A Code Profiling Model for StART

IOP Conf. Series: Materials Science and Engineering **769** (2020) 012057 doi:10.1088/1757-899X/769/1/012057

A Code Profiling Model for StART

Roslina Mohd Sidek

Faculty of Computing, College of Computing and Applied Sciences,
Universiti Malaysia Pahang, Lebuhraya Tun Razak,
26300 Kuantan Pahang, Malaysia

E-mail: roslinams@ump.edu.my

Abstract: Test case selection in software testing is one of the process that support to quality in finding bug. Decreasing number of bug may increase quality of software. Many strategies have been proposed by researcher for software testing. Specially to avoid exhausted testing which all the testers take into account which may increase time and performance. One of the strategy is StART. The code profiling is a part of StART, which is a selection test cases strategy using code profiling. The strategy is based on the adaptive random testing and support with code profiling in the strategy. The aim of this paper is to show the code profiling model. The selection of domain area to test is based on the highest probability in the code profiling result. The probability of in the code profiling is calculated and the highest value to be chosen as the first area to test. The value to be chosen as a domain area for the first to be tested in SUT. This model used AspectJ as case study to show the model can be implemented for a new programming paradigm. Tetris selected as case study to show the model flow. From the case study of Tetris, the model shows the area to be tested first is in the package Gui, for aspect menu and advice after the probability value shows the highest result. For future study, this model need to test their efficiency of their strategy.

Key words: code profiling, aspect, StART, model.

1. Introduction

There are many strategies in selection test case in testing phase such as random and systematic approach. The test cases are the essence of software testing which need clarification of information should be tested. Random approach [1], [2] selects the test cases without consider any information behind the software under test (SUT). Random testing is very simple and easy strategy to apply due to unbiased [3] in a way of selection normally happened from human mistake. This reason random testing is very popular testing strategy to be used in selection the test cases when to test software. Meanwhile, systematic approach [4], [5] will use certain strategy to get the best test cases to find the error or bug [6]. The information behind the software in systematic approach is considered to select which part that need to test to get potential error. Nevertheless, both approaches have similar objectives which are to avoid more effort to test all test cases known are known as exhausted testing.

Test cases is a set of conditions or a specification of the input, condition or variables under which a tester will determine whether a system under test satisfies requirements or works correctly. Test cases underlie

testing that is methodical rather than haphazard. A battery of test cases can be built to produce the desired coverage of the software being tested. Formally defined test cases allow the same tests to be run repeatedly against successive versions of the software, allowing for effective and consistent. In testing, to test all test cases is impossible. The problem is to test all combination of input is time consuming [7]. Such as, if test Internet Explore shows in [7] for Advance Option which have 53 condition produce 108,086,391,056,891,904 possible combinations of conditions.

Selection test cases using model [8] can be used random or systematic approach and even combination of both random and systematic. Actually, no standard for test case selection. Thus, selection is depending on strategy that tester try to apply during testing. Offering many strategies may support the testing process more option to be chosen and give more testing to find bug.

In this study, code profiling had been chosen as strategy that tester need to be considered. This proposed model uses AspectJ as a code to be tested. The code involved consists of core concern and crosscutting concern. The crosscutting concern is the area of study. Basically, source code can be many package, module and line of code such as [9] measuring the time spent on each line of code, code coverage or memory usage during its execution, [10] using probability calculation and focus on line of code, and package used. [11] gathered profiling information is structured in line with the application structure in terms of packages, classes, and methods. In this paper, the elementary of code is the advice. The advice consists of after, before and around.

Detail explanation for the model in code profile with the probability calculation organized according to the sections. Section 2 discuss about related work, section 3 research background, section 4 proposed model and section 5 is about discussion and section 6 conclusion and future work.

2. Related Work

In this study, code profile used to help in selection test case in software testing especially to avoid biasness [12],[10], [11]. In [12] , the author used the profiling in forecasting model selection and the accuracy. Performance has been assessed in terms of a model's ability to generate both unbiased and accurate forecasts, and accuracy has been examined using both error magnitude and directional change error criteria. In [10], the author generate code profile using statistical testing to increase the reliability of test case selection. [11] code profile generated to easily build profilers and visualize profiling information. It is also can be used to help in many areas such as to show usage model [13]. It provides the most cost- effective use of the testing budget and the only direct statistical method for making inferences concerning reliability of software in operational use.

3. Research Background

Test case selection in aspect-oriented domain still need more strategy especially in random approach. [14] researcher said the exclusive random testing may to cater this testing strategy. As a software tester to test all (known as exhausted testing) is impossible due to time limitation and expensive. That is the reason test cases to be selected wisely to be test to minimize the cost.

3.1 StART Strategy

StART is a strategy to select the test cases [10]. The strategy is implemented in aspect-oriented program domain area. So, the process model based on AspectJ code. The process code profiling involved with AspectJ code and focus on advice: after; before and around [10]. In this paper, model proposed only for code profiling in the strategy. StART also consider about the distance between test cases [15]. The whole strategy discusses in different research paper.

3.1.1 Code Profiling. This code profiling is referring to the consideration of element used in code. It has been discussed in [10]. If the code is object-oriented the classes and the method will take as code profile with using operational profile. Operational profile is to determine the execution frequencies of various thread and to use this information to select thread for system testing. One way to determine the operational profile of a system is to use decision tree. The decision tree shows the transition probabilities value. The

value for any state need is to be found or estimated of each going transition. The sum of these must be 1. The example of operational profile shown in [16].

3.1.2 Probability. There two type of probability theory in the study of software testing: the probability that a particular path of statements executes; and the other generalized to a popular industrial concept called operational profile [17]. It is a tool for the design and analysis of probabilistic and randomized algorithms.

The probability concept explain detail in [6]. It is defined as a sample space S, which is a set of elements called elementary events. The elementary event can be viewed as a possible outcome this study. The element in this study are packages, aspects and advices.

4. Proposed Models

According to the subject program as data set used throughout this paper, the main requirements are code profiling at the first step of the proposed method. This operation already discussed in [10].

Definition in this step are provided as follows: A: the number of aspect; a: the elements of A; F(a): the set of aspect in A. Given a program, let $a \epsilon A$ be an aspect. Denote that A(a) the set of aspect in A and $F(A_1) = \cup_{a \epsilon a_1} F(a), F(a_i)$ contain aspects which are in a_i. Suppose that P is a probability of aspect from 0 to 1; equivalent there is a distribution of aspect. Then the aspect size of a_1 is defined as

$$size\ (A_l) = \frac{\sum_{a \epsilon F(A_l)} P(a)}{\sum_{a \epsilon A} P(a)}$$

The highest of value will select as the first domain to be tested. Figure 1 shows the Code Profiling process model. Firstly, read the input source code. In this process, aspect is counted. Certain program has package. The package also need to be list down and counted. If the aspect found, then check whether there is advice or not. Thus, advice can be after or before or around. If more than one of each advice such as after. The name for after is *after 1, after 2* until *after n*. Same case to other advice. The process continues until finish the code. Then, probability calculation needed. In this calculation, line of code need to consider. In this study, tool used to ensure the number is correct. From this calculation the value for each of advices are determined.

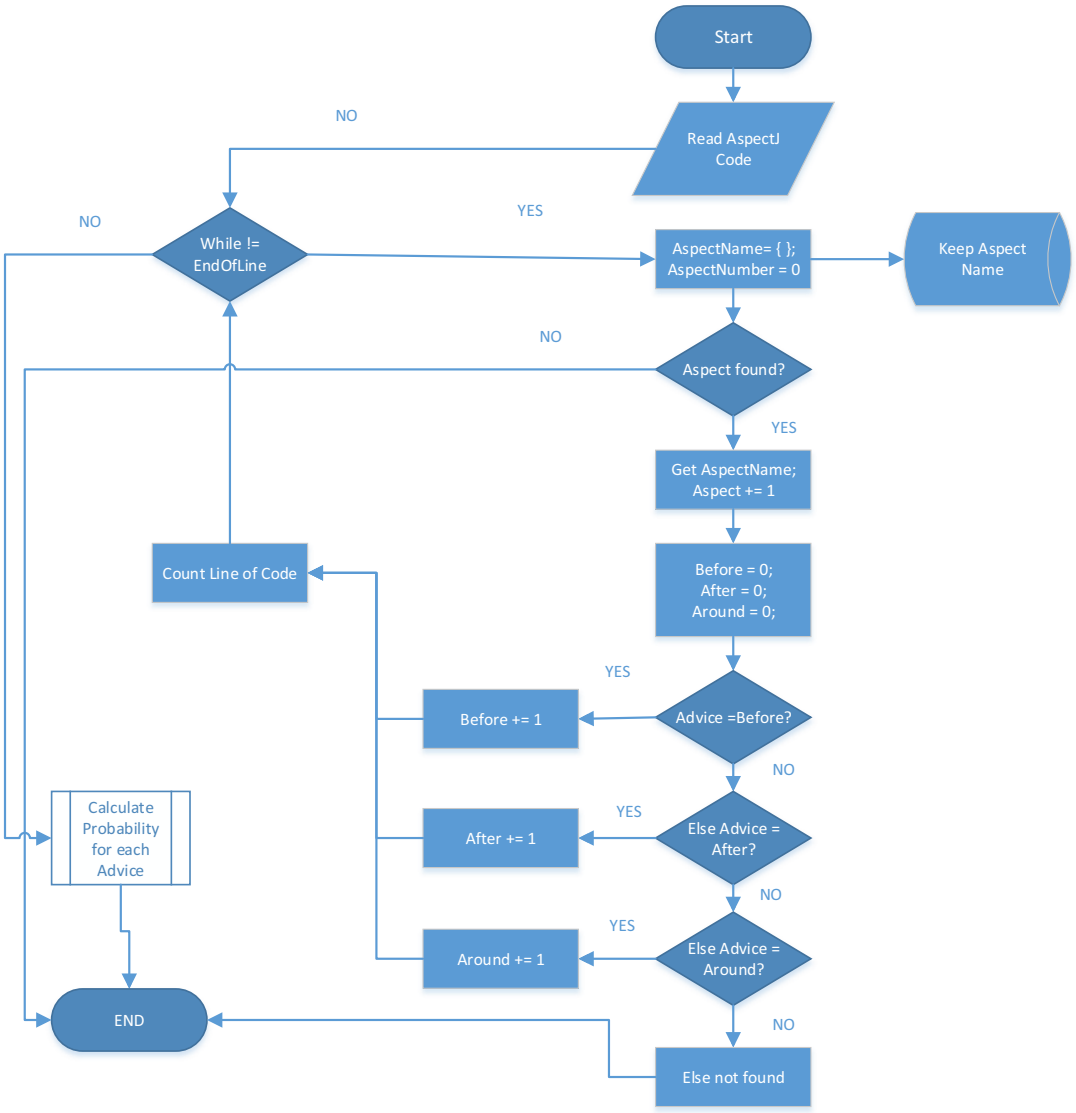

Fig. 1. Code Profiling Process Model.

A. Code profiling

According to the subject program as data set used throughout this paper, the code profiling is the first step of the proposed methods. Tetris code used as a sample in this research as case study. The Tetris code is a benchmark code from eclipse [18]. After analyzing the code using tool, there are three packages named: AspectLogic; Gui; and HighScore. For AspectLogic, two aspect involved: nextBlock with two advices; and newBlock with five advices. Package Gui, consists of two aspects: Levels with four advices; and Counters with 6 advices. Meanwhile, package HighScore consists of two aspects: Menu with two advices; and TestAspect with single advice. In Figure 2. shows the tree of Tetris code. This tree to show the profile of the code graphically. This method easy to calculate the probability for each element inside the code. The elementary of code can see clearly with package, aspect and advices.

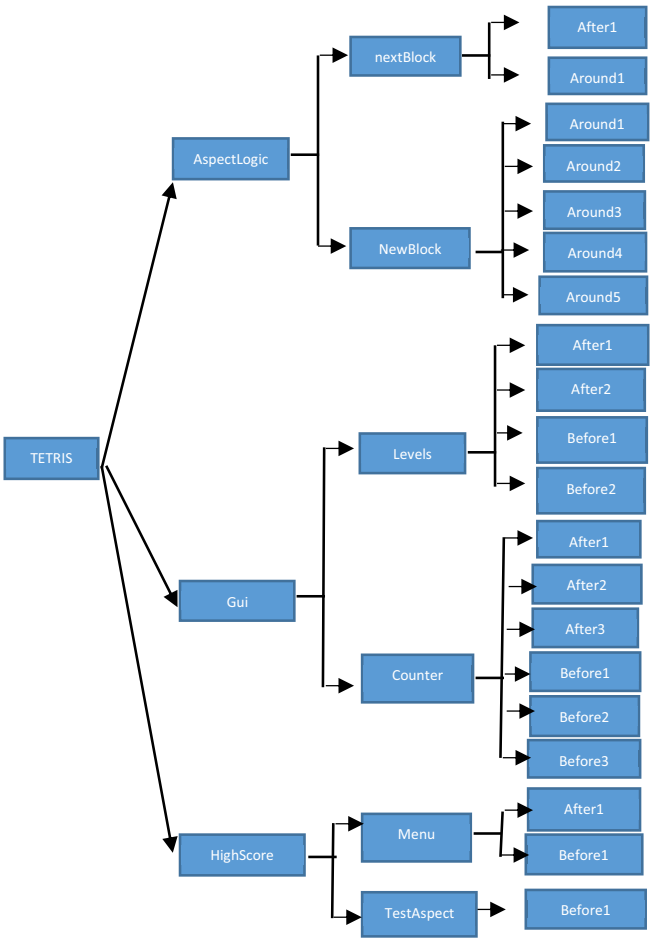

Fig 2. Tetris Tree.

B. Calculate the Probability

Then, probability calculation based on the profile analysis for the code. The calculation needed to get the highest value. The value will be used as the first domain to be tested in the software under test. The probability value shown in Table 1.

Table 1: Probability Value

CODE	PACKAGE	prob	ASPECT	prob	ADVICE	LOC	Prob(LOC)	Prob(a)	MAX
Tetris	aspectLogic	1/3	nextBlock	1/2	after	5	0.55556	0.09	0.141026
					around	4	0.44444	0.07	
			newBlock	1/2	around1	1	0.02222	0.00	
					around2	23	0.51111	0.09	
					around3	7	0.15556	0.03	
					around4	7	0.15556	0.03	
					around5	7	0.15556	0.03	
	highScore	1/3	levels	1/2	after	3	0.2	0.03	
					after	7	0.46667	0.08	
					before	2	0.13333	0.02	
					before	3	0.2	0.03	
			counter	1/2	after	3	0.25	0.04	
					before	1	0.08333	0.01	
					after	3	0.25	0.04	
					before	1	0.08333	0.01	
					before	3	0.25	0.04	
					after	1	0.08333	0.01	
	Gui	1/3	menu	1/2	after	11	0.84615	0.14	
					before	1	0.07692	0.01	
			testAspect	1/2	before	1	0.07692	0.01	

5. Discussion

One of the issues in code profiling is the elementary of profile. This work focus on advice part in AspectJ code. Advice is the part that execute the and can change the operation of code. The advice consists of after, before and around. In this strategy, object also is one of the factor will affected to the value of profile. Thus, calculation from the whole code that refer to the aspect should take into consideration for next study. The result in Figure below shows the highest is gui.menu.after with 17% from the aspect code. This code will be selected as the first area to be tested. If core concern or object involved in this profiling, the value may affect to the test cases selection.

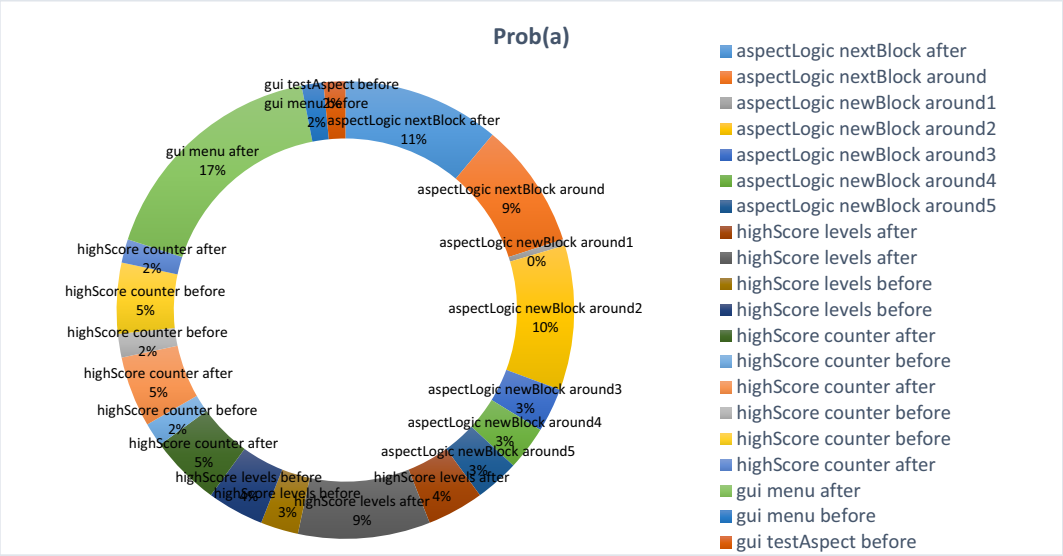

Fig 3. Percentage for aspect code in Tetris

6. Conclusion and Future Work

StART is a testing strategy for test case selection. In this strategy, information from source code is taken into account called code profiling. In this paper, shows the code profiling model support the tester to test which part the need to be test first. The model applies probability calculation to calculate elementary in AspectJ code profile.

Future work, the model can be applied to next phase of StART strategy and can be implemented to any real case study.

Acknowledgment

Appreciation conveyed to Universiti Malaysia Pahang Grant RDU170303 for financing this research.

References

[1] A. Arcuri, M. Z. Iqbal, and L. Briand, "Random Testing: Theoretical Results and Practical Implications," *IEEE Trans. Softw. Eng.*, vol. 38, no. 2, pp. 258–277, Mar. 2012.

[2] T. Y. Chen and D. H. Huang, "Adaptive Random Testing by Localization," in *11th Asia-Pacific Software Engineering Conference*, 2004, pp. 292–298.

[3] J. Mayer and C. Schneckenburger, "An empirical analysis and comparison of random testing techniques," in *Proceedings of the 2006 ACM/IEEE international symposium on International symposium on empirical software engineering - ISESE '06*, 2006, no. Empirical Software Engineering and Measurement, pp. 105–114.

[4] H. D. Mills, M. Dyer, and R. C. Linger, "Cleanroom Software Engineering Cleanroom Software Engineering," *IEEE Softw.*, 1987.

[5] S. Poulding and J. A. Clark, "Efficient Software Verification : Statistical Testing Using Automated Search," *IEEE Trans. Softw. Eng.*, vol. 36, no. 6, pp. 763–777, 2010.

[6] P. C. Jorgensen, *Software Testing: A Craftman's Approach*. 2013.

[7] "Exhaustive Testing: Definition, Principles, and Examples," *Resources Test Automation*, 2018. [Online]. Available: https://testautomationresources.com/software-testing-basics/exhaustive-testing-fundamentals/.

[8] S. Fujiwara, G. v. Bochmann, F. Khendek, M. Amalou, and A. Ghedamsi, "Test selection based on finite state models," *IEEE Trans. Softw. Eng.*, 1991.

[9] A. Rubio and F. Villar, "Code Profiling in R: A Review of Existing Methods and an Introduction to Package GUIProfiler," *R J.*, 2015.

[10] Roslina *et al.*, "A Code Profiling Using Statistical Testing in StART," *Adv. Sci. Lett.*, vol. 24, no. 10, pp. 7295–7299, 2018.

[11] A. Bergel, F. Ba??ados, R. Robbes, and D. R??thlisberger, "Spy: A flexible code profiling framework," *Comput. Lang. Syst. Struct.*, vol. 38, no. 1, pp. 16–28, 2012.

[12] S. F. Witt, H. Song, and P. Louvieris, "Statistical Testing in Forecasting Model Selection," *J. Travel Res.*, vol. 42, no. 2, pp. 151–158, Nov. 2003.

[13] gwendolyn h. Walton, J. h. Poore, and carmen j. Trammell, "Statistical Testing of Software Based on a Usage Model," *Softw. – Pract. Exp.*, vol. 25, no. January 1994, pp. 97–108, 1995.

[14] R. M. Parizi, A. Azim, and A. Ghani, "On the Preliminary Adaptive Random Testing of Aspect-Oriented Programs," in *ICSEA 2011: The Sixth Interntaional Conference on Software Engineering Advances*, 2011, no. c, pp. 49–57.

[15] M. S. Roslina, A. Azim, Abdul Ghani, S. Baharom, and H. Zulzalil, "A Preliminary Study of Adaptive Random Testing Techniques," *Int. J. Inf. Technol. Comput. Sci. (IJITCS)*, vol. 19, no. 1, pp. 116–127, 2015.

[16] J. D.Musa, "Operational Profiles in Software Reliabilitv Engineering," *IEEE Comput. Soc.*, vol. 10, no. 2, pp. 14–32, 1993.

[17] Paul C. Jorgensen, *Software Testing A Craftsman's Approach*. 2014.

[18] S. Apel, "How aspectj is used: An analysis of eleven AspectJ programs," *J. Object Technol.*, 2010.