Correlation between Test Suite Effectiveness and Code Coverage

ICSE 2014

Fahmid Morshed Fahid
North Carolina State University
ffahid@ncsu.edu

ABSTRACT

The effectiveness of a test suite is often measured with coverage and used as a proxy for its ability to detect faults. But unfortunately, previous studies failed to reach a consensus about the nature and strength of such relationship.

To study the relationship between test suite size, coverage and effectiveness, Inozemtseva et al. [4] used large Java programs with 31000 test suits for five systems consisting of up to 724,000 lines of source code. They measured different types of coverage and used mutation testing to evaluate their fault detection effectiveness.

They found that, there is a low to moderate correlation between coverage and effectiveness when the number of test cases in the suite is controlled for. Also, stronger forms of coverage do not provide greater insight into the effectiveness of the suite. Their results suggest that, coverage, while useful for identifying under-tested parts of a program, should not be used as a quality target because it is not a good indicator of test suite effectiveness.

KEYWORDS

Universal Defect Prediction Model, Test Suite Coverage, Bug

1 INTRODUCTION

Testing is an important part of producing high quality software, but its effectiveness depends on the quality of the Test Suite. Testing textbooks often recommend coverage as one of the metrics for measuring effectiveness of a test suite.

But unfortunately, previous works found contradictory results. For example, Gligoric et al. [2] found that, effectiveness of each test suite using mutation testing has a Kendall's τ correlation from 0.452 to 0.757 when size of the suite is not considered and from 0.585 to 0.958 when considered. On a different experiment, Gopinath et al. [3] found that, coverage is correlated with effectiveness across project with for all coverage types and suite size do not improve the results.

Due to different assumptions such as small or synthetic programs, confounding influence of test suite size and using impractical adequate suites, no consensus was found in code coverage as measure of effectiveness in a test suite. That's why, Inozemtseva et al [4] proposed three research questions.

- **RQ1:** Is The Size Of A Test Suite Correlated With Effectiveness?
- **RQ2:** Is The Coverage Of A Test Suite Correlated With Effectiveness When Suite Size Is Ignored?
- **RQ3:** Is The Coverage Of A Test Suite Correlated With Effectiveness When Suite Size Is Fixed?

Table 1: Important characteristics of the subject programs.

Property	Apache POI	Closure	HSQLDB	JFree Chart	Joda Time
Total Java SLOC	283,845	724,089	178,018	125,659	80,462
Test SLOC	68,832	93,528	18,425	44,297	51,444
Statement coverage	67%	76%	27%	54%	91%
Decision coverage	60%	77%	17%	45%	82%
MC coverage	49%	67%	9%	27%	70%
# mutants	27,565	30,779	50,302	29,699	9,552
# detected mutants	17,835	27,325	50,125	23,585	8,483
Equivalent mutants	35%	11%	0.4%	21%	11%

2 METHODOLOGY

2.1 Terminology

- **Master suite:** Test suite written by developers of a subject program. The all other suites are strict subsets of this suite.
- **Mutant and Equivalent Mutant:** A mutant is a new version of the program that is created by making a small syntactic change to the original program. If resulting mutant produce the same output, it is called an equivalent mutant.

2.2 Subject Program

They have used a number of criteria to select these projects.

- Large projects on Java (on the order of 100,000 SLOC), actively developed.
- Large number of test methods (on the order of 1,000).
- Ant as a build system and JUnit as a test harness, to automate data collection.

2.3 Generate Faulty Program

Inozemtseva et al. [4] used the open source tool **PIT** [1] to generate large number of mutants and run the programâĂŹs test suite on each one. If the test suite fails when it is run on a given mutant, the suite is said to kill that mutant, else it is equivalent. A test suiteâĂŹs mutant coverage is the fraction of mutants that it kills.

2.4 Generate Test Suite

For each program, they have identified all of the test methods in the master suite. Then generated new test suites of fixed size by randomly selecting a subset of these methods without replacement.

Foundations of Software Science, Fall 2018, Computer Science

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$9999.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

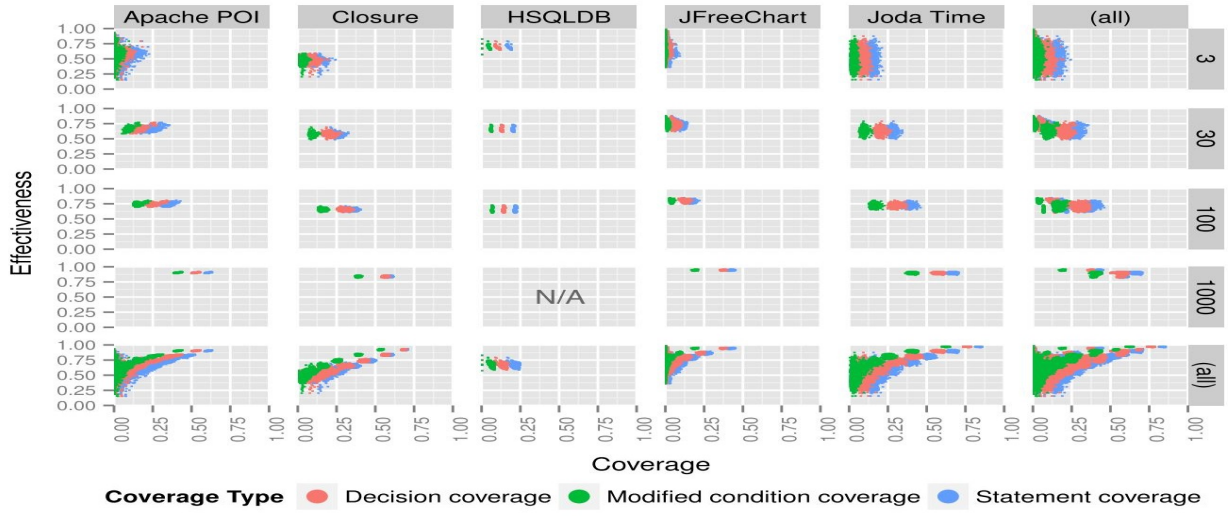

Figure 1: Normalized effectiveness scores (left axis) plotted against coverage (bottom axis) for all subjects. Rows show the results for one suite size; columns show the results for one project.

They made 1,000 suites for the following sizes: 3 methods, 10 methods, 30 methods, 100 methods, and so on. This resulted in a total of 31,000 test suites.

2.5 Measuring Coverage

Inozemtseva et al. used the open source tool **CodeCover** [5] to measure coverage. They used two effectiveness measurements in this study: normalized and non-normalized. The normalized effectiveness measurement is the number of mutants a test suite detected divided by the number of non-equivalent mutants it covers. For non-normalized results, see [4]

3 RESULTS

RQ1: Is size correlated with effectiveness?
There is a moderate to very high correlation between the effectiveness of a test suite and the number of test methods it contains (Pearson's r value range of 0.51 to 0.97).

RQ2: Is Coverage Correlated With Effectiveness When Size Is Ignored?
There is a moderate to high correlation between the effectiveness and the coverage of a test suite when the influence of suite size is ignored. For normalized measurement, the Kendall's τ ranges from 0.50 to 0.80.

RQ3: Is Coverage Correlated With Effectiveness When Size Is Fixed?
The correlation between coverage and effectiveness drops when suite size is controlled for, typically ranges from low to moderate (from 0.1 to 0.4 in most cases), meaning it is not safe to assume that effectiveness is correlated with coverage. Additionally, Table 2 shows that, the type of coverage used had little influence on the strength of the relationship.

Table 2: The Kendall τ and Pearson correlations between different types of coverage for all suites from all projects.

Coverage Types	Kendall's τ	Pearson's r
Statement/Decision	0.92	0.99
Decision/MCC	0.91	0.98
Statement/MCC	0.92	0.97

4 CONCLUSION

They studied the relationship between the number of methods in a programâĂŹs test suite, statement, decision, and modified condition coverage, and mutant effectiveness measurement. These results imply that high levels of coverage do not indicate that a test suite is effective. Consequently, using a fixed coverage value as a quality target is unlikely to produce an effective test suite. In addition, complex coverage measurements may not provide enough additional information about the suite.

REFERENCES

[1] Henry Coles, Thomas Laurent, Christopher Henard, Mike Papadakis, and Anthony Ventresque. 2016. Pit: a practical mutation testing tool for java. In *Proceedings of the 25th International Symposium on Software Testing and Analysis*. ACM, 449–452.
[2] Milos Gligoric, Alex Groce, Chaoqiang Zhang, Rohan Sharma, Mohammad Amin Alipour, and Darko Marinov. 2015. Guidelines for coverage-based comparisons of non-adequate test suites. *ACM Transactions on Software Engineering and Methodology (TOSEM)* 24, 4 (2015), 22.
[3] Rahul Gopinath, Carlos Jensen, and Alex Groce. 2014. Code coverage for suite evaluation by developers. In *Proceedings of the 36th International Conference on Software Engineering*. ACM, 72–82.
[4] Laura Inozemtseva and Reid Holmes. 2014. Coverage is not strongly correlated with test suite effectiveness. In *Proceedings of the 36th International Conference on Software Engineering*. ACM, 435–445.
[5] T Scheller. 2008. CodeCover. http://codecover. org. *Retrieved on* 12 (2008), 31.