Journal of Robotics, Networking and Artificial Life
Vol. **8**(**1**); June (2021), *pp.* 24–28
DOI: https://doi.org/10.2991/jrnal.k.210521.006; ISSN 2405-9021; eISSN 2352-6386
https://www.atlantis-press.com/journals/jrnal

Research Article

Measuring Test Suite Reusability Based on the Usage Frequency and Coverage of Reused Test Cases

Mochamad Chandra Saputra[1,*], Tetsuro Katayama[1], Yoshihiro Kita[2], Hisaaki Yamaba[1], Kentaro Aburada[1], Naonobu Okazaki[1]

[1]*Interdisciplinary Graduate School of Agriculture and Engineering, University of Miyazaki, 1-1 Gakuen-Kibanadai Nishi, Miyazaki 889-2192 Japan*
[2]*Department of Information Security, Faculty of Information Systems, Siebold Campus, University of Nagasaki, 1-1-1 Manabino, Nagayo-Cho, Nishi-Sonogi-gun, Nagasaki 851-2195, Japan*

ARTICLE INFO

Article History

Received 25 November 2020
Accepted 06 April 2021

Keywords

Test suite reusability
reused frequency
code coverage
reused test cases

ABSTRACT

Test suite reusability measurement is important to obtain the value of reusability as the degree of effectiveness of reused test suite. The measurement in this experiment considers not only the frequency of the successful test suite to examine different objects but also the code coverage as the criteria of a good test suite. The combination of the frequency and code coverage in the measurement reports the current condition of test suite reusability. The research confirms the test suite reusability measurement provides useful information to know the degree of effectiveness of reused test suite, especially in regression testing and automated testing.

1. INTRODUCTION

White-box testing knows as structural testing is one of the software testing approaches to test the program based on the internal structures of the program [1]. It is used to analyze internal logic and behavior, data structure, and code coverage. The important asset of software testing is the test suite that contains a set of test cases. On the white-box testing approach, the test suite is important to guarantee all independent paths or statements within a program executed at least once. Test cases in the test suite should execute all true or false logical decisions and all loops at their or within the boundaries and ensure the validity of internal data structures [2].

Currently, reusing assets in the software development process have a great purpose. A reusable test suite during the testing phase is decreasing testing time and cost [3]. The research is considered to code coverage has been conducted. The research on test cases reusability measurement considers to code coverage is reported by modifying the program with reducing branches, the test suite is highly reusable but, modifying the program such as splitting, and loop, the test suite is less reusable [4]. The research has shown test suite reusability measurement is needed to consider the code coverage.

The use of code clones is increased day by day due to the growth of the use of open-source software and variants [5]. Code clones are efficient in reducing the cost and time on software development that have similar requirements. Testing code clones is needed a strategy to achieve efficiency on it. Test suite reusability is one of the best strategies to test the code clones.

This research proposes the formula for the test suite reusability score to measure the degree of test suite reusability based on reused test cases on another program. The formula is considered to test suite reusability frequency and distinct of code coverage. The test suite reusability score measurement is including good information to evaluate the efficiency of the test suite on reusability.

The rest of the paper is organized as follows. Section 2 describes the principle of test suite reusability and code clones method. Section 3 describes the test suite reusability measurement considering frequency and code coverage of successful reused test cases in the test suite. Section 4 describes the experimental activity and its result. Section 5 describes the results and discussion of the research. Section 6 describes the conclusion and future work of the research.

2. TEST SUITE REUSABILITY AND CODE CLONES

The test suite contains a set of test cases that helps the software tester to examine the object of tested and reporting the test execution status. A test case is a set of test inputs, execution conditions, and expected results developed for a particular objective, such as to exercise a particular program path or to verify compliance with a specific requirement [6].

The principle of software reusability is defined as the capability of an attribute to be reused in various objects [7]. This research uses the terms of software reusability to interpret the context of test suite reusability. The test suite reusability defines as the capability of test cases in the test suite to examine several or all paths of method should be tested on diverse objects.

*Corresponding author. Email: chandra@earth.cs.miyazaki-u.ac.jp

The test suite reusability additionally needs to consider code coverage on the reusability measurement. One of the criteria of good test cases in the test suite related to white box testing is that the test cases can achieve 100% code coverage. The case of test suite reusability measurement on white box testing considers not only the capability of the test suite to examine different objects but also code coverage as the criteria of a good test suite.

The asset is defined as any kind of product from any part of the software process [8]. The core asset in software testing is the test suite. The test suite consists of the set of tests for a module in the program such as a class or method. Reusing assets on software testing will help to reduce the cost of software testing. The test suite is not very valuable if it is not reusable in testing another program. Test suite reusability measurement is important to obtain the value of reusability as the degree of effectiveness of reused test suite.

The test suite reuse can boost productivity at least as much as reuse of code and it is referred to similar features of a software system under testing [9]. Code clones are indicated as the highest opportunity to reuse the test cases in the test suite. The test suite reusability measurement is applied on code clones because the reusability of the test cases in the test suite needs to use the same characteristic. This research uses several types of code clone [10,11] such as code clone type 1 (exact clones) are identical clones and the second type is code clone type 2 which the differences from the original code are renamed identifiers, literals, types, layout, and comments but the structurally and syntactically are similar. The code clones type 3 are modified the statement such as statement insertions/deletions in addition to changes in identifiers, literals, types, and layouts. Code clone type 4 has been modified on code fragments to perform the same objective but different syntactic variants.

3. TEST SUITE REUSABILITY MEASUREMENT

The research on test suite reusability has been conducted such as function calling path based for test cases reuse method is determined the correlation function by investigating with manual analysis of the change path to be tested and the change of test case in test case reuse method is completed [12]. The method is testing the part of code that affected by the modification but has not measurement of the test suite reusability that related to the number of test cases reduction on regression testing, the workload of the tester, and the test cost.

The test suite which consists of the test cases is implemented on Junit testing. Figure 1 shows the test suite reusability measurement activity. The several information of test suite execution then uses for the test suite reusability measurement. The first information used on this measurement is the number of successful reused test cases in the test suite. The successful reused test cases are the examination result of the test cases on Java program that could achieve the objective of the testing without any error. The next information is distinct code coverage. The test cases in the test suite are possible to execute similar lines of code.

The similar lines of code executed by the test cases in the test suite then count as one line of code executed or called distinct code coverage.

To simplify the formula, the research uses the following notation.

- SRTC: Successful reused test cases
- DCC: Distinct code coverage
- OT: Objects tested
- OLOC: Original line of code
- TC: Test cases

By using the notation, the formula for test suite reusability score as follows.

$$\text{Test suite reusability score} = \frac{\sum \text{SRTC} + \sum \text{DCC}}{\left(\sum \text{OT} \times \sum \text{TC}\right) + \sum \text{OLOC}} \quad (1)$$

The proposed formula for test suite reusability score has considering frequency and code coverage of successful reused test cases in the test suite.

4. EXPERIMENT

The research uses the *parallelogram* Java program as shown in Figure 2 with two given test suites. The test suite reusability measurement on this experiment is excluding the result of the original test suite examination data. The experiment is focused on the

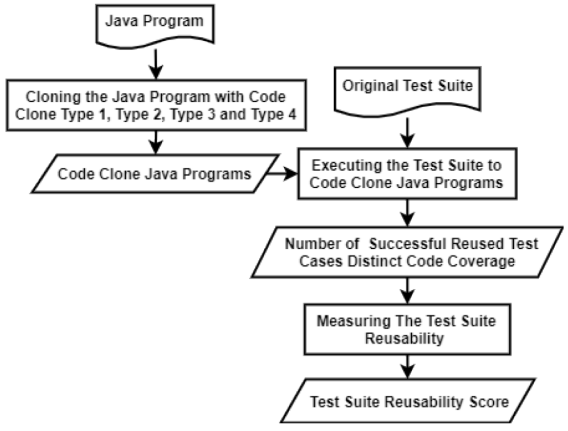

Figure 1 | Test suite reusability measurement activity.

```
package parallelogram;
import java.io.BufferedReader;
import java.io.InputStreamReader;
public class Parallelogram {
  public static void main(String[] args) {
    double base,height;
    BufferedReader br = new BufferedReader(new
InputStreamReader(System.in));
    System.out.println("This program calculate value the
area of a trapezium");
    System.out.println("Enter the base and the height of
the parallelogram");
    try{
     base=Double.parseDouble(br.readLine());
     height=Double.parseDouble(br.readLine());
      }catch (Exception e){
        System.out.println("An error occurred");
        return;
        }
       if(base<=0 || height<=0){
        System.out.println("Wrong Input");
        return;}
      System.out.println("Area = " + base*height );
    }
}
```

Figure 2 | Parallelogram source code.

test suite reusability measurement on another Java program that is reused the test suite to test the program. The test suite examines the code clones to collect the information of the number of successful reused test cases and distinct code coverage.

This research has two given test suites. The first test suite contains three test cases and the second test suite contains four test cases. The original Java program is cloning by code clones type 1–4 approach.

5. RESULTS AND DISCUSSION

The parallelogram java program is cloned by using code clones type 1–4. The given test suite is examined to collect information such as the lines of code executed by the test cases in the test suite and the number of successful reused test cases. The result of the examination of the test suite is shown in Table 1. Table 1 shows that the given test cases on the test suite successfully examine the code clones without any error.

Table 1 | The result from code coverage information on code clones

	Test suite-1				Test suite-2				
No. LOC	**TC-1**	**TC-2**	**TC-3**	**Distinct code coverage**	**TC-1**	**TC-2**	**TC-3**	**TC-4**	**Distinct code coverage**
Code clone type-1									
1	1	1	1	1	1	1	1	1	1
2	1	1	1	1	1	1	1	1	1
3	1	1	1	1	1	1	1	1	1
4	1	1	1	1	1	1	1	1	1
5	1	1	0	1	1	1	1	1	1
6	1	1	0	1	1	1	1	1	1
7	0	0	1	1	0	0	0	0	0
8	0	0	1	1	0	0	0	0	0
9	0	0	1	1	0	0	0	0	0
10	1	1	1	1	1	1	1	1	1
11	1	1	0	1	1	1	1	1	1
12	0	1	0	1	1	1	1	0	1
13	0	1	0	1	1	1	1	0	1
14	1	0	0	1	0	0	0	1	1
15	1	0	0	1	0	0	0	1	1
Code clone type-2									
1	1	1	1	1	1	1	1	1	1
2	1	1	1	1	1	1	1	1	1
3	1	1	1	1	1	1	1	1	1
4	1	1	1	1	1	1	1	1	1
5	1	1	0	1	1	1	1	1	1
6	1	1	0	1	1	1	1	1	1
7	0	0	1	1	0	0	0	0	0
8	0	0	1	1	0	0	0	0	0
9	0	0	1	1	0	0	0	0	0
10	1	1	1	1	1	1	1	1	1
11	1	1	0	1	1	1	1	1	1
12	0	1	0	1	1	1	1	0	1
13	0	1	0	1	1	1	1	0	1
14	1	0	0	1	0	0	0	1	1
15	1	0	0	1	0	0	0	1	1
Code clone type-3									
1	1	1	1	1	1	1	1	1	1
2	1	1	1	1	1	1	1	1	1
3	1	1	1	1	1	1	1	1	1
4	1	1	1	1	1	1	1	1	1
5	1	1	0	1	1	1	1	1	1
6	1	1	0	1	1	1	1	1	1
7	0	0	1	1	0	0	0	0	0
8	0	0	1	1	0	0	0	0	0
9	0	0	1	1	0	0	0	0	0
10	0	0	1	1	0	0	0	0	0
11	1	1	1	1	1	1	1	1	1
12	1	1	0	1	1	1	1	1	1
13	0	1	0	1	1	1	1	0	1
14	0	1	0	1	1	1	1	0	1
15	1	0	0	1	0	0	0	1	1
16	1	0	0	1	0	0	0	1	1

(*Continued*)

Table 1 | The result from code coverage information on code clones—*Continued*

	Test suite-1				Test suite-2				
No. LOC	**TC-1**	**TC-2**	**TC-3**	**Distinct code coverage**	**TC-1**	**TC-2**	**TC-3**	**TC-4**	**Distinct code coverage**
Code clone type-4									
1	1	1	1	1	1	1	1	1	1
2	1	1	1	1	1	1	1	1	1
3	1	1	0	1	1	1	1	1	1
4	1	1	0	1	1	1	1	1	1
5	1	1	0	1	1	1	1	1	1
6	1	1	0	1	1	1	1	1	1
7	1	1	0	1	1	1	1	1	1

Table 2 | Result of the number of successful reused test cases

	Code clones	Number of test cases	Number of successful test cases tested
Test suite-1	Type-1	3	3
	Type-2		3
	Type-3		3
	Type-4		3
Test suite-2	Type-1	4	4
	Type-2		4
	Type-3		4
	Type-4		4

Table 3 | Result of test suite reusability measurement

	Code clones	Test suite reusability score	Average
Test suite-1	Type-1	100%	100%
	Type-2	100%	
	Type-3	100%	
	Type-4	100%	
Test suite-2	Type-1	84%	87%
	Type-2	84%	
	Type-3	80%	
	Type-4	100%	

The unsuccessful test case is represented by the value 0 on all lines of code and in this experiment there is no unsuccessful test case.

The result of code coverage information is shown in Table 1. The value 1 means this line is executed and 0 is not executed. Distinct code coverage information is shown in Table 1. The value 1 means this line is executed and 0 is not executed during the testing by using one or more test cases. The distinct code coverage on code clones shows that several lines of code on code clones type 1–3 are not executed by test cases on the test suites during the testing indicated by value 0.

The result of the number of successful reused test cases is shown in Table 2. The result obtains from test cases examination on code clones. Table 3 shows the result of test suite reusability measurement uses formula (1). The test suite reusability measurement is very common information to reduce the cost of software testing. The different number of lines of code is possible for code clones type 3 and 4, for code clones type 1 and 2 usually the number lines of code are the same. The distinct code coverage is used to selecting the redundant line of code executed by the test cases on the test suite. By using the distinct code coverage, the calculation of percentage code coverage is represented the actual condition.

The test cases in the test suite are possible to execute similar lines of code. The number of successful reused test cases on the test suite as shown in Table 2 is important for test suite reusability measurement. The number of successfully reused test cases is one of the important parameters for the test suite reusability measurement.

The information of code coverages and number of successful reused test cases in the test suite is used for test suite reusability measurement. The result of test suite reusability measurement uses formula (1) as shown in Table 3. The result of test suite reusability score shows the different scores for several test suites because they have a different number of distinct code coverage by the test cases as shown in Table 1. The number of distinct code coverage is important to ensure the capability of the test suite to achieve 100% code coverage. The score of test suite reusability 100% means the test suite has the perfect capability to achieve 100% code coverage and reused on another program. The test suite reusability measurement provides useful information to know the degree of effectiveness of reused test suite, especially in regression testing and automated testing.

6. CONCLUSION

This research confirms the test suite reusability measurement is calculated by combining the number of successful reused test cases and code coverage. Code coverage is enriching the information on test suite reusability measurement. The result of test suite reusability measurement is valuable information to reduce the cost of software testing, especially in regression testing and automated testing.

Our future works will focus on other measurement for test suite quality measurement.

CONFLICTS OF INTEREST

The authors declare they have no conflicts of interest.

REFERENCES

[1] I. Sommerville, Software engineering 9th ed. in: M. Hirsch (Ed.), Software engineering, Addison-Wesley Publishing Company, USA, 2010, pp. 133–170.
[2] R.S. Pressman, B.R. Maxim, Software engineering: a practitioner's approach, McGraw-Hill, NY, USA, 2009.
[3] A.G. Sutcliffe, The domain theory: patterns for knowledge and software reuse, Lawrence Erlbaum Associates, Inc., Publishers, Mahwah, NJ, 2002.

[4] Y. Dong, Y. Wang, M.F. Lau, Sy. Lin, Experiments on test case reuse of test coverage criteria, 2010 7th International Conference on Ubiquitous Intelligence & Computing and 7th International Conference on Autonomic & Trusted Computing, IEEE, Xi'an, China, 2010, pp. 277–281.
[5] N. Saini, S. Singh, Suman, Code clones: detection and management. Procedia Comput. Sci. 132 (2018), 718–727.
[6] The Institute of Electrical and Electronics Engineers (IEEE), IEEE standard computer dictionary: a compilation of IEEE standard computer glossaries, IEEE, New York, 1990, pp. 1–217.
[7] S. Younoussi, O. Roudies, All about software reusability: a systematic literature review. J. Theor. Appl. Inf. Technol. 76 (2015), 64–75.
[8] M. Ezran, M. Morisio, C. Tully, Practical software reuse, Springer, London, 2002.
[9] S.P.R. Asaithambi, S. Jarzabek, Towards test case reuse: a study of redundancies in android platform test libraries, in: J. Favaro, M. Morisio (Eds.), Safe and Secure Software Reuse, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 7925, Springer, Berlin, Heidelberg, 2013, pp. 49–64.
[10] S. Bellon, R. Koschke, G. Antoniol, J. Krinke, E. Merlo, Comparison and evaluation of clone detection tools, IEEE Trans. Softw. Eng. 33 (2007), 577–591.
[11] C.K. Roy, J.R. Cordy, A survey on software clone detection research, Technical Report No. 2007-541, School of computing, Queens University, Kingston, ON, Canada, 2007.
[12] Y. Mu, X. Gao, M. Shen, Research of reuse technology of test case based on function calling path, Chinese J. Electron. 27 (2018), 768–775.

AUTHORS INTRODUCTION

Mr. Mochamad Chandra Saputra

He received the Master's degree from the University of Miyazaki, Japan, and Brawijaya University, Indonesia on Double Degree Program on 2014. Since 2015, he has been a lecturer on the Faculty of Computer Science, Brawijaya University. Currently, he is persuing the Doctoral Study at the University of Miyazaki. His research interests include software testing, software quality, and software project management.

Mr. Tetsuro Katayama

He received the PhD degree in engineering from Kyushu University, Fukuoka, Japan in 1996. From 1996 to 2000, he has been a Research Associate at the Graduate School of Information Science, Nara Institute of Science and Technology, Japan. Since 2000, he has been an Associate Professor at Faculty of Engineering, Miyazaki University, Japan. He is currently a Professor with the Faculty of Engineering, University of Miyazaki, Japan. His research interests include software testing and quality. He is a member of the IPSJ, IEICE, and JSSST.

Mr. Yoshihiro Kita

He received a PhD degree in systems engineering from the University of Miyazaki, Japan in 2011. He is currently an Associate Professor with the Faculty of Information Systems, University of Nagasaki, Japan. His research interests include software testing and biometrics authentication.

Mr. Kentaro Aburada

He received the B.S., M.S., and PhD degrees in computer science and system engineering from the University of Miyazaki, Japan in 2003, 2005, and 2009, respectively. He is currently an Associate Professor with the Faculty of Engineering, University of Miyazaki, Japan. His research interests include computer network and security. He is a member of IPSJ and IEICE.

Mr. Hisaaki Yamaba

He received the B.S. and M.S. degrees in chemical engineering from the Tokyo Institute of Technology, Japan in 1988 and 1990, respectively, and the PhD degree in systems engineering from the University of Miyazaki, Japan in 2011. He is currently an Assistant Professor with the Faculty of Engineering, University of Miyazaki, Japan. His research interests include network security and user authentication. He is a member of SICE and SCEJ.

Mr. Naonobu Okazaki

He received his B.S., M.S., and PhD degrees in electrical and communication engineering from Tohoku University, Japan in 1986, 1988, and 1992, respectively. He joined the Information Technology Research and Development Center, Mitsubishi Electric Corporation in 1991. He is currently a Professor with the Faculty of Engineering, University of Miyazaki since 2002. His research interests include mobile network and network security. He is a member of IPSJ, IEICE, and IEEE.