International Journal of Computer Science and Engineering (IJCSE)
ISSN 2278-9960
Vol. 2, Issue 3, July 2013, 1-4

International Academy of Science, Engineering and Technology
Connecting Researchers; Nurturing Innovations

SOFTWARE ARCHITECTURE BASED REGRESSION TESTING IMPLEMENTATION

HARSH BHASIN[1], ANKUSH GOYAL[2] & DEEPIKA GOYAL[3]

[1]Department of Computer Science Engineering, DTU, Delhi, New Delhi, India

[2]M.Tech Scholar, Department of Computer Science Engineering, AITM, Palwal, Haryana, India

[3]Assistant Professor, Department of Computer Science Engineering, AITM, Palwal, Haryana, India

ABSTRACT

The use of Software Architecture in Regression Testing has always got the "step brotherly treatment" by software fraternity. The present endeavor focuses on the amazing capability of this fascinating technique in order to accomplish the tedious task of regression testing. The work is a continuation of our earlier attempt and has been successfully applied to a large Enterprise Resource Planning system. The work is good enough to compete with the existing techniques. It promises to bring the advantages of software architecture at the fore. The initial results are encouraging and pave way for the technique to be used for larger systems.

KEYWORDS: Regression Testing, Software Architecture, Test Case Selection

INTRODUCTION

Regression testing is, perhaps, one of the most important types of testing in existence. It is done when changes are made in the software. It may be noted that since software changes continuously during its life span, therefore, regression testing becomes an ongoing process. As per the literature, there are three types of regression testing techniques: selection, minimization and prioritization [1]. Regression test case selection focuses on modification in the software, regression test case prioritization prioritize the test cases so that the task of selecting test cases is accomplished in limited time, minimization reduces the test case suit based on certain criteria. There are many techniques in order to accomplish the above task.

Software architecture technique relies on the specifications and requirements of software to accomplish the task. The technique is good, in the sense, that it does not require the whole code. It may be stated at this point that, most of the times code is not available to the tester. The present work focuses on the use of software architecture in order to reduce the test case suite. The use of software architecture overcomes many problems posed by various conventional techniques [2]. The work is a continuation of our earlier attempt to apply Software Architecture to regression testing [3]. It is in line with the endeavor to comprehensively examining various testing techniques [4- 7].

The architecture of purposed work has already been presented in our earlier paper [3]. The present work focuses on the application of the proposed architecture. It may be stated at this point that, the results obtained so far are encouraging. The rest of paper has been divided as follows. The second section presents the literature review, the third section presents the background of work, the fourth section proposes the work, the fifth section presents verification and the sixth section concludes.

LITERATURE REVIEW

An extensive literature review has been carried out in order to find gaps in the existing techniques and propose a new technique. The review has been carried out in accordance with the guidelines proposed by Kitchenham [13]. It has

been observed that Software Architecture based technique has seldom been applied on regression testing. However, Muccini and Harrold were the pioneers of the concept.

It was observed by Muccini that regression test selection technique selects a few tests from older test suite [2]. However the premise is inapt as the changes in the software render the previous test cases useless. However the software architecture approach overcomes the problem. Since, it maps ATC (Architecture-level Test Cases) to code level test cases according to testing criteria previously identified.

It may be noted that, if the changes are made in the portion of the software which is not covered by test cases, then test cases might not traverse these changes. Another point that must be taken into account is that, if the architecture covers only those components which have not been changed while moving to new version then it might be possible that the changes are not observed.

The solution suggested in order to handle the problem was that checking the new ATC might not be feasible if the path becomes large. It is therefore important to identify the new test cases with the help of software architecture based model.

It may be stated here that selective retest strategies must cater to inclusiveness, precision, efficiency, generality and accountability as noted by Rothermel [8].

The work by Muccini was different in the sense that the existing techniques assumed the availability of source code. Reverse engineering could, therefore, be applied to extract specification. The concept has been used to find out the changes in the object oriented environment as well [9-11]. The problem of these approaches is not just the non availability of code but also the problems that would crop up if the implementation is incorrect.

During the literature review it was also observed that Harrold also employed software architecture based technique for regression testing [12].

BACKGROUND

Premises of the Paper

This proposed technique is based on the concept proposed by Muccini [2]. The behavior of the software architecture is exposed by the components, connectors and configuration, which represents the topology in SA (Software Architecture) based model. In SA based model the base is taken as its specifications, whereas the test oracle is determined by its behavior. State machine based model shows abstraction because non relevant actions are hidden in the software architecture. In order to accomplish the above task, a mapping function is used to map SA level tests to code level tests, which exists in architecture level test cases.

Goals

The following goals are to be accomplished. The first goal checks the conformance of the modified program with the initially suggested software architecture. For testing the conformance, generate the test cases that use the previous test information. This will reduce the test cases in the regression testing.

The second goal tests the conformance of the evolved software architecture with the source code by reusing the architecture – level test cases. In the first goal, architecture of the software and program, implemented on this architecture, tests and checks whether the program is correctly implemented or not. After changing the program by modifying the components or adding an extra component, it will be again tested with the initial software architecture. For the above purpose, the control graph is generated and compared with the previous graph. The result is stored in history. Now, the test

cases are selected on the basis of modified program and test history. Using the selective testing technique for both the code level and the architecture based regression testing, generate the test cases on the basis of architecture. The stored history and expected behavior are mapped with expected output.

The changes are determined by comparing the old and the new graphs. The earlier attempts were not suitable, because in some conditions change in the program might not affect the graph but these changes may affect the software architecture. It may also be noted that coupling changes one portion of the program owing to change in another portion [1]. Earlier techniques did not consider the above issues.

This work intends to propose a new technique by merging of software architecture concept with the concept of regression testing. The proposed technique will be used to find out the errors via software architecture. The concept has been compared with the technique proposed in the work by Muccini [2]. 20 programs have been selected. These programs have been divided into three categories small, medium and large.

PROPOSED TECHNIQUE

The section represents a new framework which is an amalgamation of regression testing and software architecture. This work is different from the base work as behavior of model has been taken into account.

Software Architecture Based Regression Testing with Behavioral Blend

Step 1: SA Specification: SA based conformance testing is a combination of behavioral specification of SA and topology of SA. We use ADL (Architecture Description Language) for describing the topology and TS (Transition System) for behavior of software architecture. Topology includes the components, connectors and configurations.

Step 2: Testing Criterion: We select the architecture level test cases which is the sequence of events. For selecting and designing of architecture level test cases, the behavior and specifications of the model are determined.

Step 3: Test Cases: For checking the correct functioning of the system test cases are generated. The behavior of the program and system can change by adding or modifying the components. So, a mapping function is used to map SA level test cases to code level test cases and also perform the checking function on behavior of the model.

Step 4: Test Execution: Many test cases are generated with the help of the above technique and each test case is checked and mapped with the corresponding result.

VERIFICATION

The technique has been verified on an Enterprise Resource Planning System developed by Sahib Soft. The version of the system, being tested, has 60 modules. The system is developed in Visual C#, .NET framework 4.0. The system has been chosen as its architectural specifications are well defined and the system is a professional system, developed for a company. The system has undergone many changes, since its inception, so it is an apt case for applying regression testing.

CONCLUSIONS

The above work is a continuation of our previous work. It is a part of a larger endeavor to study testing. The concept of software architecture has seldom being used in regression testing.

The above work finds gap in the existing methodology and has been successfully able to propose and verify a new technique. It may be stated here, that SA will be future of regression testing owing to its ability to perform regression testing without having the code.

REFERENCES

1. Harsh Bhasin, Manoj, Regression Testing Using Coupling and Genetic Algorithms, International Journal of Computer Science and Information Technology, Vol. 3, No. 1, pp. 3255 – 3259, 2012.
2. Henry Muccini, Marcio Dias, Debra J. Richardson, Software Architecture-Based Regression Testing, the Journal of Systems and Software 79, pp.1379–1396, 2006.
3. Harsh Bhasin, Ankush Goyal, Deepika Goyal, Software Architecture Based Regression Testing, International Journal on Computer Science and Engineering , Vol. 5, No. 04, Apr 2013.
4. Harsh Bhasin, Shailja Gupta, Mamta Kathuria, Regression Testing Using Fuzzy Logic, International Journal of Computer Science and Information Technologies, Vol. 4, No.02 , 2013.
5. Harsh Bhasin, Harish Kumar, Vikas Singh, Orthogonal Testing Using Genetic Algorithms, International Journal of Computer Science and Information Technologies, Vol. 4 No.02 , 2013.
6. Harsh Bhasin, Shewani, Deepika Goyal, Article: Test Data Generation using Artificial Life, International Journal of Computer Applications, vol. 67 No.12, April 2013
7. Harsh Bhasin et. al., Implementing Regression testing using Fuzzy logic, Communicated.
8. Rothermel, G., Harrold, M.J., A framework for evaluating regression test selection techniques, Proceedings of the 16th International Conference on Software Engineering, ICSE, pp. 201–210, May 1994.
9. Kung, D., Gao, J., Hsia, P., Toyoshima, Y., Chen, C., Kim, Y., Song, Y., Developing an object-oriented software testing and maintenance environment, Communications of the ACM, October 1995.
10. Wu, Y., Chen, M.H., Kao, H.M., Regression testing of object oriented programs, Proceedings of the International Symposium on Software Reliability, 1999.
11. Beydeda, S., Gruhn, V., Integrating white- and black-box techniques for class-level regression testing, Proceedings of the COMPSAC, pp. 357–362, 2001.
12. M. J. Harrold, Architecture-Based Regression Testing of Evolving Systems, Int. Workshop on the Role of Software Architecture in Testing and Analysis (ROSATEA), CNR-NSF, pp. 73-77, July 1998.
13. Barbara kitchenham, Systematic literature reviews in software engineering - A systematic literature review, Journal Information and Software Technology, Vol. 51 No 1, pp. 7-15, January 2009.