A GENERIC FRAMEWORK FOR SOFTWARE TESTING TECHNIQUES

Ms. V. Sathyavathy[1], Dr.D. Shanmuga Priyaa[2]

ABSTRACT

Software testing is an important role in the software development life cycle. There are various types of software testing techniques to ensure the quality of the product. So it is very much essential to find the right testing technique at right time. The proper usage of efficient testing technique at any stage of SDLC is very crucial. Selection of right testing technique at any stage is one of the crucial problems. Comprising of testing technique is a multi-criteria decision making problem and propose an efficient solution.

In this paper, a review is made on several past empirical studies along with their result, conducted with subjects to examine both functional and structural testing techniques. Finally, the overall performance of various testing techniques has been observed with respect to effectiveness and efficiency in existing experiments.

Index Terms : code reading, empirical study, evaluation, functional testing, structural testing

I. INTRODUCTION

Cryptography is art of hiding information or keeping the information secret from external environment Software testing is the process that involves various activities that aims at detecting the errors in the software product to improve the product quality through the entire lifecycle of software development. The objective of software testing is to find the faults and failures to ensure that the software is error-free. Inadequate and improper selection of testing leads to various software related issues. So it is important to choose effective software testing techniques.

The main objective of software testing is to detect and correct the fault that occurs during the software development. The improper and inadequate testing leads many issues. It is very important to create the error free software product to ensure the quality. For this purpose, testing techniques are to be selected it is likely to choose effective testing technique.

To evaluate the effectiveness and efficiency, the resource utilization and the parameters are taken into consideration. There are variety of software testing techniques available which includes lots of resources and time. Many testing techniques results in the same types of faults which leads to the duplication of efforts.Therefore,it is necessary to evaluate the effectiveness of software testing techniques.

II. CLASSIFICATION OF TESTING TECHNIQUES

Software testing techniques are divided into two broad categories: Static testing and dynamic testing.

[1]*Ph.D. Research Scholar, Karpagam University*

[2]*Professor, Department of Information Technology, Karpagam University*

A. Static Testing Techniques :

A static testing technique focuses on testing that tests the software product without the actual execution of source code. It includes the all the areas of representations of system such as requirement analysis, design and source code of the system without executing it. It is possible to do it either manual testing or automated testing.

Automatic testing is mainly based on testing program or program related documents by using software tools. In static automatic testing, static automated tool is used for testing the program; the source code is given as the input into this static tool and evaluated to ensure the quality. Manual testing is based on testing the program and program related document without using any types of tool.

B. Dynamic Testing Technique

Dynamic testing techniques focus on the execution of software to test the software product. The software product is tested in real environments, for the possible set of inputs, in order to validate how the system responds to different types of inputs.. A system is tested dynamically by examining the result of execution; the level of quality set for dynamic evaluation can be decided.

There are two categories of Dynamic testing techniques: White Box testing and Black Box testing

White Box Testing

In white box testing, source code knowledge is required for designing test cases. It concentrates on both the logic and internal structure of program code, and corresponds to the requirements of software which is under test. It is also called as structural testing. It mainly focuses on examining the logic of program or software product. In these types of testing techniques, test cases includes the coverage of code written in terms of branch coverage, conditions, statement coverage, and internal logic of program code etc.

To implement the white box testing method, knowledge of internal logical structure of the software systems must be taken and results are evaluated based on a set of coverage criteria. There are several types of white box testing techniques evaluated in the mentioned past empirical studies.

Statement Coverage: It requires that test cases to be designed in such a way that each statement in a program is executed at least once.

Branch Coverage: It requires to generate the test cases to cover each branch condition in the program that is both true and false value in turn.

Condition Coverage: It aims to design the test cases so as to apply each component of the conditional expression which returns either true or false value.

Loop Testing: It aims to design test cases so as to execute the loop continuously until the condition becomes satisfied or not.

Path testing: It aims to design test cases so as to execute and test all possible paths in the program at least once.

Black Box Testing

In Black Box testing, the source code knowledge is not required; test cases are designed from the analysis of input/output values only. Black Box testing affects only the behavior of software system. It is also called as functional testing. To implement this testing strategy, knowledge of functional specification of system must be given, so that all functionality of system is tested at least once. Black Box testing mainly based on testing functionalities and requirements of system. Functional testing becomes the most important factor to meet the required functionality and behavior. It specifies the external behavior according to the user requirements and specifications. Functional testing mainly focuses on "what" instead of "how". There are various types of functional testing methods to ensure the quality of the end product. Those tests are unit test, system tests, regression test and acceptance test

Black box testing can be classified as follows: Equivalence partitioning and boundary value analysis.

Equivalence partitioning: The equivalence classes for program are identified and test cases are generated for each equivalence class that is identified.

Boundary value analysis: This aims to design test cases using the values at boundaries of equivalence class that are identified

The overall process of evaluation of test design technique is represented in Fig.1.Prepare faulty program by injecting faults in unadulterated program. The experiment conducted with subjects, they applied different testing techniques to faulty program. The next step consists of evaluation of testing techniques, based on results achieved.

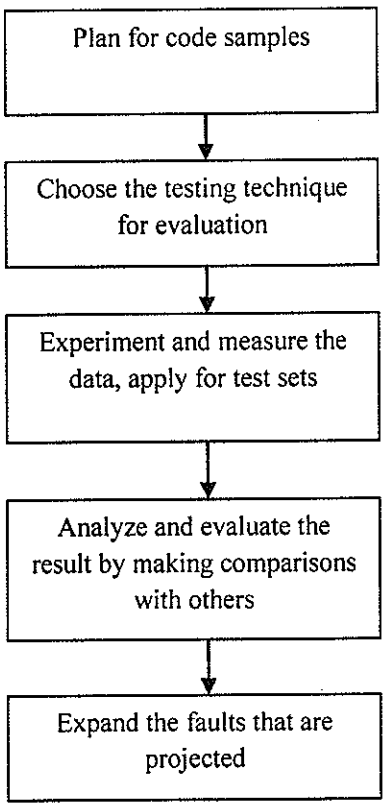

Figure 1: Overall process of testing technique

III. Measuring The Test Efficiency

Following is the method to calculate the efficiency of a testing process based on the number of defects by the Customer and to the number of defects identified by the Testing Team.

1. Assign Ranking to each Severity level.

a. if the type of error is Critical, the severity level 4

b. if it is Serious, the value which may be 3

c. if the error type is Moderate ,then assumed as 2

d. if the fault type is Minor then value 1 can be assigned

2. The list of defects reported by the Testing and Customer team based on Severity levels.

For Instance :

The Customer has Reported 3 Critical, 3 Serious, 2

Moderate and 5 Minor faults. The Testing Team should have identified these defects.

The Testing Team has Reported 5 Critical, 5 Serious, 10 Moderate and 10 Minor faults.

The Test Efficiency can be calculated as follows :

Tester's number of Defects = 4*5+5*3+10*2+10*1 = 65

Customer's number of Defects = 3*4+3*3+2*2+5*1 = 30

So Test Efficiency is (65/65+30) * 100 = 68.42%

In case if the Customer did not identified any fault or defects in the above example then the Test Efficiency will become 100%. If there is a small project in which the testing team and customer did not find any defects then also the Test Efficiency will be 100%. If the Testing Team Failed to find any Defects and the Customer finds them then the efficiency of the testing will be 0%.

IV. Measuring The Test Effectiveness

Software Test Effectiveness :

The effectiveness of a testing technique or a system is the ability to find the defects and isolate them, from a product or deliverable. Test effectiveness is to ensure quality of the software product and minimize the two quality gaps that are producer's quality gap and customer's quality. Software quality is both process and product quality which meets the customer requirements and conformance to product requirement specification. These metrics should be examined and quantified, as they closely likely relate to quality.

Software Test Effectiveness (for different factors):

Test effectiveness = Number of defects found / Number of test cases executed.

Test effectiveness = (total number of defects injected+ total number of defect found) / (total number of defect escaped)* 100

Test Effectiveness = Loss due to problems / Total resources processed by the system

Test effectiveness can be calculated for particular set of test activities.

For e.g. Test preparation efficiency will be the time taken for "X" number of Test cases to be prepared, reviewed and reworked to finalize them. There is a catch here, the quality standards of the test cases should be predefined by using defining standards, as I have tried to state a few of them here.

The Test cases are complete with respect to Use Cases on in which they are reffered.

A tester should be able to execute this test case.

Table: 1 Software Testing Levels

Testing Technique	Specification	Coverage	Tester
Unit Testing	Low level design	Small part of code	Developer
Integration Testing	High level and low level design	Large project with many modules	Deveoper
Functional	External Specification	Logic coverage criteria	External Tester
System	Analysis	The whole product	Developer / External Tester
Acceptance	Analysis	The whole product	End User
Beta	Situational	The whole product	End User

V. How To Chose Best Testing Technique

There are some **internal factors** that affect the decisions about which technique to use :

- **Models used in developing the system**– Testing techniques are mainly based on the models that are used to develop that system, which will say which technique to be chosen. For example, if the specification contains a state transition diagram, state transition testing would be an efficient technique.
- **Testers knowledge and their experience** – How much testers know about the system and about testing techniques will clearly influence their choice of testing techniques. This knowledge will in itself be influenced by their experience of testing and of the system under test.
- **Similar type of defects** – Knowledge of the same kind of defects will be very helpful in choosing testing techniques (each technique is better at finding a particular type of defect). This knowledge could be gained through experience of testing a previous version of the system and levels of testing which was tested on the current version.
- **Test objective** – If the test objective is used to gain confidence that the software will cope with typical operational tasks then use cases would be a useful approach. If the objective is for very complete testing then more crucial and detailed techniques (that includes structure-based techniques) should be chosen.
- **Documentation** –If the documentation is available or not (e.g. a requirements specification) exists and whether or not it is updated that will affect the choice of testing techniques. The content and style of the documentation will also influence the choice of techniques (for example, if decision tables or state transition graphs can be used then the associated test techniques should be used).
- **Life cycle model used** – A sequential life cycle model can be used that leads to the selection of more formal techniques whereas an iterative life cycle model may be well suited for using an exploratory testing approach.

There are some of the **external factors** that affects the decisions about which technique to use are :

- **Risk assessment** – if the risk is greater the risk (e.g. critical systems), the need for more

thorough and more formal testing will be greater. Technical risk may be influenced by quality issues (so more thorough testing would be needed) or by time-to-market issues (exploratory testing would be a more better choice).

- **Customer and contractual requirements** – Sometimes customers specify particular testing techniques to use (most commonly statement coverage or branch coverage techniques).
- **Type of system used** – The type of system (e.g. embedded, graphical, financial, etc.) will reflect the choice of techniques. For example, a banking application involving many calculations would benefit from boundary value analysis.
- **Regulatory requirements** – Some industries have professional standards or guidelines that exhibits the testing techniques used. For example, the mission critical application requires the use of equivalence partitioning, boundary value analysis and state transition testing for high integrity systems together with statement, decision or modified condition decision coverage depending on the level of software integrity required.
- **Time and budget of the project** – how much time and cost there is available will always affect the choice of testing techniques. When more time is available more techniques can be selected and when time is severely limited to those that is known have a good chance of helping us find just the most important defects.

Each technique is good in its own way in finding out the certain kind of defect, and not as good for finding out the other kind of defects. For example, one of the benefits of structure-based techniques is that they can find out the defects or things in the code.

Hetzel,conducted an experiment with 39 subjects for comparing effectiveness of three testing techniques i.e. functional testing, code reading and structural testing. The experiment was based on testing three program coded in PL/I. The result of experiment was that functional testing and structural testing was equal in effectiveness, while code reading was less effective.

Aspect	Experiment Results
Effectiveness	The subjects that applied the testing technique performed more effectively than those who applied the reading technique
	There are no much difference in the effectiveness

Figure 2

Roper et al. replicated the experiment of Kamsties and Lott The experiment had conducted with 47 subjects to evaluate the effectiveness of testing techniques and combination of techniques. The testing techniques were functional testing using boundary value analysis, structural testing using branch coverage and code reading by stepwise abstraction. The experiment used three program coded in C, to which testing techniques were applied. The result of the experiment was that effectiveness of techniques was dependent on program to which testing techniques were applied, and on the type of faults.

Aspect	Experiment Results
Effectiveness (Detection) combination of techniques	-Depends on the technique/program combination. -Depends on nature of faults. -Higher number of faults combining techniques

Figure 3: Roper's experiment on effectiveness

Farooq et al. [7] replicated the experiment of Kamsties and Lott [4] has compared the software testing techniques i.e. static testing techniques (code reading) and dynamic testing techniques(functional testing and structural testing). The experiment was conducted with 18 subjects which was based on testing three programs coded in C. The testing techniques has evaluated in terms of effectiveness, and efficiency.

Effectiveness has measured in terms of number of faults detected and isolated. Efficiency has measured in terms of time required to detect and isolate faults. The result of the experiment was that effectiveness depends on program. Efficiency depends on program and techniques.

Aspect	Experiment Results
Effectiveness (Detection)	Depends upon program, not on technique.
Effectiveness (isolation)	Depends upon program
Efficiency (detection)	Depends on program
Efficiency (isolation)	Depends on technique

Figure 4: Farooq's experiment on effectiveness

Juristo and Vegas replicated the experiment of Roper et [5] to evaluate the effectiveness of static testing techniques(code reading by stepwise abstraction) and dynamic testing techniques(functional testing using equivalence class partitioning and structural testing using branch coverage). The experiment was based on testing four program coded in C. They performed two replication of their experiment. In replication-I and replication-II the experiment was conducted with 195 subjects and 46 subjects respectively. The result of the experiment was that effectiveness of techniques depends on program, techniques and fault type. Functional testing and structural testing behave identically with respect to fault type. Code reading behave worse.

Aspect	Experiment Results
Effectiveness (Detected and observable)	1) Depends on technique, program and fault. 2) Code reading behaves worst than functional testing and structural testing, indistinctly for the defect type. With regard to functional testing and structural testing, both behaves identically. The number of subjects that detect a defect influence the program version

Figure 5 : Juristo's and Vegas's experiment on effectiveness

Empirical studies can be performed with subjects and without subject and based on inductive reasoning and logic. Empirical studies without subjects examine test-case generation and compare efficiency and effectiveness of different testing techniques. Some studies examine approaches for selection of test-case. Empirical studies with subjects is a simulation of real situation. It takes into account how the subject influence technique behaviour. Most of the studies conducted to evaluate static and dynamic testing techniques in terms of efficiency and effectiveness.

Analytical studies resembles with theoretical studies in nature and produce generalized results, the results which are applicable to any experimental perspective. The conclusions of analytical comparisons are based on statistical terms.

Selby computed the effectiveness and efficiency of three software testing techniques. The experiment was extended by including a fault isolation phase after fault detection phase

Aspect	Results of experiment
Effectiveness (Detection)	-Depends upon program, not on technique.
Effectiveness (isolation)	-Depends upon subject and program, not on technique.
Efficiency (Detection)	1) Condition coverage takes more time than boundary value analysis. 2) Time spent on finding faults also depends on subjects. 3) Condition coverage has lower fault rate than boundary value analysis.
Efficiency (isolation)	1) Depends upon subject and program, not on technique. 2) For inexperienced subjects: boundary value analysis takes longer than Condition coverage.
Efficiency (total) Fault type	1) For inexperienced subjects: Condition coverage takes more time than boundary value analysis 2) Time also depends on subject. -For both isolated and detected: there is no difference between techniques.

Figure 6: Selby's experiment on effectiveness

VI. Results

In this paper, the main results from above experiments are listed. From the above experiments it can be concluded that the functional testing is most effective than structural testing is less effective and code reading is least effective, in almost all experiments i.e.

CR<ST<FR(with respect to effectiveness).

FR- functional testing

ST- structural testing

CR- code reading

VII. Conclusion

Several past empirical studies are reviewed conducted with subjects, to evaluate the effectiveness and efficiency of testing techniques. The techniques includes both functional testing and structural testing. Functional testing includes boundary value analysis and equivalence partitioning and Structural testing (statement coverage, branch coverage condition coverage, loop and relational operator coverage) were evaluated.

Finally functional testing is most effective and efficient, structural testing is less effective and efficient and code reading is least effective and efficient.

In other words, static testing is less effective and efficient than dynamic testing

REFERENCES

[1] W. C. Hetzel, *"An experimental analysis of program verification methods,"* PhD thesis, University of North Carolina at Chapel hill, 1976.

[2] G. J. Myers, *"A controlled experiment in program and code walkthrough/ inspection,"* Communication of ACM, 21(9):760-768, September 1978.

[3] V. Basili and R. Selby, *"Comparing the effectiveness of software testing strategies. Software Engineering,"* IEEE Transcation on (12):1278-1296, 1987.

[4] E. Kamsties, and C. Lott, *"An empirical evaluation of three defect detection techniques,"* software Engineering ESEC95, pages 362-383, 1995.

[5] M. Roper, M. Wood, and J. Miller, *"An empirical evaluation of defect detection techniques,"* Information and Software Technology, 39(11): 736-775, 1997

[6] N. Juristo and S. Vegas, *"Functional testing, Structural testing and Code Reading: what fault type do they each detected?,"* Empirical Methods and Studies in Software Engineering Pages 208-232, 2003

[7] S. U. Farooq, S. M. K. Quadri and N. Ahmad, *"A controlled experiment to evaluate effectiveness and efficiency of three software testing methods,"* IEEE Conference on Software Testing, Verification and Validation, 2013

[8] T. Y. Chen, Y .T. Yu, *"On the relationship between partition and random testing,"*. IEEE Transaction on Software Engineering, 20(12): 977-980, 1994.

[9] A. J. Offutt, S. D. Lee, *"An empirical evaluation of weak mutation,"* IEEE Transactio on Software Engineering, 20(5):337-344, 1994

[10] A. J. Offutt, A. Lee, G. Rothermel, RH. Untch, C. Zapf, *"An experimentation determination of sufficient mutant operators,"* ACM Transaction on Software Engineering and Methodology, 5(2): 99-118. 1996.

[11] L.A. Clarke, A. Podgurski, D.J. Richardson, S.J. Zeil. *"A formal evaluation of data flow path selection criteria."* IEEE Transaction on Software engineering. 15(11): 1318 -1332, 1989.

[12] D. S. Rosenblum, E. J. Weyuker, *"Using coverage information to predict the cost effectiveness of regression testing strategies,"* IEEE Transaction on Software Engineering. 23(3): 146-156, 1997.

[13] G. Rothermel, M. J. Harrold, *"Analyzing regression test selection techniques,"* IEEE Transaction on Software Engineering. 22(8): 529-551, 1996.

[14] G. Rothermel, M. J. Harrold, *"Empirical studies of safe regression test selection technique,"* IEEE Transaction on Software Engineering. 24(6): 401-419, 1998.

[15] T. L. Graves, M. J. Harrold, J. Kim, A. Porter, G. Rothermel, *"An empirical study of regression test selection techniques,"* ACM Transaction on Software Enginering and Methodology, 10(2):184-208, 2001.

[16] N. Juristo, A. Moreno, and S. Vegas, "Reviewing 25 years of testing techniques experiments," Empirical Software Engineering, 9(1): 7-44, 2004.

[17] S. Eldh, H. Hansson, S. Punnekkat, A. Petterson and D. Sundmark, *"A framework for comparing efficiency, effectiveness and applicability of Software testing techniques,"* In testing: Academic and industrial conference –Practice and research technique,2006, TAIC PART 2006.proceedings, Pages 159-170, IEEE, 2006.

[18] N. Juristo, S. Vegas, M. Solari, S. Abrahao and I. Ramos, *"Comparing the effectiveness of equivalence partitioning, branch testing and code reading by stepwise abstraction applied by subjects,"* IEEE Conference on Software Testing, Verification and Validation, 2012.

Author's Biography

V. Sathyavathy is currently working as an Assistant Professor in the department of Computer Technology at KG College of Arts and Science, Coimbatore. She has completed her under graduation in Bachelor of Science in Computer Technology at Ramakrishna Engineering College, Coimbatore. She has completed her Post Graduation in Computer Application at KGISL-IIM.

Dr. D. Shanmuga Priyaa, working as a Professor in the department of Information Technology at Karpagam University Coimbatore. She had completed Ph.D in Computer Science at Karpagam University in August 2014 and she got 16 yrs of experience. At present she is guiding 8 Ph.D and 1 M.Phil scholars. She has published more than 15 international journals and more than 2 international conferences.