International Journal of Scientific Engineering and Research (IJSER)
www.ijser.in
ISSN (Online): 2347-3878
Volume 2 Issue 2, February 2014

Approaches Used for Prioritization of Test Suites

Gurdiksha[1], Janpreet Singh[2]

[1]M. Tech Student, Lovely Professional University, School of Computer Science and Engineering,
Jalandhar-Delhi G.T Road,National Highway 1, Phagwara 144411, India

[2]Assistant Professor and COD, School of Computer Science and Engineering, Lovely Professional University,
Jalandhar-Delhi G.T Road,National Highway 1, Phagwara 144411, India

Abstract: ***Regression testing certifies that alteration or augmentation performed is not going to impact the original functionality. To make this sure test case must rerun from the present set of test suites, although it is not possible to conclude that how much retesting is required. Various regression test selection techniques have been developed for making regression testing more effective and efficient. As test suites becomes large in size when the software progresses also tend to increase the cost and effort of re-executing the entire test suite randomly. Test Suite Prioritization approach has been studied to exploit the test suites that have been built. According to it, test suites are run in a particular order to maximise the fault detection.***

Keywords: Regression Testing, Test case, Test Suite, Test Suite Prioritization, Average Percentage of faults detected.

1. Introduction

Software testing is a procedure of testing or comparing the actual outcome with the expected outcome. Testing of the software is done in order to check the correct functionality of the system or project. If the testing will not be performed then system may lead to catastrophic or improper results in the field. So it’s better to test the system earlier, so that the excellent results can be produced. Also it ensures quality of the system. As testing is performed in order to detect faults but we can’t make guarantee that 100% faults will be detected and removed.

a) Software Testing includes:
- Verification: Under verification, we check whether we are going to make right product means we ensure before the actual implementation of the software or project. Basically it is performed phase wise.
- Validation: Under validation, we check whether we made the right product means we ensure after the actual implementation of software or project. Basically it is performed on end product.

b) Software testing can be performed using two models:
- White box testing model: This type of testing is based on the internal working of the system that how the actual flow of programs is going on. One must have good programming skills in order to test with the white box testing model. Logical paths, loops, conditions, branches are mainly tested in it.
- Black box testing model: In this type of model no internal working of the system is needed to be known, also no programming skills are required. Input is given to the system and output is manipulated that whether it is wrong or right. Only the external behaviour is observed, without any knowledge of internal functioning.

Now there exists various types of testing but our focus is on Regression Testing as we are going to prioritize test cases during regression testing. Regression Testing is performed when any module, unit, component or system is modified in order to change the functionality or for some other reasons. It ensures that any modification done in specified components will not lead to any discrepancy in the actual output of the system. It also ensures that no other modules are being affected due to performed modifications. It maintains the quality of the system. Regression testing basically provides the assurance that modification or enhancement made will not infect the existing functionality or system. It may be a more complex and challenging process if retest-all approach is going to be used. In this approach all the test cases of all test suites must be rerun. Though it may become very expensive to execute all the test cases, various techniques have been studied to assist regression testing procedure that may include majorly test suite minimisation, Test case selection and Test suite Prioritization.

This paper surveys the work carried out in the field of test suite prioritization.

2. Test Case Prioritization

Prioritization of test cases is the process of arranging the test cases in a logical manner and then t run them according to their logical order. It is useful because it helps in better utilization of expensive resources like disk space. It is considered when we need test cases having higher priority to be run earlier than the other test cases present in the test suite. Also with this approach defects can be found earlier. The problem of resource lacking can be resolved as resource can be better utilized.

Priorities are not assigned randomly but got assigned according to some principle. And then test cases will run according to their own order of execution. If faults can be detected former, then release of the product will be before on time as well. And we can detect the faults former by prioritizing test cases.

Need for prioritization of test cases: whenever we have inadequate number of resources then it may happen that the test cases that are not so much required to get run would consume the resources. That will results in lack of resources for the test cases which are urgently required to run. So to avoid this kind of condition to occur, we can prioritize the test cases. With the help of which we can give high priority to the test cases which are urgently required to run and lower priority to those test cases which are less needed. Test case prioritization will also decrease the time consumption and will lead to early detection of faults.

3. Approaches of Prioritization

Approaches used for prioritization describes various ways on the basis of which prioritization can take place. Different ways can be used to prioritise the test cases and they can be described as:

3.1 Prioritization Using Dependency Structure

Definition 3.1 (Test Case Prioritization): Test case prioritization is the process of scheduling test cases to be executed in a particular order so that test cases with a higher priority are executed earlier in the test sequence. The priority is defined relative to some test criterion; for example the number of code statements covered [3].

Test Case Prioritization is considered when we need test cases having higher priority to be run earlier than the other test cases present. The First approach that can be used is discovering the "functional dependencies". As Scenarios that will be having more dependencies will produce more faults as well. Functional dependency exists when some interactions can't take place until or unless some other interactions occur first. We can say interaction I1 is dependent on interaction I2 if and only if interaction I1 must take place before I2 anywhere in the program. . Functional dependencies in the software are present in the requirement phase. These dependencies exactly can be inherited in the test cases. For example, if a requirement R1 is dependent on requirement R2 then the test cases developed for R1 must be dependent on test cases developed for R2.

In following first figure, the test order is t1-t2-t3-t4. 20 percent faults have been detected after t1 has executed. And in second figure, the test order is t4-t1-t2-t3. 60 percent faults have been detected after the first test case has run which is t4. Clearly the execution sequence has greater impact on the percent of fault detection.

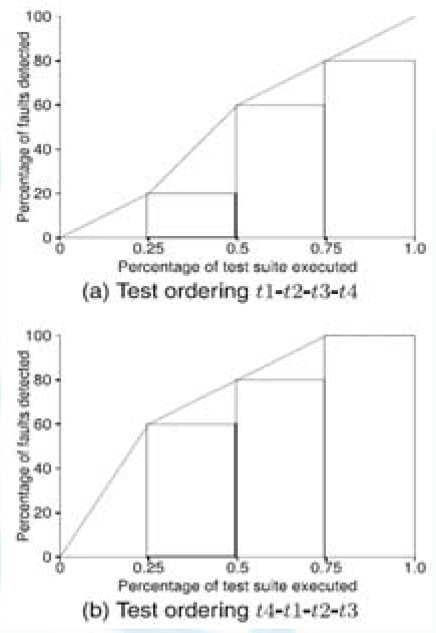
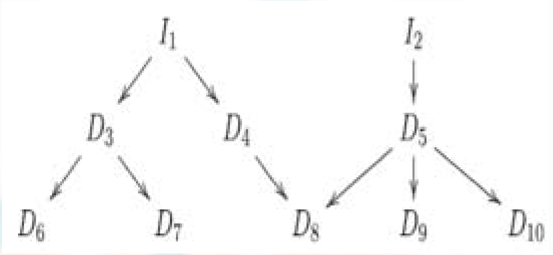


Figure 1: Test ordering [3]

Techniques can be:

- Open dependency structure: It can be described as one test case should be executed prior to other test case anywhere in the program.

- Closed dependency structure: It can be described as one test case should be executed just before the other test case means the other test case must follow the first test case.
- Structures of dependencies (TD): we can say there exist dependencies between scenarios when one must take place before the other.



$$\begin{tikzcd}
 & I_1 \arrow[dl] \arrow[dr] & & & I_2 \arrow[d] & \\
D_3 \arrow[d] \arrow[dr] & & D_4 \arrow[dr] & & D_5 \arrow[dl] \arrow[d] \arrow[dr] & \\
D_6 & D_7 & & D_8 & D_9 & D_{10}
\end{tikzcd}$$

Figure 2: Dependency structure [3]

Or we can say the processing of one test case must be halted till the test case on which it is dependent will get executed. Only after that test case will take place its execution will be started.

- Independent test cases: These test cases can be executed without taking into the concerns of another test case's completion into account. Means they can be executed freely as no any test case needed to be finish before them.
- Dependent test cases: These test cases can't be executed without taking into the account another test case's execution as they are dependent on it. Means they can't execute freely because one or more test cases needed to be finish before them.
- Algorithms used for open dependency structure: DSP volume: Higher weight will be assigned to the test cases whosoever will be having greater number of dependents.

It can be calculated by finding all of the direct and indirect dependents.

- DSP height: Higher weight will be assigned to the test cases whosoever will be having dependents in depth. Altitude of all paths needs to be measure in order to calculate it. Longest path will be having the greatest weight.

3.2 Model Based Prioritization

In Model based prioritization of test cases, test cases are generated from the UML diagrams (activity diagram, sequence diagram, component diagram, and state chart diagram) and then prioritization takes place based on some model information. Traditional test case prioritization techniques consider the knowledge about how system has been previously used, like fault proneness of different modules of code in a program, capability of detecting faults, information regarding control flow and data flow. Almost all of these techniques are code based and used for the post implementation phase of development of software. Therefore model based technique is required for pre-implementation testing. Terms need to understand for model based testing:

Definition 3.2(Activity Graph): *Activity graph G=(V,E) is a directed graph where each node in V represents one action in the activity model and each edge in E represents a control flow from one node to other [1].*

Definition 3.2.2(Activity Flow): *Activity flow (or flow) in an activity graph is defined as sequence edges starting from the root (start) node of the activity graph to the target node [1].*

Test case generation process:

Step 1: Create activity diagram: activity diagram can depict the behaviour of the system. It can be used to recognize the work flow performed by an object or component. Also interaction between different use cases can be visualised. The main vigour of activity diagram is that, it can represent concurrent activities. Concurrency can be shown using:

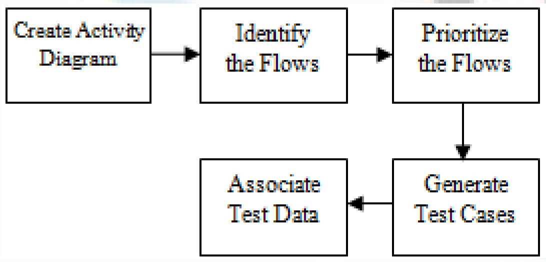


Figure 3: Generation of test cases [1]

- Fork: It has multiple outgoing transitions and one incoming transition.
- Join: It has one outgoing transition and multiple incoming transitions.
- Rendezvous: It has multiple outgoing transitions and multiple incoming transitions.

Step 2: Identify the flows: All the flows in the activity diagram are recognized once the activity diagram is created. Flows in the activity diagram represents major and substitute scenarios for that use case. The system should cover as many as flows it can.

Step 3: Prioritise the flows: At this point, prioritization of flows takes place based on coverage of all transitions in the activity diagram.

Step 4: Generate test cases: Test cases get generated from the prioritised flows using manual or automated approach.

Step 5: Associate test data: Association of test data with the appropriate test cases is done in order to guarantee that accurate activity flow is executed for a test case.

3.3 Requirement-Based Prioritization

In requirement based test case prioritization test cases gets generated from the requirement specifications produced by the customer which lead to the earlier detection of faults and user satisfaction with quality software. There are three factors that need to be considered while doing prioritization based on the requirements.

1. Customer assigned priority of requirements: It defines the significance of some requirements to the customer. Customer can assign any priority value from 1-10 to all the requirements. 1 represents the lowest priority and 10 represent the highest priority. Customer can assign any number to the requirement based on the importance of that requirement to the customer. Motivation to include this factor: As the customer satisfaction is very important in any development, so to make customer satisfy, the requirements of higher priority must be tested first.
2. Implementation Complexity: It represents the difficulty faced by the developer in order to develop or implement the system. Each and every requirement can be scrutinized and provided a value from 0-10. Developer can assign a value based on difficulty of development.
   Motivation to include this factor: Requirements having more complex tend to be producing more faults.
3. Requirement changes: It shows how many number of times a requirement has been changed during the development time. Again the value from 1-10 can be assigned by the developer itself. It can be calculated as:
   $RC_i = N/M*10$
   Where, RC is Requirement Change [5].
   N is number of changes for requirement i.
   M is highest number of changes for any requirement among all the project requirements.
   Motivation to include this factor: It is seen that roughly 50% of the faults introduced in the project are due to the errors exists in the requirement phase. And requirement changes can lead to these faults.

3.4 Prioritization Using Clustering

In this approach, clustering of test cases is done (based on some criteria) of the prepared test cases after prioritization is performed on those clusters. As an example we can consider that first of all, clustering is done using the agglomerative hierarchical method. According to which, pair-wise distance is calculated among two test cases. The closest the two test cases in terms of their code coverage similarity are merged into one cluster. Then the table of pair-wise similarity is recomputed.

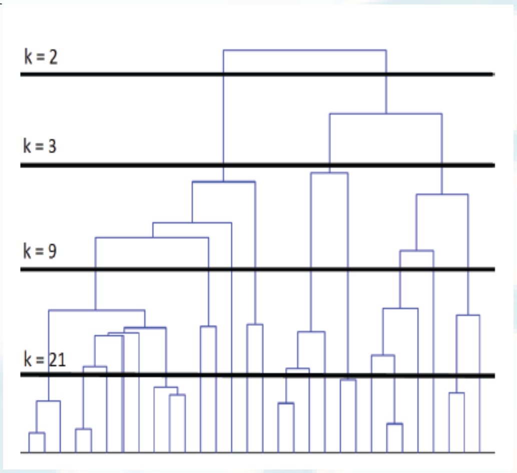


Figure 3.4.1: Hierarchical tree for clustering [6]

The vertical lines at a meticulous level represent the number of test clusters at that level. This approach gives flexible path to adjust size and number of clusters. After the preparation of clusters, First step is to perform prioritization within each cluster. And to perform prioritization, reorder of test cases is done. The second step is to generate complete prioritised list of test cases by selecting test cases from each cluster.

4. Conclusion and Proposed Work

Test case prioritization is procedure of prioritizing and scheduling test cases with the help of which test cases of higher priority run in order to minimise the testing effort, time and cost. The literature review shows that many researchers proposed many methods to achieve this. We can take some of them collectively in order to improve the prioritization process.

By concluding this, In future we are going to perform:

Step1: Generate test cases from models
Step2: Find the dependencies between generated test cases.
Step3: Based on those dependencies clustering will be performed.
Step4: And finally the prioritization will take place.

By performing these steps we can achieve:

- Elevation of rate of fault detection.
- Decreased time of testing process.
- Decreased pair-wise comparison.
- Increased performance of coupled modules.

Reference

[1] Gantait, A. (2011). Test Case Generation and Prioritization from UML Models. *IEEE*, pp. 345-350.
[2] Gregg Rothermel, R. H. (2001). Prioritizing Test Cases For Regression Testing. *IEEE vol. 27 no.10* , pp.929-948.
[3] Miller, S.-e.-Z. H. (2013). Using Dependency Structures for Prioritization of Functional Test Suites. *IEEE Transation on Software Engineering, Vol. 39, No. 2*, pp.258-275.
[4] Quart-ul-an-farooq, M. Z. (2010). A Model Based Regression Testing Approach For Evolving Software Systems With Flexible Tool Support. *IEEE*, pp. 41-49.
[5] R.Kavitha, V. K. (2010). Requirement Based Test Case Prioritization. *IEEE*, pp 826-829.
[6] Ryan Carlson, H. D. (n.d.) (2011). A clustering Approach to Improving Test Case Prioritization: An Industrial Case Study. *IEEE*, pp. 382-391.

Author Profile

Gurdiksha is pursuing M. Tech from Lovely Professional University, School of Computer Science and Engineering, Jalandhar-Delhi, India.

Janpreet Singh is working as Assistant Professor and COD, Lovely Professional University, School of Computer Science and Engineering, Jalandhar-Delhi, India.