Journal of Information Technology Management

ISSN #1042-1319

A Publication of the Association of Management

AN EMPIRICAL FRAMEWORK FOR CHOOSING AN EFFECTIVE TESTING TECHNIQUE FOR SOFTWARE TEST PROCESS MANAGEMENT

SUDIP MISRA
CARLETON UNIVERSITY
smisra_@scs.carleton.ca

ABSTRACT

The choice of a software testing technique can have severe implications on the quality and efficiency of a testing project. One of the fundamental steps in managing a testing project should be selecting an effective testing technique. This paper discusses a methodology that can be used for experimentally assessing the effectiveness of two or more testing techniques in practice. The work should help quality managers and related researchers to learn how two or more testing techniques can be compared experimentally when one needs to choose an effective testing technique for a project. The *primary motivation* behind this paper is to advocate selecting an effective testing technique in practice in testing projects and providing a framework that can be readily adopted in practical testing projects. The testing techniques used to demonstrate the above are trivial in practice, and we did not intend to report the results of their comparisons here.

Keywords: software testing, empirical, test process.

INTRODUCTION

One of the important activities of test process management is ensuring that a testing strategy that is implemented into the project is efficient, effective in terms of factors such as cost, time, and the number of bugs that can be detected. Thus, the choice of a software testing technique in a testing project can have severe implications on both process and product quality. Selecting an effective testing technique is one of the key activities that a test management program should emphasize on during the initial stages of test planning. This paper discusses a framework that can be used for experimentally assessing the effectiveness of two or more testing techniques.

For testing of software there exists different techniques that have varied characteristics in terms of their effectiveness in uncovering faults, repeatability and cost. Some of the techniques are based on the white-box methodology of analysis of control flow of programs [9], [10], some on the analysis of data flow [12], [19], while some others on the analysis of all executable statements [1], [16]. In test process management, it is important to assess the effectiveness of different testing techniques, in terms of their ability to expose errors and also the size of test sets required by them. Although some work has been done to address this issue [2], [3], [4], [5], [7], [11], [17], there is no definitive and easy answer to the problem [5]. Most of the early studies in this regard have been using theoretical and formal techniques [3], [5], [17]. While theoretical results are indeed considered helpful, it is required to experimentally assess the different testing techniques. Some experimental studies have been done in this area [3], [4], [11] but the approaches mentioned in them are highly research-oriented with in-depth knowledge of Advanced Statistics required to apply them. Those litera-

tures do not provide a simple, easy-to-understand and implement, practitioner-oriented framework that can be readily used by test managers in the real-life projects, with little knowledge of advanced research methodologies. Also, further studies are required with more techniques and also to validate the results obtained so far.

The work involved, providing a simple and easy-to-use methodology that can be used in practice for experimentally comparing two or more testing techniques, in testing projects to see which technique would be more effective. For illustrating the use of the framework, two simple and widely accepted testing techniques, decision and all paths testing, have been studied. However, the methodology is not restricted to these techniques, and is generic enough to be used in other situations. Before proceeding further, it should be reminded to the readers that the *primary motivation* behind writing this paper is to advocate selecting an effective testing technique in practice in testing projects and providing a framework that can be readily adopted in practical test management projects. The intention behind writing this paper is *not merely* to show how decision testing compares with all paths testing. However, the results obtained in this paper should be helpful for those who are interested in applying these two testing techniques in a testing project and thus want to observe the empirical results of comparison of these techniques.

With the above motivation behind writing this paper, the experimental study was designed with the following specific four-fold goal:

To select an experimental methodology that allows the error-detecting ability of different testing techniques to be compared. It should be possible to use the experimental framework for assessing other testing techniques.

To use the experimental design to measure and compare the two testing techniques (decision and all-paths) for a few subject programs.

To study the relationship, if any, between coverage and effectiveness, using statistical correlation analysis technique.

To develop statistical regression models from the coverage, faults and size of test sets data, that should help in predicting the effectiveness from coverage information.

This study involving the comparison of the two testing techniques should help the testing practitioner to provide insight into the type of testing technique to be chosen. In other words, which technique should be adopted if the number of test cases required is not an important criterion, and on the other hand, if the number of faults to be detected is not important, which technique should be adopted.

This paper is organized in the following way. First we provide short descriptions of methodologies and techniques that are used in the study. Then we detail the design of the empirical study. Next, we describe the experimental results and the analysis of results obtained. Following that, the results are summarized and directions for future work are shown. Finally, the references used in this study are documented at the end of the paper.

OVERVIEW

Testing Techniques Considered

The testing techniques considered in this study are classified in the literature as *white-box,* because to generate the test cases for these techniques, a thorough understanding of the source-code of the programs are needed. The following describes, briefly, the two white-box techniques that have been used for comparison in this empirical study:

All-Paths Testing: This requires the execution of all possible paths that can exist in a program. Faults or defects are determined if the parts containing them have not been executed. The paths should have distinct branches from the start to end of a control flow graph of a program. Although thorough testing is possible using this technique, in practice, the number of such paths can be too large in large programs.

Decision Testing: This technique requires the coverage of all decision paths (i.e., both *true* and *false* points) for each decision point in a program. This technique uncovers which path(s) that have not been executed in a program and detects faults that arise from incorrect decision paths in a program.

Effectiveness of Testing Techniques

In this paper, the *effectiveness of* testing techniques is used to

(1) determine in terms of a testing technique's ability to detect errors at a certain stage in the testing process, and to,

(2) describe the cost, in terms of the number of test cases required to achieve coverage of a program at certain level.

The more *effective* a testing technique is at uncovering errors, the more confidence one can have in the delivered product that employed the technique. Similarly,

the higher the number of test cases required to achieve 100 % coverage, the lesser effective the technique is in terms of cost.

THE EXPERIMENTAL FRAMEWORK

The goal of the experiments conducted in this project was to obtain meaningful information about the effectiveness of the testing techniques to be compared and to measure and compare their effectiveness using different programs. In these studies, it is quite understandable that the programs used and the nature of faults embedded in them is important. They are described below.

Subject Program

The following are the brief descriptions about the *two* subject 'C' programs considered in this study. Table 1 provides a summary of the subject program. (In Table 1, LOC, refers to the number of lines of code in a program, NSTAT is the number of executable statements and NDEC is the number of decision statements).

The program consists of a function, *maths*, which performs some basic operations, such as square, square root, on inputs. The function returns a value, indicating the success-level of the operation performed.

Table 1: Subject Program

Prog	LOC	NSTAT	NDEC	Description
maths	68	15	7	Math functions

Fault Seeding / Fault Data

The fault space was restricted to artificially seeded faults into each of these programs. Faults were seeded manually into the subject programs. In the experiment, 35 faulty versions of the program, *maths*, were created. The results of this study should be interpreted, keeping in mind this limited representation of the faults and the size of the program considered. The faults were seeded using different 'C' mutation operators [8]. Some of the mutation operators considered are, "*while* replacement by *do-while*", "Statement Deletion", "*break* replacement by *continue*", "Scalar Variable Replacement", "Move Brace Up or Down", "Unary Operator Mutations" and "Binary Operator Mutations". Several types of faults were considered to make the data realistic to real-life situations where broad ranges of faults may occur.

Tools used

Two testing tools, Cantata (http://www.iplbath.com) and GCT (http://www.testing.com), were used in this study for measuring test coverage and also for executing test cases.

The Experimental Procedure

To following major steps were undertaken to measure the effectiveness of the two testing techniques considered in this study:

Generating a large number of faulty versions, or *mutants,* of each subject program one considers in the study, e.g., 35 mutants for *maths*. Table 2 summarizes the number of mutants, used in this study.

Table 2: Number of mutants

Base Program	Number of mutants
maths	35

Generating and executing test cases for each version or the mutant. Coverage was measured right from the beginning.

1. Checking the outputs and obtaining the values of all-paths coverage, decision coverage, percentage faults detected, and the number of test cases required.
2. Applying coverage: After the test cases were run at each step, the coverage values were increased in small amounts, by adding a few more test cases, until 100 % coverage is obtained.

The results obtained from the above study are summarized in Table 3. These results are then analyzed in a later section, for effectiveness of the criteria.

Investigating why certain mutants were not killed

This experimental process was carefully monitored to check whether all mutants were killed. If any mutants were not killed, it was investigated why they were not. A common cause for mutants remaining alive is that some mutants are equivalent to other mutants. This investigation was done to ensure that the testing results are complete and accurate.

DATA ANALYSIS

The data obtained from the experiments were the number of faults detected and the required number of test cases at different coverage level. Each subject program's data was treated as that of a separate experiment. The basic goal was to observe how fault detection and the number of test cases varied as coverage levels were increased towards 100%.

Methodology

Let N_c be the total number of coverage conditions satisfied by a testing technique and let X_c be the total number of coverage conditions satisfied by the technique. Then, the value, $P_c = X_c/N_c$, at any instance, is the proportion of the total number of coverage conditions satisfied by the technique. It is an estimator of the obtained coverage level, at any instance. Let N_f be the total number of faults in a program and F the number of faults detected at any instance. Then, the ratio $P_f = F/N_f$, is the estimator of the faults detected, at any instance. Similarly, let S be the size of the test set at any instance. The values, P_c, N_f, and S will be used as estimators to study the effectiveness.

Let the proportion of coverage conditions satisfied by all-paths testing be denoted by $P_{cp} = X_{cp}/N_{cp}$ while that of decision testing be denoted by $P_{cd} = X_{cd}/N_{cd}$. If for a particular value of P_c, the value of P_f is higher than its' value, the former technique will be supposed to be more effective than the latter in detecting faults, at that instance. On the contrary, if the size of test cases is lower for the former criterion than the latter criterion, the former will be supposed to be more effective in terms of number of test cases required than the latter.

Table 3 summarizes the results of the experiments. The obtained data were required to examine the relationships among the coverage level, size and fault detection attributes. Figures 1-2 show the plots for coverage and percentage faults detected and number of test cases required.

Table 3: Summarized Results

Sl No	P_{cd}	P_{cp}	P_{fd}	P_{fp}	S
1	0.16	0.26	0.2	0.32	2
2	0.30	0.42	0.36	0.51	3
3	0.58	0.55	0.59	0.68	5
4	0.66	0.74	0.71	0.73	6
5	0.72	0.88	0.75	0.76	8
6	0.89	0.96	0.82	0.81	9
7	1.00	1.00	0.91	0.91	10

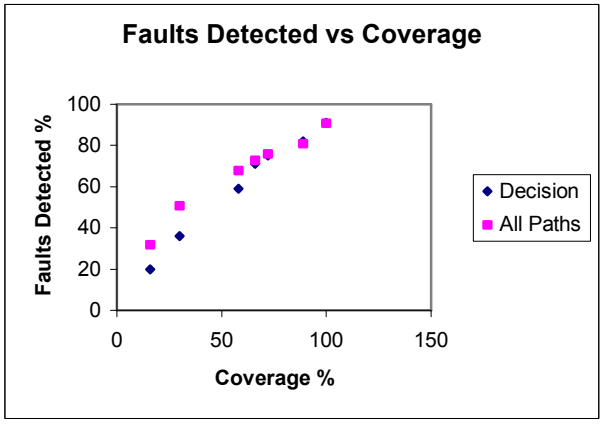

Figure 1: Fault Detection versus Coverage

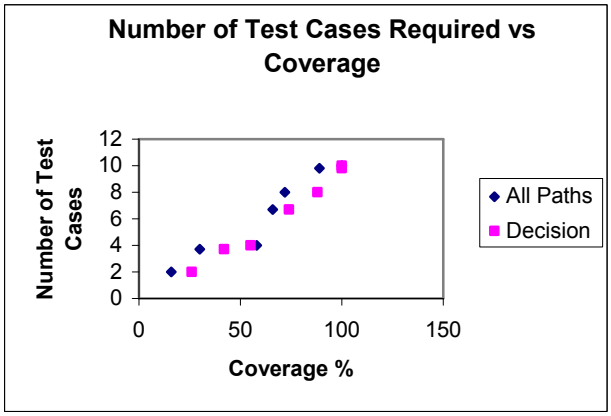

Figure 2: Number of Test Cases vs Coverage

From a close inspection of the plots in Figures and 2 we see that, for the example, in terms of the fault detection ability, all-paths coverage out-performs decision coverage until the value of 70-75%, after which both of them perform almost equally. However, at least for this example, if not for all possible examples the difference in performance levels is not too significant. In terms of size, the number of test cases covered by decision coverage is slightly less than that of all-paths coverage, although the difference is not too high. Considering the overall picture, we see that all-paths coverage is *more effective* than decision coverage for detecting faults; however, the number of test cases required to achieve 100 % coverage for decision testing is slightly less than that of all-paths testing.

Relationship between Coverage and Effectiveness

It is necessary to determine if there is a relationship between effectiveness and coverage, in other words, whether there is a relationship between the extent to which a test set satisfies statement or decision coverage and the probability that the test set will expose an error. Statistical *Correlation Analysis* technique is used for this purpose. The experimental data is then used to build *Least-Squares Regression Models*. Statistical Analysis was performed using SPSS ™ (http://www.spss.com).

If an increase (or decrease) of one variable (e.g. coverage) results in an increase or decrease in the other (e.g. percentage faults detected), the correlation is said to be *positive*. If the increase (or decrease) in one corresponds to decrease (or increase) in the other, the correlation is said to be *negative*. If there is no relationship indicated between the variables, the variables are *uncorrelated*. In this study, P*earson's Correlation Coefficient* is calculated. Higher the value of the correlation coefficient, the higher is the degree of association between the variables plotted and vice-versa.

A *Regression Model* gives the line of best fit for a given distribution of data. The line of regression of y on x is given by:

$$y-y' = r\,(\sigma_y/\sigma_x)\,(x-x')$$

Here x' and y' are the mean values of x and y respectively. The slope of this line is called the *Regression Coefficient* of y on x.

In this study, we assume *y* to be the % faults detected (denoted by *f*), *x* the coverage (denoted by *d*, for decision coverage and *s* for statement coverage), and *t* the size of test sets. The results of Correlation and Least-Square Regression Analysis are summarized in Tables 4 and 5 respectively. It can be observed from Table 4, that all the relationships were significant at the 5% level. The regression equations summarized in Table 5 will be helpful for predicting the number of faults or the number of test cases required when the coverage value is known.

Table 4: Results of Pearson's Correlation Analysis

Relationship	Correl Coeff	Stats. Signif?	Signif Level
Decision Coverage and faults detected	0.980	yes	1%
All-paths Coverage and faults detected	0.948	yes	1%
Decision Coverage and Size	0.997	yes	1%
All-paths Coverage and Size	0.990	yes	1%

Table 5: Regression Equations

Relationship	Regression Equation
Decision Coverage and faults detected	$f = 0.0133\ d^2 - 1.6638\ d + 118.385$
All-paths Coverage and faults detected	$f = 1.4\ p^2 - 171.31p + 12021.32$
Decision Coverage and Size	$t = -0.0022\ d^2 + 0.5119\ d + 0.059$
All-paths Coverage and Size	$t = -18.01p^2 + 60.19\ p + 4.98$

CONCLUSIONS

The paper provided a simple practitioner-oriented framework for experimentally assessing the effectiveness of two or more testing techniques in a testing project. This study experimentally evaluated and compared the effectiveness of two important and frequently used testing techniques, viz., decision testing and all-paths testing. Interesting results were obtained from the study, some of them supported the proposed hypotheses and some did not. The results are summarized as follows:

- Globally, all-paths coverage was found to be more effective than decision coverage in terms of the ability to detect faults.
- In terms of the number of test cases required, both of them performed very closely, although all-paths was slightly higher than decision.
- At higher ranges of coverage values (values approximately > 75 %), both all-paths and decision coverage performed almost equally.
- There was a high degree of positive correlation between effectiveness in terms fault detection or number of test cases required and percentage of coverage.
- Regression models for the different relationships shown in Table 6, should help one to predict faults when the value of coverage is known.
- None of the three techniques can guarantee 100 % fault detection at 100 % coverage levels. In both the programs there were many mutants that were not killed, even after achieving 100 % coverage values. Thus *100 % coverage doesn't guarantee fault-free software*.

It should be alerted here that, there were no assumptions made in this study about the characteristics of the sample programs. Thus it is not claimed that the results are representative of all kinds of software possible. Future work in this area may include, validating the results obtained with software having varied characteristics. Also, the promising and interesting results obtained in the paper should motivate readers to conduct other similar studies for experimentally comparing different pairs of testing techniques. It would be also interesting to study the cost in terms of time in employing the different testing techniques.

REFERENCES

[1] Beizer, B. *Software Testing Techniques*, 2nd edn., Van Nostrand Reinhold, 1990.

[2] Frankl, P. and Weiss, S. "An Experimental Comparison of the Effectiveness of Branch Testing and Data Flow Testing", *IEEE Transactions on Software Engineering* 1993, 19:8, pp. 774-787.

[3] Frankl, P. and Weyuker, E. J. "A Formal Approach of the Fault Detecting Ability of Testing Methods", *IEEE Transactions of Software Engineering*, 1993, 19:3, pp. 202-213.

[4] Frankl, P. "All-Uses versus Mutation Testing: An Experimental Comparison of Effectiveness", *Journal of Systems and Software*, Sept. 1997.

[5] Frankl, P. and Iakounenko, O. "Further Empirical Studies of Test Effectiveness", *ACM SIGSOFT Sixth International Symposium on the Foundations of Software Engineering*, Nov. 1998.

[6] Gottfried, B. *Programming with C*, Schaum's Outline Series, Tata McGraw-Hill Publishing Co., New Delhi, 1991.

[7] Hamlet, D. "Theoretical Comparison of Testing Methods", *Proceedings ACM SIGSOFT Third Symposium on Software Testing, Analysis, and Verification,* 1989; pp. 28-37.

[8] Agrawal H. et al., "Design of Mutation Operators for the C Programming Language", *Technical Report, SERC, Department of Computer Science, Purdue University*, 1989.

[9] Huang, J. "An Approach to Program Testing", *ACM Computing Surveys*, 1975, 7:3, pp. 113-128.

[10] Howden, W.E. "A Survey of Dynamic Analysis Methods, Tutorial: Software Testing and Validation Techniques", *IEEE Computer Society Press,* 1978, pp. 209-213.

[11] Hutchins et al., M. "Experiments on the Effectiveness of Dataflow- and Control flow-Based Test Adequacy Criteria", *Proceedings of the International Conference on Software Engineering*, IEEE ICSE-16, 1994.

[12] Laski, J.W. and Korel, B.A. "Data Flow Oriented Program Testing Strategy", *IEEE Transactions on Software Engineering,* 1983, SE-9:3, pp. 347-354.

[13] Marick, B. "Experience With the Cost of Different Coverage Goals for Testing", *Reliable Software Technologies*, 1997

[14] Marick, B. "How to Misuse Code Coverage", *Reliable Software Technologies*, 1997.

[15] Marick, B. The Craft of Software Testing, PTR, 1995.

[16] Myers, G. The Art of Software Testing, Wiley, 1979.

[17] Offutt, A.J. "Investigations of the Software Testing Coupling Effect", *ACM Transactions on Software Engineering Methodology*, 1992, 1:1, pp. 15-20.

[18] Pressman, R. *Software Engineering: A Practitioner's Approach*, 5th edn., McGraw Hill, New York, 2000.

[19] Rapps, S. and Weyuker, E.J. "Selecting Software Test Data Using Data Flow Information", *IEEE Transactions on Software Engineering*, 1985; SE-14:4, 367-375.

AUTHOR BIOGRAPHY

Sudip Misra earned his Ph.D. degree from Carleton University, in Ottawa, Canada. Prior to this, he received his Masters and Bachelors Degrees from the University of New Brunswick (Fredericton, Canada), and the Indian Institute of Technology (Kharagpur, India) respectively. He has several years of experience working in academia, government, and the private sectors. Dr. Misra has worked on R&D projects in project management, architecture, software design, and product engineering roles at Nortel Networks (Ottawa, Canada), Atreus Systems Corporation (Ottawa, Canada), and the Government of Ontario (Toronto, Canada).