ISSN No. 0976-5697

Volume 2, No. 5, Sept-Oct 2011

International Journal of Advanced Research in Computer Science

CASE STUDY AND REPORT

Available Online at www.ijarcs.info

Comparison of Metamoprhic Testing and Special Value Testing using the properties of SUT

Amanjot Singh*
M.tech Student of CSE Dept.
CEC, Landran. Mohali, INDIA
amanjotmundi@gmail.com

Sandeep Kang
Assistant professor of CSE Dept.
CEC, Landran. Mohali, INDIA
Cecm.cse.skang@gmail.com

Parminder Singh
Assistant professor of IT Dept
CEC, Landran, Mohali, INDIA
Singh.parminder06@gmail.com

Abstract: When testing a program, correctly executed test cases are seldom explored further, even though they may carry useful information. Metamorphic testing proposes to use special test cases to check important properties of the target function. It does not need a human oracle to verify, compare or predict output . An "oracle" in software testing is a procedure by which testers can decide whether the output of the program under testing is correct. In some situations, however, the oracle is not available or too difficult to apply. This is known as the "oracle problem". The manual prediction and verification of program output greatly decreases the efficiency and increases the cost of testing. A metamorphic testing method has been proposed to test programs without the involvement of an oracle. It employs properties of the target function, known as metamorphic relations, to generate follow-up test cases and verify the outputs automatically.

Keywords: Metamorphic testing, Metamorphic relation, software under test(SUT), oracle, successful test cases

I. INTRODUCTION

It is impractical, if not impossible, to test a program with all conceivable inputs we should aim at selecting test cases with higher probabilities of revealing program failures. A successful *test case* is one on which the program computes correctly. Since successful test cases do not reveal any failure, they are conventionally considered useless and thus discarded by testers or merely retained for reuse in regression testing later, but in Metamorphic testing only these successful test cases are taken ton generate the follow up test cases. Another limitation of software testing is the *oracle problem* .An *oracle* is a mechanism against which people can decide whether the outcome of the program on test cases is correct. In some situations, the oracle is not available or is too expensive to be applied, even when manual prediction and comparison of testing results are possible, they are often time consuming and error prone A metamorphic testing (MT) method has been proposed with a view to making use of the valuable information in successful test cases. It does not depend on the availability of an oracle.

It proposes to generate follow up test cases based on metamorphic relations, or properties among inputs and outputs of the target function. Metamorphic relation (MR) is a relation over a series of distinct inputs and their corresponding results for multiple evaluations of a target function [20]. Consider, for instance, the sine function. We have the following relation: *If* $x2 = \pi - x1$, *then* $\sin x2 = -\sin x1$. We note from this example that a metamorphic relation consists of two parts. The first part (denoted by r in the definition below) relates $x2$ to $x1$. The second part (denoted by r') relates the results of the function. If the MR above is not satisfied for some input, we deem that a failure is revealed.

II. THE CONCEPT OF METAMORPHIC TESTING

Metamorphic testing (MT) is a technique to generate follow-up test cases based on existing test cases that have not revealed any failure. MT should be applied in conjunction with other test case selection strategies that generate the initial set of test cases. Let us consider a program p implementing function f on domain D. Let S be the test case selection strategy adopted by the tester, such as data flow testing or branch coverage. According to S, a test set $T = \{t1, t2, \ldots, tn\} \subset D$, where $n \geq 1$, can be generated. Running the program on T yields the outputs $p(t1), p(t2), \ldots, p(tn)$. When there is an oracle, these test results can be verified against $f(t1), f(t2), \ldots, f(tn)$; Otherwise the tester may still have some way to identify some outcomes that are obviously wrong. For example, an execution that runs too long can be considered a failure; when a trigonometric function computing $\cos x$ returns a value greater than 1, a failure can also be found immediately.

When a failure has been detected, testing can stop and the program will be debugged; otherwise T is a set of successful test cases. In the latter case, MT can be applied to automatically generate follow-up test cases $T' = \{t'1, t'2, \ldots, t'n\} \subset D$ based on the initial successful test set T, so that the program can be further verified against some necessary properties. MT is useful because the vast majority of test cases are successful ones, although they have not revealed any

failure, these test cases do carry useful information ignored in conventional testing. MT generates follow-up test cases by making MT generates follow-up test cases by making reference to "metamorphic relations" (MR). For program *p*, an MR is a property of its target function *f* . For a successful test case *ti* and a chosen MR, we can construct follow-up test case(s), say *t'I* , and run the program again. Let *p* denote the program under test. We check *ti*, *p*(*ti*), *t'i* , and *p*(*t'i*) against the MR.

If MR cannot be satisfied, the program must have failed. Consider, for instance, a program that computes the sine function. The property $\sin x = \sin(180-x)$ can be used as a metamorphic relation. Let $t = 57.3$ be one of the test cases chosen according to a selection strategy such as branch coverage. Suppose the output is 0.8415. This output may not be verified easily if an oracle is not available. On the other hand, regardless of whether an oracle exists, MT suggests testing the program with a follow-up test case $180-57.3$. The program is run on this test case to produce a second output, say 0.8402. The two outputs are then compared. Obviously, they do not satisfy the expected MR and hence a failure is detected before MT is applied, a test case selection strategy *S* and a set of test cases *T* corresponding to *S* must exist in the first place. If no failure is revealed by *T*, then MT can be applied to generate a new set of test cases as a partner accompanying *T*, so that the program can be further verified against some necessary metamorphic relations.

This is regardless of whether an oracle is available. Another characteristic of MT is that Metamorphic relations are not limited to identity relations. Any expected relation involving inputs and outputs of two or more executions of the program can be taken as an MR. MT does not check the correctness of individual outputs. Instead, it checks the relations among several executions. Since no manual output predictions and comparisons are required, MT can be efficient and fully automated.

III. APPLICATION OF METAMORPHIC TESTING

A. Computer Graphics:

When the outputs of a program involve a large amount of data, they are expensive to verify. For example, computer graphics software generates graphics and prints them on the screen. It is, however, practically impossible for the tester to manually check whether each and every pixel is displayed properly. In this situation, a practical approach is that after checking the correctness of certain amount of individual outputs, we apply MT to verify all the outputs in a more cost effective way. For the tester, it is not easy to verify whether all the pixels in the screen are displayed properly because the generation of realistic graphics involves complicated computation and there is a huge amount of pixels.

Nevertheless, some metamorphic relations can be identified. For example, if the position of the light source for an image changes, then the brightness of all the points that become closer to the light source will increase according to a certain formula; similarly, all the points that become farther will become darker. This is an easy approach to check all the displayed pixels quickly and automatically.
Following this way, many other metamorphic relations can be identified as well.

B. Other Areas of Application:

Many applications in the field of scientific computing - such as computational biology, computational linguistics, and others - depend on Machine Learning algorithms to provide important core functionality to support solutions in the particular problem domains. However, it is difficult to test such applications because often there is no "test oracle" to indicate what the correct output should be for arbitrary input. In such cases also metamorphic testing is efficient for validating the system. Many applications in the field of scientific computing -such as computational physics, bioinformatics, etc. depend on supervised Machine Learning (ML) algorithms to provide important core functionality to support solutions in the particular problem domains. For instance, lists over fifty different real-world computational science applications, ranging from facial recognition to computational biology, that use the Support Vector Machines classification algorithm alone. As these types of applications become more and more prevalent in society, ensuring their quality becomes more and more crucial.

Quality assurance of such applications presents a challenge because conventional software testing processes do not always apply: in particular, it is difficult to detect subtle errors, faults, defects or anomalies in many applications in these domains because there is no reliable "test oracle" to indicate what the correct output should be for arbitrary input. The general class of software systems with no reliable test oracle available is sometimes known as "non-testable programs"; the fact that such programs exist is often referred to as "the oracle problem".

IV. CASE STUDY

In our case study we take a matrix operation function which performs the multiplication of matrices, this function is our target function which is being tested. We have particularly chosen this function is because , it has been largely used in scientific and engineering problems, it has a large number of universal properties associated, which we can use as metamorphic relations and for operations on large order matrices oracles are not available.

Now, we will use the universal properties of matrices as metamorphic relations against which we verify the system, the more is number of metamorphic relations the better results we will get, but it still depends on the quality of the metamorphic relations used as strong MR's are more effective in revealing the faults than weak MR's.

The proposed methodology for conducting the test is as followed:

a. Creating a function for matrix operation i.e matrix multiplication
b. Creating the mutants
c. Creating the special test cases
d. Identifying the metamorphic relations

e. Performing the test
f. Comparing the results of test

The following mutants are used in the testing process and the results are evaluated on the basis of fault detection ratio with them.

MUTANT 1	Replacing the operator * with +
MUTANT 2	Replacing the operator * with -
MUTANT 3	Replacing the operator + with *
MUTANT 4	Replacing the line of code 18 with C(i,j) = C(i,j) * A(i,k)

In the next step after inducing the mutant and identifying the metamorphic relations we perform the testing with random test cases as well as specially designed test cases. The specially designed test cases are those for which we already know the expected output or it is very easy to calculate, basically eliminating the need for oracle to verify the output of system.

Test case	Initial test	MR1	MR2	MR3	MR4	MR5
T1	T	T	T	F	F	F
T2	T	T	T	T	T	F
T3	T	T	T	F	F	T
T4	T	T	T	F	F	F

Results with Mutant 1

Test case	Initial test	MR1	MR2	MR3	MR4	MR5
T1	T	T	F	F	F	F
T2	T	T	F	F	T	F
T3	T	T	T	T	F	F
T4	T	F	F	F	F	F

Results with Mutant 2

Test case	Initial test	MR1	MR2	MR3	MR4	MR5
T1	T	T	T	T	T	T
T2	T	T	T	T	T	T
T3	T	T	T	T	T	T
T4	T	T	T	T	T	T

Results with Mutant 3

V. RESULTS AND CONCLUSION

Metamorphic testing is capable of exposing the errors in the system, where other testing techniques fail. This is because of the use of metamorphic relations in the testing procedure, many testing techniques which use oracles to verify the result may not detect a minor fault in the program which may or may not affect the outcome of the system for most cases. In such cases the metamorphic testing will detect the fault by testing the system with different relations. This is proved by comparing the metamorphic testing with other random and special value testing using a mathematical function 'matrix' in the above case study.

VI. FUTURE WORK

There is a lot of work to be done in this field, such as identifying the strong metamorphic relations among all the available relations to get better results by using less number of relations, optimization of test data with respect to the properties of the target function for metamorphic testing also have a lot of scope for future work. Metamorphic testing is now widely used in mathematical and scientific applications, it also have a lot of scope in the field of bioinformatics. Moreover a special type of metamorphic testing called n-iterative testing have a great scope for future work.

VII. REFERENCES

[1]. Beizer, B. Software Testing Techniques, Van Nostrand Reinhold, New York, '1990'.

[2]. T. J. Cheatham, J. P. Yoo, and N. J. Wahl. Software testing: a machine learning experiment. In Proc. of the ACM 23rd Annual Conference on Computer Science, '1995'.

[3]. T. Y. Chen, S. C. Cheung, and S. 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'.

[4]. Chen, T.Y., Cheung, S.C., and Yiu, S.M. Metamorphic testing: a new approach for generating next testcases, Technical Report HKUST-CS98-01, Department of Computer Science, Hong Kong University of Science and Technology, Hong Kong, '1998'.

[5]. Chen, T.Y., Feng, J., and Tse, T.H. Metamorphic testing of programs on partial differential equations: a case study, In Proceedings of the 26th Annual International Computer Software and Applications Conference (COMPSAC 2002), IEEE Computer Society Press, Los Alamitos, California, '2002'

[6]. Chen, T.Y., Kuo, F.-C., Liu, Y., and Tang, A. Metamorphic testing and testing with special values, In Proceedings of the 5th International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel /Distributed Computing (SNPD 2004), International Association for Computer and Information Science, Mt. Pleasant, Michigan, '2004'.

[7]. Chen, T.Y., Tse, T.H., and Zhou, Z.Q., Fault-based testing without the need of oracles, Information and Software Technology, 45 (1), '2003'.

[8]. Gotlieb, A. and Botella, B. Automated metamorphic testing, In Proceedings of the 27th Annual International Computer Software and Applications Conference(COMPSAC 2003), IEEE Computer Society Press, Los Alamitos, California, '2003'

[9]. T.Y. Chen, F.-C. Kuo , T.H. Tse , Zhi Quan Zhou Metamorphic Testing and Beyond *Proceedings of the International Workshop on Software Technology and Engineering Practice* (*STEP 2003*), IEEE Computer Society Press, Los Alamitos, California (2004)