VOL. 10, NO. 3, FEBRUARY 2015 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences


www.arpnjournals.com

MATRIX MULTIPLICATION PROGRAM: A CASE STUDY OF METAMORPHIC TESTING

Arlinta Christy Barus
Del Institute of Technology, Jl. Sisingamangaraja Sitoluama, Laguboti, North Sumatera, Indonesia
E-Mail: arlinta@del.ac.id

ABSTRACT

Software testing is one of phases in software engineering process that has a very important role to determine the quality of software under test. In software testing, after generating and selecting test cases, and executing them, the outputs need to be checked against a test oracle to determine whether any failures detected or not.

Oracle problem is one of the biggest problems in Software testing. It is a condition where a test oracle can not be obtained or too expensive to be used in testing. Metamorphic Testing is a new testing approach designed to alleviate oracle problem. This approach makes use the crucial properties of software under testing, to determine some Metamorphic Relations (MRs). MRs is used to generate follow-up test cases based on original test cases, known as source test cases. The relations are also used to verify whether test passes or fails.

This paper presents a use of Metamorphic Testing in testing a program implementing a matrix multiplication. Five Metamorphic Relations are identified and implemented to test five Mutant programs having intentionally bug inserted. All Mutants have been successfully killed by test cases generated by the five Metamorphic Relations. It showed that the generated MRs have been effective enough in conducting Metamorphic Testing for this case study.

Key words: oracle problem, metamorphic testing, multiplication matrix.

INTRODUCTION

Computer-based application has been widely used all over the world. Hence, the roles of software systems have been increased exponentially. This causes, at the same time, the increasing reports of software faults.

To guarantee the quality of software used is handled by software quality assurance process. It has become one of the most important areas in the software industry as well as in the academic sectors. Software testing, an important approach in software quality assurance, is widely reflected as a critical activity and now is one of main research focus in software engineering (Hailpern et al., 2002). One objective of software testing is to detect as quickly as possible, as many software faults as possible (Myers, 2004).

Software testing is one of phase in software engineering process that has a very improtant role to determine the quality of software under test. The general steps in software testing is generating test cases, selecting appropriate set of test cases based on certain criteria, executing them, and checking the outputs against a test oracle to determine whether any failures detected or not.

A test oracle is a mechanism to check whether the output of executing a program under testing using a test case is according to the expected output or not. In other words, it is used to verify whether the progam has passed the test or not (Hierons, 2012).

The presence of oracle testing is very important in conducting testing. However, in most situation, oracle testing is impractical to be found or too expensive which is known as an oracle problem (Manolache et al, 2001).

Chen et al designed a new testing method, called Metamorphic Testing (MT) which was aimed to alleviate the oracle problem (Chen et al, 1998). This method is approached based on the property of program under test. Based on the properties, tester is expected to generate some Metamorphic Relations that mainly have two functions: (i) to generate new test cases from the original test cases, and (ii) to verify whether test passes or fails based on the relations of the inputs and or outputs of original test cases and new test cases.

This paper aims to introduce the use of MT in a case study of matrix multiplication. This case is chosen as matrix multiplication proogram can face oracle problem particularly when the size of matrices are large. However the case is quite common and widely used so that it will be easier to understand in explaining the concept used in MT.

This paper consists of several parts as follows: (i) a literature study of MT, (ii) a presentation of the case study used in this paper, (iii) the explanation of the Metamorphic Relations identified in this study, (iv) the experimental design, (v) the experiment result, (vi), some discussions on the result, and ended by (vii) the conclusion of the paper.

METAMORPHIC TESTING

Metamorphic Testing (MT) is property bases testing which aims to find some useful relations (called Metamorphic Relations) to alleviate the oracle problems (Chen et al, 2003). As explained by Asrfai et al. (Asrafi et al, 2011), a metamorphic relation (MR) is an expected relation of the program under test which should be valid over a set of distinct input data and their corresponding output for multiple executions. Figure-1 sumarizes the relations in MT which involve source and follow-up inputs and outputs.MT checks the validity of MRs by multiply executing of the target program. The steps of MT are as folllowings: (i) determining specific properties of the SUT to construct MRs, (ii) generating source test case by some traditional testing techniques (such as random testing), (iii)

generating follow-up test cases based on source test cases according to the MRs, (iv) executing the test cases, and (v) verifing the outputs of the test cases against MRs. If the outputs of the source and follow-up test cases do not match their relations in corresponding MR, then the test fails.

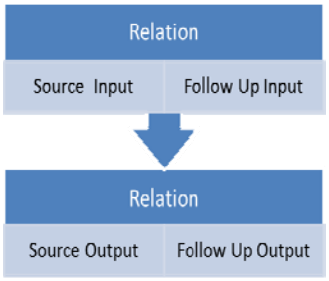

Figure-1. Relation in MT.

Asrafi et al (Asrafi et al, 2011) presented a simple example of MT in a sorting program as follows. The program sorts a set of integers in the ascending order. Suppose S is a set of numbers to be sorted. If the set S is rearranged in reverse order the output of the sorting program will still remain same. This MR can be denoted by Sort(S) = Sort (reverse(S)). Suppose S = {35, 15, 32, 25}, Sort(S) will yield {15, 25, 32, 35}.We reverse the set S to generate the follow-up test case reverse(S) = {25, 32, 15, 35}. If Sort (reverse(S)) ┐ {15, 25, 32, 35}, we can say a fault is detected. MT has been widely used in solving many oracle problems (Barus et al, 2009; Chen et al, 1998; Chen et al, 2009; Chen et al, 2004).

CASE STUDY

In mathematics, matrix multiplication is a binary operation that takes a pair of matrices, and produces another matrix (Coppersmith et al., 1990). As shown in Figure 1, if **A** is an n × m matrix and **B** is an m × p matrix. The matrix product **AB** is defined to be the n × p matrix, where each i, j entry is given by multiplying the entries A_{ik} (across row i of **A**) by the entries B_{kj} (down column j of **B**), for k = 1, 2, ..., m, and summing the results over k.

This operation is simple if the size of matrices are small. However, if the sizes are large, it may be difficult to verify whether the output of the operation is correct or not. A program implementing matrix multiplication is potentially facing oracle problem. Therefore, we choose this case study in this paper, to present the use of Metamorphic Testing as the concept can be easily presented and understood.

$$\mathbf{A} = \begin{pmatrix} A_{11} & A_{12} & \cdots & A_{1m} \\ A_{21} & A_{22} & \cdots & A_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ A_{n1} & A_{n2} & \cdots & A_{nm} \end{pmatrix}, \quad \mathbf{B} = \begin{pmatrix} B_{11} & B_{12} & \cdots & B_{1p} \\ B_{21} & B_{22} & \cdots & B_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ B_{m1} & B_{m2} & \cdots & B_{mp} \end{pmatrix}$$

$$\mathbf{AB} = \begin{pmatrix} (\mathbf{AB})_{11} & (\mathbf{AB})_{12} & \cdots & (\mathbf{AB})_{1p} \\ (\mathbf{AB})_{21} & (\mathbf{AB})_{22} & \cdots & (\mathbf{AB})_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ (\mathbf{AB})_{n1} & (\mathbf{AB})_{n2} & \cdots & (\mathbf{AB})_{np} \end{pmatrix}$$

Figure-2. The operation of matrix multiplication[1].

METAMORPHIC RELATIONS

Below are some notations to be used to illustrate the metamorphic relations in matrix multiplication program:

1. A, B: a pair of source inputs of matrices
2. A', B': a pair of corresponding follow-up inputs of matrices
3. O: output of multiplication of source inputs of matrices
4. O': output of multiplication of follow-up inputs of matrices

In other words:

O = A x B and
O' = A' x B'.

Based on the properties of matrix multiplication operation, some Metamorphic Relations are identified as follows:

1) MR-1: if A' is equal to B and B' is equal to A then O' is not equal to O.
This MR corresponds to the property of matrix multiplication operation where A.B is not equal to B.A

2) MR-2: if A' is equal to A and B' is equal to B multiplied by I (Identity Matrix) then O is equal to O'.
This MR corresponds to the property of matrix multiplication operation where A.B is equal to A. (B.I) where I is a identity matrix.

3) MR-3: if A' is equal to A * N where N is a positive integer and B' is equal to B then O*N is equal to O'.
This MR corresponds to the property of matrix multiplication operation where A. N. B is equal to A.B.N where N is a positive integer.

4) MR-4: if A' is equal to A * N where N is a negative integer and B' is equal to B then O*N is equal to O'.
This MR corresponds to the property of matrix multiplication operation where A. N. B is equal to A.B.N where N is a negative integer.

[1] Source: http://en.wikipedia.org/wiki/Matrix_multiplication

5) MR-5: if A' is a negative of A and B' is a negative of B then O' is equal to O.
This MR corresponds to the property of matrix multiplication operation where A.B is equal to A'.B' where A' is (-1).A and B' is (-1). B.

DESIGN EXPERIMENT

To conduct the experiment, two steps of preparation below are required to do:

1) Creating test pools

A hundred of test pools are randomly generated where each test pool contains 100.000 (a hundred thousand) of test cases. A test case contains a pair of matrices with random size of rows and columns. For each test pool, seventy percent of test cases are valid (the size of row of first matrix is equal to the size of column of the second one) and the remaining pairs are invalid (the size of row of first matrix is not equal to the size of column of the second one).

2) Creating mutant versions

A program implementing matrix multiplication is made by a programmer excluded from the author of the paper. Then five Mutant version of the program are generated randomly, using a technique intoruduced by Mu et al. (Mu et al, 2005). The generated errors for each Mutant can be seen in Table-1 below.

Table-1. Mutated lines of codes.

Mutant	Correct version	Mutant version
Mut-1	for (c = 1 ; c < p ; c++)	for (c = 0 ; c < p ; c++)
Mut-2	if (n < p)	if (n != p)
Mut-3	for (d = 1 ; d < q ; d++)	: for (d = 0 ; d < q ; d++)
Mut-4	If (n <= p)	if (n != p)
Mut-5	sum = 1	sum = 0

For each metamorphic relation, a new set of test cases (follow-up test cases) are generated from the source test cases generated in each test pool. Then the source and follow-up test cases are executed againts five different Mutants, and the results are recorded to see whether failures are detected or not. These steps are repeated 100 (one hundred) times for 100 (one hundred) different test pools.

EXPERIMENT RESULT

After conducting the experiments, the result can be found in Table-2 and Table-3 below.

Table-2. Average number of pair test cases revealing errors.

MR/ Mutant	MR-1	MR-2	MR-3	MR-4	MR-5
Mut-1	32350.38	97239.49	97520.02	97742.12	15001.46
Mut-2	32359.93	29999.87	29999.63	29999.44	15001.46
Mut-3	32393.28	67285.10	67565.90	67788.20	67285.10
Mut-4	64753.21	82286.69	82567.49	82789.79	82286.96
Mut-5	32393.28	67285.90	67565.90	67588.20	67285.10

Table-3. Average percentage number of pair test cases revealing errors.

MR/ Mutant	MR-1	MR-2	MR-3	MR-4	MR-5
Mut-1	32,35%	97,24%	97,52%	97,74%	15,00%
Mut-2	32,36%	30,00%	30,00%	30,00%	15,00%
Mut-3	32,39%	67,29%	67,57%	67,79%	67,29%
Mut-4	64,75%	82,29%	82,57%	82,79%	82,29%
Mut-5	32,39%	67,29%	67,57%	67,59%	67,29%

Table-2 presents data of average number of pair test cases that are able to reveal failures for every Mutants. It means the corresponding relations of each MR cannot be verified by the pairs of inputs hence the failures are detected. Table-3 presents the same data however in percentage instead of in numbers. Figure-1 displays the data in Table-3 grouped by each MR.

Table-2 shows that the largest numbers of pair test cases revealing errors for Mutant-1 are ones generated by MR-2, MR-3, and MR-4. The largest numbers of pair test cases revealing errors for Mutant-2 are ones generated by MR-1, MR-2, MR-3, and MR-4. The largest numbers of pair test cases revealing errors for Mutant-3 are ones generated by MR-2, MR-3, MR-4, and MR-5. The largest numbers of pair test cases revealing errors for Mutant-4 are ones generated by MR-2, MR-3, MR-4, and MR-5. For Mutant-5, the largest numbers of pair test cases revealing errors are generated by MR-2, MR-3, MR-4, and MR-4.

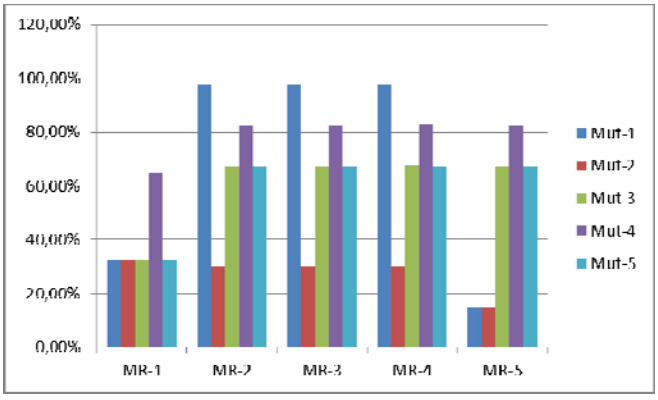

Figure-3. Average percentage number of pair test cases revealing errors.

DISCUSSIONS

Based on experiment results shown in Table-1 and Table-2, we can see there are a number of pairs of test cases revaling failures for each Mutant. This shows that all Mutants can be "killed" by the five MRs. Or in other words, all MRs are effective in detecting failures in all Mutants.

By comparing the number of test cases revealing the failures, as shown in Figure-2, overall we can see that all MRs performs good as they are able to kill all mutated version. In particular, MR-2, MR-3, and MR-4 perform quite similar and are the best among all MRs. MR-1 performs worse than others in testing Mutant-1, Mutant-3, and Mutant-5. MR-5 performs worse than others particularly in testing Mutant-1 and Mutant-2.

The similar performance between MR-2, MR-3, and MR-4 can be understood as their corresponding properties are relatively similar. This may cause the same response against the same mutated versions under testing.

The effectiveness of MRs is indeed a very interesting study to be conducted. Relevant studies have been conducted by Asrafi et al (Asrafi et al., 2011) and Liu et al. (Liu et al., 2014). There are some criteria can be assessed to see the performance of MRs, such as the excution behaviour of the pairs of source and follow-up test cases (Asrafi et al., 2011).

This paper is a preliminary study of an application of Metamorphic Testing. It is good to be continued in the future to study the efectiveness of each MR introduced in this paper according to some assessment criteria such as been investigated in (Asrafi et al., 2011).

CONCLUSIONS

An implementation of Metamorphic Testing (MT) has been studied in this paper, particularly for testing a program implementing multiplication matrix. Five different Metamorphic Relations (MRs) have been identified and also implemented. Using 5 (five) different mutation versions of the program, MT has been conductes using a hundred set of test pools, each containing 100.000 (a hundred thousand) of test cases.

From the conducted experiment, it is found that all MRs have been successfully generating test cases that are able to reveal faliures of every Mutant. The performance of three MRs which are MR-2, MR-3. MR-4 are quite similar and the best among all MRs. MR-1 performs worse than others in testing Mutant-1, Mutant-3, and Mutant-5. MR-5 performs worse than others particularly in testing Mutant-1 and Mutant-2.

The similar performance between MR-2, MR-3, and MR-4 are suspected due to their corresponding properties are relatively similar. This may cause the same response against the same mutated versions under testing.

FUTURE WORK

Currently the effectiveness of MRs are widely investigated. However it is not covered by this study. In the future, it is interesting to continue this study by investigating the effectiveness of all MRs identified in this study. One criterion of assessment such as the execution behaviour of the pairs of source and follow-up test cases, can be investigated as the causeof different performance of MRs in this study. The similarity performance of MR-2, MR-3, and MR-4 are worth to be investigated further. It is as well as their differences to MR-1 and MR-5.

In addition to the effectiveness of MRs, the increase number of MRs needs to be considered. Considering the operation of matrix multiplication, there will be a lot of potential MRs to be identified. This can be added in the future study of investigating the effectiveness of MRs of this study.

The expansion case studies to more operations in matrix can be also the extension of the future work of this study.

REFERENCES

R. M. Hierons (2012). Oracles for distributed testing. IEEE Transactions on Software Engineering, 38(3), pp 629–641.

A. C. Barus., T. Y. Chen D..,Grant F.-C. Kuo. and M.-F. Lau. Testing of heuristic methods: A case study of greedy algorithm. In Proceedings of the 3rd IFIP TC 2 Central and Eastern European Conference on Software Engineering Techniques (CEE-SET 2008), volume 4980 of Lecture Notes in Computer Science, 2011, pp. 246–260.

Chen T. Y., Cheung S. C. and Yiu S. M (1998). Metamorphic testing: a new approach for generating next test cases.Technical Report HKUST- CS98-01, Department of Computer Science, Hong Kong University of Science and Technology.

Manolache L. I. Kourie D.G (2001). Software testing using model programs. Software: Practice and Experience. Vol. 31, pp. 1211–1236.

Chen T. Y., Tse T. H., Zhou Z. Quan (2003). Fault-based testing without the need of oracles. Information and Software Technology. Vol. 45(1). pp 1-9.

Asrafi M., Kuo F.-C., Liu H. (2011). On Testing Effectiveness of Metamorphic Relations: A Case Study. 2011 Fifth International Conference on Secure Software Integration and Reliability Improvement, pp 147-156.

Coppersmith D., Winograd S. (1990). Matrix multiplication via arithmetic progressions, J. Symbolic Comput. 9, p. 251-280.

Ma Y.-S., Offutt J., Kwon Y.-R (2005): MuJava: An Automated Class Mutation System. Journal of Software Testing, Verification and Reliability. 15(2): pp 97-133

H. Liu F.-C. Kuo D. Towey T.Y. Chen (2014). How Effectively does Metamorphic Testing Alleviate the

Oracle Problem, IEEE Transactions on Software Engineering, 40(1). pp. 4-22.

B. Hailpern and P. Santhanam (2002). Software debugging, testing, and verification. IBM Systems Journal, 41(1) pp: 4–12.

G. J. Myers (2004). The Art of Software Testing. John Wiley and Sons, second edition. Revised and updated by T. Badgett and T. M. Thomas with C. Sandler.

T. Y. Chen., S. C. Cheung. and S. M. Yiu. Metamorphic testing: A new approach for generating next test cases. Technical Report HKUST-CS98-01, Department of Computer Science, Hong Kong University of Science and Technology, 1998.

T. Y. Chen., J. W. K. Ho. H. Liu. and X. Xie. An innovative approach for testing bioinformatics programs using metamorphic testing. BMC Bioinformatics, 10:24:1–24:12, 2009.

T. Y. Chen, D. H. Huang T. H. Tse. and Z. Q. Zhou. Case studies on the selection of useful relations in metamorphic testing. In Proceedings of the 4th Ibero-American Symposium on Software Engineering and Knowledge Engineering (JIISIC 2004), pages 569–583, 2004.