Imperial Journal of Interdisciplinary Research (IJIR)
Vol-3, Issue-3, 2017
ISSN: 2454-1362, http://www.onlinejournal.in

Methods to Choose Best Metamorphic Relations

Dr.Amit Verma & Manindra kaur
Computer Science and Engineering, CEC, Landran, Mohali, India

Abstract—*Software is an arrangement of directions used to perform particular undertaking. Errors happen in the product program and impact its quality, to beat this issue we utilize Software testing which guarantees its quality. Test prophet is utilized to recognize issues, yet it is not generally conceivable as it is regularly occupied and entirely costly. Without oracle, the viability of Software testing gets blocked. This is known as oracle issue. For instance when a project's input output relation is perplexing and difficult to catch formally. Metamorphic Testing has been proposed which is extremely pertinent in recognizing bugs without oracle and it depends on the properties of target capacity. These properties infer a few relations which creates follow-up experiments taking into account unique experiments that is source test cases. These relations are called metamorphic relations. The presentation of metamorphic relations in 1998, numerous commitments on metamorphic testing have been made, and the strategy has seen fruitful applications in an assortment of areas, going from web administrations to PC design. Metamorphic relations are generally in light of distinguished transformative relations and changeable relations are recognized physically by analyzers as there is no appropriate procedure to recognize them. Metamorphic relations are utilized to perlustrate a few projects viably. For any problem, more than one metamorphic relation is clarifying, distinctive transformative relations have curious viability in descry flaws, yet illustrating transformative relations can be strenuous. Good Metamorphic relations give viability in distinguishing bugs to lessen the expense. It is imperative to know how to develop and produce great metamorphic relations to spare time and assets. This paper audits its difficulties and strategies like experimental study, code scope, arrangement of metamorphic relations, mechanized metamorphic testing, change investigation to develop and produce likely Metamorphic relations. This paper has a substantial degree and gives rules to the development of good metamorphic relations and era of likely metamorphic relations to upgrade the capacity of uncovering disappointments and decrease the expense of testing report.*

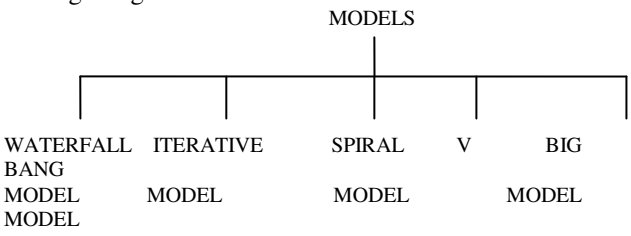
Index Terms—Software testing, oracle problem, metamorphic testing, metamorphic relation.

I.INTRODUCTION

Software engineering is a field of engineering, for designing and writing programs for computers or other electronic gadgets. A product engineer, or programmer, composes programming (or changes existing programming) and accumulates programming utilizing techniques that improve it quality. Better quality software is easier to utilize, and the code is easier to comprehend, to maintain , and to include new elements. Turning into a software engineer requires college level classes and work on composing code. Software engineering can be exceptionally troublesome work. Software engineering is regularly done as a feature of a group.

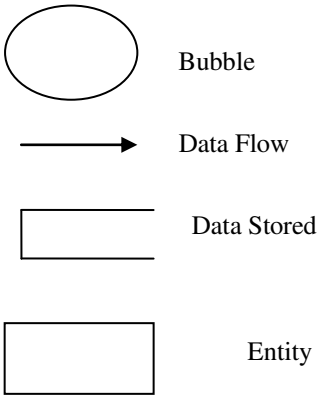
A Software process model is a dynamic representation to depict the procedure from a specific point of view. There are quantities of general models for programming forms, similar to: Waterfall model, Evolutionary development, Formal systems development and Reuse based improvement, and so forth. This examination will see the accompanying models:

1. Waterfall model.
2. Iteration model.
3. Spiral model.
4. V model
5. Big Bang Model

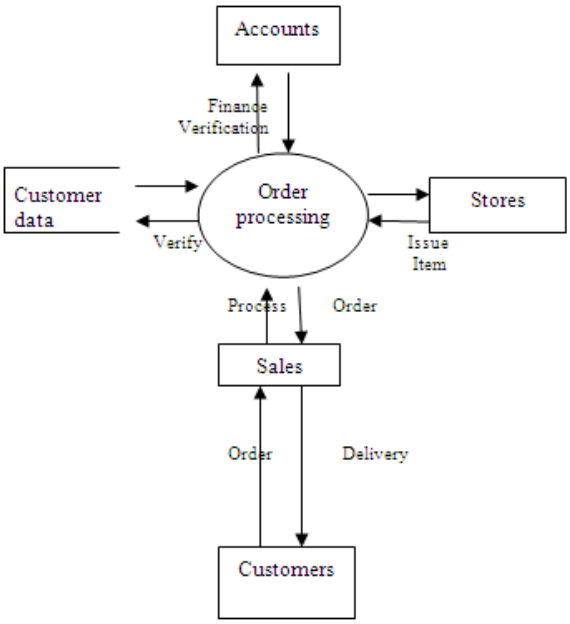


DFD OF SOFTWARE TESTING

SYMBOLS :-



For example :-



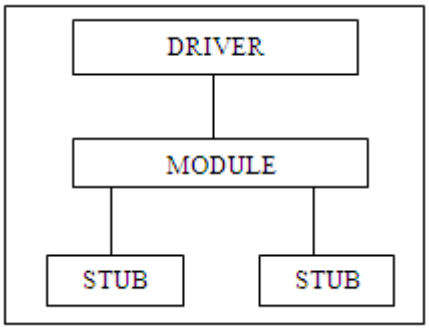
SOFTWARE TESTING TECHNIQUES:-

Black box testing – Internal framework configuration is not considered in this sort of testing. Tests depend on prerequisites and usefulness.

White box testing – This testing depends on information of the inside rationale of an application's code. Otherwise called Glass box Testing. Interior programming and code working ought to be known for this kind of testing. Tests depend on scope of code articulations, branches, ways, and conditions.

Unit testing – Testing of individual programming parts or modules. Ordinarily done by the software engineer and not by analyzers, as it requires nitty gritty information of the inside project outline and code. may require creating test driver modules or test tackles .

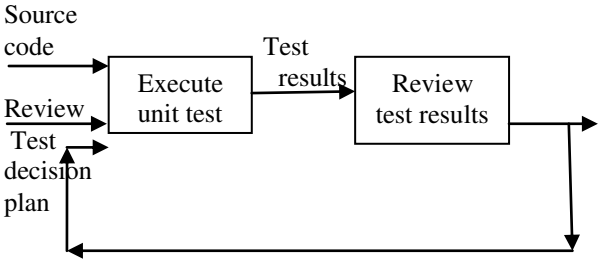
Integration testing – Testing of coordinated modules to check consolidated usefulness after reconciliation. Modules are ordinarily code modules, singular applications, customer and server applications on a system, and so forth. This kind of testing is particularly applicable to customer/server and circulated frameworks.



System testing – Entire framework is tried according to the prerequisites. Discovery sort testing that depends on general necessities details, covers every single joined part of a framework

Regression testing – Testing the application in general for the adjustment in any module or usefulness. Hard to cover all the framework in relapse testing so commonly mechanization devices are utilized for these testing sorts.

Metamorphic Testing - Metamorphic testing is a technique for detecting the failures without the need of oracle.



Metamorphic testing has been introduced in 1998. Since its introduction, the writing on Metamorphic testing has flourished with various systems, applications and appraisal concentrates on that have not been completely audited as of recently. The oracle problem has two limitations as following :-

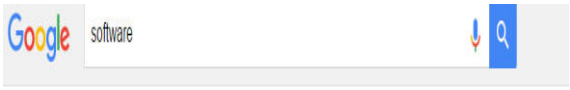
1. In most situations, oracle is unavailable. In fact when it is available, it cannot guarantee the correctness of program.
2. It is quite expensive as it is mostly unavailable and that's why it is difficult to apply.

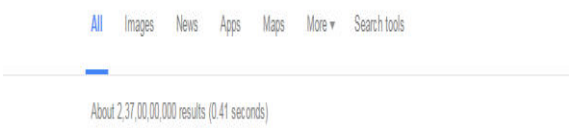
Metamorphic Testing is a technique to detect faults without oracle, that employs the expected properties of target function and these properties are called Metamorphic Relations. Metamorphic Relations generate follow-up test case based on source test case.

1.1 BASIC STEPS TO BE FOLLOW IN METAMORPHIC TESTING:

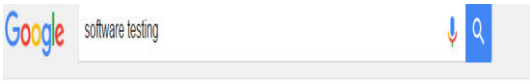
STEP 1.Determine Properties of the target function.
STEP 2.Generate source test case.
STEP 3.Follow-up test case is generated from source test case.
STEP 4.Execution of test cases takes place.
STEP 5.It verifies whether test fails or pass on the basis on input output relations.

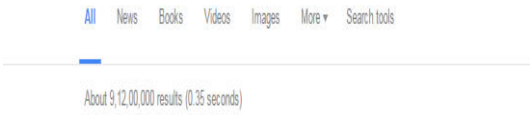
Let us consider an example from goggle [15] , consider source test case as "software" results in 2,37,00,00,000 whereas its follow-up test case as "software testing" results in 9,12,00,000. Follow-up test case results are less than source test case. Thus worthwhile result has been found . Otherwise it destroys the metamorphic relation by giving defect.





Source test case





Follow-up test case

1.2 PROPERTIES TO SELECT METAMORPHIC RELATIONS:

1. To define good metamorphic relations, one should have thorough knowledge of the program.
2. Metamorphic relations should make multiple parallel executions of source test case as different as possible to reduce the cost to perform better in detecting faults.
3. Good metamorphic relations should contain as much as semantics of SUT.
4. Good metamorphic relations are derived from those parts of the program which are easy to construct and more effective in detecting faults.

PROPERTIES TO SELECT GOOD METAMORPHIC RELATIONS ARE:
Thorough knowledge
Make multiple executions
Contain semantics
Derived program

The essential procedure for the utilization of metamorphic testing can be compressed as takes after:
1) Construction of metamorphic relations
Distinguish important properties of the system under test and speak to them as metamorphic relations among various test cases inputs and their normal outputs , together with some strategy to produce a follow–up test cases in view of a source test cases. Note that metamorphic relations might be connected with preconditions that confine the source test cases to which they can be connected.

2) Generation of source test cases
Produce or select an arrangement of source test cases for the project under test utilizing any customary testing system (e.g., irregular testing).

3) Execution of metamorphic experiment: Utilize the metamorphic relations to create follow–up test cases, execute source and follow–up test cases, and check the relations. In the event that the outputs of a source test cases and its follow–up test cases damage the metamorphic relation, the test cases is said to have fizzled, showing that the project under test contains a bug.

1.3 METHODS FOR THE SELECTION OF GOOD METAMORPHIC RELATIONS:

It is very crucial to develop some methods for the effectiveness of metamorphic relations to reduce the cost and select good metamorphic relations for revealing failures more effectively.

Empirical Study enable us to elaborate the usefulness of metamorphic testing to derive the good metamorphic relations. Metamorphic Relations can be identified from the algorithms or specification of software under test. Mutation analysis is used to classify metamorphic relations by applying mutants in the program. Different multiple execution of the program for better performance. Metamorphic relations with high code coverage performs better in detecting bugs. These methods helps to design good metamorphic relations .

1.4 CONSTRUCTION OF METAMORPHIC RELATIONS:-

We need creativity for constructing good metamorphic relations. Construction of metamorphic relations can be done manually by having thorough knowledge of the problem domain.

There are three main challenges in constructing metamorphic relation:

1. Manual metamorphic relations are needed for metamorphic testing.
2. Input data is sophisticated.
3. ”Fake errors” occurs in comparison.

To cope up with these challenges we use automated metamorphic testing but these are unrealized and scope of improvement if further study.

Another way of constructing metamorphic relations is CMR i.e. composite metamorphic relations. CMR combines various relations like if follow-up test case of one relation can be source test

case of other. This method reduces the cost of the problem domain.

1.5 GENERATION OF LIKELY METAMORPHIC RELATIONS:

It is difficult to find good metamorphic relation and generation of likely metamorphic relations is the biggest challenge in metamorphic testing . The generated metamorphic testing is used to reveal faults more effectively by using mutation analysis having fault detection ratio and mutation score. Fault detection ratio is the ratio that calculates test cases to reveal errors and the mutation score is the ratio of detected faults and total number of faults. But Generation of likely metamorphic relations is unable to define its previous metamorphic relation. The research on generation of likely metamorphic relations need further investigation.

II. REVIEW

We have reviewed many papers for the construction and generation of likely metamorphic relations but there is still no systematic way to construct and generate likely metamorphic relations. It has a large scope of improvement which we will carried out for my future work. In this segment, we audit proposed elective approaches to make metamorphic relations, either by consolidating existing relations or producing them consequently.

Liu et al. [16] proposed a strategy named Composition of Metamorphic Relations (CMR) to build new Metamorphic relations by joining a few existing relations. Liu et al. defined two Metamorphic relations as "compositable" if the follow–up test cases of one of the relations can simply be utilized as source test cases of the other.

Kanewala and Bieman [2] proposed a technique to figure out if a numerical program shows certain metamorphic relations out of a predefined set of relations that they trust exists in numerous numerical programs.

Zhang et al. [3] proposed a search–based approach for the derivation of polynomial metamorphic relations. All the more specifically, the algorithm searches for metamorphic relations as linear or quadratic equations (e.g., $\cos(2x) = 2\cos2(x)-1$).

Carzinaga et al. [4] proposed to produce oracle by misusing the excess contained in programs.

Goffi et al. [5], [6] displayed a search–based algorithm for the automated amalgamation of likely–equivalent technique groupings in object–oriented programs. The authors recommend that such likely–equivalent arrangements could be utilized as metamorphic relations during testing.

We review applications of metamorphic testing to specific problem domains, and outline approaches that utilization metamorphic testing to improve other testing methods.

WEB SERVICES AND SERVICES

Sun et al. [7], [8] proposed to manually get metamorphic relations from the WSDL portrayal of web administrations. Their method naturally creates random source test cases from the WSDL specification and applies the metamorphic relations.

COMPUTER GRAPHICS

Just and Schweiggert[9]used mutation analysis to evaluate the effectiveness of test data generation methods and metamorphic relations for a jpeg2000 image encoder.

EMBEDDED SYSTEM

Kuo et al. [10] reported a contextual analysis on the utilization of metamorphic testing for the location of shortcomings in a remote metering framework. A metamorphic relation was identified and used to test the meter perusing capacity of a business gadget from the electric business in which two genuine imperfections were revealed.

SIMULATION AND MODELLING

Many authors utilized code coverage criteria to control the determination of successful metamorphic relations and the making of test cases .

MACHINE LEARNING

Murphy et al. [11] identified six metamorphic relations that they trust exist in most machine learning applications, in particular: additive, multiplicative, permutative, invertive, inclusive , and exclusive relations. The adequacy of the relations was evaluated on three specific machine learning apparatuses in which some genuine bugs were identified.

VARIABILITY AND DECISION SUPPORT

Kuo et al. [12] displayed a metamorphic testing approach for the automated detection of flaws in decision support systems .

NUMERICAL PROGRAM

Prasad [13] displayed a few metamorphic relations for augmentation and division of multi accuracy math programming applications. The work was assessed with four real–time numerical activities and mutation analysis.

ENCRYPTION

Sun et al. [14] identified several metamorphic relations for encryption program.

WORK DONE BY VARIOUS AUTHORS ON CONSTUCTION OF LIKELY METAMORPHIC RELATIONS ARE ILLUSTRATED IN THE TABLE BELOW :

AUTHOR	PAPER	TECHNIQUE USED	PROS	CONS
Liu et al.	"A new method for constructing metamorphic relations",2012	Composition of Metamorphic Relations.	Cost effective	Need to generate more metamorphic relations
Kanewala and Bieman	"Using machine learning techniques to detect metamorphic relations for programs without test oracles," 2013	Numerical Program	Automatically detect likely metamorphic relation	Needs to evaluate the effectiveness.
Zhang et al.	"Search-based inference of polynomial metamorphic relations," 2014	Search based approach	First automatic approach to MR inference.	Further investigation is needed to extend types of MR.
Carzinaga et al	"Cross-checking oracles from intrinsic software redundancy,"2014	Redundant method to generate oracle	Reduce redundancy.	Identification of redundant method is manual.
A. Goffi, A. Gorla, A. Mattavelli, M. Pezze, and P. Tonella	"Search-based synthesis of equivalent method sequences,"2014	Search based algorithm	Fault tolerance and self healing.	Prototype implementations have some limitation.
C. Sun, G. Wang, B. Mu, H. Liu, Z. Wang, and T. Y. Chen	"A metamorphic relation-based approach to testing web services without oracles,"2012	WSDL	Automatically generate test cases.	Needs further investigation.
R. Just and F. Schweiggert	"Evaluating testing strategies for imaging software by means of mutation analysis,"2009	Mutation analysis	Good at detection.	Require more accuracy.
F.-C. Kuo, T. Y. Chen, and W. K. Tam	"Metamorphic testing of decision support systems: a case study,"2011	Meter reading	Detect faults	Two real effects were uncovered.

III. CONCLUSION AND FUTURE SCOPE

In this specialized Paper, we exhibited a writing survey on metamorphic testing..We broke down proportions and patterns demonstrating the principle propels on the procedure, its application spaces and the qualities of experimental evaluations. We additionally discovered confirmation of the materialness of the system to different areas a long ways past numerical projects, and in addition its combination with other testing methods. Moreover, we identified an expanding number of papers reporting the identification of flaws in genuine projects. In spite of the fact that our strategy still needs some examination with the goal that it can develop more metamorphic relations deliberately from the given metamorphic relations..The real test is the manner by which to choose, build and create great

metamorphic relations which has a large scope and needs further investigation. We assume that this work may turn into an accommodating reference for future advancement on metamorphic testing.

ACKNOWLEDGEMENT

We are very grateful to S.Segura, T.Y.Chen, Z.hiu and S.Huang, D.H.Huang, T.H.Tse, Z.Zhou, J.Mayer, R.Guderlei, M.Ashrafi, H.Liu, F.C.Kuo, X.Liu, Arlinta Christy Barus to get all the necessary resources from their work.

REFERENCES

[1] U. Kanewala and J. M. Bieman, "Using machine learning techniques to detect metamorphic relations for programs without test oracles," in IEEE 24th International Symposium on Software Reliability Engineering (ISSRE), 2013, Nov 2013, pp. 1–10.

[2] U. Kanewala, "Techniques for automatic detection of metamorphic relations," in IEEE Seventh International Conference on Software Testing, Verification and Validation Workshops (ICSTW), 2014, March 2014, pp. 237–238.

[3] J. Zhang, J. Chen, D. Hao, Y. Xiong, B. Xie, L. Zhang, and H. Mei, "Search-based inference of polynomial metamorphic relations," in Proceedings of the 29th ACM/IEEE International Conference on Automated Software Engineering, ser. ASE '14. New York, NY, USA: ACM, 2014, pp. 701–712.

[4] A. Carzaniga, A. Goffi, A. Gorla, A. Mattavelli, and M. Pezz`e, "Cross-checking oracles from intrinsic software redundancy "in Proceedings of the 36th International Conference on Software Engineering, ser. ICSE 2014. New York, NY,USA: ACM, 2014, pp. 931–942.

[5] A. Goffi, A. Gorla, A. Mattavelli, M. Pezze, and P. Tonella, "Search-based synthesis of equivalent method sequences," in Proceedings of the 22Nd ACM SIGSOFT International Symposium on Foundations of Software Engineering, ser. FSE 2014. New York, NY, USA: ACM, 2014, pp. 366–376.

[6] . Goffi, "Automatic generation of cost-effective test oracles," in Companion Proceedings of the 36th International Conference on Software Engineering, ser. ICSE Companion 2014. New York, NY, USA: ACM, 2014, pp. 678–681.

[7] C. Sun, G. Wang, B. Mu, H. Liu, Z. Wang, and T. Y. Chen, "Metamorphic testing for web services: Framework and a case study," in IEEE International Conference on Web Services (ICWS), 2011, July 2011, pp. 283–290

[8] C. Sun, G. Wang, B. Mu, H. Liu, Z. Wang, and T. Y. Chen, "A metamorphic relation-based approach to testing web services without oracles," International Journal of Web Services Research, vol. 9, no. 1, pp. 51–73, Jan. 2012

[9] R. Just and F. Schweiggert, "Evaluating testing strategies for imaging software by means of mutation analysis,"inInternational Conference on Software Testing,Verification and Validation Workshops, 2009. ICSTW '09, April 2009, pp. 205–209.

[10] F.-C. Kuo, T. Y. Chen, and W. K. Tam, "Testing embedded software by metamorphic testing: A wireless metering system case study," in IEEE 36th Conference on Local Computer Networks (LCN), 2011, Oct 2011, pp. 291–294.

[11] C.Murphy,G.Kaiser,andL.Hu,"Propertiesofmachinelearning applications for use in metamorphic testing," Department of Computer Science, Columbia University, New York NY, Tech. Rep., 2008.

[12] F.-C. Kuo, Z. Zhou, J. Ma, and G. Zhang, "Metamorphic testing of decision support systems: a case study," Software, IET, vol. 4, no. 4, pp. 294–301, August 2010.

[13] C. Aruna and R. S. R. Prasad, "Metamorphic relations to improve the test accuracy of multi precision arithmetic software applications," in International Conference on Advances in Computing, Communications and Informatics (ICACCI, 2014, Sept 2014, pp. 2244–2248.

[14] C. Sun, Z. Wang, and G. Wang, "A property-based testing framework for encryption programs," Frontiers of Computer Science, vol. 8, no. 3, pp. 478–489, 2014.

[15] Google,www.google.com

[16] H.Liu, X.Liu, and T.Y.Chen,"A new method for constructing metamorphic relations," in 12th International Conference on Quality Software (QSIC),2012,Aug 2012, pg.59-68