# Coverage Is Not Strongly Correlated with Test Suite Effectiveness

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, mostly due to small or synthetic programs, confounding influence of test suite size and using impractical adequate suites.

To study the relationship between test suite size, coverage and effectiveness, Inozemtseva et al. [1] used large Java programs with 31000 test suits for five systems consisting of up to 724,000 lines of source code. They measured the statement coverage, dicision coverage and modified condition coverage of these suites and used mutattion testing to evaluate their fault detection effectiveness.

Inozemtseva et al. [1] 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, Rank Transformation, Bug

## 1 INTRODUCTION

In this study, the authors propose a context-aware rank transformation to address the variations in the distribution of predictors before fitting them to the universal defect prediction model. They used 21 code metrics, five process metrics, and six context factors as predictors (i.e., programming language, issue tracking, the total lines of code, the total number of files, the total number of commits, and the total number of developers). The context-aware approach stratifies the entire set of projects by context factors, and clusters the projects with similar distribution of predictors. They applied every tenth quantile of predictors on each cluster to formulate ranking functions. After transformation, the predictors from different projects have exactly the same scales. The universal model was then built based on the transformed predictors.

They applied their approach on 1,398 open source projects hosted on SourceForge and GoogleCode. They examined the generalizability of the universal model by applying it on five external projects also.

They claimed that their main contributions are:

**(1) Context-aware rank transformation**
**(2) Context factors as predictors of the universal model**

*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

**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

In this study, the authors propose a context-aware rank transformation to address the variations in the distribution of predictors before fitting them to the universal defect prediction model. They used 21 code metrics, five process metrics, and six context factors as predictors (i.e., programming language, issue tracking, the total lines of code, the total number of files, the total number of commits, and the total number of developers). The context-aware approach stratifies the entire set of projects by context factors, and clusters the projects with similar distribution of predictors. They applied every tenth quantile of predictors on each cluster to formulate ranking functions. After transformation, the predictors from different projects have exactly the same scales. The universal model was then built based on the transformed predictors.

They applied their approach on 1,398 open source projects hosted on SourceForge and GoogleCode. They examined the generalizability of the universal model by applying it on five external projects also.

They claimed that their main contributions are:

**(1) Context-aware rank transformation**
**(2) Context factors as predictors of the universal model**

## 3 CASE STUDY RESULTS

**RQ1: Can a context-aware rank transformation provide predictive power comparable to the power of log transformation?**

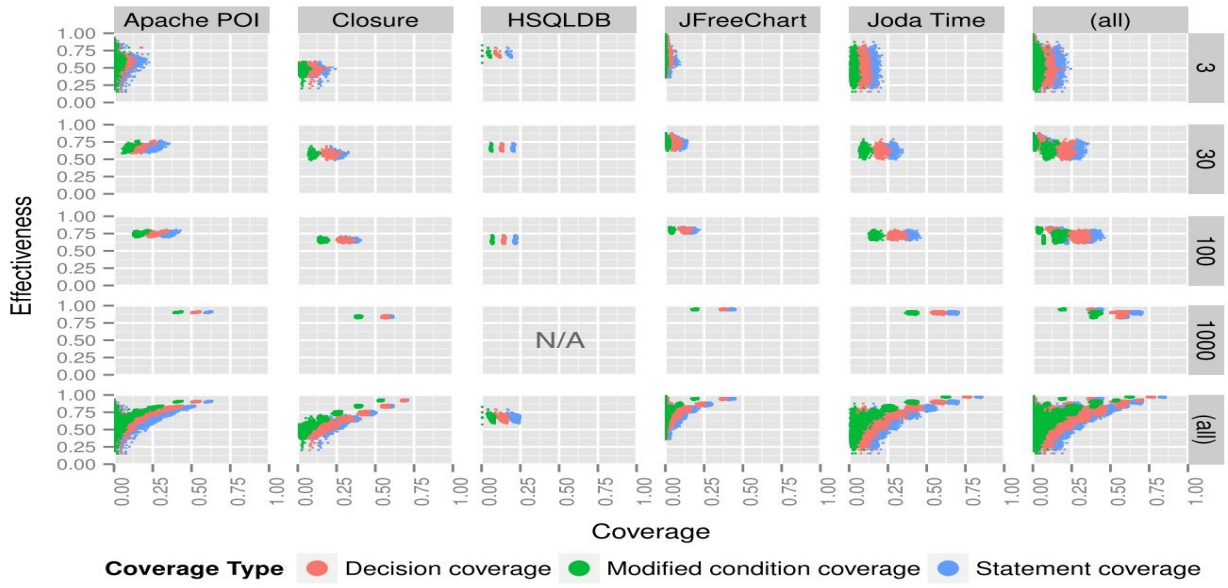


**Figure 1: Normalized effectiveness scores (left axis) plotted against coverage (bottom axis) for all subjects. Rows show the results for one suite size; columns show the results for one project. N/A indicates that the project did not have enough test cases to fill in that frame.**

**Table 2: The Kendall $\tau$ 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 |

rank transformations to the models built using log transformations. Table 1. presents the mean values of the six performance measures of both log and rank transformations, and the corresponding p-values of Wilcoxon rank sum test. The results show the difference between the two transformations is small (i.e., less than 0.10). They concluded that rank transformation achieves comparable performance to log transformation. It is reasonable to use the proposed rank transformation method to build universal defect prediction models.

**RQ2: What is the performance of the universal defect prediction model?**

and the AUC value. Hence, the context factors are good predictors for building a universal defect prediction model.

**RQ3: What is the performance of the universal defect prediction model on external projects?**

ble predictive power.

## 4 CONCLUSION

In future, their plan is to evaluate the feasibility of the universal model for commercial projects. They have also thought about making a plugin for a version control system or IDE.

## REFERENCES

[1] 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.