Automatic Test Case Generation using Multiple Condition Control Flow Graph

Hyun Seung Son[1], Woo Yeol Kim[1], Jae Seung Kim[1], and Robert Young Chul Kim[1]

[1] Dept. of CIC(Computer and Information Communication), Hongik University, Sejong Campus, 339-701, Korea
{son,john,jskim,bob}@selab.hongik.ac.kr

Abstract. No test can ever be performed perfectly; thus, in risk-based testing, testing efforts and resources are allocated in order of increasing risk. There are strategies for applying this approach to high-level coverage where risk is high and to low-level coverage where risk is low. However, while this approach can be implemented effectively in general overall testing, methods for generating test cases become complicated and difficult to automate, which increases the cost of testing. In an effort to resolve this issue, we use a Unified Modeling Language (UML) sequence diagram as a basis for building a model to propose a method for automatically generating test cases capable of generating a variety of different coverages.

Keywords: UML Sequence Diagram, CFG(Control Flow Graph), Test case Generation, Multiple Coverage, Testing

1 Introduction

Testing is necessary as a means of minimizing problems in software. Recently, with increasing importance of software quality, the software sector has developed a need for more diverse and improved methods of testing. This stems from the fact that, as embedded systems and software systems targeted for testing change from singularities into complex varieties, the scale of software is also ever-growing [1]. Testing has also evolved from testing for existing single systems to testing for complex systems. This has resulted in a need for greater efforts to further improve the quality of software.

There is a need for testing in order to improve software quality in this manner. However, it is impossible to perform a perfect test; this is because of infinite input values, paths, and timings. Testing capable of addressing all possibilities is impossible, with the exception of simplified software. As such, instead of perfect testing, efforts have been focused on testing activities according to the level of priority. For this reason, risk-based testing has recently become a concern. Risk-based testing is a method for inputting testing efforts and resources in order of increasing risk. After the elements of the risk items have been determined and risk has been determined with the participation of the stakeholders, the amount of effort inputted into the testing is determined on the basis of this risk [2,3]. In such a case, the level of risk is the basis for determining the technique or coverage of the testing. However, though this

method is effective when used to carry out testing, it requires several different testing techniques to be applied; therefore, the test case generation method is complicated. In addition, these reasons also render automation complicated and lead to an increase in testing costs.

In this paper, we propose a test case generation method for automatically creating test cases of different levels for a variety of coverages. A Unified Multiple Language (UML) sequence diagram is used as an input model, and a multiple condition control flow graph (MCCFG) is used in order to apply a variety of different testing levels. The MCCFG is a mode for expanding an existing CFG. The test case generation process converts a sequence diagram into a MCCFG and applies coverage criteria [4] to automatically generate various test cases for each respective coverage.

The proposed test case generation method can test different strengths with respect to the testing priority. Furthermore, mixing several different models, as in existing test case generation methods, makes it possible to generate tests for a variety of levels using a single model without the need to perform complex testing.

This paper is organized as follows: Section 2 deals with risk-based testing in related studies. Section 3 deals with a test case generation method using the MCCFG. Section 4 shows the test case generation process by means of the proposed method. Finally, Section 5 presents the results as well as future research.

2 Related Work

Risk-based testing is an effective strategy for allocating test resources based on priority when a comparison of test subjects shows that test resources are insufficient. This method has a process to define risk, analyzes what is defined as risk, establishes avoidance strategies for the analyzed risk, and performs tests in accordance with the strategies.

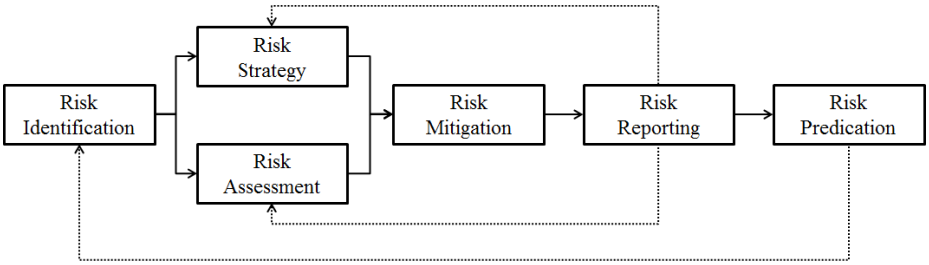

Fig. 1. Risk management process [5]

The risk management process, as shown in Figure 1, seeks risk identifiers from the functions and features of a system to be tested. The identified risks in a system are used as input data for the test plan, which comprises a risk strategy. These items of information are also used to assess risk through a matrix. Risk mitigation allows the tester to inspect the risk items. Finally, test metrics are used to report and predict risks [6].

3 Proposed Automatic Test Case Generation using Multiple Condition Control Flow Graph

The proposed test case generation method generates test cases by converting a sequence diagram model to MCCFG and applying coverage to the MCCFG. Herein, CFG is a publicly available white-box test design technique used to carry out unit or integrated tests as a means of testing program structures. CFGs contain levels that vary depending on the specifications of the test. These facts describe the test cases as consecutive sets of syntaxes depending on the selected flow. For all parts of a test subject, test coverage is increased as the depth of testing increases. However, there is also an exponentially higher number of test cases, which is a problem. An ordinary CFG is decision-based; however, in this paper the concept of multiple conditions is applied to an existing CFG to present the newly conceived MCCFG.

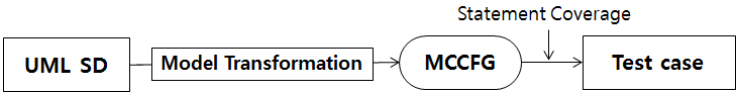

Fig. 2. MCCFG-based test case generation method

Figure 2 shows the overall process of the applied test case generation method. As shown in the figure 2, the elements of the control flow are extracted from the UML sequence diagram, and the model is transformed into an MCCFG that uses test coverage to generate test cases. Meta-models of the UML sequence diagram and MCCFG are defined in order to carry out model conversion. The model conversion rules are defined by means of a model transformation language. Finally, in the generation of test cases, the MCCFG meta-model and existing coverage [4] are used to generate test cases.

4 Case study

This section describes the process of generating test cases from the sequence diagram. Figure 3 shows the conversion of the MCCFG from the sequence diagram.

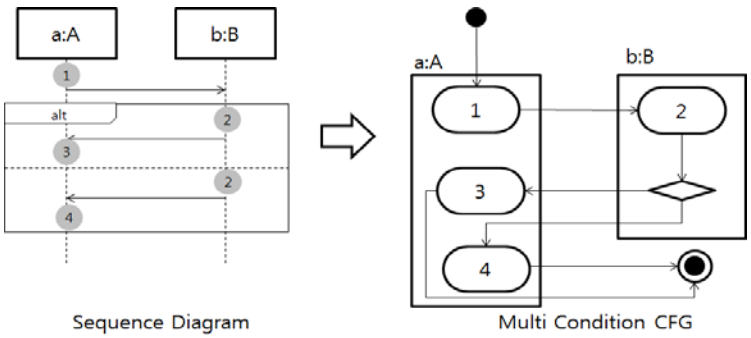

Fig. 3. Transformation of UML sequence diagram to MCCFG

Table 1 shows how the MCCFG in Figure 3 is used to apply the statement coverage and generate test cases. With the statement coverage, all of the syntaxes in the program code need to be run at least once. Because the test case generation method requires that all of the nodes in the tree be run at least once, paths that are already run are excluded, and a depth-first search is used to generate test cases.

Table 1. Test case of statement coverage

Testcase ID	*Inflow*	*Event*	*Condition*	*Outflow*
TC1	Node1	-	N/A	Node2
TC2	Node2	-	N/A	Node3
TC3	Node2	-	N/A	Node4

5 Conclusion

We propose an automatic test case generation method based on a sequence diagram. The proposed method generates test cases by using a multiple condition CFG, which is an extension of an existing CFG. The reason for using MCCFG in particular is to allow flexible application to diverse models and coverages. Certain applied cases are employed to generate test cases.

The proposed method has the advantage of being able to generate test cases of different intensities based on the priority of the test. This makes it possible to reduce the effort expended in testing and removes the need for complex models in test design; it can thus be possible to reduce the costs incurred in testing.

The creation and application of test case coverage for generating test cases as a means for improvement remains a subject for future study.

Acknowledgments. This research was supported by the MKE(The Ministry of Knowledge Economy), Korea, under the ITRC(Information Technology Research Center) support program supervised by the NIPA(National IT Industry Promotion Agency)(NIPA-2012-(H0301-12-3004)) and the Ministry of Education, Science Technology (MEST) and National Research Foundation of Korea(NRF) through the Human Resource Training Project for Regional Innovation.

References

1. Heungnam Kim, Seungmin Park, Doo Hyun Kim, “Current Technology Trends in Embedded Software”. Communications of the Korean Institute of Information Scientists and Engineers, Vol. 24, No. 8, pp. 5-11 (2006)
2. Wonil Kwon: The necessity of software testing in improving software quality. FKII Digital 365, pp. 66-69 (2008)
3. Eunyoung Lee, Miso Yoon, Byoungju Choi, “A Risk-based Test Case Priority Management System”. Journal of KIISE : Computing Practices and Letters, Vol. 18, No. 6, pp. 439-449 (2012)

4. John Joseph Chilenski and Steven P. Miller, “Applicability of modified condition decision coverage to software testing”. Software Engineering Journal, pp. 193-200 (1994)
5. Karolak, D.W., “Software Engineering Risk Management”. IEEE Computer Society Press, Silver Spring (1996)
6. Stale Amland, “Risk-based testing: Risk analysis fundamentals and metrics for software testing including a financial application case study”. The Journal of Systems and Software, Vol. 53, pp. 287-295 (2000)