Journal of Theoretical and Applied Information Technology
31st July 2015. Vol.77. No.3
© 2005 - 2015 JATIT & LLS. All rights reserved·
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

YOKING OBJECT ORIENTED METRICS THROUGH MUTATION TESTING FOR MINIMIZING TIME PERIOD RAMIFICATION

[1]Chandu P.M.S.S., [2]Dr.T.Sasikala
1. Research Scholar, Department of CSE, Sathyabama University, Chennai, India.
2. Principal, SRR Engineering College, Chennai, India.
E-mail: [1]chandupmss@gmail.com, [2]sasi_madhu2k2@yahoo.co.in

ABSTRACT

Object Oriented Programming largely depends on the testing processes. Testing is of vital importance and is being carried out at three levels namely system, unit and integration verification. The three approaches Structural, black-box, error-driven approach is normally considered as efficient approaches. In the proposed system, we probe into object-oriented metrics which are forcefully yoked through mutation testing processes using Selenium tool and the primary focus is on checking the time consumption in the effective evoking of results. However maximizing the effectiveness of software has been principle objective of the proposed work. To derive a non-existing software test case and to ascertain quality of programs Mutation method is used. Selenium an automatic tool through open source technology is implemented in this process which provides various set programming equipments. Each and every equipment evokes an unique mode of support testing being generated by the tool. In this proposed system the mutation mode of testing the objects is carried out by means of selenium tool.

Keywords: *Testing, Implementing, Metrics, Automation, Mutation*

1. INTRODUCTION:

Testing process is the nucleus for a software program development. The existing testing processes depict an unbalanced application and invariable time period hence it is prominent to have a system that evokes effective, balanced output. The result is more effective and reaches the client within a specified span of time with minimum or no errors. Entire functionality of the program and the software is tested whereas the oldest software building approaches gives a detailed description about the usual analyzing set that leads to comprehend outcome of the building phase.

2. SOFTWARE TESTING:

Program building process depends on testing processes to determine the output without errors. [1] The accuracy of the Test cases and its logical perfection leads the effective functioning of the software. To validate software and to eke out the flaws in software, software testing is usually carried out. The testing process takes into account a span of time allotted and available resources and hence perfection of the software cannot be ascertained completely using such testing. Software development is a vast area whereas software testing is applied in most cases in lieu of developing it. The principle objective of life cycle testing is to check effort testing from the beginning and to limit the time involved in evolving out debugging and the development of a testing phase.

3. SOFTWARE METRICS:

Functionality and source code testing is the existing technique. The relationship between two variables is often denoted with a number termed as metrics. A given attribute, its system, component its quantitative measure are all analyzed. The ultimate goal of software metrics is to hold predominance in testing phase which also acts as a fault proneness and software quality indicators [4]. Equivalence partitioning testing, error guessing, decision table analysis and state transition analysis are the main arguments to be tested. These are done through black box examination whereas in the white box testing Path analysis, branch coverage examination, statement analysis, data flow analysis are tested. This sort of testing is done through a separate tool through which the functionality of the verification process is maximized. The software verification process is utilized in the testing mechanism that has its own merits and demerits. The tools' usage is practiced but in certain occasions it may not bring out the result which we desire. Object oriented metrics provides the maintenance of the software.

[5] The data members and the activities that are related to the software programs are determined using the metrics mechanism. The testing includes main factors such as coupling measurement, number of lines of code, program execution time and cohesion metrics are measured quantitatively. A hoard is delivered to evoke and analyze the class that should be verified and there is an occurrence of stint to complete evaluation. [7] Data flow testing is integrated as a first step and subjected for both individual and interacting with the member functions to contribute to child class and parent class analysis by hierarchical testing methodology. In the next phase a testing approach is included.

Object oriented programs are nothing but larger procedural components and their methods [4] an aspect based terminology is used in Object oriented fundamentals with modularizing and crosscutting concerns into object oriented programs are provided by vitally combined equipments. Fundamental communication elements and the developing rules are applied to different methods and are measured by more than one model. Some methods depend on software metric evaluation and it can be applied to programs by using various paradigms or in a multi paradigm environment. The verbal skill tactics is not related to the member's functions or language in some programming.

For detecting the programming errors the output as a sequence is generated in an effective way. In most of the cases it generates the sequence that covers the constraints at least once in an effective way of fault identification in the coding. [1] Testing techniques like totaling and checking the condition at the end of each statement and a test database or also the combination of both increases the bugs identification mechanism

The capability of judging the building phase of the software and its quality is measured by the program and it is the most important phase of the software construction life cycle. [6] The important complications in now a day's software practices are the density, information and control questions the software in a very effective way. In addition to this, program metrics was observed that initiates the tester to evaluate the non-recurring execution flow paths through the program that adds value to the value of the software.

4. MUTATION TESTING

A testing process in which the source code is analysed and test cases are capable of denoting the errors is often termed as mutation testing. Unit testing is done through this white box testing genre. The overall objective of the program should not be disturbed hence mutant program is kept small. To access the quality of the test cases is the principle goal of mutation testing this is also robust to fail mutant code. It is also denoted as fault based testing strategy which was proposed earlier in 1971. It was felt expensive and so lost its popularity. In this modern era it has regained its momentum and hence Java and XML largely depend on this mode. The substitution of simple code and syntactic operation is the key for it. Re execution of test cases in the mutated program for generating mutants that are un captivated previously is an unique feature of this mode of testing. [5].

5. EXISTING METHODOLOGY:

In the existing testing technique testing done by metrics includes the maintenance and management effort that is based on the evaluation of the system quality and the user interface management system[2]. Another inheritance procedure is used by iterating the testing data for a base class again and again and it is updated to the sub class. In the later stages they also utilize six type of metrics for the purpose of evaluation they are response for a class, coupling between objects, methods per class, number of sub class, lack of cohesion in between methods and the depth of inheritance tree.

DEMERITS OF EXISTING METHODOLOGY:

- This existing technique does not work well with the real time experiments.
- These techniques do not focus on all object oriented concepts other than inheritance
- Time consumption for examining the code is more in the existing techniques.

6. PROPOSED METHODOLOGY:

Selenium tool is used to test the web application and mutation testing is performed in the proposed methodology. A test complexity estimator is used to help the tester to evaluate how much testing is needed in this process and this also asses the process of retesting or verifying. The sub class methods in an object are counted and added to the difficulty evaluation process only when they are not examined at the parent class level.

7. ARCHITECTURE DIAGRAM:

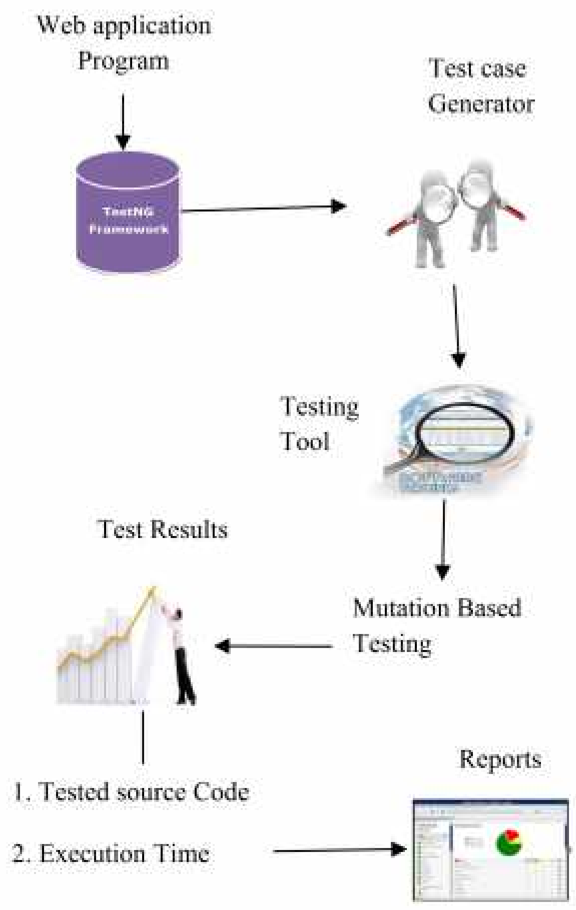

Fig 1 Architecture diagram of the Testing process

The Fig 1 explains clearly about the whole testing process carried out in this project.

7. MODULES IDENTIFICATION:

7.1 Design Oops Based Application Program:

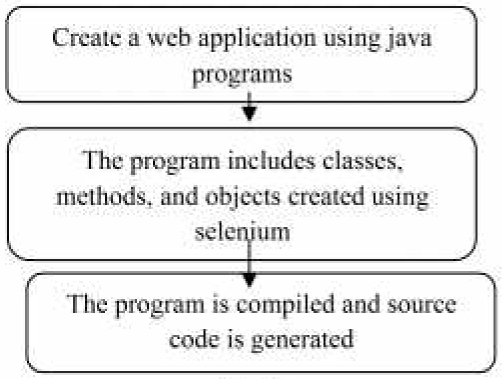
Create a web application using java programs
The program includes classes, methods, and objects created using selenium
The program is compiled and source code is generated

Fig 2.1

In fig 2.1 a web application is created using java with class, methods and objects using selenium tool. The program is complied and the source code ` generated to fulfill the needs of test case neration.

7.2 Testing: Mutation Based Method:

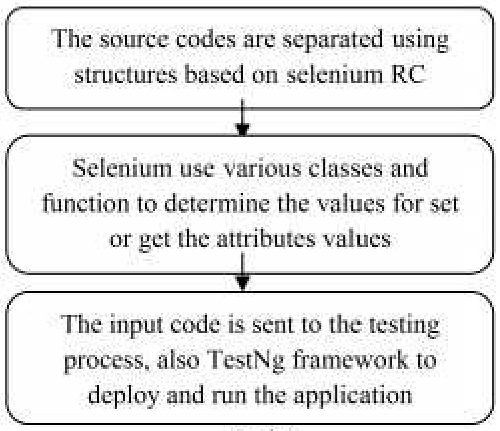
The source codes are separated using structures based on selenium RC
Selenium use various classes and function to determine the values for set or get the attributes values
The input code is sent to the testing process, also TestNg framework to deploy and run the application

Fig 2.2

Fig 2.2 explains that the generated source code is separated using structures based on selenium remote control. Selenium tool uses various classes and member functions to determine the values for getting and setting the attributes. The input code is sent to the testing process and also to the TestNg frame work to deploy and run the application.

7.3 Testing: Mutation Based Testing:

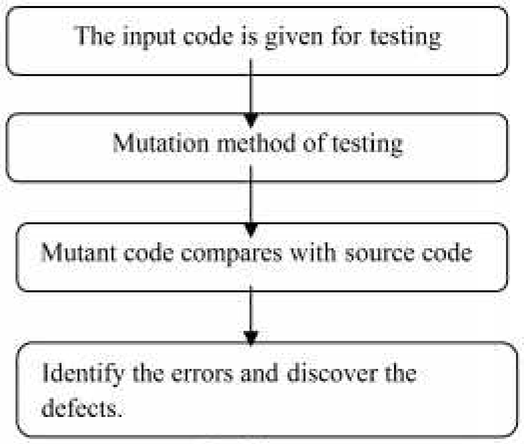
The input code is given for testing
Mutation method of testing
Mutant code compares with source code
Identify the errors and discover the defects.

Fig 2.3

Fig 2.3 depicts that mutation testing is the system testing in which the source code is deliberately manipulated by applying some mutant code with source code and the result is produced to find the errors. This method of testing to source code is to imitate common programming errors.

7.4 Software Metric Evaluation:

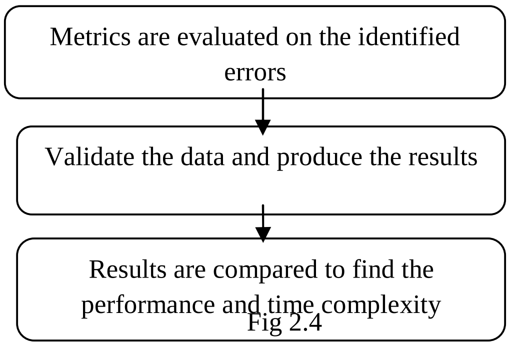
Metrics are evaluated on the identified errors
Validate the data and produce the results
Results are compared to find the performance and time complexity

Fig 2.4

In fig 2.4 evaluation of Software metric will validate the tested code and identify the errors in the code. This will improvise the quality assurance and development of new software. The results of the tested code will be generated and the performance of testing is also mentioned.

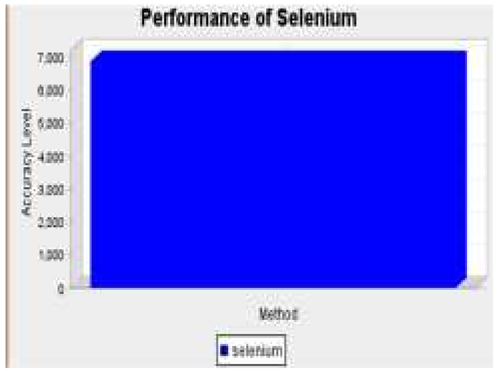

Fig 3 Performance of selenium

In Fig 3 is the output of the fourth module where performance of the selenium is obtained in the form of graph.

8. GRAPH:

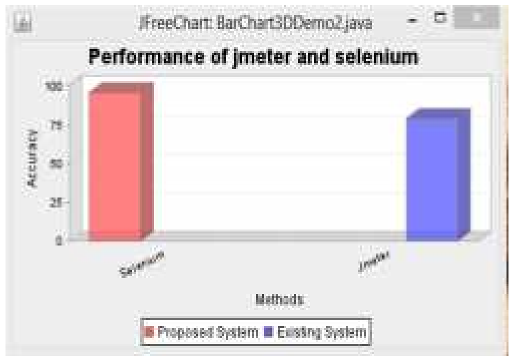

Fig 4 Comparision of selenium and jmeter

In Fig 4 is the final output of the process is the graphical representation where x-axis indicates the methods and y-axis indicates the accuracy, where the existing system is compared with the proposed system. In the existing system Jmeter is the tool which is used and in the proposed system selenium tool is used. The above graph depicts that selenium i.e., proposed system has greater accuracy than the existing system which uses Jmeter.

9. FUTURE ENHANCEMENT:

Selenium tool is used to test the web application in order to prove the efficiency of mutation testing to improve the proposed system. In the proposed system we have implemented selenium tool with mutation testing for the process of examination, in future some additions tools are utilized in mutation testing examination process to test the object oriented code.

CONCLUSION:

The final conclusion of the research work is hereby delivered that the automation testing like selenium and Jmeter takes less amount of time than manual testing but manual testing can be effectively fast enough to cope up with automation testing. Whereas the selenium performance is little faster than jmeter .The accuracy on both the testing is slightly enhanced by the selenium when compared to the jmeter . However The testing process utilized in this project is hereby concluded that it is quick, efficient, has more accuracy and also reduces the time complexity. This methodology increases the performance and reduces the maintenance of the software.

REFERENCES:

[1] S. Pasupathy, DR. R. Bhavani, ‘ Analysing The Efficiency Of Program Through Various OOAD Metrics’ ,Vol. 61 No.2 , March 2014.,JATIT.

[2] Ruchi Kulkarni, Samidha Diwedi Sharma , “Object Oriented Software Modularization Quality Measurement Based On API and Information Theoretic Metrics” , Vol. 01,Issue 04,pp 550-553 june 2014.

[3] Richard Baker ,Ibrahim Habli (2013), ‘An Empirical Evaluation of Mutation Testing for Improving the Test Quality of Safety-Critical Software’ , IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 39, NO. 6.

[4] Gao, J. San Jose State Univ., San Diego, CA, USA (2012), 'Cloud Computing Technology and Science (CloudCom)', 4th International Conference,pp.464-471

[5] Norbert Pataki, Adam Sipos, and Zoltan Porkolab" Measuring the Complexity of Aspect-Oriented Programs with Multiparadigm Metric" Supported by GVOP-3.2.2.-2004-07-0005/3.0. 2004

[6] Mary Jean Harold and John D. McGregor" Incremental Testing of Object-Oriented Class Structures"1989

[7] F.J. Daniels K.C. Tai" Measuring the Effectiveness of Method Test Sequences Derived from Sequencing Constraints" supported by a NASA and in part by NSF grant CCR-9320992.

[8] Wei li, salli henry" Object Oriented Metrics Which Predict Maintainability"TR-93-05.1993.

[9]S. Gambir, "Testing Strategies for Object Oriented Systems", IRACST - International Journal of Computer Science and Information Technology & Security (IJCSITS), Vol. 2, No. 2, pp. 459-462, April 2002.

[10] S. R. Chidamber, C. F. Kemerer, "A Metrics Suite for Object-Oriented Design", IEEE Transactions on Software Engineering, Vol.20, No. 6, pp. 476-492, June 1994.

[11] E. Weyuker, "Evaluating Software Complexity Measures", IEEE Transactions on Software Engineering, Vol. 14, No. 9, pp. 1357-1365, September 1988.

[12] T. JI. McCabe, "Design Complexity Measurement and Testing", ACM 32:1415-1425, 1989.

[13] S. C. Ntafos, "On Required Element Testing", IEEE Transactions on Software Engineering", SE-10(6):795-803, 1984.

[14] S. Rapps, E. J. Weyuker, "Selection Software Test Data using Data Flow Information, IEEE Transactions on Software Engineering, SE-11(4):367-375, 1985.

[15] W. Li and S. Henry, Object-oriented metrics that predict maintainability, J. Systems and Software 23:111-122, 1993.

[16] M. Barrold, J. D, McGregor and K. Fitzpatrick, "Incremental Testing of Object-Oriented Class Structure", Proceedings of the 14th International Conference on Software Engineering, pp. 68-80, 1992.