Testing of A Grade Calculation Program: A Case Study of Metamorphic Testing

Arlinta Christy Barus1), Rich Simamora2), Hanna Tobing3), Sumiati Hutagalung4)

Politeknik Informatika Del

Desa Sitoluama, Laguboti, Kabupaten Tobasa 22381, Sumut, Indonesia

{1)arlinta3),hanna, 4)sumiati}@.del.ac.id, 2)if09015@students.del.ac.id

Abstract—Metamorphic Testing (MT) is a testing method that involves input, output, and relation between the input and output of a program under test that generated by relationships of the program known as called metamorphic Relations (MRs). The inputs are used to generate new inputs and to be checked whether it is appropriate with the functional relationships of the relevant MR. The application that is used as a case study in this paper is a program to calculate grades of students as a part of Academic Information System of Del Polytechnic of Informatics. After testing the application, it is proven that MT was able to reveal failures. If failure of case study is revealed, the oracle problem of case study is reduced. MRs that are used in this paper are considered to be still simple. Therefore, the authors will conduct more study to generate more complex MR to reveal more failures.

***Keywords*—Software Testing, Metamorphic Testing (MT), Oracle problem, Metamorphic Relation (MR)**

I. INTRODUCTION

There are two problems in software testing, reliable test set problem and oracle problem [1]. Reliable test set problem is a problem where the tester never be able to find a subset of the input domain that can represent the whole input domain except the whole input domain itself. In this paper, we will focus on the oracle problem. An “oracle” in software testing is a procedure by which testers can decide whether the output of the program under testing is correct. [2]. In some situations, the oracle is not available or too difficult to apply. This is known as the "oracle problem" [2]. That situation may be caused by the complexity of the software to be tested or the number of input domains are too large to be tested so it is not possible to test all possible inputs (input domain). Input domain is a range of input fields of software under test.

The oracle, generally, is not available in the program that implements a mathematical function. The mathematical functions are normally implemented by complex algorithms so it is difficult to verify the output manually and hard to find the tools that can check the truth of the output program. The difficulty is known as the “oracle problem”.

Metamorphic testing (MT) is a testing method developed by Chen et al to alleviate the oracle problem [3]. In MT involved variety of inputs and outputs that can be used as a reference. Relation in MT is called metamorphic relations (MR) [4]. By making use the information from the executed test cases, MR is used to generate the follow-up test cases and to validate the compute outputs of the software [4].

In this paper, the application that is used as a case study to applying the MT is a program to calculate grades of students as a part of Academic Information System of Del

Polytechnic of Informatics The functions contained in this case study involve a computational process that needs to be proven whether the results is true or not. Checking the results of computing manually will be too difficult and inefficient. Therefore, the author will use the MT for reducing the oracle problem. The use of MT is done by generating a lot of MR to strengthen testing. MR will be applied on a small program that will be used to test case study directly.

This paper is presented in the following sections: a literature review, a description of case studies that will be reviewed, analysis, experimental design, results and discussion and conclusion.

II. LITERATURE STUDY

There are two problems in software testing, Reliable Test Set Problems and Oracle Problem [1]. Reliable Test Set Problem is a problem when the tester never found a subset of the input domain that can present domain input except the input domain. For example: P is a program. D is a domain input (all possible input values for P). After testing P using all input D and no failure is found, then P is a correct program. We need to find a reliable test set T with T is a subset of D, and if P (T) is true, then P (D) is correct. Problem: T is unlikely to be achieved unless T = D, known as the Reliable Test Set problem.

Oracle problem is a problem when the tester difficult to determine whether the output of a program is correct or not. The problem is caused by the complexity of the input domain to be tested. Input domains are all inputs that may be used for a program. Test oracle is a mechanism by the tester to determine if there is a failure in a program or not, by comparing the expected output with the actual output [9]. One of the software testing techniques that try to solve problems by applying a test oracle is metamorphic testing (MT). Oracle Problem is a problem that will be discussed in this paper.

A. EXAMPLE OF ORACLE TEST PROBLEM

One of oracle problem occurs in the search engines. An example occurs in Google with the address www.google.com. On the Google page, enter input 'ACM Transactions'. In accordance with the existing specifications on Google, spaced typed recognized as relations 'AND'. The system will return about 8,660,000 results for these searches. How can testers determine the correct output of very much result according to the words the user entered? Suppose again, a user inputs ‘ACM Transactions “World Wide Web”’. According to specifications of Google, quotation marks ("") means the search will return results that match with words in quotation without split. System will return about 834.000 results. How can testers determine that just 834.000 pages are relevant to the inputs? It is known as the oracle problem [11].

B. METAMORPHIC TESTING

Metamorphic testing (MT) is a testing method developed by Chen to alleviate the oracle problem [3]. Metamorphic testing is a technique to generate follow-up test cases based on existing test cases (test case source) that have not reveal any failure. Follow-up test cases generated by making reference to "metamorphic relations" (MR) [2]. Metamorphic Testing verify metamorphic relation of the functions that contained in the program that being tested [10].

By the concept of Metamorphic Relation have some characteristics below. First, MR is an important property of the function to be tested. If one of the properties of functions is not satisfied during the test, the program

called p can be said as the wrong program. Second, a metamorphic relation is a relationship between the source input with the follow-up input and source output with the follow-up output of a function that the relationship is going to be tested. If the relationship is not satisfied, it can be said that the tested function contained an error. For example, the sine function has a property that can be used for some inputs such as x1 and x2, if $x1 + x2 = \pi$, then sin (x1) = sin (x2). x1 is the source input and x2 is the follow-up input. If the sine function is run with x1 and x2 as input and found that the output is not equal, it can be said that the function is contain errors [10].

III. CASE STUDY

In this paper, the real application that will be used as a case study to applying the MT is a program to calculate grades of students as a part of Academic Information System of Del Polytechnic of Informatics. The Academic Information System of Del Polytechnic of Informatics is a system used to display academic information in Del Polytechnic Informatics. In addition, the system also has a feature to count the grade of all students. The program to calculate grades of students as a part of Academic Information System of Del Polytechnic of Informatics has a function that process and display the data related to all grades of students. Processing student's grade involves many calculation processes so it is difficult to check the results of the processing manually, especially for a lot number and variety of data. It is not efficient that the tester must check the output one by one. This is the oracle problem of this case study.

IV. DESCRIPTION OF THE FUNCTIONS TESTED

This part explains the analysis of the functions on case studies.

- AddKomposisi Function (component, weights)

This function is used to insert the components and weights of each course by teacher of each course.

- AddRentangNilai Function (IndeksNilai)

This function aims to provide an grade index for each subject.

- AddNilai Function (Nim, GradeValue)

This function aims to put the grade value by students NIM (student's ID) into each component of certain subjects.

- HitungTotalNlai Function (KODE_MK)

This function aims to calculate the final grade value of each component on the subject by multiplying weight with the grade value per component.

- GenerateGrade Function (FinalGradeValue)

This function aims to generate the grade of each course based on the final grade of the course.

- HitungNilaiRata Function (NIM, SEM)

This function aims to calculate IP (performance index). The output of this function is IP of students.

- Function hitungIPK (NIM, TotalSem)

This function aims to calculate the GPA (Grade Point Average) of students.

- SelisihTertinggi Function (IP2, IP1)

This function aims to generate the highest difference of IP even and odd semesters. The output is the student's name that has the highest difference of IP even and odd semester.

- GenerateCumlaude Function (IP)

This function aims to generate student data were obtained the cum laude. The output is the name of the student who obtained the cum laude.

- GenerateBeasiswa Function (IP)

This function aims to generate student's data that receive scholarships.

By all of above functions that used for calculating the student's grade, the chosen function to be tested is a function that has

complex computing process, there are: HitungNilaiRata function (NIM, SEM) as Fungsi1, HitungTotalNlai (KODE_MK) Function as fungsi2, Function hitungIPK (NIM, TotalSem) as Fungsi3

A. METAMORPHIC RELATION

This part describes the metamorphic relations that generated for the case study. Here is the explanation of each MR that forms by the relationship between the features of grade calculation program.

- 1st Function - hitungNilaiRata (NIM, SEM)

1. MRSik1 = "if gradeSource> gradeFollow on SKScourseSource = SKScourseFollow then IPSource> IPFollow". The gradeSource is variable to source input and gradeFollow is variable grade to follow-up input.
   Constraint: the same grade for the other courses.
2. MRSik2 = "if gradeSource <gradeFollow on SKScourseSource> SKScourseFollow then IPSource <IPFollow".
   Constraint: the same grade for the other courses.
3. MRSik3 = "if gradeSource <gradeFollow on SKScourseSource = SKScourseFollow then IPSource <IPFollow".
   Constraint: the same grade for the other courses.

- 2nd Function - hitungTotalNlai (KODE_MK)

The MR that used to test this function is the relation between the components, weight per component, and the grade value of a component that will affect the final grade per subject. MR that generated for this function is:

1. MRSik5 = "if gradeSource > gradeFollow on componentWeightSource = componentWeightFollow, then FinalGradeSource > FinalGradeFollow".
   Constraint: the same grades for other components and subjects to be tested must have at least two components with different weights.
2. MRSik6 = "if gradeSource < gradeFollow on componentWeightSource > componentWeightFollow, then FinalGradeSource < FinalGradeFollow".
   Constraint: the same grades for other components and subjects to be tested must have at least two components with different weights.
3. MRSik7 = "if gradeSource> gradeFollow on componentWeightSource < componentWeightFollow, then FinalGradeSource > FinalGradeFollow".
   Constraint: the same grades for other components and subjects to be tested must have at least two components with different weights.

- 3rd Function - HitungIPK (NIM, TotalSem)

The MR that used to test this function is the relation between the IP and the number of credits that will affect GPA. MR that resulted from this relation is:

1. MRSik8 = "if IPSource> IPFollow on totalSKSSemesterSource = totalSKSSemesterFollow then IPKSource > IPKFollow".
   Constraint: the same IP for the other one and have more than one semester.
2. MRSik9 = "if IPSource > IPFollow on totalSKSSemesterSource > totalSKSSemesterFollow then IPKSource > IPKFollow"
   Constraint: the same IP for the other one and have more than one semester but the number of credits is not same.

3. MRSik10 = "if IPSource > IPFollow on totalSKSSemesterSource < totalSKSSemesterFollow then IPKSource <IPKFollow".
   Constraint: the same IP for the other one and have more than one semester but the number of credits is not same.

V DESIGN AND EXPERIMENTS

A. MR IMPLEMENTATION

The testing for Fitur Penghitung Nilai of the Academic Information System of Del Polytechnic of Informatics does not fully automatically.

This is because Fitur Penghitung Nilai involves two different applications. The first application is a small program that was built with the programming language PHP. This small program is used to test the relation between input and output source with input and output follow-up whether it meets the MR. The second application is Microsoft Office Excel package, used to generate the random input and then made them in to CSV form file. Inputs that have been raised are used to test Fitur Penghitung Nilai to be able to produce output with a particular function.

B. MUTANT GENERATION

Mutant is a program that is seeded an error intentionally. Process of giving an error on the original program should be able to insert automatically through mutant operator. Selection of mutant operator is done randomly and independently [13]. However, due to time limitations in this paper the addition of mutant performed semi-automatically. Selection of mutants made randomly / done by the programs, but implementation at the program under test is done manually.

In this paper, the mutant is adding to Fitur Penghitung Nilai. Giving mutants Fitur Penghitung Nilai was done because in his original program testing, failure was not found. Mutants that are used to test Fitur Penghitung Nilai were added to the function to be tested. Every function has several mutants with specific versions.

Types of mutations using in this paper is the operator mutation. Operator mutation means the addition of the mutant to the original program by changing the existing operators of the program. Types of operators that will be modified on this TA is the arithmetic operator like "+", "-", "*" and "/". Each operator will be replace by "+", "-", "*", or "/".

If there are several operators in the function to be tested, each operator will be replaced with very operator in his own range. The selection operator which will be used to the program is based on the results of random mutant operators. Mutant operator is a small program / console application built using Java programming language. Outputs of the application are the operator will be changed and what will change it.

The author record the results then applied in the program. Each function to be tested was given five different mutants. Each mutant only changes one operator.

C. METRIC

Metric is a measure used to measure off the success of an experiment. The expected result in software testing is the ability to reveal the failure. In MT, the success reveal failure occurs when the output source test cases and the output follow-up test case does not meet the MR relation. In this case, it can be concluded that the experiment was successfully.

When failure is not revealed, there are two possible causes. Firstly MR is ineffective to uncover failure (too simple) or the application under test is already quite good (free of failure). In this condition, the next step is proving the possibility by adding mutant to the program.

Program mutants tested again using the same relation to the MR. If the failure successfully revealed it can be concluded that the actual MR is effective to reveal the failure. No revealing failure in the experiment because the application is the correct one. But when failure is not successfully revealed by the mutant program, it proves that MR is not effective to expose the failure.

VI RESULTS AND DISCUSSION

Tests were implemented on Fitur Penghitung Nilai of the Academic Information System of Del Polytechnic of Informatics performed on mutant program. This is because the failure in the original program is not reveal, so insert the mutant is needed. The test results of the application using MR in every function can be found in [12].

Experiments to the third function hitungIPK were done by inserting the code for MR8 only. In Chapter III, there are three MR they are MR8, MR9 and MR10 as implementation of this function. However, the MR9 and MR10 cannot be inserted mutants because there are code errors on the program so the GPA for more than one semester cannot be correctly calculated using Fitur Penghitung Nilai. This is because IP does not affect GPA.

A ANALYSIS OF TEST RESULTS

This section describes the analysis of the results of experiments conducted to the case study. An analysis of the experiments result performed using XLSTAT tools. Tools are used to compute analysis of variance (analysis of variance) and analysis of the differences between the categories by using the 95% confidence interval. The value of the confidence interval is a confidence value which is used in XLSTAT.

The analysis conducted on the MR function aims to find the most suitable among all MR for use in testing the function. The analysis conducted on the application aims to find the most appropriate MR among all the MR.

The analysis of this case study is divided into two types based on functionality and MR analysis of the overall application. After doing the calculations using XLSTAT the results of calculations using ANOVA and grouping categories confidence interval can be seen in the description below.

B TEKNIK ANALISIS ANOVA

ANOVA calculations performed for functions that have more than one MR in order to analyze the average - the average population of the different groups who have different treatments - also vary based on experiments that have been done before.

Therefore, technical analysis cannot be performed on HitungIPK() function because MR is used only one so that the data of the experimental results are homogeneous or similar.

Grouping Categories Based on 95% Confidence Intervals

Grouping Categories is done to get the category for each MR of every function and overall functions on a case study based on experiments that have been done before. Grouping can be seen in [12].

VII CONCLUSIONS AND RECOMMENDATIONS

There are the conclusions obtained for this paper.

After testing in both case studies and see the results of experiments, it turns out that MT method was able to generate MR that can be applied to reveal the failure.

The small program that is used to assist the experiments is able to applying MR to reveal failure.

When failure is not revealed in the experiments, there are two causes that occur, they are MR is used to expose the failure is not effective (too simple) or the application under test is already quite good (free of failure). In this condition, the next step to do to prove it is adding mutant into the program.

After doing experiments and failure revealed in the case studies the oracle problem in the case study is reduced. MR is used in this paper is a simple MR. Therefore, the authors hope that further development can generate more complex MR so the failure that revealed more and more.

REFERENCES

[1] Barus Arlinta Christy, "An In-depth Studi of Adaptive Random Testing for Testing Program with Complex Input Types", Ph.D. dissertation, Swinburne University of Technology, 2010.

[2] Zhou Zhi Quan, Huang D. H., Tse. T. H., Yang Zongyuan, Huang Haitao, dan Chen T. Y., " Metamorphic Testing and Its Applications" , 2004.

[3] T. Chen, S. Cheung, and S. Yiu. Metamorphic testing: a new approach for generating text test cases. Technical Report HKUST-CS98-01, Department of Computer Science, Hong Kong University of Science and Technology, Hong Kong, 1998.

[4] Barus Arlinta Christy, T. Y. Chen, D. Grant, F.C Kuo and M.F. Lau " Testing of heuristic methods: A case study of greedy algorithm", in In Proceedings of the 3rd IFIP CEE Conference on Software Engineering Techniques(CEE-SET 2008).

[5] Hetzel, W. (1973) Program Testing Method, Prentice-Hall.

[6] IEEE(1983) IEEE standard for software testing documentation, IEEE Std 829-1983.

[7] IEEE(1990) IEEE standard for software testing documentation, IEEE Std 610.12-1990.

[8] Pfleeger, S.L.(2001) Software Engineering: Theory and Practice, 2nd ed. Prentice-Hall.

[9] "Software Testing", diakses pada tanggal 09 Januari 2012 dari http://en.wikipedia.org/wiki/Software_testing.

[10] Chen, Tsong Yueh, Ying Liu, dan Antony Tang, Metamophic Testing and Testing with Special Values, Swinburne University of Technology, Australia.

[11] Zhou Zhi Quan, Tse T. H., Kuo F. –C., Chen T.Y., "Automated Functional Testing of Web Search Engine in the Absence of an Oracle", 2007.

[12] Rich, Hanna, Sumiati, "Penerapan Metamorphic Testing (MT) Studi Kasus: Fitur Penghitung Nilai Pada Sistem Informasi Akademik Politeknik Informatika Del dan Aplikasi Tebak Kendaraan", Laporan Tugas Askhir Politeknik Informatika Del, 2012

[13] Agrawal, H., DeMillo, R.A., Hathaway, R., Hsu, W., Hsu, W., Krauser, E.W., Martin, R.J., Mathur, A.P., Spafford, E.H. : Design of mutant operators for the C programming language. Technical Report SERC-TR-41-P, Software Engineering Research Center, Purdue University, West Lafayette, Indiana, USA (March 1989).