Automation of Model-based Regression Testing

Mr. Rohit N. Devikar[1], Prof. Manjushree. D. Laddha[2]

[1]Department of Computer Engineering, Dr. Babasaheb Ambedkar Technological University, Lonere, India
[2] Department of Computer Engineering, Dr. Babasaheb Ambedkar Technological University, Lonere, India

Abstract- Model-Based Testing is the automatic generation of efficient test procedures using models of system requirements and specified functionality. Model-based regression testing is an important activity that ensures the reliability of evolving software [1]. One of the major issues in this type of testing is the optimal selection of test-cases to test the affected portion of the software. Model-based selective regression testing promises reduction in cost and labour by selecting a subset of the test suite corresponding to the modifications after system evolution. Identification of such changes in modification and selection of test cases is very difficult task.

In this paper, MBRT (Model Based Regression Testing) tool has been developed, which is a java base testing tool 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. In this paper use of class diagram and state machine diagram has been made for presenting the idea of regression testing and also use of flow graph to generate the test cases.

Index Terms- Regression testing; model based testing; UML diagrams; State machine; flow graph.

I. INTRODUCTION

Regression testing is an essential part of an effective testing process for ensuring software quality. The purpose of regression testing is to test a new version of a system so as to verify that existing functionalities have not been affected by new system features. Regression test selection is the activity that consists in choosing, from an existing test set, test cases that can and need to be rerun to ensure existing, unmodified functionalities are still working correctly. Reducing the number of regression test cases to execute is an obvious way of reducing the cost associated with regression testing. Current practice of regression testing of applications using complete system test suite developed during system testing, called retest-all technique [13], is costly and time consuming, hence, leading to delays in deploying software upgrades.

To minimize the time and cost involved in regression testing, a number of regression test suite selection techniques that recommend smaller test suites to validate modified software. Model based regression testing has several advantages compared to the conventional code-based regression testing approaches. The traditional code- based testing approaches fail to scale when the size of the system increases. MBT supports the generation of test cases from models and the demonstration of model and source-code compliance. Models evolve, much like source code. Thus, an important activity of MBT is selective regression testing, which selects test cases for retest based on model modifications, rather than source-code modifications[1]. This activity explores relationships between model elements and test cases that traverse those elements to locate re-testable test cases. In model-based development, several artefacts are inter-related. A change in one artefact can affect other related artefacts; 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. 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. The relationship between class diagram, state machine and the corresponding test suite as discussed in [1] required mapping for test case generation at each time from class diagram to state machine diagram to show the structural and behavioral approach respectively. In this paper we have generated the test cases automatically by using the flow graph as shown in [12]. Figure 1 shows the control flow graph for ATM system, use for generating the test cases shown in Table 1.

Similarly by using MBRT Tool we generate similar type of test cases and also classify the test cases into obsolete, re-testable and reusable test cases.

The rest of the paper is organized as follows: Section 2 describes the different regression test case selection technique proposed in the literature. Section 3 describes the proposed technique. Section 4 describes the change definition for change identification and analyzing its impact on other system. Section 5 describes the MBRT tool and result analysis. Section 6 describes the conclusion and future scope.

II. LITERATURE REVIEW

This section describes the existing work on UML based regression and state based regression testing.

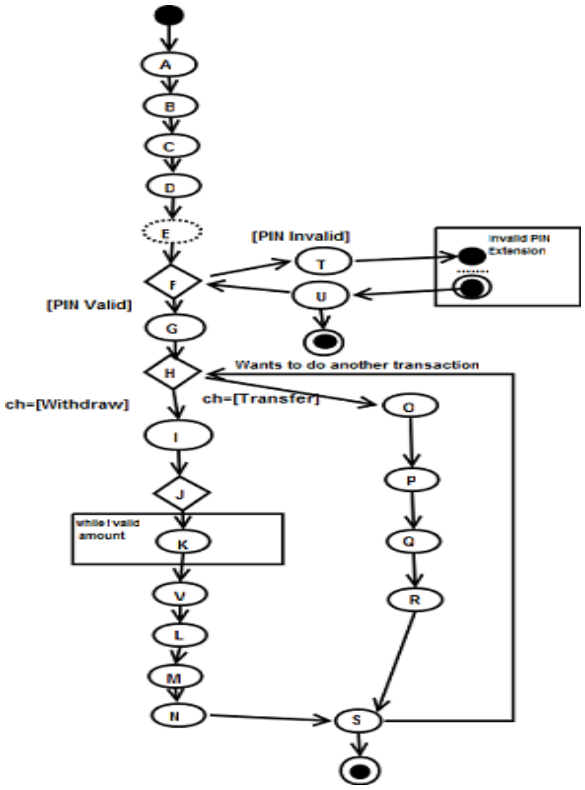

Figure 1: Control Flow Graph for ATM System

TABLE 1: Test cases for the Control Flow Graph of ATM System

	Test Cases
T1	A,B,C,D,E,F,G,H,O,P,Q,R,S
T2	A,B,C,D,E,F,T,U
T3	A,B,C,D,E,F,T,U,G,H,O,P,Q,R,S
T4	A,B,C,D,E,F,T,U,G,H,I,J,K,V,L,M,N,S
T5	A,B,C,D,E,F,G,H,I,J,K,V,L,M,N,S
T6	A,B,C,D,E,F,T,U,G,H,O,P,Q,R,S,H,O,P,Q,R,S
T7	A,B,C,D,E,F,T,U,G,H,O,P,Q,R,S,H,I,J,K,V,L,M,N,S

Briand et al. [2] present a process of using Use Cases, Sequence Diagrams, and Class Diagrams to regression test code. Briand et al. create a mapping between design changes and code changes to classify code test cases. Briand et al. draw upon the work of Leung and White [8] who classifies test cases according to if and how they can be reused.

Chen et al. [6] use UML activity diagrams to detect changes in design and then use a traceability matrix between activity diagram and the test suite. It covers activities at an abstract level and does not cover the attributes of a class.

Wu et al. [4] provide a methodology for regression testing of components using class, collaboration and state diagrams. It does not state any methodology for test cases selection; major emphasis is on change identification only.

Pilskalns et al. [10] propose a regression testing technique based on UML sequence and class diagrams. Their approach does not take into account the pre and post conditions of the operations which affect behavior of a class. Also, their approach does not handle concurrency.

Naslavsky et al. [14] presented an idea for regression testing using class diagrams and sequence diagrams for model driven architecture (MDA). They make use of model transformations for regression testing.

Deng et al. [7] presented a rule-based approach for regression testing using use case diagrams, class diagrams, sequence diagrams and activity diagram. The test cases are generated based on activity diagrams. However, rework is required to apply the technique to a real scenario due to fewer details.

Ali et al. [12] presented an approach for regression testing using class and sequence diagrams. They construct an extended control flow graph using both diagrams. The change identification is based on comparison of both diagrams and then the selection of corresponding test cases is performed.

Qurat-ul-ann Farooq [1] presented the regression testing approach using class diagram and state machine diagram. They use the flexible tool support approach for the selection of test cases.

Our proposed work is based on class diagram and state machine diagram. In this research, we developed our own MBRT Tool that is useful for selection of test cases and classification of test cases on the basis of state and transitions.

III. THE PROPOSED APPROACH

Our approach is based on the existing work on model based regression testing described in [1] with some extensions to fulfill the purpose of regression testing. We included class diagram to get more information while testing and capturing the changes.

Figure 2 shows the flow of our work for regression testing. Control flow graph is constructed using a State machine diagram. Figure 2 shows two comparators, Class Comparator and State Comparator. The Class Comparator compares the Original class diagram (baseline version) and Modified class diagram (delta version) and generate corresponding difference between both the diagrams. This difference we refer as Class Difference. The input to the State Comparator is Original State diagram (baseline version) and Modified state diagram (delta version) and Class Difference. Class differences are used to obtain 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 refer to these changes as State Difference. Finally, the set of affected test cases from the baseline test suite are selected using Regression Test Selector by tracing the State Difference to the corresponding test-cases. Test suit are classified into Obsolete, Reusable and Re-testable test cases [8]. Obsolete test cases corresponds to deleted transition, and no more valid for modified version. Reusable test cases have corresponds to unchanged part of the system.

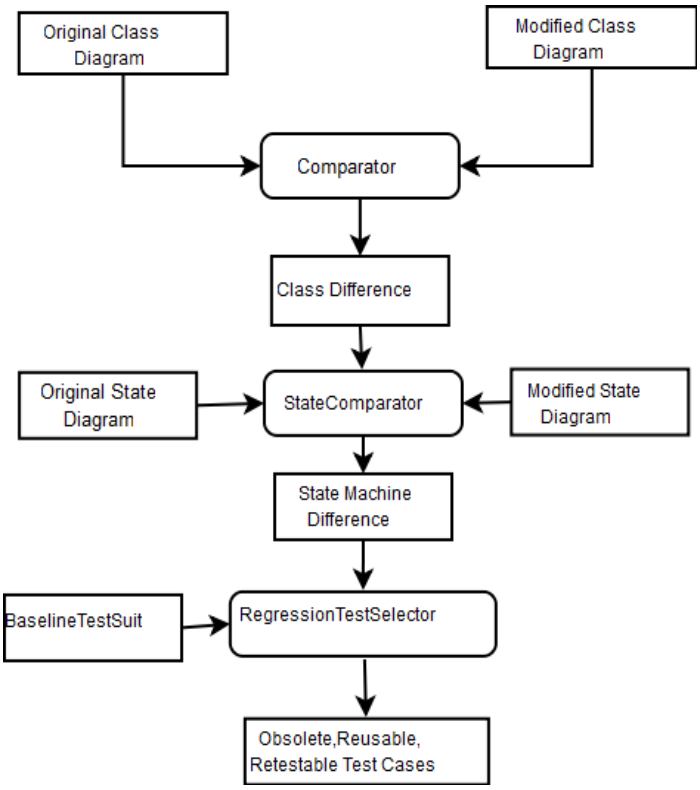

Figure 2: Abstract Model of State Based Regression Testing Approach

These test cases are valid and no need to re-execute for regression testing.

In this paper we concentrate more on Retestable test cases as it corresponds to modified part of the system.

For the generation of test cases we have used control flow graph (CFG), for that we used sequence diagram and class diagram as inputs for the construction of CFG. State machine diagram is used to get the flow of a complete event state and class diagram is used to get information of the attributes from the class of object that receives a message in the state machine diagram. In the next section we discus about the change definition required for our research.

IV. CHANGE DEFINITIONS

In this section we discus about the changes in original class and modified class differences, also changes in original state machine and modified state machine differences.

A. *Class Difference:*

Changes in class diagram are found out by comparing original class diagram (baseline version) and modified class diagram (delta version).The difference between classes can be carried out by using class comparator. The comparator compares the classes attributes, methods and finds the differences.

The figure 3 shows the class diagram comparator. The text area in the comparator shows the result of differences between the two classes. 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.

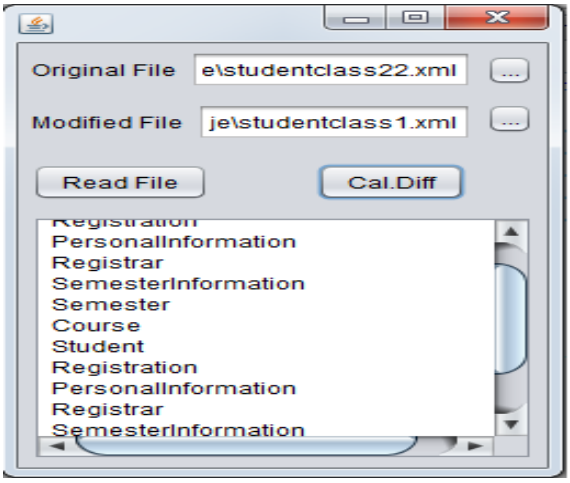

Figure 3: Class Diagram Comparator.

Sometimes these changes are not reflected on the state-machine and this information can be obtained only by analyzing the class diagrams [1].

B. *State Machine Difference*:

Changes in state machine diagram can be found out by providing Original state machine diagram,

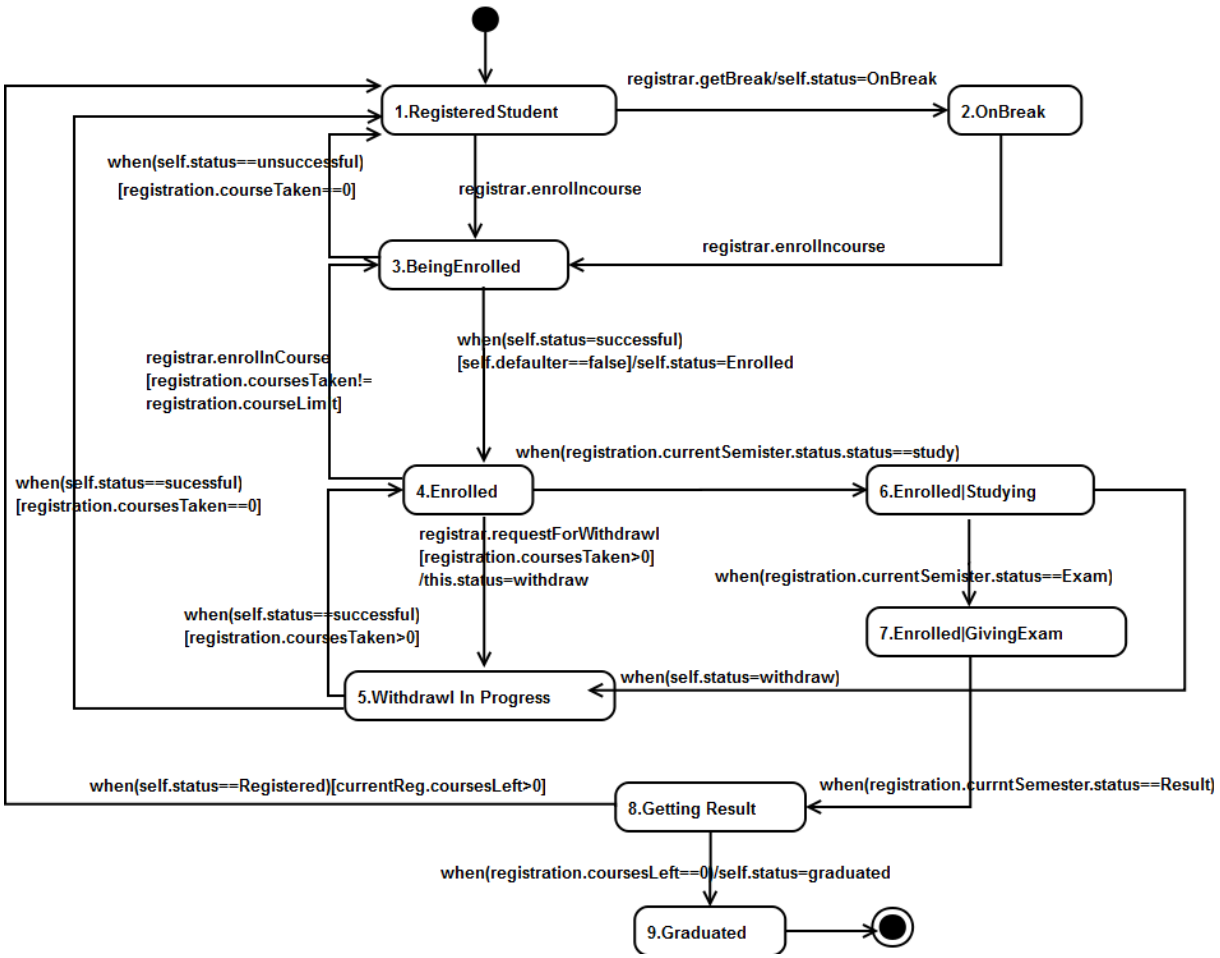

Figure 4: State Machine Diagram for Modified Version

Modified state machine diagram and difference between class diagram as an inputs to the comparator.We use the state reader, state handler, statedifference type of classes in the state comparator. Figure 5 shows the state machine comparator. The class state reader reads the two version of the state machine diagram. The class state handler stores the information of the two state machine diagram. The class state difference finds the differences between the state, transition of the two version state diagram.

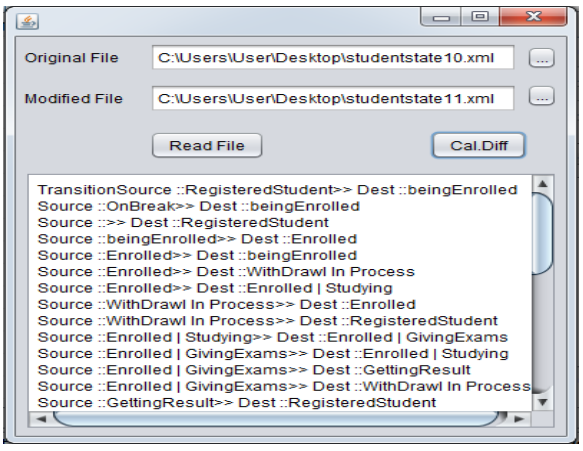

Figure 5: State Machine Comparator.

V. MODEL-BASED REGRESSION TESTING TOOL (MBRT) AND RESULT ANALYSIS:

MBRTP tool is build on java platform.This tool consist of Parser, Comparator, Regression test selector.

The XMI parser use the XMl files of the class and state machine diagram for parsing. Two types of comparator are used class diagram comparator and state machine comparator as discussed in section IV.

The regression test selector block uses the information of State Machine difference and baseline test suite and classifies the baseline test suite into obsolete, reusable and re-testable test cases [8]. Test cases are generated on the basis of state and transition of the state diagram.

Figure 4 shows the modified version of the student system, from this state machine 31 test cases are generated, from that 18 test cases need to be re-executed for regression testing, 4 test case become obsolete and is required to be excluded from the baseline test suite and 9 test cases are preserved for the future use.

Table 2 consists of the test path from the stating state of the system. From table the test path from 1 to 4 shows the Obsolete test cases, 5 to 22 are the Re-testable test cases and 23 to 31 are Reusable test cases for the modified version.

TABLE 2: Test Path Result

N0.	*Test Path*	No.	*Test Path*
1.	**R-B-E-E/S-E/G-W**	**17.**	**R-O-B-E-E/S-W**
2.	**R-B-E-E/S-E/G-W-E**	**18.**	**R-O-B-E-E/S-W-R**
3.	**R-B-E-E/S-E/G-W-R**	**19.**	**R-O-B-E-E/S-W-E**
4.	**R-B-E-E/S-E/G-E/S**	**20.**	**R-O-B-E-E/S-E/G**
5.	**R-B-R**	**21.**	**R-O-B-E-E/S-E/G-GR**
6.	**R-B-E-E/S-W**	**22.**	**R-O-B-E-E/S-E/G-GR-Grad**
7.	**R-B-E-E/S-W-E**	**23.**	**R-B-E**
8.	**R-B-E-E/S-W-R**	**24.**	**R-B-E-B**
9.	**R-O**	**25.**	**R-B-E-W**
10.	**R-O-B**	**26.**	**R-B-E-E/S**
11.	**R-O-B-R**	**27.**	**R-B-E-E/S-E/G**
12.	**R-O-B-E**	**28.**	**R-B-E-E/S-E/G-GR**
13.	**R-O-B-E-B**	**29.**	**R-B-E-E/S-E/G-GR-Grad**
14.	**R-O-B-E-W**	**30.**	**R-B-E-W-R**
15.	**R-O-B-E-W-R**	**31.**	**R-B**
16.	**R-O-B-E-E/S**		

Notations:-

R- Registered Student E/S-Enrolled | Studying
O-On Break E/G-Enrolled | Giving Exam
B-Being Enrolled GR-Getting Result
E-Enrolled Grad-Graduate
W-Withdrawl In Progress

VI. CONCLUSION AND FUTURE SCOPE

In this paper, the methodology for regression test selection using UML state machines diagrams and class diagrams has been presented. Paper also discusses the existing literature work on UML based regression testing. In this paper a model based approach of regression testing has been provided that addresses limitations of existing work on UML based regression testing.

In this paper changes in baseline and delta versions of the class diagram (Class differences) and state machines (State machine differences) are defined. The change definitions presented in the paper conform to UML version 2.1. Classification of baseline test suite into Obsolete, Reusable, and Re-testable test cases is also carried out.

In this paper, MBRT tool has been developed for selection, generation as well as classification of test cases. Also the tool developed is useful for dividing the test cases into obsolete, reusable and re-testable test cases. MBRT, a Java base testing tool, has been used for reduction in the test cases, which results in reduction in time, cost and in improving software quality. In this paper results based on a case study carried out have been specified. Mutation analysis for regression testing is also planned as future work.

Future work aims at determining the effectiveness of the proposed regression testing strategy by applying it on industrial scale case studies of various sizes. In future, it is possible to include analysis of impact of changes in sequence diagram on state machine to integrate the sequence diagram with implemented approach.

Application of implemented approach using the model driven development methodology can also be investigated as one of the possibility in future. It is also possible in future to investigate the required transformations and to see how the relationships between models at different levels of abstractions can affect original methodology. Also, prioritization of test cases on the basis of genetic algorithm can be viewed as future work.

REFERENCE

[1] Qurat-ul-ann Farooq, Muhammad Zohaib Z.Iqbal, 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

[2] L.C. Briand, , Y. Labiche, S. He, *Automating regression test selection based on UML designs*, Information and Software Technology, Volume 51 , Issue 1, January 2009.

[3] Thierry Jéron, Jean-Marc Jézéquel, Yves Le Traon, Pierre Morel, *Efficient Strategies for Integration and Regression Testing of OO Systems*, 10th International Symposium on Software Reliability Engineering, p. 260, 1999.

[4] Wu, Y., Offutt, J. *Maintaining Evolving Component based software with UML*, In Proceeding of 7th European conference on software maintenance and reengineering, IEEE, pp: 133-142, ISBN:0-7695- 1902-4, Publisher: IEEE Computer Society, 2003.

[5] Beydeda, S., and Gruhn, V. *Integrating White- and Black-Box Techniques for Class-Level Regression Testing*, Proceedings of the 25th International Computer Software and Applications Conference on Invigorating Software Development, pp. 357 – 362, 2001.

[6] Chen, Y., Probert, R.L., and Sims, D.P. *Specification-based Regression Test Selection with Risk Analysis*, Proceedings of the conference of the Centre for Advanced Studies on Collaborative research, pp: 1, 2002.

[7] Deng, D., Sheu, P.C.Y., and Wang, T. *Model-based testing and maintenance*, Proceedings of IEEE Sixth International Symposium on Multimedia Software Engineering, Volume 00, pp. 278- 285, ISBN:0-7695-2217-3, 2004.

[8] Leung, H.K.N. White, L. , *Insights into regression testing software testing*, Proceedings of Conference on Software Maintenance., , On page(s): 60-69, ISBN: 0-8186-1965-1, Oct 1989.

[9] Korel, B., Tahat, L.H., and Vaysburg, B. (2002). *Model Based Regression Test Reduction Using Dependence Analysis*, Proceedings of the International Conference on Software Maintenance (ICSM.02).

[10] Pilskalns, O., Andrews, A. *Regression Testing UML Designs*, In Proceedings of 22nd IEEE International Conference on Software Maintenance (ICSM'06), Publisher: IEEE Computer Society, Pages: 254 - 264 , ISBN :1063-6773, 2006.

[11] Sajeev, A.S.M., and Wibowo, B. *Regression test selection based on version changes of components*, Tenth Asia-Pacific Software Engineering Conference, pp. 78- 85, 2003.

[12] Atifah Ali, Aamer Nadeem, Muhammad Zohaib Z. Iqbal, Mohammad Usman, *Regression Testing based on UML Design Models*, Proceedings of the 13th Pacific Rim, International Symposium on Dependable Computing, pp. 85-88, 2007.

[13] Ravi Prakash Gorthi, Anjaneyulu Pasala, Kailash KP Chanduka and Benny Leong, *Specification-based Approach to Select Regression Test Suite to Validate Changed Software*, Proceedings of the 2008 15th Asia-Pacific Software Engineering Conference, pp 153-160, Year of Publication: 2008.

[14] Leila Naslavsky, Debra J. Richardson, *Using Traceability to Support Model-Based Regression Testing*, Proceedings of the twenty-second IEEE/ACM international conference on Automated software engineering, pp. 567-570, 2007.

AUTHORS

First Author – Mr. Rohit N. Devikar,
Second year M.Tech, Department of Computer Engineering
Dr. Babasaheb Ambedkar Technological University, Lonere.
Email:- devikarrohit@gmail.com.
Second Author – Prof. Manjushree D. Laddha,

Assistant Professor, Department of Computer Engineering,
Dr. Babasaheb Ambedkar Technological University, Lonere.
Email: manjuladdha@gmail.com.