The Effect of Code Coverage on Fault Detection under Different Testing Profiles

Xia Cai
Dept. of Computer Science and Engineering
The Chinese University of Hong Kong
xcai@cse.cuhk.edu.hk

Michael R. Lyu
Dept. of Computer Science and Engineering
The Chinese University of Hong Kong
lyu@cse.cuhk.edu.hk

ABSTRACT

Software testing is a key procedure to ensure high quality and reliability of software programs. The key issue in software testing is the selection and evaluation of different test cases. Code coverage has been proposed to be an estimator for testing effectiveness, but it remains a controversial topic which lacks of support from empirical data. In this study, we hypothesize that the estimation of code coverage on testing effectiveness varies under different testing profiles. To evaluate the performance of code coverage, we employ coverage testing and mutation testing in our experiment to investigate the relationship between code coverage and fault detection capability under different testing profiles. From our experimental data, code coverage is simply a moderate indicator for the capability of fault detection on the whole test set. However, it is clearly a good estimator for the fault detection of exceptional test cases, but a poor one for test cases in normal operations. For other testing profiles, such as functional testing and random testing, the correlation between code coverage and fault coverage is higher in functional test than in random testing, although these different testing profiles are complementary in the whole test set. The effects of different coverage metrics are also addressed in our experiment.

Categories and Subject Descriptors

D.2.5 [**Software Engineering**]: Testing and Debugging—*testing tools (e.g., data generators, coverage testing)*; D.2.8 [**Software Engineering**]: Metrics—*complexity measures, performance measures, product metrics*

General Terms

Measurement, Reliability

Keywords

Code Coverage, Fault Detection, Software Testing

A-MOST'05, May 15–16, 2005, St. Louis, Missouri, USA.

1. INTRODUCTION

As the main fault removal technique, software testing is one of the most effort-intensive activities during software development [1]. The key issue in software testing is test case selection and evaluation. An effective test set should detect software faults that do not easily lead to failure by other test cases. To improve the test resource allocation, code coverage has been proposed as an indicator of testing effectiveness and completeness for the purpose of test case selection and evaluation [9, 13, 14]. Code coverage is measured as the fraction of program codes that are executed at least once during the test. Various code coverage criteria have been suggested [8], including block coverage, decision coverage, C-use coverage and P-use coverage, etc.

However, it remains a controversial issue about whether code coverage is a good indicator for fault detection capability of test cases. Some previous studies show that high code coverage brings high software reliability and low fault rate [6, 8, 13, 15]. Such experimental data indicate that both code coverage and fault detected in programs grow over time, as testing progresses. For example, [4] observed this correlation between the code coverage and software reliability using experimentation with randomly generated flow graphs. In [17], it is reported that the correlation between test effectiveness and block coverage is higher than that between test effectiveness and the size of test set. [7] showed that an increase in reliability comes with an increase in at least one code coverage measures, and a decrease in reliability is accompanied by a decrease in at least one code coverage measures.

Furthermore, considering code coverage is a positive indicator for software reliability and quality, some researchers try to model the relationship between code coverage and code quality by hypergeometric distribution modeling [16] (under the assumption of a uniform probability and a random distribution of defects in the unit code, and independence between defects). Some suggest code coverage as an additional parameter for the prediction of software failures in operation [3]. Some model the relation among testing time, coverage and reliability altogether [11].

On the other hand, despite the observations of correlation existing in code coverage and fault coverage, a question is raised [2]: Can this phenomenon of concurrent growth be attributed to a causal dependency between code coverage and fault detection, or is it just coincidental due to the cumulative nature of both measures? A simulation experiment involving Monte Carlo simulation was conducted on the assumption that there is no causal dependency between code

coverage and fault detection. The testing result on published data did not support a causal dependency between code coverage and defect coverage.

Overall, the relationship between code coverage and fault detection is very complicated. More empirical data and theoretical insight are needed to explore the causal dependency between the two measures. In our previous work, we performed mutation testing and code coverage testing [10]. The results indicate that in most situations additional coverage of the code was achieved when the mutants were killed by a new test case. It observes that the increase in code coverage is related to more fault detections by a large portion (61.5%) among 21 program versions, although the range (from 22.2% to 94.7%) is very wide among different versions. In this paper, we will further study the relationship between code coverage and fault detection capability under different testing profiles.

The remainder of this paper is organized as follows. In Section 2, we hypothesize the effectiveness of code coverage varies under different situations. An experiment is set up as described to evaluate the actual performance of code coverage. The main discovery and data analysis are presented in Section 3, followed by further discussions in Section 4 and conclusions in Section 5.

2. EFFECTIVENESS OF CODE COVERAGE IN DIFFERENT TESTING PROFILES

As we have mentioned above, the relationship between code coverage and fault coverage is very complicated according to former empirical observations. The correlation between the two measures varies in different experiments, thus causing the question on whether a causal-effect dependency exists in code coverage and reliability. Both theoretical insight and empirical data are needed to clarity this question.

As code coverage is measured as the portion of program code, which is defined by different coverage criterion. The four popularly used code coverage criteria are: block coverage, decision coverage, C-use and P-use. The definitions for different coverage are given in [8]. We give brief descriptions of each as follows:

Block coverage is measured as the portion of basic blocks executed. Basic blocks are maximal code fragment without branching, containing no internal flow of control change;

Decision coverage is measured as the portion of decisions executed. A decision is a code fragment associated with a branch predicate.

C-use coverage is measured by computational uses covered. It refers to a pair of definition and computational use of a variable.

P-use coverage is measured by predicate uses covered. It refers to a pair of definition and predicate use of a variable.

From the definitions of these four coverage metrics, block coverage and C-use contain no control flow change while decision coverage and P-use are related to branch predicates.

According to previous work from others and ourselves, we notice that the effect of code coverage on fault coverage is positive in general. However, this correlation varies a lot in different reports. The intuitive reason of using code coverage as an indicator for software reliability is that code constructs not exercised during test may contain faults. But considering the requirements in specification, on the one hand, test set with additional code coverage is more effective in detecting faults; on the other hand, some test cases with less code coverage can still detect more program faults, when new code fragments are exercised which are not covered by other test cases.

Based on these considerations, we hypothesize that: 1) the effect of code coverage on fault detection varies if different testing profiles are examined; 2) different code coverage metrics may have influence on such correlation.

To investigate the above effect of different code coverage metrics under different testing profiles, empirical data are seriously needed. It requires a software project with bug history recorded, so that real faults can be studied, code coverage can be measured, test effectiveness can be quantified and test cases can be analyzed. Moreover, in such experiment, the development process should be controlled, the population of program versions should be large enough, and the application should be complicated as real-world projects in practice to ensure the software complexity.

Motivated by the lack of experimental data satisfying the requirement above, we conducted a experiment adopting the RSDIMU avionics application [10]. The application was part of the navigation system in an aircraft or spacecraft, and was first engaged in [5] for NASA-sponsored 4-university multi-version software project.

We employ mutation testing in our investigation. *Mutation testing* is one of the main schemes for test case selection and evaluation [12]. It starts with creating many versions of a program. Each version is "mutated" to introduce a single fault. These "mutant" programs are run against test cases with the purpose of causing each faulty version to fail. Each time when a test case causes a faulty version to fail, the mutant is considered "killed". An effective test case always kills more mutants than a less effective test case does.

Based on the detected software faults, we selected 21 program versions and created 426 software mutants, and conducted coverage testing [9] and mutation testing [12]. The contribution of each test case in block coverage of the total 426 mutants, measured across all executed mutants, is recorded and depicted in Figure 1. The decision, C-use and P-use coverage measures expose almost exactly the same pattern except for their absolute values, and thus omitted here.

The contribution of each test case in covering (killing) the mutant population is shown in Figure 2. Figure 1 and Figure 2 clearly portray certain patterns between block coverage and fault detection under six different test profiles, as delimited by A-E in the figures. On the one hand, test coverage and mutant coverage show similar capability in revealing patterns in the test cases. On the other hand, higher and more stable code coverage, e.g., that achieved by test cases 1001-1200, may result in lower and unstable fault coverage.

For the overall test set, the former 800 test cases are designed according to the specification, which are named as *functional testing*. To latter randomly generated 400 test cases are so-called *random testing*.

Other detailed descriptions of the test set as well as the experiment can be found in [10].

3. EXPERIMENTAL EVALUATION AND TESTING RESULTS

Based on our former experimental data, we further explore

the relationship between code coverage and fault detection capability for the current 1200 test cases which fall into six regions according to the various patterns revealed in Figure 1 and Figure 2. As described above, these test cases can be classified as functional testing (1-800) and random testing (801-1200). They can also be categorized by the system status: normal operation testing and exceptional operation testing. In this study, we examine the relationship in all these classifications and survey their similarities and differences.

To answer the question: Is code coverage a good indicator of fault detection capability? We investigate the statistical relationship between code coverage and fault coverage using linear regression model. In our experiment, each mutant stands for one real fault in the software development process. Thus the terms "fault" and "mutant" are used interchangeably in this paper.

In the following, we will examine the different relationship on three aspects: 1) the situations in overall test set and different regions; 2) the situations in functional testing versus random testing; and 3) the situations in normal operational testing versus exceptional operational testing.

3.1 The relationship revealed in different test case regions

As mentioned before, the former 800 test cases were designed to target different functions of the system, and the latter 400 test cases were randomly generated to simulate the operational environment. Moreover, as shown in Figure 1 and Figure 2, block coverage and fault coverage show different patterns in different parts on the whole test set. We divide the whole test set into six regions according to their patterns (see Table 1). These six clusters also reflect the underlying design principles of different test cases. After applying linear regression model on current data, we get the parameters and the quality of fit of linear models in various regions as well as in the whole test case space, as illustrated in Table 1. The results show that the relationship between block coverage and mutant coverage can be predicted by linear model at the whole test case space with the value of R-square of 0.781 (see Figure 3). But as a measure of the quality of fit, R^2 ranges dramatically from 0.189 (in Region IV) to 0.98 (in Region VI) in different test case regions, as shown in Figure 4 and Figure 5.

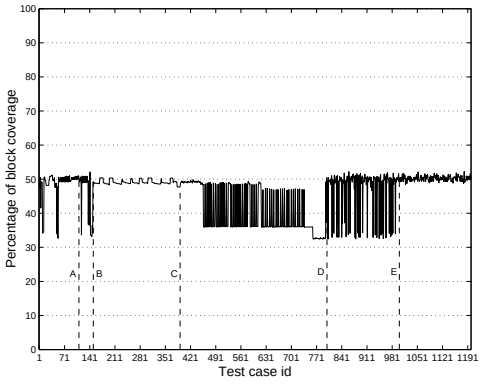

Figure 1: Test Case Contribution on Block Coverage

Figure 3 indicates that code coverage is a moderate indicator for fault detection capability of given test cases. Generally, the larger of the code coverage that a test case executes, the more mutants it kills in program versions. But different phenomenon can be observed if we view the whole figure as a combination of two clusters: one with block coverage at about 35% and mutant coverage at 90-150, and the other with block coverage at about 50% and mutant coverage at 150-270. In each cluster, the relationship between block coverage and mutant coverage is not always a positive correlation. Test cases with larger block coverage may kill less mutants, while test cases with smaller block coverage may cause more mutants to fail.

However, if we look into the linear regression relations

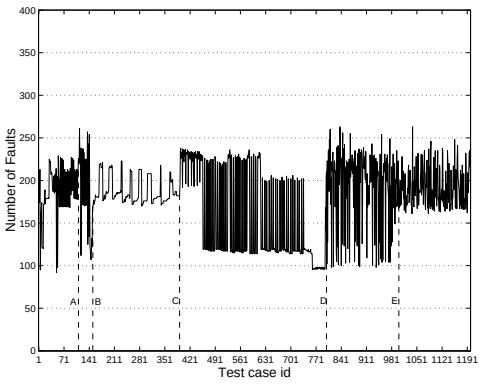

Figure 2: Test Case Contribution on Mutant Coverage

Table 1: Parameter and fitness of linear models in different test case regions

Test case region	R-square
Overall (1-1200)	0.781
Region I (1-111)	0.634
Region II (112-151)	0.724
Region III (152-392)	0.672
Region IV (393-800)	0.981
Region V (801-1000)	0.778
Region VI (1001-1200)	0.189

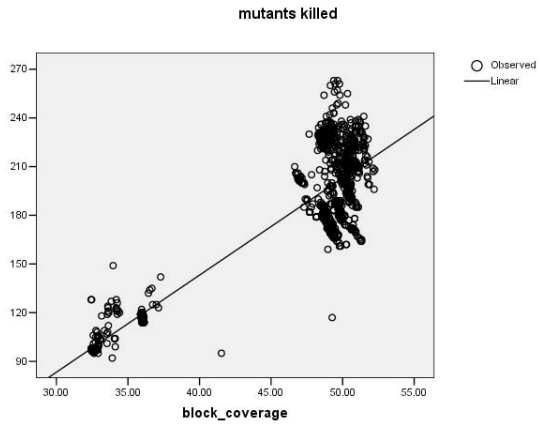

Figure 3: Linear Regression Relations between Block Coverage and Defect Coverage

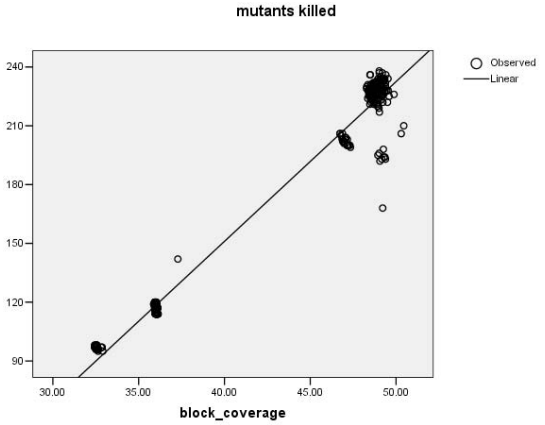

Figure 4: Linear Regression Relations between Block Coverage and Defect Coverage in Region IV

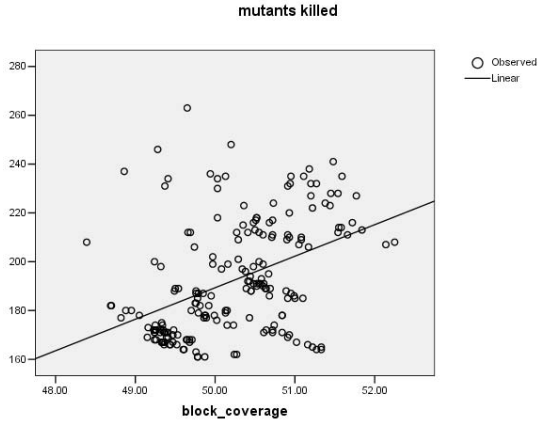

Figure 5: Linear Regression Relations between Block Coverage and Defect Coverage in Region VI

between block coverage and mutant coverage in each of the six regions, we can find the most fit in Region IV and the least fit in Region VI. Note that test cases in Region IV are designed with various combinations of the system status, while test cases in Region VI are randomly generated with a single initial random seed. One of the reasons behind this phenomenon may be because of the design principle of test cases in Region IV, which targets at the main control flow of the program. The more program code portion they execute, the more likely that program versions fail. This agrees with the traditional assumption and observation that more code coverage brings more fault coverage. The other reason lies that for Region VI, all the 200 test cases have very close block coverage (from 48% to 52%). It agrees with our earlier observation in two clusters: If the code coverage is in a small range, the linear correlation between code coverage and fault coverage may be insignificant. Furthermore, as shown in the latter analysis, we believe the strong correlation in Region IV lies in the fact that large number (277/373) of exceptional test cases contained in this region.

3.2 The relationship revealed in functional testing versus random testing

Functional testing and random testing are two basic methods employed in test case generation. In our test set, 800 test cases are functional test cases based on the basic operational requirements in the specification. The other 400 test cases are randomly generated with different seeds to simulate the large data set in real operations. The linear correlation in functional testing and random testing can be seen in Table 2. The correlation in functional testing is larger than that in random testing, but the difference is not significant. In general, functional test cases are designed to increase their code coverage (i.e., to cover more code fragments), while random test cases are generated to simulate real operational environment and not likely to improve code coverage. From our results, some functional test cases inherit the strong linear correlation between code coverage and fault coverage (e.g., in Region IV), while some random test cases show little correlation between the two measures (e.g., in Region VI). The underlying reason may be that there is no exceptional test cases in Region VI, while a large number of exceptional test cases (277 in Region IV while 373 in total test set) in Region IV. For another random test region, i.e., Region V, positive correlation is also observed with R^2 0.778 as there are 56 exceptional test cases in this region.

However, in average, the correlations between code coverage and fault coverage vary from 0.837 in functional testing to 0.558 in random testing. In both situations, code coverage is a moderate indicator for fault detection capability.

Table 2: R-square value in testing profiles

Testing profile(size)	R-square
Whole test set(1200)	0.781
Functional test(800)	0.837
Random test(400)	0.558
Normal test(827)	0.045
Exceptional test(373)	0.944

The effectiveness of random testing has been a controversial [17]. For the question whether random testing is an effective testing approach, we can see some positive signs from our statistical data. First, although random test cases are not designed to improve code coverage, they can still achieve similar code coverage as those functional test cases, e.g., the similar code coverage (around 50%) obtained in Region VI as that in Region IV. Secondly, random test can kill mutants whose faults are hard to detect, i.e., with small number of failure occurrence. If we examine the failure details of mutants that failed at less than 20 test cases (which means these mutants inherit low failure occurrence), we find that there are 94 random test cases and 169 functional test cases that can detect these faults. The percentage 35.7% ($\frac{94}{94+169}$) shows that random test cases are effective to detect hard-to-kill mutants as well as functional test cases. The numbers and failure occurrence of mutants that failed in only functional testing as well as in random testing are listed in Table 3. The figures indicate that there are 382 mutants killed in functional testing and 371 mutants killed in random testing. Among all these mutants, 362 mutants failed at both testing, 20 mutants (with mean failure number of 4.5) killed by functional testing only and nine mutants (with 3.67 failures in average) failed at random test cases only. This means that random testing may miss 5.2% (20/382) faults compared with functional testing, but it kills 2.4% (9/371) ad-

ditional faults which are not detected by functional testing. These nine newly-killed mutants inherit pretty low failure occurrence.

Table 3: The failure number of mutants that failed in different testing

Test case type	Mutants killed	Mean failure number	Std. deviation
Functional testing	20/382	4.50	3.606
Random testing	9/371	3.67	2.236
Normal testing	36/371	120.00	221.309
Exceptional testing	20/355	55.05	99.518

Overall, random testing is a necessary complement to functional testing. Code coverage is still a good indicator of fault detection capability for functional as well as random test cases.

3.3 The relationship revealed in normal operational testing versus exceptional testing

Test cases are designed to detect and remove residual faults in program versions which are developed to satisfy the requirements in the software specification. There are two major system status according to the specification: normal operation and exception handling. A test set should contain test cases designed according to these two system operation scenarios to hit all kinds of faults. The classification of normal and exceptional status is application-dependent and defined by the specification. Particularly in RSDIMU application, normal operations refer to those situations where at most two sensors fail as the input and at most one sensor fails during the test. All the other cases, which cause the difficult conditions where acceleration of the instrument unable to be estimated, are viewed as exceptional operations.

As shown in Table 2, the linear correlation of code coverage and fault coverage changes dramatically from normal testing (0.045) to exceptional testing (0.944). It clearly indicates the strong correlation of the two measures in exceptional testing, but no correlation in normal testing, as seen in Figure 6 and Figure 7, respectively.

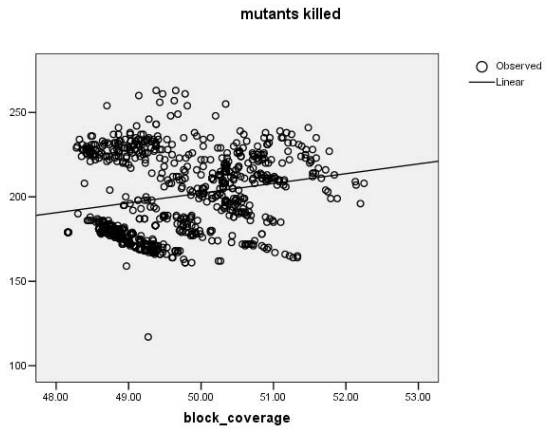

Figure 6: Linear Regression Relations between Block Coverage and Defect Coverage in normal testing

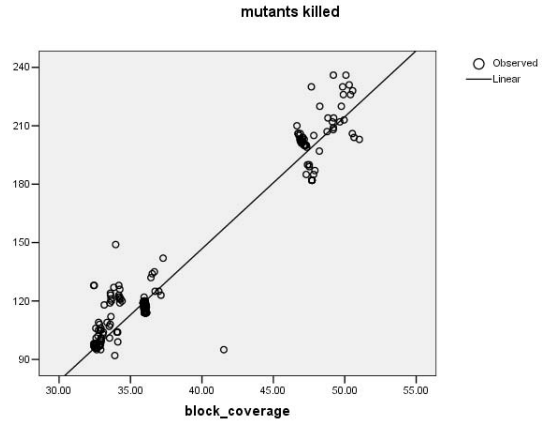

Figure 7: Linear Regression Relations between Block Coverage and Defect Coverage in exceptional testing

In normal testing, the code coverage range is relatively small (see Figure 6), between 48% and 52%. This agrees with the design principle of normal test cases. The normal operations should execute the major part of the program versions. In such a situation, although high code coverage may be obtained, it cannot be employed to predict the fault detection capability of a normal operational test case. On the contrary, in the case of exceptional testing, the value of R-square of 0.944 indicates an obvious positive correlation between code coverage and fault coverage.

Figure 7 contains two main clusters. We examine the exceptional test cases and find that these two clusters are caused by the specific application. Because of the complexity of the RSDIMU application, some functions such as acceleration estimation, contain large-scale computational code. In some exceptional cases, part of these functions can be executed but others be skipped (e.g., When four sensors on exactly two faces have failed before the test, and no additional sensor fails during the test); while in other cases, all these computational codes are skipped according to the system status. This explains why the code coverage shows two different ranges and a big gap exists between the two clusters. Although this phenomenon is application specific, the strong correlation pattern provides a positive support for the code coverage. We postulate that even in other applications, since different exceptional test cases simulate different exceptional situations, a variation of code coverage are achieved although the ranges of code coverage may be larger or smaller compared with our application. Test cases with higher code coverage are likely to detect more faults, i.e., the correlation between code coverage and fault coverage may still hold. Of course, this needs further empirical investigation.

According to Table 3, the mutants killed by exceptional testing only fail less frequently (with 55 failures in average) than those failed at normal testing only (with 120 failures in average). Considering the total numbers of test cases in normal testing and exceptional testing are 827 and 373, the normalized failure occurrences of these two classes of mutants are similar (120/827 vs. 55/373). Normal testing can detect more faults than exceptional testing (371 vs. 355), yet it contains larger test set than exceptional testing.

Table 3 also reveals that mean failure numbers under functional testing and random testing are significantly different from those under normal testing and exceptional testing. It may imply the different features and relationship among the four testing profiles. Functional testing (which designed according to the specification) and random testing (which designed according to operations) have a big overlap. Most cases under the two testing profiles can detect similar faults. Only a small amount of function-specific faults or faults under some extreme situations can be detected by functional testing or random testing only. But for normal testing and exceptional testing, the two testing profiles are parallel, i.e., they contain no overlap. A fault only occurring under normal operations may fail at many normal test cases, but it cannot be detected by exceptional testing, and vice versa. The different features and relationship among testing profiles can also explain the various patterns they inherit in terms of the correlation between code coverage and fault coverage: there is a similarity between functional testing and random testing, but a major difference between normal testing and exceptional testing.

In summary, both normal operational testing and exceptional testing are important for software testing. But code coverage is clearly a good indicator of fault detection capability of exceptional test cases, rather than normal test cases. This can also give some hints on designing the exceptional test cases: increasing the code coverage of such test cases will gain benefits on fault detection capability.

3.4 The relationship revealed in different combinations for various coverage metrics

From the data shown above, we observe that the effect of code coverage on fault coverage is significant in exceptional testing, while weak in normal testing. The difference between functional testing and random testing is not obvious, but still code coverage is a moderate indicator for test effectiveness. To further illustrate such effect, we examine the correlation pattern in different testing profile combinations. The linear regression fit in the four combinations are listed in Table 4. It is shown that the combinations containing exceptional testing (random/exceptional and functional/exceptional) indicate strong correlation, while the combinations containing normal testing (random/normal and functional/normal) inherit weak correlation.

Table 4: Linear Regression Fitness for combinations

Testing Combination	R-Square
random & normal	0.045
random & exceptional	0.949
functional & normal	0.076
functional & exceptional	0.950

To investigate the correlation pattern between different code coverage metrics and test effectiveness under various testing profiles, the R-square values of linear regression for decision coverage, C-use and P-use are listed in Table 5, compared with that of block coverage. The other three coverage metrics show similar patterns as block coverage. There is an insignificant difference between block coverage/C-use and decision coverage/P-use under normal testing. One possible reason may be that the variation of decision coverage and P-use coverage are larger under normal operations, as they are related to the control flow change in the program code. According to our previous observation, larger variation in code coverage implies more correlation in terms of the relationship among different clusters.

Table 5: R-square value in different code coverage and testing profiles

Testing profile(size)	block coverage	decision coverage	C-use	P-use
Whole test set(1200)	0.781	0.832	0.774	0.834
Functional test(800)	0.837	0.880	0.830	0.881
Random test(400)	0.558	0.646	0.547	0.648
Normal test(827)	0.045	0.368	0.019	0.398
Exceptional test(373)	0.944	0.952	0.954	0.954

4. DISCUSSIONS

Based on our project data, we investigate the effect of different code coverage metrics under different testing profiles. we focus on the following two questions: 1) Does the effect of code coverage on fault detection vary under different testing profiles? 2) Do different code coverage metrics have various effects on such relationship?

For the first question, based on above experimental data, our answer is supportive. The correlation varies under different testing profiles. In particular, there is a significant correlation between code coverage and fault detection capability for exceptional test cases. Positive linear correlation holds with an overall R-square of 0.944. The relationship shows no correlation for normal operational test cases. The phenomenon of different correlation revealed in different test case regions can be explained by the effect under exceptional testing. The strong positive correlation in Region IV is caused by large number (277/373) of exceptional test cases contained in this region.

On the other hand, code coverage implies fault detection capability moderately in both functional testing and random testing. The difference between the two testing profiles is not obvious.

For the second question, we cannot give conclusive answer according to our data. The correlation pattern seems similar for all coverage metrics under various testing profiles. There is a small discrepancy between block coverage/C-use and decision coverage/P-use. It may be caused by the control flow diversion related to decision predicate. But as the difference is not statistically significant, we cannot tell whether the coverage metrics have influences on the concerned correlation.

As our project data are based on RSDIMU application, which is computation-intensive, the size of some functions is very large compared with other applications. We find that there is a gap between the coverage of different exceptional test cases, which is determined by the execution of these functions. It is the reason behind the two clusters shown in some of the patterns. As RSDIMU is a real-world application from critical avionics industry, the correlations and patterns that are observed in our experiment should be representative to a certain degree. However, since this is only a case study of investigation, further real-world empirical data are still needed.

The significance of the clear positive correlation in exceptional testing is that it can provide guidelines for selection

and evaluation of exceptional test cases. Test cases with high code coverage tend to detect more faults, although it does not necessarily mean that test cases with low coverage are useless. For functional testing, test cases with low coverage may detect faults related to specified operations. For random testing or operational testing, code coverage can estimate the fault detection capability for exceptional test cases.

5. CONCLUSIONS

Software testing is a key procedure to ensure high quality and reliability of software programs. The key issue in software testing is the selection and evaluation of different test cases. Code coverage has been proposed to be an estimator for testing effectiveness, but it remains a controversial topic and lack of support from empirical data.

In this paper, we employ coverage testing and mutation testing to investigate the relationship between code coverage and fault detection capability for test cases selection and evaluation purpose. A unique contribution of our work is an innovative approach is establishing the relationship according to different testing profiles. We conduct a large-scale project with real-world application to address such relationship with different coverage metrics under different testing profiles. From our experimental data, code coverage is a moderate indicator for the capability of fault detection on the whole test set. The effect of code coverage on fault detection varies under different testing profiles. The correlation between the two measures is high with exceptional test cases, while weak in normal testing.

Furthermore, there is no sign on various influence of different coverage metrics. All the four coverage metrics show similar patterns on the linear relationship between code coverage and fault detection. Moreover, the data support the effectiveness of random test cases due to its significant fault detection capability.

The new finding about the effect of code coverage on fault detection can be used to guide the selection and evaluation of test cases under various testing profiles, although this still needs supports and evaluations from more empirical data.

6. ACKNOWLEDGMENTS

The work described in this paper was fully supported by a grant from the Research Grants Council of the Hong Kong Special Administrative Region, China (Project No. CUHK4205/04E).

The authors would like to greatly appreciate Lorenzo Strigini for his detailed discussions in both the technical content and research direction that shape up this paper.

7. REFERENCES

[1] B. Beizer. *Software Testing Techniques*. Van Nostrande Reinhold Co., New York, 1990.

[2] L. Briand and D. Pfahl. Using simulation for assessing the real impact of test coverage on defect coverage. *IEEE Transactions on Reliability*, 49(1):60–70, March 2000.

[3] M. H. Chen, M. R. Lyu, and W. E. Wong. Effect of code coverage on software reliability measurement. *IEEE Transactions on Reliability*, 50(2):165–170, June 2001.

[4] M. H. Chen, A. P. Mathur, and V. J. Rego. Effect of testing techniques on software reliability estimates obtained using time domain models. In *Proceedings of the 10th annual software reliability symposium*, pages 116–123, Denver, Colorado, June 1992.

[5] D. E. Eckhardt, Caglavan, Knight, Lee, McAllister, Vouk, and Kelly. An experimental evaluation of software redundancy as a strategy for improving reliability. *IEEE Transactions on Software Engineering*, 17(7):692–702, July 1991.

[6] P. G. Frankl and E. J. Weyuker. An applicable family of data flow testing criteria. *IEEE Transactions on Software Engineering*, 14(10):1483–1498, October 1988.

[7] F. D. Frate, P. Garg, A. P. Mathur, and A. Pasquini. On the correlation between code coverage and software reliability. In *Proceedings of the 6th International Symposium on Software Reliability Engineering*, pages 124–132, Toulouse, France, October 1995.

[8] J. R. Horgan, S. London, and M. R. Lyu. Achieving software quality with testing coverage measures. *IEEE Computer*, 27(9):60–69, September 1994.

[9] M. R. Lyu, J. R. Horgan, and S. London. A coverage analysis tool for the effectiveness of software testing. *IEEE Transactions on Reliability*, 43(4):527–535, December 1994.

[10] M. R. Lyu, Z. Huang, K. S. Sze, and X. Cai. An empirical study on testing and fault tolerance for software reliability engineering. In *Proceedings 14th IEEE International Symposium on Software Reliability Engineering (ISSRE'2003)*, pages 119–130, Denver, Colorado, November 2003.

[11] Y. K. Malaiya, M. N. Li, J. M. Bieman, and R. Karcich. Software reliability growth with test coverage. *IEEE Transactions on Reliability*, 51(4):420–426, December 2002.

[12] J. Offutt and S. D. Lee. An empirical evaluation of weak mutation. *IEEE Transactions on Software Engineering*, 20(5):337–344, May 1994.

[13] S. Rapps and E. J. Weyuker. Selecting software test data using data flow information. *IEEE Transactions on Software Engineering*, 11(4):367–375, April 1985.

[14] S. K. Sze and M. R. Lyu. ATACOBOL: A COBOL test coverage analysis tool and its applications. In *Proceedings of the 11th International Symposium on Software Reliability Engineering (ISSRE'2000)*, pages 327–335, San Jose, California, October 2000.

[15] E. J. Weyuker. The cost of data flow testing: an empirical study. *IEEE Transactions on Software Engineering*, 16(2):121–128, February 1988.

[16] T. W. Williams, M. R. Mercer, J. P. Mucha, and R. Kapur. Code coverage: what does it mean in terms of quality? In *Proceedings of the Annual Reliability and maintainability Symposium*, pages 420–424, Philadelphia, PA, USA, January 2001.

[17] W. E. Wong, J. R. Horgan, S. London, and A. P. Mathur. Effect of test set size and block coverage on the fault detection effectiveness. In *Proceedings of the 5th International Symposium on Software Reliability Engineering*, pages 230–238, Monterey, CA, November 1994.