Automated Model Based Regression Testing using Behavioral Aspects

Miss. Yogita P. Sonawane
Department of Computer Engineering
Dr. Babasaheb Ambedkar Technological University,
Lonere, India.
yogita.sonawane13@gmail.com

Prof. Manjushree D. Laddha
Department of Computer Engineering
Dr. Babasaheb Ambedkar Technological University,
Lonere, India
manjuladdha@gmail.com

Abstract— Important activity of MBT is a selective regression testing, it selects test cases for retest based on model modification. It is challenging task to identify modifications in the system and selection of test cases due to interdependencies among the models. It achieves higher level of automation. State based testing is an important approach to test the system behavior.

In this paper we concentrate on behavior of system using state chart and activity diagram. We developed AMBT tool which is automated, java based tool. It uses XMI v1.3 interchange format for UML model. AMBT tool generate test cases and classify it into obsolete, reusable and retestable test cases.

Keywords- *Regression Testing; Model Based Testing; UML diagram class; state char, activity*.

I. INTRODUCTION

Regression testing is a testing process which is applied after a program is modified. It involves testing the modified program with some test cases in order to re-establish our confidence that the program will perform according to the (possibly modified) specification. Regression testing is a major component in the maintenance phase where the software system may be corrected, adapted to new environment, or enhanced to improve its performance.

Model-centric development creates opportunities for software testing. In particular, it creates opportunity to drive the testing process at a higher level of abstraction and to demonstrate compliance of models with source code, by means of Model-Based Testing (MBT). MBT employs models as basis for test generation, generating test cases based on what is expected from the system (in the form of models). Hence, it tests what the system is supposed to do, rather than what the system does [6].

Leung and White categories test cases into three classes.

1) *Reusable:* reusable test cases only execute the parts of the program that remain unchanged between two versions, i.e. the parts of the program that are common to P and P'. It is unnecessary to execute these test cases in order to test P'; however, they are called reusable because they may still be retained and reused for the regression testing of the future versions of P.
2) *Retestable:* retestable test cases execute the parts of P that have been changed in P'. Thus retestable test cases should be re-executed in order to test P'.
3) *Obsolete:* test cases can be rendered obsolete because 1) their input/output relation is no longer correct due to changes in specification 2) they no longer test what they were designed to test due to modifications to the program, or 3) they are `structural' test cases that no longer contribute to structural coverage of the program [5].

In model-based development, several artifacts are inter-related. A change in one artifact can affect other related artifacts hence, it is necessary to cater for these relationships and dependencies for effective regression testing. In UML-based development, the class diagram shows the structural aspects of classes and state machines reflect the behavioral aspects of the classes. State machine-based testing is widely used in practice to test class behavior. When a class is modified, due to any change in the specification, both structural and behavioral aspects can be modified and it is necessary to retest the corresponding test cases [2]. The state chart diagram models the different states that a class can be in and how that class transitions from state to state.

Activity diagrams are one of the important UML models used in representing the workflows of stepwise activities and actions with support for choice, iteration and concurrency. Moreover, Activity diagrams can be utilized to describe the business and operational step-by-step workflows of components in a system [4]. It shows the overall flow of control between activities as well as the activity-based relationships among objects as it has all the characteristics that can improve the quality of the automatically generated test cases as well as using these test cases for system, integration, and regression testing.

Finding test information from activity diagram is a formidable task. Reasons are attributed as follows:(a) activity diagram presents concepts at a higher abstraction level compared to other diagrams like sequence diagrams, class diagrams and hence, activity diagram contains less information compared to others, (b) presence of loop and concurrent activities in the activity diagram results in path explosion, and practically, it is not feasible to consider all execution paths for testing.

Activity diagram starts with initial node and end with final node. Activity can be written in rectangle or box with

round corner. Detail code, pre condition and post condition for activity and action can be included with activity diagram. Notes for each action can be specified through dog eared rectangle box. The nodes represent processes or process control including action states, activity states, decisions, swim lanes, fork, join, objects, signal senders and receivers [7][1].

In this paper, we focus on automating regression test selection based on architecture and design information represented with the Unified Modeling Language (UML). State chart and activity diagrams specify the desired behavior of the system and class diagram specify the structural view of the system. Using this structural and behavioral view we generate test cases and classify it into obsolete, reusable and retestable test cases which result in reduction in time and cost of testing.

The rest of the paper is structured as follows: Section 2 literature review Section 3 describes previous state-based regression testing methodology. Section 4 discusses the proposed approach for selective regression testing. Section 5 provides the change definition for change identification. Section 6 shows required Model based tool information. Finally we conclude the paper in Section 7.

II. LITERATURE REVIEW

This section describes the existing work on UML based regression testing and test case generation technique.

Pakinam N. B et al. [10] uses Activity Dig and the proposed model introduces an algorithm that automatically creates a table called Activity Dependency Table (ADT), and then uses it to create a directed graph called Activity Dependency Graph (ADG). The ADT is constructed in a detailed form that makes the generated ADG covers all the functionalities in the activity diagram. Finally the ADG with the ADT are used to generate the final test cases.

Devikar R. et al [9] developed MBRT (Model Based Regression Testing) tool, which is used for generation, reduction of test cases and also classify the test cases into obsolete, reusable and re-testable test cases which results in reduction in time and cost. It uses class diagram and state machine diagram for presenting the idea of regression testing and also use of flow graph to generate the test cases.

Ms.Thanki H. et al [7] focuses on test case generation techniques and describes proposed approach for improvement in test case minimization and prioritization. Generation of optimal test cases us model driven testing. Changes in modified activity diagram element will identify Reusable and retestable test case.

Atifah Ali. et al [11] presents a methodology for identifying changes and test case selection based on the UML designs of the system.UML class diagram and sequence diagrams, which are used to generate an extended concurrent control flow graph (ECCFG) .which is further used for regression testing.

Leila Naslavsky. et al [12] contribute an approach and prototype to model-based selective regression testing, whereby fine-grain traceability relationships among entities in models and test cases are persisted into a traceability infrastructure throughout the test generation process.

S. Shanmuga priya et al [14] presents a model based testing approach from which the test paths are automated and obtained before or during the development process and so. Based on UML sequence diagram generate sequence dependency graph and finally derive test path.

Kundu et al [3] present an approach of generating test cases from activity diagrams using UML 2.0 syntax and with use case scope. Consider a test coverage criterion, called *activity path coverage* criterion. The test cases generated in this approach are capable of detecting more faults like synchronization faults, loop faults unlike the existing approaches.

Shin Yoo et al [13] presented survey each area of minimization, selection and prioritization technique and discusses open problems and potential directions for future research.

Wang Linzhang et al [15] propose an approach to generate test cases directly from UML activity diagram using gray-box method, test scenarios are directly derived from the activity diagram modeling an operation. At last, the possible values of all the input/output parameters could be generated by applying category-partition method, and test suite could be systematically generated to find the inconsistency between the implementation and the design. A prototype tool named UMLTGF has been developed to support the above process.

III. PREVIOUS STATE-BASED REGRESSION TESTING METHODOLOGY

In this section, we discuss previous state-based regression testing methodology. It uses class diagram and state machine diagram and to the corresponding test suite to deal with change propagation.

First it obtain changes in the class diagrams by comparing the baseline version of class diagrams and the delta version of class diagrams along with class invariants and operation contracts. The Class Diagram Comparator performs this comparison task and fined the changes. They refer this change set as class-driven changes.

After computing the class-driven changes the State Machine Comparator compare baseline version of state machine diagrams and the delta version of state machine diagrams along with state invariants and obtain the changes in both versions and class-driven changes are also used to obtain the affected elements of the state machine. Finally, the set of affected test cases from the baseline test suite are selected using Regression Test Selector by tracing the state-driven changes to the corresponding test-cases. It classifies those test cases into obsolete, reusable and retestable test cases.

IV. THE PROPOSED APPROACH

Our proposed approach is based on the existing work on model based regression testing described in [9] with some extensions to fulfill the purpose of regression testing. We include activity diagram to concentrate more on behavior of system.

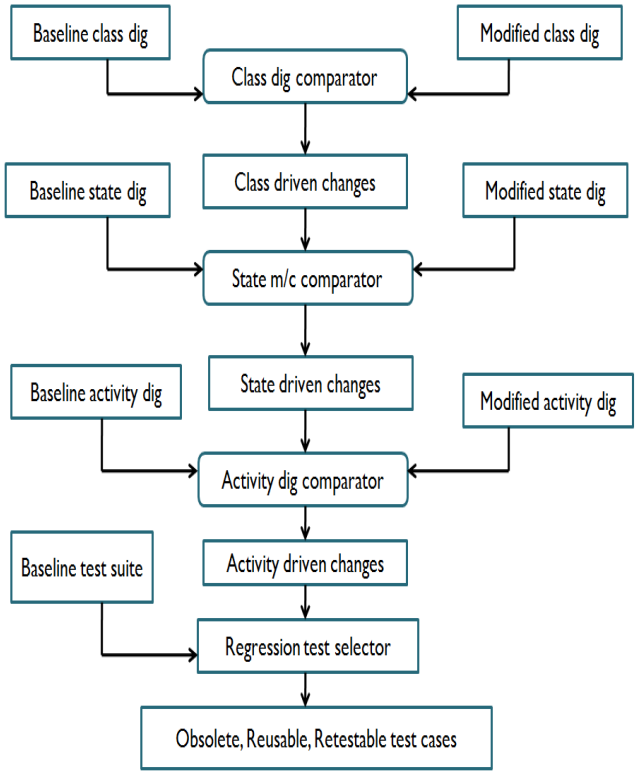


Figure 1: Architecture of Proposed Model

Figure 1 show that the class diagram comparator takes the input from baseline class diagram (original class diagram) and modified class diagram and compares them to find the differences between both the diagrams. We called these differences as class driven changes. If class diagram having polymorphism then it shows that polymorphism message after the class XMI reading. The state machine comparator take input from baseline state diagram (original state diagram) and modified state diagram and class difference. Class differences are used because it obtains the affected elements of the state machine. For example, a state transition will be marked as affected if it is using any changed attribute or operation of the corresponding class in its guards, events or actions. We called these differences as state driven changes. The baseline activity diagram (original activity diagram) and modified activity diagram are comparing by activity comparator based on activity action, activity event and control flow and find the differences between them. Called these activity differences are activity driven changes. Finally, the set of affected test cases from the baseline test suite are selected using Regression Test Selector by tracing the activity driven changes to the corresponding test-cases. Test cases are classified into obsolete, reusable and retestable test cases.

Obsolete test cases are no more valid for the modified version. They usually correspond to elements in the system that are deleted and are not accessible in the modified version. Re-testable test cases need to be executed for regression testing as they correspond to modified parts of the system. Reusable test cases correspond to unchanged parts of the system. They are valid but they are not required to be re-executed for regression testing. In the next section we discus about the change definition required for our research.

V. CHANGE DEFINITION

In this section we discuss about the three types of definitions that we required in our approach, like class driven changes, state driven changes, and activity driven changes.

A. Class Driven Changes:

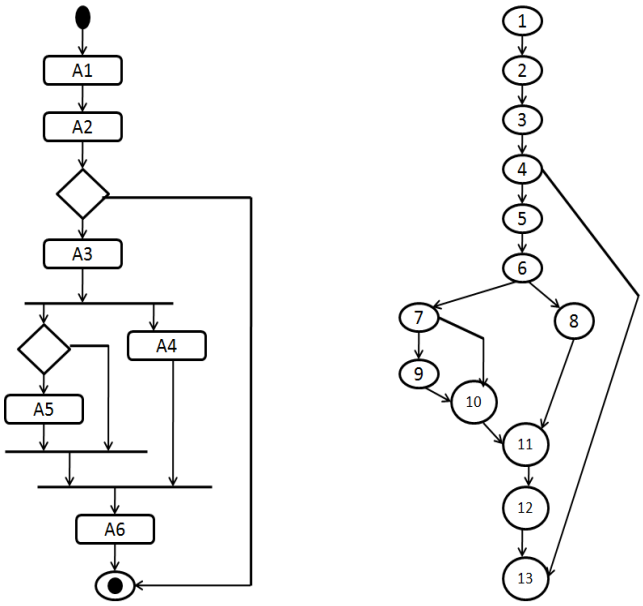
The class diagram comparator compares the original class diagram and modified class diagram and found the differences. The comparator found those changes by comparing the class name, attributes, methods, and classifier. These changes are sometimes directly visible on the state machines, such as deletion of an attribute from a class may cause change in an action using this attribute on the corresponding state machine. Sometimes these changes are not reflected on the state-machine and this information can be obtained only by analyzing the class diagrams [2].

B. State Driven Changes:

The state diagram comparator compares the original state diagram and modified state diagram, also take input from class driven changes and found the differences. We use the state reader, state handler and state difference type of classes in the state comparator. The state reader reads the two version of the state machine diagram. The state handler stores the information of the two state machine diagrams. The state driven changes found the differences between the state, transitions, event, and guard of the two version state diagram.

C. Activity Driven Changes

The original activity diagram and modified activity diagram are comparing for change identification. The activity diagram contains activity states that are made up of smaller actions which are represented in the implementation of a statement in a process or the performance of an activity in a workflow. It compares activity action, event and flow. It also takes input from state driven changes.



Activity Diagram Activity Graph

Figure 2: Example of activity diagram with its activity graph

VI. AUTOMATED MODEL BASED TOOL (AMBT)

This tool uses XMI v1.3 is an interchange format for UML models. Tool is built on Java platform. It consists of three major components; a Parser, a comparator and a Test suite analyzer. Parser is used to parse the XMI files of class, state, and activity diagrams. Parser consists of different sub components like XMI Parser, Class Diagram Parser, State Diagram Parser, and Activity Diagram Parser. The comparator component compare two versions diagram for change identification. It consists of class diagram comparator, state diagram comparator, and activity diagram comparator.

The Regression test selector uses baseline test cases and the change information obtained from the activity comparator. It classify test suite into three types of test cases; obsolete, reusable and re-testable.

```
<?xml version="1.0" encoding="UTF-8" ?>
- <XMI xmi.version="1.1" xmlns:UML="href://org.omg/UML/1.3" timestamp="Thu Jan 08 14:51:35 2015">
  - <XMI.header>
    - <XMI.documentation>
        <XMI.owner />
        <XMI.contact />
        <XMI.exporter>StarUML.XMI-Addin</XMI.exporter>
        <XMI.exporterVersion>1.0</XMI.exporterVersion>
        <XMI.notice />
      </XMI.documentation>
      <XMI.metamodel xmi.name="UML" xmi.version="1.3" />
    </XMI.header>
  - <XMI.content>
    - <UML:Model xmi.id="UMLProject.1">
      - <UML:Namespace.ownedElement>
        - <UML:Model xmi.id="UMLModel.2" name="Use Case Model" visibility="public" isSpecification="false" namespace="UMLProject.1" isRoot="false" isLeaf="false"
            isAbstract="false">
          - <UML:Namespace.ownedElement>
              <UML:Stereotype xmi.id="X.62" name="useCaseModel" extendedElement="UMLModel.2" />
            </UML:Namespace.ownedElement>
          </UML:Model>
        - <UML:Model xmi.id="UMLModel.3" name="Design Model" visibility="public" isSpecification="false" namespace="UMLProject.1" isRoot="false" isLeaf="false" isAbstract="false">
          - <UML:Namespace.ownedElement>
            - <UML:Class xmi.id="UMLClass.4" name="Student" visibility="public" isSpecification="false" namespace="UMLModel.3" clientDependency="UMLDependency.43"
                isRoot="false" isLeaf="false" isAbstract="false" participant="UMLAssociationEnd.32 UMLAssociationEnd.33 UMLAssociationEnd.35 UMLAssociationEnd.36
                UMLAssociationEnd.41" isActive="false">
              - <UML:Classifier.feature>
                  <UML:Attribute xmi.id="UMLAttribute.5" name="status" visibility="public" isSpecification="false" ownerScope="instance" changeability="changeable"
                    targetScope="instance" type="X.65" owner="UMLClass.4" />
                  <UML:Attribute xmi.id="UMLAttribute.6" name="defaulter" visibility="public" isSpecification="false" ownerScope="instance" changeability="changeable"
                    targetScope="instance" type="X.65" owner="UMLClass.4" />
```

Figure 3: XMI file of use case model

VII. CONCLUSION

In our paper we present a model driven testing approach with improved minimization of test cases. In this paper, the methodology uses UML class diagrams, state diagrams and activity diagrams. Paper discusses the existing literature work on model based regression testing.

In this paper the differences are defined between original UML diagram and modified version UML diagram (class, state and activity diagrams). The component of parser, comparator and test suite analyzer are important for change identification.

In this paper, AMBT tool has been developed for selection, generation and classification of test cases. This tool classifies the test cases into obsolete, reusable and retestable test cases. AMBT tool uses UML 2.1 class diagram, state diagram and activity diagram. It is a java based testing tool, which minimize the test cases and improve the software quality.

In future, we work on prioritization of test cases on the basis of genetic algorithm. In future, it is also possible to include sequence diagram for behavior testing. It analyzes the impact of change in sequence diagram on state machine.

REFERANCES

1. Puneet E. Patel, Nitin N. Patil (2013). *Test cases Formation using UML Activity Diagram*. 2013 International Conference on Communication Systems and Network Technologies, IEEE 884-889.
2. Qurat-ul-ann Farooq, Muhammad Zohaib Z., Zafar I Malik, Matthias Riebisch, *A Model-Based Regression Testing Approach for Evolving Software Systems with Flexible Tool Support* , 17th IEEE International Conference and Workshops on Engineering of Computer-Based Systems,2010.
3. Debasish Kundu, Debasis Samanta*: A Novel Approach to Generate Test Cases from UML Activity Diagrams*, in *Journal of Object Technology*, vol. 8, no. 3, May-June 2009, pp.65 83, http://www.jot.fm/issues/issue 2009 05/article1/ .
4. G. Booch, J. Rumbaugh, I. Jacobson. *The Unified Modeling Language User Guide*, Addison-Wesley, 1999.
5. Leung H.K.N, White L., *Insights into regression testing software testing, and Proceedings of Conference on Software Maintenance, on page*: 60-69, ISBN: 0-8186-1965-1, Oct 1989.
6. Leila Naslavsky, Hadar Ziv, Debra J. Richardson, *MbSRT2: Model-based Selective Regression Testing with Traceability*, 3rd International Conference on Software Testing, Verification and Validation(ICST 2010).
7. Ms. Hetal J. Thanki,Prof. S.M.Shinde *Test case generation using UML activity diagram - A survey*, International Journal of Computer Application Issue 4, Volume 3 (May-June 2014).
8. Amitranjan Gantait *Test case Generation and Prioritization from UML Models*, 2011 Second International Conference on Emerging Applications of Information Technology.
9. Mr. Rohit N. Devikar, Prof. Manjushree. D. Laddha *Automation of Model-based Regression Testing*, International Journal of Scientific and Research Publications, Volume 2, Issue 12, December 2012.
10. Pakinam N. Boghdady, Nagwa L. Badr, Mohamed Hashem and Mohamed F.Tolba, *A Proposed Test Case Generation Technique Based on Activit Diagrams*, International Journal of Engineering & Technology IJET-IJENS Vol: 11 No: 03.
11. Atifah Ali, Aamer Nadeem, Muhammad Zohaib Z, Iqbal, Mohammad Usman, *Regression Testing based on UML Design Models*, 13th IEEE International Symposium on Pacific Rim Dependable Computing.
12. Leila Naslavsky, Hadar Ziv, Debra J. Richardson, *MbSRT2: Model-based Selective Regression Testing with Traceability*, 2010 Third International Conference on Software Testing, Verification and Validation.
13. Shin Yoo, Mark Harman, *Regression Testing Minimisation, Selection and Prioritisation : A Survey*, King's College London, Centre for Research on Evolution, Search & Testing, Strand, London.
14. S. Shanmuga Priya,P. D. Sheba Kezia Malarchelvi, *Test Path Generation Using Uml Sequence Diagram*, International Journal of Advanced Research in Computer Science and Software Engineering, Volume 3, Issue 4, April 2013.
15. Wang Linzhang, Yuan Jiesong, Yu Xiaofeng, Hu Jun, Li Xuandong and Zheng Guoliang, *Generating Test Cases from UML Activity Diagram based on Gray-Box Method*, Asia-Pacific Software Engineering Conference 2004,Pages 284-291.