Appendix To: Coverage Is Not Strongly Correlated with Test Suite Effectiveness

Laura Inozemtseva and Reid Holmes
School of Computer Science
University of Waterloo
Waterloo, ON, Canada
lminozem,rtholmes@uwaterloo.ca

ABSTRACT

This document expands on the brief discussion of related work in the paper *Coverage Is Not Strongly Correlated With Test Suite Effectiveness*. Specifically, it discusses a number of studies that considered the relationship between test suite size, test suite coverage and test suite fault detection ability.

1. RELATED WORK

Most of the previous studies that investigated the link between test suite coverage and test suite effectiveness used the following general procedure:

1. Created faulty versions of one or more programs by manually seeding faults, reintroducing previously fixed faults, or using a mutation tool.
2. Created a large number of test suites by selecting from a pool of available test cases, either randomly or according to some algorithm, until the suite reached either a pre-specified size or a pre-specified coverage level.
3. Measured the coverage of each suite in one or more ways, if suite size was fixed; measured the suite's size if its coverage was fixed.
4. Determined the effectiveness of each suite as the fraction of faulty versions of the program that were detected by the suite.

Table 1 summarizes thirteen studies that considered the relationship between the coverage and the effectiveness of a test suite, eleven of which used the general procedure just described. Ten of them found that at least one type of coverage has some correlation with effectiveness; however, not all studies found a strong correlation, not all studies controlled for suite size, and most studies found that the relationship was highly non-linear. In addition, some found that the relationship only appeared at very high levels of coverage.

The earliest work on this topic was done by Frankl and Weiss [8,9], who wanted to determine the relative effectiveness of all-use adequate suites and decision adequate suites. As a control, the authors generated random suites that contained approximately the same number of test cases as the adequate suites. As subjects, the authors used nine Pascal programs, ranging from 22 to 78 SLOC in length, that had naturally occurring faults. Suite coverage was measured with the ASSET tool [11]. The authors found that all-use adequate suites were often more effective than decision adequate suites, and decision adequate suites were often more effective than suites without a coverage criterion. However, these results did not control for the effect of test suite size. The authors therefore grouped the suites by size to compare the effectiveness of suites of similar size that were constructed in different ways. When size was fixed, they found that all-use adequate suites were still more effective than decision adequate suites; however, they found that decision adequate suites were no more effective than random suites. This suggests that all-use coverage is related to effectiveness independently of size but decision coverage is not. Finally, the authors tried measuring the effectiveness of test suites with less than 100% coverage. They found that coverage was somewhat correlated with effectiveness for both coverage types but that the correlation was not particularly strong and the relationship was highly non-linear. This indicates that, even when coverage is independently related to effectiveness, the quality of a test suite should not be assumed to be proportional to its level of coverage.

Frankl et al. [10] later extended this study to mutation adequate suites, comparing them with all-use adequate suites and random suites. The subjects were the same nine Pascal programs used in the earlier study. However, the mutation tool the authors used, Mothra [6], only works with Fortran programs, so the nine subjects were also ported to Fortran. Mutants were generated from the Fortran programs with Mothra; all-use coverage was measured on the Pascal programs with ASSET. The authors found no significant difference between mutation-adequate suites and all-use adequate suites: both were more effective for some of the programs, and for two of the programs there was no significant difference between the two criteria. However, the authors reported that mutation-adequate suites were much more difficult to make. Following this, the authors tried to isolate the effect of size by generating suites with a fixed number of test cases. They found that, while effectiveness improved with coverage, it generally did not improve until very high levels of coverage were reached (greater than 80%), and even then effectiveness did not increase by much.

ICSE '14, May 31–June 7, 2014, Hyderabad, India

Table 1: Summary of the findings from previous studies.

Citation	Languages	Largest Program	Coverage Types	Findings
[8, 9]	Pascal	78 SLOC	All-use, decision	All-use related to effectiveness independently of size; decision is not; relationship is highly non-linear
[10]	Fortran Pascal	78 SLOC	All-use, mutation	Effectiveness improves with coverage but not until coverage reaches 80%; even then increase is small
[7]	C	5,905 SLOC	All-use, decision	Effectiveness is correlated with both all-use and decision coverage; increase is small until high levels of coverage are reached
[21]	C	<2,310 SLOC	Block	Effectiveness is more highly correlated with block coverage than with size
[15]	C	512 SLOC	All-use, decision	Effectiveness is correlated with both all-use and decision coverage; effectiveness increases more rapidly at high levels of coverage
[4]	C	4,512 SLOC	Block, c-use, decision, p-use	Effectiveness is moderately correlated with all four coverage types; magnitude of the correlation depends on the nature of the tests
[1]	C	5,000 SLOC	Block, c-use, decision, p-use	Effectiveness is correlated with all four coverage types; effectiveness rises steadily with coverage
[17]	C C++	5,680 SLOC	Block, c-use, decision, p-use	Effectiveness is correlated with all four coverage types but the correlations are not always strong
[12, 20]	C Java	72,490 SLOC	AIMP, DBB, decision, IMP, PCC, statement	Effectiveness correlated with coverage; effectiveness correlated with size for large projects
[2]	C	4,000 SLOC	Block, c-use, decision, p-use	None of the four coverage types are related to effectiveness independently of size
[13]	Java	$O(100,000)$ SLOC	Block, decision, path, statement	Effectiveness correlated with coverage across many projects; influence of project size unclear

Frankl and Iakounenko [7] later extended Frankl and Weiss's work [8, 9] to a C program with 11,640 LOC. Faulty versions of the program were created by reintroducing faults that had been previously found and fixed. Test suites of various sizes were created by randomly selecting test cases from a large test case pool. The ATAC tool [16], which accounts for the use of pointers in C, was used to measure the suites' all-use and decision coverage. The authors found that, when the number of test cases in the suite was fixed, effectiveness was correlated with both decision coverage and all-use coverage. This contradicts their earlier result that decision coverage is not independently related to effectiveness; however, they again found that high levels of coverage were needed to see a large increase in effectiveness.

Wong et al. [21] also studied this topic, evaluating ten C programs with a total of 2,310 SLOC. Test suites were generated by random selection from a pool; their block coverage was measured with ATAC. Faulty versions of the subjects were made by asking graduate students to manually inject faults. The authors found that effectiveness was more highly correlated with block coverage than with the number of test cases in the suite; in other words, knowing the block coverage of a suite tells you more about its effectiveness than knowing its size does.

Hutchins et al. [15] studied this question using the Siemens suite of seven C programs ranging from 141 to 512 SLOC. Faulty versions were generated by asking ten experienced programmers to manually seed faults. The authors used two different coverage measurements: decision coverage and a slight variant of all-use coverage that accounts for the use of pointers in C. These were both measured with the Tactic tool [18]. The authors found that effectiveness does rise with increased coverage, but that high coverage does not guarantee a high level of effectiveness. They suggest instead that low coverage should be taken as a sign that the test suite is inadequate, even if it seems comprehensive. They also found that effectiveness increased more rapidly when coverage was already at a very high level; in other words, increasing the coverage of the test suite from 40% to 50% had less impact than increasing it from 90% to 100%. While this paper did not focus on the influence of suite size, the authors did note that suites with high coverage tend to be large and that this may be a confounding factor. They therefore took test suites with coverage in the 90 to 100% range, generated random suites that contained the same number of test cases, and compared the fault detection ability of these two types of suites. The high coverage suites were quite a bit more effective, suggesting that decision coverage and all-use coverage are both related to effectiveness independently of size.

Cai and Lyu [4] studied this question using 21 C programs that ranged from from 1,455 to 4,512 SLOC. These programs were written by fourth year university students as part of a software engineering course. Faulty versions of the programs were made by reintroducing faults that had been found and fixed during development. Before a program was accepted

at the end of the course, it had to pass 1,200 test cases, 800 of which were designed using the program's operational specification and 400 of which were randomly generated. These test cases were reused on the mutated versions of the programs during the study. The authors used ATAC to measure block, decision, p-use and c-use coverage. They found that effectiveness was moderately correlated with all four types of coverage; however, they found that the magnitude of the correlation depended on the nature of the test case. Specifically, the authors found that the correlation between coverage and effectiveness was somewhat higher for the 800 operational test cases than it was for the 400 random test cases, though the difference was not statistically significant. In addition, the correlation was quite a bit stronger for test cases designed to test erroneous situations than for those designed to test normal operation.

A later comprehensive study by Andrews et al. [1] addressed a number of questions related to mutation testing. They studied a C program with approximately 5,000 SLOC and used ATAC to measure four different kinds of coverage: c-use, p-use, decision and block. Test suites were generated in two different ways. One set of suites was generated by randomly selecting test cases from a pool. The other set was also created by random selection from a pool, but with the added condition that every test case increase the overall coverage. In other words, if a randomly selected test case did not increase the suite's coverage, it was not added; a different test case was selected instead. Plotting effectiveness against the number of test cases in the suite for both types of suites and all four coverage measures showed that suites built to maximize coverage were more effective. The authors confirmed this result by running a regression analysis with size and coverage as covariates, revealing that both were statistically significant and showed a positive regression coefficient. This suggests that coverage is related to effectiveness independently of size. In contrast to the results described earlier, the authors found that effectiveness rose steadily with coverage, meaning that it may be valid to assume that effectiveness is proportional to coverage.

A recent study by Namin and Andrews [17] worked with the Siemens suite of seven C programs, which range from 137 to 513 SLOC. The suite includes a comprehensive test case pool for each program, so the authors generated test suites by randomly selecting test cases from this pool. The authors used Proteum [5] to generate mutants and ATAC to measure four different types of coverage: block, decision, c-use and p-use. They found that, for all coverage types, both coverage and the number of test cases in the suite independently influenced effectiveness, but the correlations were not always very strong. Using linear regression, they found that the best predictor of effectiveness was a combined measurement: $\log(\text{size}) + \text{coverage}$. The authors then attempted to replicate their findings on two additional programs, one in C with 5,680 SLOC and one in C++ with 966 SLOC. They found that their results held for these programs, suggesting that their conclusions may generalize to slightly larger programs.

A study by Briand and Pfhal [3] was the one of two experiments that did not use the procedure outlined earlier. Instead, they proposed a general statistical method to determine the relationship between coverage, size and effectiveness. Specifically, they suggested using Monte Carlo simulation to estimate the effectiveness of a test suite, but without using any information about the coverage of the suite. This estimate is then compared to the actual effectiveness. If the difference between the two is statistically significant, then coverage has influenced effectiveness. The authors tested this method on data from an earlier study [14] that used twelve versions of a C program ranging from 900 to 4,000 SLOC. This study measured decision, block, c-use and p-use coverage with ATAC. Using their statistical method, Briand and Pfhal found that none of the four coverage measures influenced the effectiveness of the test suites independently of the number of test cases in the suite.

At the time of writing, no other study considered any subject program larger than 5,905 SLOC[1]. However, a recent study by Gligoric et al. [12] and a subsequent master's thesis [20] partially addressed this issue by studying two large Java programs (JFreeChart and Joda Time) and two large C programs (SQLITE and YAFFS2) in addition to a number of small programs. The authors created test suites by sampling from the pool of test cases for each program. For the large programs, these test cases were manually written by developers; for the small programs, these test cases were automatically generated using various tools. Suites were created in two ways. First, the authors specified a coverage level and selected tests until it was met; next, the authors specified a suite size and selected tests until it was met. They measured a number of coverage types: statement coverage, decision coverage, and more exotic measurements based on equivalent classes of covered statements (dynamic basic block coverage), program paths (intra-method and acyclic intra-method path coverage), and predicate states (predicate complete coverage). They evaluated the effectiveness of each suite using mutation testing. They found that the Kendall τ correlation between coverage and mutation score ranged from 0.452 to 0.757 for the various coverage types and suite types when the size of the suite was not considered. When they tried to predict the mutation score using suite size alone, they found high correlations (between 0.585 and 0.958) for the four large programs with manually written test suites but fairly low correlations for the small programs with artificially generated test suites. This suggests that the correlation between coverage and effectiveness in real systems is largely due to the correlation between coverage and size; it also suggests that results from automatically generated and manually generated suites do not generalize to each other.

A study by Gopinath et al. [13] accepted to the same conference as the current paper was the second study that did not use the aforementioned general procedure. The authors instead measured coverage and test suite effectiveness for a large number of open-source Java programs and computed a correlation across all programs. Specifically, they measured statement, block, decision and path coverage and used mutation testing to measure effectiveness. The authors measured these values for approximately 200 developer-generated test suites – the number varies by measurement – then generated a suite for each project with the Randoop tool [19] and repeated the measurements. The authors found that coverage is correlated with effectiveness across projects for all coverage types and for both developer-generated and automatically-generated suites, though the correlation was stronger for developer-written suites. The authors found that including test suite size in their regression model did not improve the

[1]In this paper, source lines of code (SLOC) refers to executable lines of code, while lines of code (LOC) includes whitespace and comments.

results; however, since coverage was already included in the model, it is not clear that this implies that size does not influence effectiveness. This finding may be an artifact of multicollinearity, since the amount of variation 'explained' by a variable will be less if it is correlated with a variable already included in the model than it would be otherwise.

As the above discussion shows, it is still not clear how test suite size, coverage and effectiveness are related. Most studies conclude that effectiveness is related to coverage, but there is little agreement about the strength and nature of the relationship.

2. REFERENCES

[1] J. H. Andrews, L. C. Briand, Y. Labiche, and A. S. Namin. Using mutation analysis for assessing and comparing testing coverage criteria. *IEEE Transactions on Soft. Eng.*, 32(8), 2006.

[2] L. Briand and D. Pfahl. Using simulation for assessing the real impact of test coverage on defect coverage. In *Proc. of the Int'l Symposium on Software Reliability Engineering*, 1999.

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

[4] X. Cai and M. R. Lyu. The effect of code coverage on fault detection under different testing profiles. In *Proc. of the Int'l Workshop on Advances in Model-Based Testing*, 2005.

[5] M. E. Delamaro and J. C. Maldonado. Proteum – a tool for the assessment of test adequacy for C programs. In *Proc. of the Conf. on Performability in Computing Systems*, 1996.

[6] R. A. DeMillo, D. S. Guindi, W. M. McCracken, A. J. Offutt, and K. N. King. An extended overview of the Mothra software testing environment. In *Proc. of the Workshop on Software Testing, Verification, and Analysis*, 1988.

[7] P. G. Frankl and O. Iakounenko. Further empirical studies of test effectiveness. In *Proc. of the Int'l Symposium on Foundations of Soft. Eng.*, 1998.

[8] P. G. Frankl and S. N. Weiss. An experimental comparison of the effectiveness of the all-uses and all-edges adequacy criteria. In *Proc. of the Symposium on Testing, Analysis, and Verification*, 1991.

[9] P. G. Frankl and S. N. Weiss. An experimental comparison of the effectiveness of branch testing and data flow testing. *IEEE Transactions on Soft. Eng.*, 19(8), 1993.

[10] P. G. Frankl, S. N. Weiss, and C. Hu. All-uses vs mutation testing: an experimental comparison of effectiveness. *Journal of Systems and Software*, 38(3), 1997.

[11] P. G. Frankl, S. N. Weiss, and E. J. Weyuker. *ASSET: A System to Select and Evaluate Tests.* Courant Institute of Mathematical Sciences, New York University, 1985.

[12] M. Gligoric, A. Groce, C. Zhang, R. Sharma, M. A. Alipour, and D. Marinov. Comparing non-adequate test suites using coverage criteria. In *Proc. of the Int'l Symp. on Soft. Testing and Analysis*, 2013.

[13] R. Gopinath, C. Jenson, and A. Groce. Code coverage for suite evaluation by developers. In *Proc. of the Int'l Conf. on Soft. Eng.*, 2014.

[14] J. R. Horgan, S. London, and M. R. Lyu. Achieving software quality with testing coverage measures. *Computer*, 27(9), 1994.

[15] M. Hutchins, H. Foster, T. Goradia, and T. Ostrand. Experiments of the effectiveness of dataflow- and controlflow-based test adequacy criteria. In *Proc. of the Int'l Conf. on Soft. Eng.*, 1994.

[16] 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), 1994.

[17] A. S. Namin and J. H. Andrews. The influence of size and coverage on test suite effectiveness. In *Proc. of the Int'l Symposium on Software Testing and Analysis*, 2009.

[18] T. J. Ostrand and E. J. Weyuker. Data flow-based test adequacy analysis for languages with pointers. In *Proc. of the Symposium on Testing, Analysis, and Verification*, 1991.

[19] Randoop. `https://code.google.com/p/randoop/`.

[20] R. Sharma. Guidelines for coverage-based comparisons of non-adequate test suites. Master's thesis, University of Illinois at Urbana-Champaign, 2013.

[21] 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 *Proc. of the Int'l Symposium on Software Reliability Engineering*, 1994.