Analysis of Model Based Regression Testing Approaches

SABAH TAMIMI
College of Computing, AlGhurair University,
Dubai, United Arab Emirates.
1.1.1 sabah@agu.ac.ae

MUHAMMAD ZAHOOR
College of Computing, AlGhurair University,
Dubai, United Arab Emirates.
1.1.2 zahoor@agu.ac.ae

ABSTRACT
One of the important phases in the life cycle of software development process is the designing phase. There are different models used in this particular phase including class diagrams, state diagrams and use cases etc. To test the conformance of the software it is very essential that test cases should be derived from these specific models. Similarly regressions testing through these models are very significant for testing of modified software. There are several regression testing approaches based on these model in literature. This survey report is the analysis of the model based regression testing techniques according to the parameter identified during this study. The summary as well as the analysis of the approaches is discussed in this survey report. In the end we concluded the survey by identifying the areas of further research in the field of model based regression testing.

Keywords
Regression testing, Model based regression testing, UML regression testing. Testing evaluation parameters.

1. INTRODUCTION
To inject quality into the software, testing is important activity in the software development process. Testing can be done in several ways, model based, code based and specification based etc. Many model based testing approaches are developed for the purpose of quality improvements of the software. In model based testing [1], the test cases are derived from a certain model (class diagrams, state diagrams, activity and use cases etc).

Since a model of software represents its requirements, the derivation of test cases from that model is to determine the quality of the software towards its conformance with requirements. Regression testing is an important part of software development life cycle since it revalidates the modified software [2,3]. The goal of the regression testing is to verify changes and to show that the changes do not harmfully affect other parts of the software.

(Overview of the regression testing is discussed in section 2)

With the existence of model based testing, the existence of the model based regression testing is essential. For state of the art regression testing, the analysis of the existing model based regression testing approaches [4, 5, 6, 7, 8, 9, 10, 11, 12, 13] is necessary, which is the purpose of this paper.

These approaches as a whole identify the differences between the original version of the software and the changed version of the software on different basis. The basis of the different identification between the original and the modified versions normally depends upon the characteristics of the models used for the regression test generation. Every model has their own characteristics and the testers try to identify the changes according to these characteristics. To avoid the ambiguities we explain the word characteristic of any model with respect to the example. In a class diagram (for example) the word characteristics refer to the attributes and operations [2]. Here we consider attributes and the operations as characteristics of a class diagram.

After the change is identified between both the old and the new version of the software, it has to be tested, whether the newly changed part is working properly and the unchanged portion of the software is not affected with this specific change.

In the next section we present background about the regression testing, in section 3 the survey of existing model based techniques is presented with classification and analysis. Model based regression testing approaches are evaluated in section 4, finally we concluded in section 5.

2. BACKGROUND
Testing can be used to build the confidence in the correctness of the software and to increase the software reliability. The major difference between the regression testing and the development testing is that during regression testing an established set of tests may be available for reuse. Regression testing is essential when software product has been changed and it is applied during maintenance. Regression testing ensures that the modified program meets its specification and new errors are uncovered [3].

Regression testing is expensive maintenance process which is responsible for revalidating the modified software. Maintenance cost of the software is high as compared to its development cost. According to the new assumptions, the maintenance cost of the software is exceeded to 70% relevant to the development cost [2].

Maintenance has three types; Perfective, adaptive and corrective [2]. Corrective maintenance is performed to correct error that has been uncovered in some part of the software. Adaptive maintenance is performed when software is modified to ensure its compatibility with the new environment in which it will operate. Perfective maintenance is performed to add new features to the software or to improve performance of the software. Regression testing is done in all these types [1].

There are two types of regression testing; Progressive regression testing and corrective regression testing. Corrective regression testing is applied when specification is not changed; probably some other change has been made i.e. correcting an error. In this case test cases can be reused. Progressive regression testing is applied when specifications have been changed and new test cases must be designed at least for the added part of the specification [1,2].
Regression testing uses two approaches to test the modified software; Retest all approach and selective test approach. Retest all approach chooses all test cases from the unchanged software to test the changed software, but the approach is time consuming as well as resource consuming approach. Selective retest approach chooses a subset of the tests from the old test suit to test the modified software [2]. In regression, testing selecting a suitable subset with an efficient algorithm is a major area of research.

3. MODEL BASED REGRESSION TESTING: A CLASSIFICATION

This paper presents an analysis of model based regression testing techniques with classification. These techniques generate regression tests using different system models. Most of the techniques are based on the UML models. The techniques in this survey use some models like, class diagrams, state machines diagrams, activity diagram, and use case diagrams etc. Most of the techniques use different types of state machines like state machine diagram, finite state machine and extended state machine diagrams. We classify the techniques presented in this paper according to kind of model the techniques are based on. Four classes are listed below:

1) State machine based approaches
2) Activity diagram based approaches
3) Model checking based approaches
4) Hybrid approaches

An evaluation of the techniques based on parameters (Identified during this survey study) is presented in section 4. Some of the parameters which we used are identified by Rothermel and Horrald [14]. Before a tabular evaluation of the model based regression testing approaches, all the techniques are classified into the unmentioned classes, briefly discussed and analyzed.

3.1 State Machine Based Approaches

The approaches classified as state machine based approaches use different types of state machines to generate regression test. We are going to present these approaches in a brief way.

3.1.1 Model Based Regression Test Suite Generation Using Dependence Analysis

The model based approach given by Chen et al [6] is based on the extended finite state machine (EFSM). This approach is based on a 9-tuple attributes of EFSM. One of them is transition and the whole idea of regression testing lies in it (transitions of the EFSM consisting of 6-tuple). For the purpose of the regression testing they identified the dependencies between these transitions. They concluded two types of dependences; Data dependence (DD) and the control dependences (CD). The data dependency is the usage of a variable by one transition that is set by another transition. While in control dependence one transition can effect the traversal of the other transition [6].

The basic structure of the EFSM is based on transitions and the data dependency is identified between the transitions. If one transition defines the data and the other transition defines the use of the data then data dependency exists between these two transitions. Basically the idea is to identify the def-use associations between the transitions as def-use associations are identified in data flow analysis. The test suite reduction is based on data dependency and the control dependency.

Control dependency also exists between two transitions, it identifies the traversal of a certain transition depends on another transition. To find out such dependencies the concept of the post domination is used. A state can post dominate another state of the any other transition if that state or transition has to traverse that certain state to reach the exit state. Control dependency exists if the transition is post dominated and the state of that transition is not post dominated.

Above mentioned dependencies are identified in three aspects, when EFSM model is modified; effect of the model on the modifications effect of the modifications on the model, and side effects to unchanged parts.

According to [6] there are three ways in which a model is considered as modified; addition of a transition, deletion of transition and change of transition. From these modifications all control dependencies and data dependencies are identified and further regression test suite is constructed on the basis of these dependencies.

The technique is tool supported and provides high efficiency and good coverage criteria. There is no case study found in this technique. Furthermore this technique is yet to be evaluated on an industrial based case study.

3.1.2 Regression Test Reduction Using Dependence Analysis

The next technique discussed here is not purely a regression test selection technique. Basically the aim of the technique is to reduce the regression tests already selected for an EFSM model. This technique is based on the technique of regression test selection using dependence analysis by Chen and Probert [6]. In this approach [7] the test cases are generated using dependence analysis as this analysis was done in [6]. Two kinds of dependencies; DD and CD (discussed in previous technique) were identified and from these dependencies the regression tests were generated. Two kinds of dependencies i.e. data dependency and control dependency are discussed here briefly.

After the modifications are made the impact of the modifications is analyzed with respect to these two kinds of dependencies. By the analysis of the elementary modifications, three interaction patterns are calculated named as affecting interaction patterns, affected interaction pattern and side-effect interaction pattern then these interaction patterns are used to reduce the test suite [7].

This approach is basically a refinement of the regression test suite. So it is much helpful in the cost reduction of the regression testing. It is cost effective, efficient and importantly tool supported. It is yet to be evaluated on a larger case study to analyze its effectiveness and efficiency more deeply.

3.1.3 Regression Test Suite Reduction Using Extended Dependence Analysis

This technique is highly based on the [6] and [7]. There is a little modification made in the approach discussed in [6]. The elementary modifications (EM) considered in [6] are addition of transition in EFSM model and the deletion of the transition in EFSM model. Now the extension is made in the approach to the reduction of regression test suite. All three types of EMs for the EFSM model [7] are considered in the reduction of the regression
test suite. Third elementary modification i.e. the change of the transition is also involved and the data dependency and control dependency analysis as well [8]. In the earlier techniques the data and control dependencies were not analyzed.

Again in this approach three interaction patterns are computed and on the basis of these patterns the test suite is reduced as it is done in [7].

The major contribution of the approach is the consideration of the change of transition of the EFSM model as EM reduces the regression test suite. Previous technique [7] was not working with this type of modification so this approach removed the flaw of that technique.

One of the major plus points of this technique is its automation; no case study is presented and also not validated on an industrial based case study.

3.1.4 Integrating White Box and Black Box technique for Class Level Testing

This is an approach for the regression testing based on the finite state machine. In this approach the researchers has proposed a class level regression testing technique by using both black box and white box testing techniques. Many techniques have been proposed for class level regression testing but most of these techniques focus on either black box testing or white box testing technique. Beydeda and Gruhu [12] have proposed a new approach at class level by integrating both white box and black box. An integrated approach can improve the efficiency of testing process in terms of cost and time.

Comparison is shown by using representation of a class called class control flow graph (CCFG). The comparison is carried out simultaneously by traversing both CCFG and comparing the nodes of the graph. In the second building block, the idea is to identify the definition and uses of each attribute and associate to other according to some data flow criteria. After association with each other test cases covering these def-use pairs are generated.

This technique generates test cases by using class state machine (CSM) graph. The proposed method combines these techniques. Integrated white and black box testing technique operates on a graph called class specification implementation graph (CSIG) [12].

3.2 Activity Diagram Based Approaches

The approaches classified as activity diagram based approaches are used to generate regression tests through activity diagram models. Brief description and analysis of these approaches is presented in this section.

3.2.1 Specification Based Regression Test Selection with Risk analysis

We consider this approach as a risk-based regression testing. In this approach the authors have considered the risks related to the software potential defects as a threat to the failure after the change as a significant factor, so a risk model is presented as well as the model of regression testing. Two type of test are to be included in the regression test suite, targeted tests and safety tests [9].

Targeted tests are the tests included in the regression testing for the reason of the change in the software while the safety test are those tests which are included into the regression tests on the basis of risk analysis. Combination of both targeted and safety tests are ultimately considered as regression tests [9].

To generate the targeted tests the activity diagram model is used. A way to generate test cases as well as regression tests from the activity diagram is presented. In the first step the regression test cases are generated by the EFSM analysis in the activity diagram.

The second and important step is to generate test cases which are risk based. The author presented an approach for this purpose consisting of four steps. In the first step the cost of every test case is assessed. The cost of every test case is ranked through 1-5, where the lowest value depicts the lower cost and the high value as higher cost. The cost is measured in the form of consequences of the fault; the second step is to calculate the severity probability for each test case. It is calculated by multiplying the number of defects and the average severity. The severity probability is obtained by counting the uncovered defects by a certain test case.

After the calculation of severity probability the risk exposure is calculated by the multiplication of then cost and severity probability of the defect. The value obtained is considered as the risk of the test case. Higher the value higher the risk is, and vice versa. Next the test cases with higher value of risk are chosen and included in the regression test suite. These are the safety tests which are run along with targeted tests for the purpose of regression testing of the software [9].

The major benefit of the technique is the risk analysis. After the change that is made in the software there may be potential risks. Test relevant to these risks should not be neglected. Those defects may result in catastrophic consequences. Furthermore, it provides high efficiency and it is cost effective as well. This technique is evaluated on a large industrial based case study, which is yet another plus point of the approach.

3.2.2 Towards Software Architecture Based Regression Testing

In this approach [13] it is described that how regression testing can be applied systematically at software architecture level. This approach is used to reduce the cost of the retesting modified system and to check the regression testability of the evolved system. Software architecture based regression testing may be used in both cases of development and during maintenance. But according to [13] the focus is on maintenance aspect. This work builds upon the general framework for software architecture based conformance testing. The goal of the approach is to test whether the system implementation works correctly.

This approach is tool supported by the LISA tool, the C2 framework, the Argus-I environment and the UNIX diff utility. They applied this technique on a case study for the validity of the approach. This technique is cost effective and tool supported. This approach seemed well evaluated and well validated.

3.3 Model Checking Based Approaches

In this section approaches are classified as model checking based approaches to generate regression test. One regression testing approach is found using model checking which is discussed below.

3.3.1 Regression Testing Via Model Checking

Lihua, Dias, and Richardson [5] have proposed an approach for the regression testing via model checking. They named their approach as Regression Testing via Model Checking (RTMC). At
the abstract level the basic idea of their approach is to take two specifications and find the differences between those specifications for the purpose of regression test generation. Both the original and the modified specifications are taken and passed to a comparator. The work of the comparator is to extract the differences between the original and modified specifications. From these differences the changed properties are identified called Extracted Prop. In the next step, all these properties are checked via a model checker to identify which properties of the specification are changed and which are not changed. Next, for the changed properties the test cases (for regression testing purpose) are generated and the specification is tested again. [5]

All the activities of this approach are performed through various components of the RTMC model. The components are RTMC controller, Comparator, Coverage Selector, Translator, and Test generator [5].

This approach supports different coverage criteria, includes a case study for the validation of technique. The technique is cost effective but not evaluated upon a large case study furthermore the efficiency of approach is not so significant.

3.4 Hybrid Approaches
The approaches which used multiple models for generating regression test cases are classified as hybrid approaches and discussed below.

3.4.1 An Approach for Selective State Machine Based Regression Testing
The approach given by Farooq et al [4] is a model based selective technique for the regression testing based on the class diagram and state diagram model of UML.

For the purpose of the regression testing the changes in the modified version of the software is to be identified so that appropriate test cases should be generated to check whether the new version is fit for the purpose and the changes did not cause any problem for the unchanged portions of the software. To identify changes they [4] categorized the changes into two classes; Class driven and state driven changes.

As for as Class driven changes are concerned they identified these changes from three major categories of changing attributes i.e. operations, and relationship. the class driven changes they identified are ModifiedExpression, ChangedMultiplicity, ModifiedProperty, ChangedMultiplicity, ModifiedAttribute, ModifiedOperationParameter, ModifiedOperation, ModifiedAssociation, Added/deleted Attribute, Added/deleted Operation, Added/deleted association [4].

The second kind of the changes identified were state-driven changes. These changes were identified from the state diagram of the software design. The state driven change categories identified were added/deleted state, modified state, added/deleted transition, modified transition, modified event, modified actions, and modified guards. After the identification of these changes, test cases can be generated according to the categories of both classes of changes, which are in fact the test suite for regression testing.

They applied their technique on a case study for the validity of the approach. [4].This approach presents a case study to validate the technique. Their coverage criterion is high. It is yet to be evaluated on a large case study and needs not only to be automated but also to be made more efficient.

3.4.2 Automated Impact analysis and Regression Test generation using UML Design
This is an approach [10] for the regression testing based on the UML models. To generate the regression test, three types of models from UML models are used for the purpose of the identification of the changes; class diagram, sequence diagram and use case diagram. On the basis of the identified changes from these UML models, the impact of the changes is also analysis to generate regression test suite [10].

First of all the change analysis is done in the two versions of class diagrams, i.e. original class diagram and the modified class diagram. There are different changes related to the class diagrams. The authors have identified that the changes may be added/deleted attribute, change attribute, added/deleted method, changed method, added/deleted relationship, changed relationship added/deleted class and the changed class [10].

The next step is to identify changes in the two versions of sequence diagrams, original sequence diagram and the modified sequence diagram. Author has identified changes in two forms i.e. use case and the methods. Changes which may occur are add/deled use case, changed use case, add/deleted method and changed method [10].

On the basis of the impact of these changes, the test cases from the original suite are classified into three categories. 1) Obsolete test cases are those not required for the regression testing. 2) Retest able test cases are those which are required for the regression testing and will be included in regression testing. 3) Reusable test cases are those which are for the unaffected portion of the models.

The author has presented a regression test selection tool for this approach named as RTSTool [10].The technique is safe, efficient, cost effective and tool supported. The technique is also evaluated on a large industrial based case study.

3.4.3 Regression Testing Using UML Design
Pilkasn et al [11] provided an approach for the regression testing based on the UML models as well. This approach is used to test the UML design model and to check the inconsistencies. A model is made which merges information from class diagrams, sequence diagram and OCL statements and further generates test cases to check inconsistencies. Furthermore, proposed approach categorizes changes found in UML design and then classifies the test cases which are based on categories of UML design.

They provide a set of rules i.e. how to reuse the existing test cases and how to generate the new test cases. New test cases will ensure that affected part of the system do not harmfully affect other parts of the software. The technique is safe, efficient, and cost effective. The case study is included but it is yet to be evaluated on a large case study.

4. ANALYSIS OF THE APPROACHES AND RESULTS
For evaluating different aspects of the model based regression testing approaches, we have identified some attributes including some attributes by Rothermel and Horváth [14]. A brief introduction of the parameters is presented and table 1 shows their acronyms, short definitions and possible values against any approach. In table 2 the acronym along with the classification of
the approaches is given while in table 3 all the approaches discussed an analyzed in section 3, are evaluated upon the following parameters. Table 3 the core part of the research indicating the areas of research and improvements.

4.1 Safety
One very important question during the analysis of any regression testing approach is its safety that whether the technique is safe of not. This parameter is the part of the framework for evaluation of regression testing approaches provided by Rutherme and Horrald [14].

4.2 Different Coverage Criteria
Some techniques are specific to only some coverage criteria, it is very important to be known for the improvement of the approaches. If an approach is not covering a specific criterion then the goal should be to extend the approach to make it possible that it should cover as many criteria as possible.

4.3 Efficiency
The landmark parameter in evaluating any kind of algorithm is its efficiency, so we are interested to set a value for the efficiency of each approach.

4.4 Risk analysis
It is more worthy if a technique is also based on the risk analysis of the system. It is necessary for the effectiveness and fault tolerant of the system.

4.5 Cost Effectiveness
The approaches should be cost effective so that these could be implemented in a real environment. We have analyzed the cost effectiveness of all the approaches for finding the areas where the cost effectiveness of the approaches should be enhanced.

### Table 1: Evaluation Parameters

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Comparison attributes</th>
<th>Possible values of attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>SF</td>
<td>Is technique safe?</td>
<td>Yes (Y), No (N)</td>
</tr>
<tr>
<td>DCC</td>
<td>Whether it satisfies different coverage criteria?</td>
<td>Yes (Y), No (N)</td>
</tr>
<tr>
<td>EFF</td>
<td>How much technique is efficient?</td>
<td>Low (&lt;30%), mod. (30%-69%), high (&gt;70%)</td>
</tr>
<tr>
<td>RB</td>
<td>Technique is risk based or not?</td>
<td>Yes (Y), No (N)</td>
</tr>
<tr>
<td>CE</td>
<td>Technique is cost effective or not?</td>
<td>Low (&lt;30%), mod. (30%-69%), high (&gt;70%)</td>
</tr>
<tr>
<td>CC</td>
<td>How much coverage criteria are satisfied?</td>
<td>Low (&lt;30%), mod. (30%-69%), high (&gt;70%)</td>
</tr>
<tr>
<td>CS</td>
<td>Whether case study is included or not?</td>
<td>Yes (Y), No (N)</td>
</tr>
<tr>
<td>AT</td>
<td>Technique is automated or not?</td>
<td>Yes (Y), No (N)</td>
</tr>
</tbody>
</table>

4.6 Volume of Coverage Criteria
Another very important factor is the volume of a coverage criterion covered by a certain approach. The aim should be to cover the whole of not only a certain coverage criteria but also multiple coverage criteria.

4.7 Case Study
Its always worthwhile that the techniques should included a case study with them so that their worth in context of efficiency, coverage criteria, safety etc could be analyzed.

4.8 Automation
Automation of the approaches depicts that it is validated to a larger extent and implemented. Furthermore the automation of the approaches leads it to be implemented in real world environment.

4.9 Checking Scalability
One of most important parameters is checking the scalability of the approaches, none of the approaches could make its place in the industry without checking its scalability worth. The approaches must be evaluated on a larger industrial based case study before their implementation to the real world environment.

### Table 2: Abbreviations of the approaches

<table>
<thead>
<tr>
<th>Group</th>
<th>Model based regression testing techniques</th>
<th>Abb.</th>
</tr>
</thead>
<tbody>
<tr>
<td>State Machine Based Approach</td>
<td>Model based regression test suite generation using dependence analysis</td>
<td>RTDA</td>
</tr>
<tr>
<td></td>
<td>Regression test suite reduction using extended dependence analysis</td>
<td>REDA</td>
</tr>
<tr>
<td></td>
<td>Model based regression test reduction using dependence analysis</td>
<td>RDA</td>
</tr>
<tr>
<td></td>
<td>Integrating white and black box technique for class level regression testing</td>
<td>CLT</td>
</tr>
<tr>
<td>Activity Diagram Based Approach</td>
<td>Specification based regression test selection with risk analysis</td>
<td>RB</td>
</tr>
<tr>
<td></td>
<td>Towards software architecture based regression testing</td>
<td>SAB</td>
</tr>
<tr>
<td>Model Checking Based Approach</td>
<td>Generating regression test via model checking</td>
<td>MC</td>
</tr>
<tr>
<td>Hybrid Approaches</td>
<td>An approach for selective state machine based regression testing</td>
<td>SSMB</td>
</tr>
<tr>
<td></td>
<td>Automating impact analysis and regression test selection based on UML design</td>
<td>IART</td>
</tr>
<tr>
<td></td>
<td>Regression testing UML design</td>
<td>UMLD</td>
</tr>
</tbody>
</table>
Table 3: Analysis of the model based regression testing approaches.

<table>
<thead>
<tr>
<th>Appr.</th>
<th>SF</th>
<th>DCC</th>
<th>EFF</th>
<th>RB</th>
<th>CE</th>
<th>CC</th>
<th>CS</th>
<th>AT</th>
<th>SC</th>
</tr>
</thead>
<tbody>
<tr>
<td>RTDA</td>
<td>N</td>
<td>N</td>
<td>High</td>
<td>N</td>
<td>Mod</td>
<td>High</td>
<td>N</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>REDA</td>
<td>-</td>
<td>N</td>
<td>High</td>
<td>N</td>
<td>High</td>
<td>Mod</td>
<td>N</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>RDA</td>
<td>-</td>
<td>N</td>
<td>Mod</td>
<td>N</td>
<td>Mod</td>
<td>Mod</td>
<td>N</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>CLT</td>
<td>N</td>
<td>N</td>
<td>High</td>
<td>N</td>
<td>-</td>
<td>-</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>RB</td>
<td>Y</td>
<td>N</td>
<td>High</td>
<td>Y</td>
<td>High</td>
<td>No</td>
<td>Y</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>SAB</td>
<td>N</td>
<td>N</td>
<td>Mod</td>
<td>N</td>
<td>High</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>MC</td>
<td>N</td>
<td>Y</td>
<td>-</td>
<td>N</td>
<td>High</td>
<td>High</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>SSMB</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>-</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>IART</td>
<td>Y</td>
<td>N</td>
<td>High</td>
<td>N</td>
<td>High</td>
<td>-</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>UMLD</td>
<td>Y</td>
<td>N</td>
<td>High</td>
<td>N</td>
<td>High</td>
<td>-</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
</tr>
</tbody>
</table>

5. CONCLUSION AND FUTURE DIRECTIONS

This survey presents the model based regression testing and their analysis with respect to the identified parameters. It can be helpful in exploring new ideas in the area of regression testing specifically model based regression testing. In table 3 we have analyzed the techniques by some parameters we have identified. This evaluation of the model based regression testing techniques can be helpful to improve the existing techniques where they lack. This evaluation can also be very helpful to evaluate and upcoming technique. The parameters are not specific for only the model based regression testing but also can be helpful for evaluating and improving other regression testing approaches.

One future recommendation is the improvements in the framework of Rothermel and Horrold [14] by including the parameters identified during this research.

6. REFERENCES


