The two calculations had been applied to

4 actual industrial products. The experimental results show that the new calculation is consistently better than the original ones in identification of code to improve code coverage.

7. Combinatorial Interaction Regression Testing: A Study of Test Case Generation and Prioritization. - Xiao Qu, Myra B. Cohen, Katherine M. Woolf (2007) They have conducted an empirical study on two software subjects, each with multiple successive versions. The first method uses branch coverage from the prior version. The second method is to use the interaction weighting method, but rather than re-generate they simply use it to order the given tests based on their weights. They first examined the effectiveness of CIT test suites compared with an exhaustive strategy. They applied prioritization and regen/ prio and compared their effectiveness on CIT test suites. They examined several different ways to control the prioritization. They used methods that leverage code coverage from prior releases, as well as one that is specification based. There results shows that the CIT test suites may be an effective way to reduce the test space and that prioritization improves the ability to detect faults early in certain subjects.
8. Search Algorithms for Regression Test Case Prioritization.- Zheng Li, Mark Harman, and Robert M. Hierons(2007) This paper focuses on test case prioritization techniques for code coverage, including block coverage, decision (branch) coverage, and statement coverage. This paper presents results from an empirical study that compared the performance of the five search algorithms applied to six programs, ranging from 374 to 11,148 lines of code. In this paper addresses the problems of choice of fitness metric, characterization of landscape modality, and determination of the most suitable search technique to apply. five search techniques are studied: two Meta heuristic search techniques (Hill Climbing and Genetic Algorithms), together with three greedy algorithms (Greedy, Additional Greedy, and 2-Optimal Greedy). The results of the empirical study show that the Additional Greedy, 2-Optimal, and Genetic Algorithms always outperform the Greedy Algorithm.
9. Test Case Prioritization: A Family of Empirical Studies. - Sebastian Elbaum, Alexey G. Malishevsky, Gregg Rothermel(2002) General prioritization that tries to pick out a action at law order that may be effective on the average over a succession of sequent versions of the computer code. In regression testing, they have a tendency to square measure involved with a specific version of the computer code and that they might need to place take a look at cases in an exceedingly manner that may be handiest for that version. A coarser granularity—for example, at the operate level, wherever instrumentation and analysis square measure a lot of economical. They expect, however, that coarse roughness techniques are less effective than fine roughness techniques and loss of effectiveness might offset potency gains. Revealed a large performance gap between the results achieved by the prioritization techniques but they have a tendency to examine and also the best results accomplishable.
10. Adaptive Random Test Case Prioritization.- Bo Jiang, Zhenyu Zhang, W. K. Chan, T. H. Tse(2009) They proposed a set of ART prioritization techniques guided by white-box coverage information. They also conducted an empirical study to evaluate their effectiveness. Rather than integrating with techniques with the class of greedy algorithms, they choose to study them in a standalone fashion so the observations drawn from the study will be independent of the latter techniques. The main contribution of this paper is twofold: (i) it proposes the first set of coverage-based ART techniques for test case prioritization. (ii) It reports the first empirical study on ART-based prioritization techniques. The generate procedure constructs a set of not-yet-selected test cases iteratively, by randomly adding remaining test cases into the candidate set as long as they can increase program coverage and the candidate set is not yet full. The empirical results show that their techniques are significantly more effective than random ordering. Moreover, the ART-br-max min prioritization technique is a good candidate for practical use because it can be as efficient and statistically as effective as traditional coverage-based prioritization techniques in revealing failures. Both genetic algorithms and their ART techniques are effective in avoiding local maximums commonly faced by greedy algorithms.

III. Conclusion Of Literature Survey

The above studies include various test case prioritization techniques which prioritize the test cases but they are not implemented in distributed environment. The ordering of test cases in the existing system vary in many ways but perfect ordering is not available. The rate of fault detection is less improved in the previous techniques but still higher rate of fault detection is required.

IV. Motivation Of Research

The motivation of our research is to provide an efficient test case prioritization technique in distributed environment.

- How to increase the rate of fault detection?
- How to implement test case prioritization technique in distributed environment?
- How to cluster the test cases based on the prioritization?

- How to avoid executing redundant test cases?
- How to reduce the rate of execution time?

V. Proposed Work

Our proposed system first generate the test case for the module testing of an application, then predict the Open and closed dependent structure of the test cases independently. Dependency structure prioritization technique is used to prioritize the test case by using four algorithm namely 1) DSP volume measures gives higher weight to test cases that have more dependents. 2) DSP height measures gives higher weight to those test cases that have deepest dependents.3) weighted_DFS is used for test case prioritization based on dependency strucutres. 4) weighted_DFS_visit is used for finding the visiting vertices in the dependency structure. Compute the similarity between the test cases by using the cosine similarity measurement.

The similarity based clustering is used for grouping the test cases in order to make testing effective. A distributed environment is generated to test the test cases. Distributed environment is nothing but the cloud environment. Distribute the test case groups in the distributed environment. Perform the testing process and generate the test results. After collecting the test results apply the integration method to reduce the results from distribute environment. This proposed technique helps to achieve less computing time for testing and also reduce the failure ratio for testing software.

In order to generate the test cases provide the necessary application for testing. Find the sequences of interactions between the software testing scenarios to generate the test cases for the software modules. Add the number of test cases to the test case list and assign ID for each test case. Additional test cases are added using the remote server to generate distinct ID.

Predict the functional dependency between the test cases to form a dependency prediction matrix. It is an n*n matrix which contains n number of test cases, the matrix value will be the cross number of dependencies between those two test cases. Compute the height and weight of the test cases which is used to form a DSP volume and DSP height matrix.

The prioritization is set based on the prediction done earlier; higher priority is set to those which have low height and high weight value. Vector matrix is formed based on the priority values to find similarity between the test cases. Cosine similarity matrix includes the vector matrix values to form the clusters based on similarity.

Cosine similarity approach is used to form the clusters. The clusters are used to implement in the distributed system for providing better performance in fault detection. The number of clusters must be less than the number of distributed environment in order to form the group of the clusters. This implementation reduced the execution time.

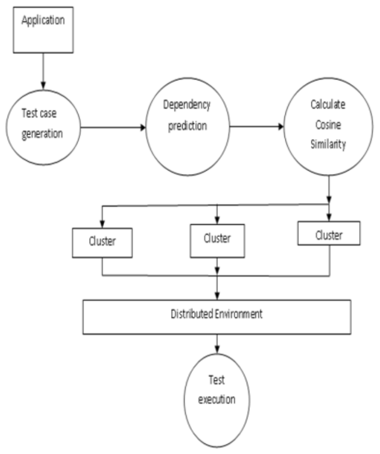

Fig 2 overall system design

Send the clustered test cases to the distributed environment for execution. Execute the test cases and evaluate the results to find the percentage of fault detection by average rate of fault detection. Compare the values with the existing system to prove that the proposed system has higher accuracy in fault detection.

COSINE SIMILARITY ALGORITHM:

Cosine similarity is a measure of similarity between two vectors of an inner product space that measures the cosine of the angle between them. The cosine of 0° is 1, and it is less than 1 for any other angle. It is thus a judgment of orientation and not magnitude: two vectors with the same orientation have a Cosine similarity of 1, two vectors at 90° have a similarity of 0, and two vectors diametrically opposed have a similarity of -1, independent of their magnitude. Cosine similarity is particularly used in positive space, where the outcome is neatly bounded in [0, 1].

$$\text{similarity} = \cos(\theta) = \frac{A.B}{||A||\,||B||} = \frac{\sum_{i=1}^{n} A_i * B_i}{\sqrt{\sum_{i=1}^{n}(A_i)2} * \sqrt{\sum_{i=1}^{n}(B_i)2}}$$

INPUT: an n*n Boolean adjacency matrix representing test case list.
OUPTUT: an n*n Boolean adjacency matrix representing the similarity matrix.

```
Begin.
Step 1: collect the unique test case identity.
Step 2: collect the properties of the test cases.
Step 3: generate vector array.
Step 4: compute similarity matrix.
End.
```

Pseudocode for cosine similarity algorithm:

```
Input: IP_List, PropertiesList
Output : Similarity matrix

N= IP_List.size();
For I = 1 ; I = 1,2,3…N do
Ip1 = IP_List.get(I);
Properties pros = Ip1.getProperties();
 Vector V1 ;
For J = 1 ; j < PropertiesList.size(); J++ do
If(Properties.contains(PropertiesList.get(J)) then
V1 [i][j] = 1
Else
V1[i][j] = 0 ;
End If
End For
End For
For I = 1; I = 1, 2, 3…N-1 do
For J= 1; J = 1, 2, 3…N-1 do
Similarity[I][J]=(v1[I]*v1[J])/((mod(V1[I]))* (mod(V2[J])))
End For
End For
Return Similarity;
```

V. Conclusion

Thus the proposed technique for prioritization of test suites that contain dependencies between test cases has been defined. Dependency structure prediction and prioritization provides effective prioritization of test cases. This cosine similarity based clustering approach helps to increase the rate of fault detection in distributed environment.

References

[1] Ryan Carlson, Hyunsook Do, Anne Denton (2011) ‘a Clustering Approach to Improving Test Case prioritization: An Industrial Case Study,’ 27th IEEE International Conference on Software Maintenance.

[2] K.P.N.V.Satya sree , Dr.J V R Murthy(2012) ‘ clustering based on cosine similarity measure,’ international journal of engineering science and advanced technology.

[3] Shifa-e-Zehra Haidry and Tim Miller (2013) ’Using Dependency Structures for Prioritization of Functional Test suites,’ IEEE transaction on software engineering.

[4] Neethi Narayanan, J.E.Judith, Dr.J.Jayakumari (2013) ‘Enhanced Distributed Document Clustering Algorithm Using Different Similarity Measures,’ IEEE Conference on Information and Communication Technologies.

[5] Vikas Thada, Dr Vivek Jaglan (2013)‘Comparison of Jaccard, Dice, Cosine Similarity Coefficient To Find Best Fitness Value for Web Retrieved Documents Using Genetic Algorithm,’ International Journal of Innovations in Engineering and Technology.

[6] J.Jenny Li (2005)’Prioritize Code for Testing to Improve Code Coverage of Complex Software,’ IEEE transaction on software reliability engineering.

[7] Xiao Qu, Myra B. Cohen, Katherine M. Woolf (2007) ‘Combinatorial Interaction Regression Testing: A Study of Test Case Generation and Prioritization,’ IEEE international conference on software maintenance.

[8] Zheng Li, Mark Harman, and Robert M. Hierons(2007) ‘Search Algorithms for Regression Test Case prioritization,’ IEEE transactions on software engineering.

[9] Sebastian Elbaum, Alexey G. Malishevsky, Gregg Rothermel(2002) ‘Test Case Prioritization:A Family of Empirical Studies,’ IEEE transactions on software engineering.

[10] Bo Jiang, Zhenyu Zhang, W. K. Chan, T. H. Tse(2009) 'Adaptive Random Test Case Prioritization,' IEEE international conference on automated software engineering.

[11] X. Qu, M.B. Cohen, and K.M. Woolf (2007), 'Combinatorial Interaction Regression Testing: A Study of Test Case Generation and Prioritization', Proceeding. IEEE Int'l Conf. Software Maintenance, pp. 255-264.

[12] Z. Ma and J. Zhao (2008), 'Test Case Prioritization Based on Analysis of Program Structure', Proceeding 15th Asia-Pacific Software Eng. Conf, pp. 471-478.