Do Pseudo Test Suites Lead to Inflated Correlation in Measuring Test Effectiveness?

Jie M. Zhang*◇, Lingming Zhang†, Dan Hao*, Meng Wang‡, Lu Zhang*
*Key Laboratory of High Confidence Software Technologies (PKU), China
◇CREST, University College London, WC1E 6BT, UK.
†Department of Computer Science, University of Texas at Dallas, 75080, USA
‡Department of Computer Science, University of Bristol, BS8 1TH, UK

***Abstract*—Code coverage is the most widely adopted criteria for measuring test effectiveness in software quality assurance. The performance of coverage criteria (in indicating test suites' effectiveness) has been widely studied in prior work. Most of the studies use randomly constructed *pseudo* test suites to facilitate data collection for correlation analysis, yet no previous work has systematically studied whether *pseudo* test suites would lead to inflated correlation results.**

This paper focuses on the potentially wide-spread threat with a study over 123 real-world Java projects. Following the typical experimental process of studying coverage criteria, we investigate the correlation between statement/assertion coverage and mutation score using both pseudo and original test suites. Except for direct correlation analysis, we control the number of assertions and the test suite size to conduct partial correlation analysis. The results reveal that 1) the correlation (between coverage criteria and mutation score) derived from pseudo test suites is much higher than from original test suites (from 0.21 to 0.39 higher in $Kendall\ \tau_b$ value); 2) contrary to previously reported, statement coverage has a stronger correlation with mutation score than assertion coverage.

***Index Terms*—test suites, coverage criteria, empirical study**

I. Introduction

Better test suites may contribute to better software quality. The evaluation of the fault-revealing ability of test suites is thus very important for fair comparison among different test suites [1], [2], [3]. Coverage criteria are the most widely adopted evaluation criteria for this purpose. They have low cost and high convenience, and are extensively used in various relevant areas (e.g., test generation [4], [5], [6], test prioritisation [7], [8], [9], incremental program testing/checking [10], [11], and fault localisation [12], [13], [14], [15], [16]).

There are two basic types of coverage criteria: input-based criteria and oracle-based criteria. Input-based criteria focus on the behaviours of test inputs by measuring the percentage of the source code that is executed as a result of running the program against the test inputs. Examples of such include statement coverage, branch coverage, and path coverage [17], [18], [19], [20], [21], [22], [23]. Among them, statement coverage is the most widely used, and is considered by some studies [23] as the most effective. Oracle-based criteria consider the oracle information [24], measuring the percentage of code that is checked by the test oracles [25]. Assertion coverage (the dynamic backward slice of assertions, also known as 'checked coverage') belongs to this category. Being relatively new, although not as popular as statement coverage, assertion coverage is gaining traction, as it appears 'obvious' that fault revealing must be related to assertions. Recently, Zhang et al. [26] also found that assertion coverage has a stronger correlation with mutation score than statement coverage.

There are many empirical studies exploring whether coverage is effective in measuring the fault-revealing ability of tests and whether some criteria are better than others. The basic setup of such experiments is to base on a set of test suites, collect both the coverage and fault-revealing data (typically through mutation testing [27], [28], [29], [30], [31], [32]), and then perform a correlation analysis of the two. Stronger the correlation, the better the indication effectiveness. A fundamental challenge in such experiments is to collect a set of suitable test suites. Almost all the previous work collect test suites by deriving multiple test suites within the same project, by randomly constructing subsets of the original test suite of this project. The subsets are treated as individual suites and are then collected to form a set. The purpose of this practice is to scale up the experiments without involving a large number of projects, therefore reducing workload. Since such suites are constructed from the original suite, we call them *pseudo* test suites in this paper.

The test suites and the selection and construction of them may directly influence the result. Experiments built on different types of test suites may yield different results, thereby causing threats to the conclusions. However, as far as we know, no previous work has systematically explored whether this threat exists or how serious this threat is. There is no empirical result on how different types of test suites under different application scenarios would impact the correlation analysis of coverage criteria, nor the effect of different construction configurations used to create pseudo test suites. If the threat does exist, the high effectiveness of current coverage criteria in evaluating test suites remains, as demonstrated by previous work, may be inflated, directly affecting the judgement and usage of coverage for developers.

In this paper, we aim to investigate this potential threat of test suites with an experimental study on 123 real-world Java projects using both pseudo test suites and original test suites.

Pseudo test suites and original test suites are investigated under different application scenarios (within and cross projects respectively) by previous work. When we compare these two types of test suites, we refer to a comparison between them containing their own application scenario. In particular, we mainly focus on statement and assertion coverage mentioned above, and perform both direct correlation analysis (without controlling any variable) and partial correlation analysis [33] (controlling variables such as assertion number and test-suite size, see more in Section III-D). The comprehensive nature of our experiment allows us to be able to study the impact of test suites collection methods, and revisit the comparison results between assertion coverage and statement coverage [26].

Our results reveal that using pseudo test suites and original test suites would yield significantly different correlation results with both direct and partial correlation analysis: the correlation using pseudo test suites may be much stronger than using original test suites (from 0.21 to 0.39 higher in $Kendall\ \tau_b$ value); When constructing pseudo test suites, the variation of test-suite size matters, while the number of test suites has little influence on the correlation results. Additionally, different from the conclusion of previous work, we find that statement coverage has a stronger correlation with mutation score than assertion coverage.

These results provide several implications: for researchers, when investigating the effectiveness of coverage criteria, it is essential to consider the impact of different types of test suites; test-suite size should be diverse; a large number of test suites may not be necessary. For developers, when evaluating test suites or adopting coverage-driven test generation, statement coverage can be superior to assertion coverage.

In summary, this paper makes the following contributions.
(1) Empirical evidence that the type of test suites used in experiments affects the correlation results between test coverage and fault-revealing ability. Pseudo test suites produce strong correlation (over 0.98 $Pearson$ value for both statement and assertion coverage), which is stronger than original test suites (0.8673/0.6777 $Pearson$ value for statement/assertion coverage). Different test suite construction configurations of pseudo test suites also have different impacts.
(2) Empirical evidence that statement coverage is superior to assertion coverage in indicating fault-revealing ability. Both direct and partial correlation analyses show that statement coverage is better correlated with test suites' overall fault-revealing ability than assertion coverage. For example, for original test suites with partial analysis, the *Pearson* coefficient is 0.7500 for statement coverage, and only 0.2090 for assertion coverage.
(3) Practical implications applicable to academia and industry. When investigating the effectiveness of coverage criteria, it is important to consider the impact of different types of test suites; if one has to construct pseudo test suites within project, large numbers of test suites may not be necessary, while the test-suite size matters. Statement coverage is superior to assertion coverage when indicating test-suite effectiveness, and might be a better choice for developers.

II. State of the Art and Motivation

A. Statistics

There has been a large body of work concerning the effectiveness of various coverage criteria. We systematically searched DBLP [34] with combinations of keywords such as '`coverage|size|assertion|criteria|checked`', '`test|oracle|suite|software`', and '`assess|evaluation|experiment|quality|measure|effect|adequacy`', where '`|`' denotes the boolean *or* operator. From the query results, we found 25 papers since 1990, including 14 papers since 2010, as closely related. We analysed these work in terms of three aspects: types of test suites, coverage criteria, and subject numbers. The analysis results are presented by Table I, where the last two columns represent the number of papers in accordance with each aspect since 1990 or 2010[1].

TABLE I
STATISTICS OF RELATED WORK.

category		since 1990	since 2010
types of test suites	pseudo	24	13
	original	1	1
coverage criteria	statement coverage	11	11
	assertion coverage	3	3
subject number	>30	1	1
	≤30	24	13
total number of papers		25	14

1) Types of Test Suites: A test suite is a collection of test cases, which is the analysis target of coverage criteria and effectiveness measurement. Two types of test suites have been adopted in the correlation analysis in literature: pseudo test suites and original test suites. In this paper, we define the these two types of test suites as follows.
Pseudo test suites: The pseudo test suites refer to a collection of artificially constructed test suites. Each pseudo test suite is usually constructed by randomly selecting a number of test cases from the original test suite of a project. When studying the effectiveness of coverage criteria, pseudo test suites are used to simulate real test suites. Each project has a collection of pseudo test suites, which are used to correlate their coverage with their ability in detecting faults inside this project.
Original test suites: The original test suites refer to a collection of the real test suites from many projects. Each original test suite comes with the project and is developed by developers as a whole. When studying the effectiveness of coverage criteria, original test suites from different projects are used for correlating their coverage with their ability in detecting faults cross many projects.

As shown in Table I, almost all the previous work used pseudo test suites. The only study that did use original test suites [23] mentioned the importance of choice, but did not systematically compare the results yielded by these two types of test suites.

[1] There may be overlap between different rows. E.g., some papers investigate both statement coverage and assertion coverage. The full details of all the related work are on our homepage [35].

2) Coverage Criteria: A variety of criteria has been studied in the literature. For the purpose of this paper, Table I lists the detailed number of papers for studying statement/assertion coverage, the two criteria that we focus on.

From the table, statement coverage is widely studied, accounting for 44% of all the papers we studied. This is not surprising because it is the most widely adopted coverage criterion in academic and industry communities. There is relatively less work on assertion coverage because of its relatively short history from 2011 [25]. Zhang et al. [26] found that assertion coverage can better indicate fault-revealing ability than statement coverage. However, they use a small number of projects (5 Java projects) with only pseudo test suites and insufficient variable controlling. In this paper, we compare statement coverage and assertion coverage with comprehensive variable controlling, a larger number of projects, and two types of test suites.

3) Subjects: Subjects refer to the projects used in the studies. The choice and scale of subjects in an experimental study may significantly influence the results. More subjects in a correlation study mean more data, and thus may lead to more reliable results. However, as we can see from Table I, the number of subjects used in existing studies are small: all but one have fewer than 30 subjects.

B. Our Study

We suspect that the type of test suites adopted under different application scenarios (within-project and cross-project) may lead to different coverage criteria effectiveness results, which remains not yet studied. Motivated by this conjecture, we propose to investigate the threat of test suites, and hope to derive more reliable conclusions concerning the effectiveness of coverage criteria based on the result.

Our work is different from the previous in terms of four aspects: 1) we use both pseudo test suites and original test suites, and investigate the influence of different test construction configurations; 2) we use significantly more subjects (123 real-world Java projects, summing up to 309,257 SLOC) than the previous studies (especially on assertion coverage); 3) we use both direct and partial analysis to validate the results and avoid spurious correlations[2]; 4) we make new comparisons between statement coverage and assertion coverage with more reliable experimental settings introduced above.

III. Experimental Setup

To find out whether the use of pseudo test suites will lead to different correlation results compared to original test suites, we design the following **key** research question.

RQ: Do PSEUDO and ORIGINAL test suites lead to different correlation results between coverage criteria and mutation score?

Building on the above explorations, we then seek to deeply explore how the construction configurations of pseudo test suites affect the comparison results, and designed the following supplementary research question:

sub-RQ: How do different test-suite construction configurations impact the comparison results between pseudo and original test suites? This research question aims to find out better practices in constructing pseudo test suites by investigating the influence of different test-construction configurations (e.g., test-suite number, test-pool size, and test-construction granularity).

A. Subjects

In this experimental study, we use 123 real open-source Java projects in total. Among these projects, five base projects come from a previous work that also focus on statement and assertion coverage [26]: JFreeChart (abbreviated as *jfree*) [36], Apache Commons Lang (abbreviated as *lang*) [37], Urban Airship Java Library (abbreviated as *jlib*) [38], lambdaj (abbreviated *lamb*) [39], and Asterisk-Java (abbreviated as *aste*) [40]. The remaining 118 projects come from the top 1,000 popular projects of GitHub[3]. In the correlation study on pseudo test suites, all the correlations are conducted within each project, and we use the five base projects alone to enable direct comparison with the previous work [26], whereas in the correlation study on original test suites we used all the 123 projects. The number of executable lines of source code of the subjects ranges from 122 to 98,334.

Figure 1 shows the distribution of statement coverage, assertion coverage of original test suites, as well as the distribution of two control variables: assertion number and test-suite size. The subjects we use differ in statement/assertion coverage, as well as the oracle, indicating the diversity of the studied subjects.

B. Test Suites

We consider both pseudo test suites and original test suites. For original test suites, each project has one real test suite that is manually written by the developers and comes with the project[4], and the correlation is performed across different projects.

For pseudo test suites, as in prior work [26], we generate 1,000 pseudo test suites from the original test suite for each of the five base projects. Each pseudo test suite is formed by randomly selecting a random number of test methods from the original suite (without replacement). Each correlation is performed within each project and one project would yield one correlation result.

To investigate the impacts of test suite construction strategy on the threat study, we also consider test suites with different construction configurations. To get test suites with different sizes, similar to previous work [21], we randomly construct 1,000 test-suites with fixed-sized test suites. Suppose that the

[2] Spurious correlations occur when two variables have clearly no causal relationship whatsoever in real life but can be statistically linked by correlation, as the correlation may be transported from other confounding variables.

[3] From the 1,000 most popular Github projects, only 118 of them are single-module and can be successfully processed by all the tools used in the experimental study.

[4] We remove the test methods that JavaSlicer [41] cannot handle, and take the remaining as the original test suite.

original test pool has n test methods, we try four fixed sizes: $0.2 * n$, $0.4 * n$, $0.6 * n$, $0.8 * n$. To investigate the influence of test-suite number, we construct different number (i.e., 50, 100, 150, ..., 1000) of random-sized test suites for correlation analysis.

C. Tools

When collecting statement coverage, we use Cobertura [42], a popular coverage tool widely used in previous work [23]. For assertion coverage, we use JavaSlicer [41] (the tool for tracing assertion coverage), which is a state-of-the-art for Java dynamic slicing tools [26], [43].

To represent the effectiveness of test suites in ensuring software quality, as in previous work [26], [21], [23], we adopt mutant killing ratio [30], [44], i.e., the ability in detecting mutation faults, because mutation faults have been reported as reliable represents of real faults and suitable for software testing experimentation [45][5]. We use *PIT* [46] as our default tool, because *PIT* has been prevalently used in both academia and industry [26], [47], [48], [49]. We also use *Major* [45] as a comparison tool to investigate the threats of using *PIT*. The distribution of the fault-revealing ability (mutant-killing ratio) of different original test suites is shown in Figure 1.

D. Correlation Analysis

In this work, we use both direct and partial correlation analysis. For direct analysis, the correlation between variables A and B is conducted directly using the values of these two variables without controlling any other variable.

The direct correlation analysis has a risk: a correlation observed between A and B may be caused by the mediation of other variables, instead of actual correlation. Partial correlation analysis [50] addresses this problem, which is widely used in statistics to control other variables [50], [51], [52][6]. Suppose that we use $r(A, B)$ to represent the direct correlation between variables A and B, and use $r(A, B|C)$ to represent the correlation between A and B when statistically controlling variable C, partial correlation analysis calculates $r(A, C)$ and $r(B, C)$, then subtracts the predictions from C and leaves only information in A and B that is independent of C. If $r(A, B)$ is relatively large but $r(A, B|C)$ is much smaller, then variable C is a mediating variable and can explain the direct correlation of A and B.

In our study, for statement coverage and assertion coverage, we control one of them and record the correlation results of the other. In addition, we also control assertion number and test-suite size. Thus, for each coverage criterion, we have three correlation results each with a different variable controlled, and we use the smallest one to represent the corresponding correlation result, because it is closer to the actual correlation discarding the mediation of all the other variables. This is different from the previous work that also performed controlled experiments [21]: the previous work only controlled the test-suite size and analysed the correlation of coverage with fixed test-suite sizes.

In summary, we perform two types of correlation analysis, i.e., direct correlation analysis and partial correlation analysis. For pseudo test suites we perform correlation analysis on the 1,000 test suites within each project, and for original test suites we perform correlation analysis on the 123 test suites across all the projects.

Furthermore, in this experimental study, we mainly present our results with the following three correlation coefficients: *(1) Pearson product-moment correlation coefficient (Pearson's r).* It is a measure of the linear correlation between two variables X and Y. The value range for Pearson's r is [-1,1]. A $Pearson$ between 0.1 to 0.3 indicates a low degree correlation, between 0.3 to 0.5 indicates a moderate degree correlation, while between 0.5 to 1 indicates a high degree correlation [53]. Especially, when $Pearson$ is around 1, it is said to be a perfect correlation [53], [54].
(2) Kendall rank correlation coefficient (Kendall's τ_b). Different from Pearson's r, Kendall's τ_b is a measure of rank correlation. For any pair of data points (x_1, y_1) and (x_2, y_2), they are denoted as *concordant* if $(x_1 > y_1) \wedge (x_2 > y_2)$ or $(x_1 < y_1) \wedge (x_2 < y_2)$, and as *disconcordant* if $(x_1 > y_1) \wedge (x_2 < y_2)$ or $(x_1 < y_1) \wedge (x_2 > y_2)$. Then, Kendall's τ is computed based on the ratio of concordant pairs and disconcordant pairs, ranging from -1 to 1. For example, 1 denotes all data pairs are concordant while -1 denotes that all data pairs are discordant.
(3) Adjusted R-squared. It is a modified version of R-squared [55] for the number of predictors in a model, which is the best estimate of the degree of relationship in the basic population, and especially proper for multiple linear regressions. The values of adjusted R-squared can be negative.

Note that in addition to the three coefficients above, we also analyze *p-values*, which is the probability of obtaining a result equal to or "more extreme" than what was actually observed, when the NULL hypothesis is true [56].

IV. Results and Analysis

In this section, we present the results and analysis. To answer the key research question of this paper, we present the experimental comparison results and analysis on how statement coverage and assertion coverage correlate with the fault-revealing ability of test suites (the mutant killing ratio). By default, we present $Pearson$ and $Kendall$ τ_b correlation, as in bivariate linear analysis R^2 is just the square of $Pearson$. As a pattern of presentation, we analyse each finding before summarising it.

A. Correlation Analysis with Pseudo Test Suites

We introduce the correlation analysis results when using pseudo test suites. We introduce the direct and partial results successively.

[5] As in prior work, we use mutant killing ratio, instead of mutation score, because equivalent mutants are undecidable, and currently there are no effective automatic detection approaches.

[6] It has been embedded into SPSS: http://www.ibm.com/analytics/us/en/technology/spss/

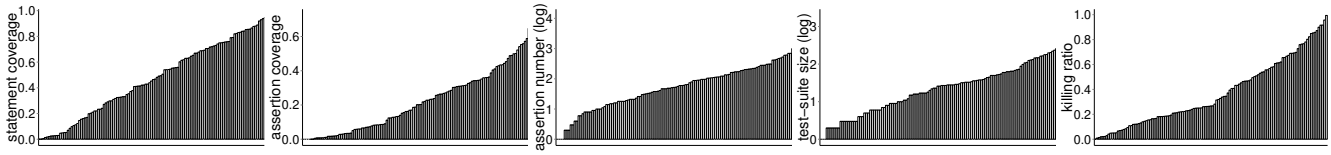

Fig. 1. Distribution of subject information. Each bar represents one subject. In each sub-figure, the projects are sorted accordingly. From the figure, the subjects differ in statement/assertion coverage, as well as the oracle and fault-revealing ability of test suites, indicating their diversity.

TABLE II
RESULTS OF DIRECT CORRELATION ANALYSIS WITH PSEUDO AND ORIGINAL TEST SUITES.

test suite		Coverage criteria	
		Statement Coverage	Assertion Coverage
pseudo	aste	0.9932 / 0.9220	0.9817 / 0.8845
	jfree	0.9979 / 0.9652	0.9876 / 0.9478
	jlib	0.9978 / 0.9568	0.9902 / 0.9141
	lamb	0.9982 / 0.9594	0.9899 / 0.9239
	lang	0.9988 / 0.9663	0.9983 / 0.9740
original		0.8673 / 0.7100	0.6777 / 0.4975

* *The* p-values *are all below 0.001.*

TABLE III
RESULTS OF PARTIAL CORRELATION ANALYSIS WITH PSEUDO AND ORIGINAL TEST SUITES.

	Control criteria for PSEUDO test suites			
criterion	StateCov	AssertCov	AssertNum	TestSize
StateCov	–	0.6900	0.7820	0.7650
AssertCov	0.5400	–	0.6720	0.6580
	Control criteria for ORIGINAL test suites			
criterion	StateCov	AssertCov	AssertNum	TestSize
StateCov	–	0.7500	0.8060	0.8060
AssertCov	0.2090	–	0.5290	0.6540

* *The* p-values *are all below 0.001.*

For pseudo test suites, to observe the correlation between their statement/assertion coverage and their fault-revealing ability, we visualise each test suite's coverage and mutant killing ratio in Figure 2. Each sub-figure represents one project. As we can see from the figure, there are strong correlations between statement/assertion coverage and the mutant killing ratio.

To quantitatively measure the linear/non-linear association shown in Figure 2, we calculate the correlation coefficients. The first five rows of Table II show the results. Each column represents the correlation coefficient values between the criterion and mutant killing ratio. In each square, the first number is the $Pearson$, the second number is the $Kendall\ \tau_b$. The *p-values* are all below 0.001, indicating that the correlations are all statistically significant at the level of 0.001.

In the table, we observe that statement coverage and assertion coverage are very strongly correlated with the fault-revealing ability of pseudo test suites in direct correlation analysis.

We speculate that the difference may be caused by the fact that pseudo test suites are constructed from the same projects, and thus may have similar features. Also, since being randomly chosen, they lack some attributes of the original real test suites, such as the coupling effect (different test methods cooperate with each other to achieve the same goal). Next, we check whether the unusually strong correlation still exists on original test suites.

We have mentioned that direct correlation analysis may lead to spurious conclusions (see more in Section III-D). In this section, we investigate this problem by using partial correlation analysis.

The first half of Table III shows the results[7]. Each column represents a control criterion, while each row represents the $Pearson$ value for this criterion after controlling the criterion in each of the columns. For example, in the first row, value 0.6900 represents that when controlling assertion coverage, the correlation coefficient between statement coverage and killing ratio is 0.6900. We do not list the *p-values*, because they are all below 0.001, indicating that the correlations are all statistically significant at the level of 0.001.

In Table III, we observe that when controlling other variables, all the correlation values decrease. This observation indicates that all the four variables affect each other in their correlation with test suites' mutant killing ratio. Different variables have different degrees of decrease. In particular, when comparing Table II and Table III, we observe that for statement coverage, the *Pearson* value drops from above 0.9900 to as low as 0.6900 for pseudo test suites. Thus, in partial correlation analysis, although the correlation becomes weaker, statement coverage still have a strong correlation with mutant killing ratio.

For assertion coverage, the *Pearson* value drops from over 0.9800 to 0.5400 for pseudo test suites. This observation implies that the strong correlation of assertion coverage in direct correlation analysis may be partially caused by other variables.

B. Correlation Analysis with Real Test Suites

In this section, we introduce the direct correlation analysis results when using original test suites and then give our results to answer our research question.

As was the case of pseudo test suites, we visualise each original test suite's coverage and mutant killing ratio in Figure 3. From the figure, we can see that the distribution patterns are noticeably different from those in Figure 2 (for pseudo test suites). In particular, the points are much more emanative. The correlation coefficients, as shown in the last

[7] For pseudo test suites we only present the results of *lang*. The results for other subjects have the same pattern and can be found on our homepage [35].

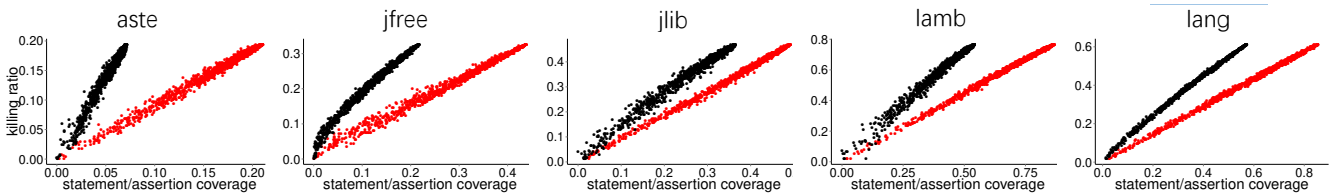

Fig. 2. Results on pseudo test suites. Red/black points are for statement/assertion coverage. In each sub-figure, the horizontal axis represents the corresponding criteria, and the vertical axis represents the fault-revealing ability of test suites (i.e., mutant killing ratio). Each point represents one pseudo test suite. As we can see from the figure, there are very strong correlations between statement/assertion coverage and mutant killing ratio.

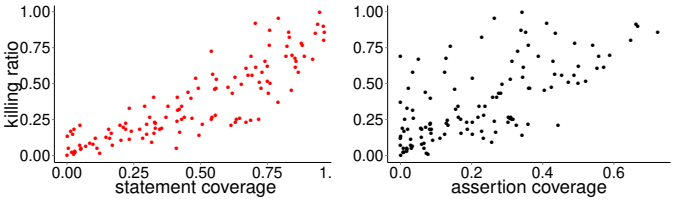

Fig. 3. Results on original test suites. Red/black points are for statement coverage/assertion coverage. From the figure, the points are much more emanative than those on pseudo test suites.

row of Table II, are significantly lower than those of pseudo test suites.

Based on the observations above, we conclude that in direct correlation analysis, the coverage criteria effectiveness observed from pseudo test suites is much higher than that from original test suites.

The second half of Table III shows the results of partial correlation results between statement/assertion coverage and mutant killing ratio.

In Table III, the same as we previously observed, all the correlation values decrease. This indicates that all the four variables affect each other in their correlation with the original test suites' mutant killing ratio. When comparing Table II and Table III, we observe that for statement coverage, the *Pearson* value drops from 0.8673 to as low as 0.7500 for original test suites. For assertion coverage, the *Pearson* value drops from 0.6777 to 0.2090 for original test suites. This observation implies that the strong correlation of assertion coverage in direct correlation analysis may be partially caused by other factors, especially statement coverage.

When reviewing the experimental results observed from pseudo test suites and original test suites, we have the following main finding:

> **Finding-1:** The correlation between coverage criteria and mutant killing ratio derived with pseudo test suites within project is stronger and easier to be influenced by other factors such as assertion number and test-suite size compared with original test suites.

There are two possible reasons for this difference. First, pseudo test suites are constructed from the same projects, resulting in similar features in their coverage and fault-revealing ability. For example, even for a project with high coverage but low fault-revealing ability, all its pseudo test suites may own the same features consistent with the whole test suite (i.e., high coverage and low fault-revealing ability). Consequently, the correlation can still be strong. Second, pseudo test suites may lack some attributes of the original real test suites. The tests in original test suites tend to have coupling effects as they cooperate with each other to better ensure software quality. For example, for an original test suite, adding a test method may increase statement coverage without revealing more faults (or vice versa), as the faults that this test method can find may likely have already been found by other methods in the suite. Such coupling effects may bring a gap between testing criteria and fault-revealing ability. Artificial test suites, however, break this coupling effect to some extent and result in spuriously strong correlations: independent test methods are reorganised together, and thus the statements covered or the faults detected by them are sparser and less likely to coincide, causing weaker coupling effects.

C. Analysis on Different Test Construction Configurations

We have observed that pseudo test suites would result in a much stronger correlation between code coverage and test suite effectiveness. In this section, we further refine our investigation and try to find out how construction configurations would impact the correlation comparison results. In particular, we consider four important configurations related to the construction of pseudo test suites: test-suite number, test-pool size, test-suite size, and construction granularity.

1) Test-pool size: We can see from Table II, as also in almost all the previous work, that the correlation results are different between different projects. We suspect that the size of the test method pool (i.e., the test method set that the test suites are constructed from) may be one of the reasons for the differences. Table IV shows how the correlation between statement/assertion coverage and fault-revealing ability changes on different sizes of test pools. From the table, it is obvious that projects with larger test pools tend to have stronger correlation values with statement/assertion coverage. The reason may be that the test suites constructed from larger test pools are more evenly distributed, which dilute the test-suite diversity. Also, these test suites may have weaker coupling effects, as the statements covered by them are sparser and have fewer opportunities to coincide.

TABLE IV
RESULTS WITH DIFFERENT TEST-POOL SIZES.

project	poolsize	StateCov	AssertCov
aste	203	0.9932 / 0.9220	0.9817 / 0.8845
jfree	1,051	0.9979 / 0.9652	0.9876 / 0.9478
jlib	278	0.9978 / 0.9568	0.9902 / 0.9141
lamb	221	0.9982 / 0.9594	0.9899 / 0.9239
lang	1,977	0.9988 / 0.9663	0.9983 / 0.9740

TABLE V
RESULTS ON DIFFERENT FIXED-SIZE TEST SUITES.

project	test size	Statement Coverage	Assertion Coverage
aste	0.2*n	0.7718 / 0.5659	0.3226 / 0.2173
	0.4*n	0.7630 / 0.5518	0.4416 / 0.2774
	0.6*n	0.7918 / 0.5810	0.4613 / 0.3117
	0.8*n	0.7984 / 0.5988	0.4813 / 0.3158
jfree	0.2*n	0.9037 / 0.7332	0.4254 / 0.2752
	0.4*n	0.8878 / 0.6878	0.5059 / 0.3369
	0.6*n	0.8615 / 0.6624	0.5213 / 0.3600
	0.8*n	0.8666 / 0.6670	0.4869 / 0.3445
jlib	0.2*n	0.9419 / 0.7864	0.6772 / 0.4899
	0.4*n	0.9089 / 0.7206	0.6215 / 0.4288
	0.6*n	0.8653 / 0.6588	0.5334 / 0.3667
	0.8*n	0.8378 / 0.6353	0.5174 / 0.3344
lamb	0.2*n	0.9183 / 0.7441	0.7470 / 0.5425
	0.4*n	0.9180 / 0.7438	0.7634 / 0.5574
	0.6*n	0.9078 / 0.7268	0.7673 / 0.5594
	0.8*n	0.8946 / 0.7122	0.7565 / 0.5662
lang	0.2*n	0.7432 / 0.5350	0.8041 / 0.6014
	0.4*n	0.7045 / 0.4902	0.8003 / 0.5937
	0.6*n	0.6776 / 0.4685	0.8174 / 0.6177
	0.8*n	0.6118 / 0.4285	0.8162 / 0.6075

Finding-2: Test-pool size would influence the correlation results, and we suggest to adopt projects of different sizes of test pools when using pseudo test suites.

2) Test-suite size: From the above observations, we found that when removing the effects of test-suite size using partial correlation analysis, the effectiveness of statement coverage and assertion coverage are not influenced much. In this section, we make further investigation into the effects of test-suite size. In particular, similar to previous work [21], we randomly construct 1,000 test-suites with fixed-sized test suites. Suppose that the original test pool has n test methods, we try four fixed sizes: $0.2 * n$, $0.4 * n$, $0.6 * n$, $0.8 * n$.

The results are shown in Table V. The second column shows the fixed test sizes. From the table, we observe the following. First, for fixed-size test suites, the effectiveness of statement coverage, assertion coverage all decreases. However, most of the correlations of statement coverage and assertion coverage remain strong (i.e., > 0.5). This observation is consistent with our previous finding, except that the decreasing rate is different, as partial correlation analysis controls test-suite size in a different way [33]. Second, different scales of fixed size would yield different degrees of correlation. When the fixed size increases from $0.2 * n$ to $0.8 * n$, the correlation with statement coverage decreases for most projects, except for *aste*. The reason is that for large pseudo test suites, many test methods tend to overlap with each other in code coverage but may detect different sets of faults, causing fluctuation to the total correlation. *aste* does not have this phenomenon due to its distinct characteristic of having a large code base (11,704 SLOC) and a very small number of test methods (203). Assertion coverage also does not have this phenomenon, because due to the lower number of covered assertions than covered statements, different test methods have a lower probability to have overlapping assertion coverage.

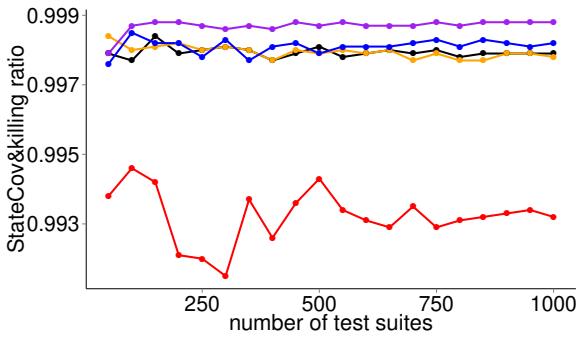

Fig. 4. Impacts of test-suite number. Each line represents one project. From this figure, there is no regular pattern between the test-suite number and correlation degree.

Finding-3: Different fixed sizes yield different correlation results, and we suggest to include different scales of fixed test suites to reduce the impacts of fixed sizes when controlling test-suite size to analyze the correlation.

3) Test-suite number: In previous results, for pseudo test suites, we use 1,000 test suites, while for original test suites, the number of test suites is 123. In this section, we check whether the different correlations between pseudo and original test suites are caused by the difference in test-suite number.

To investigate this, we construct different number (i.e., 50, 100, 150, ..., 1000) of random-sized test suites for correlation analysis. The results are shown in Figure 4[8], from which we find that there are data fluctuations (due to the bias of random selection), but there is no regular pattern between the test-suite number and correlation values. This finding also indicates that the difference in the number of test suites used in the comparison study of pseudo test suites (1000) and original test suites (123) is not a threat to the comparison result.

Finding-4: Test-suite number does not have a clear impact on the effectiveness of code coverage using pseudo test suites.

Thus, we conclude that when using random-sized pseudo test suites for correlation studies, there is no need to pursue a very high number of test suites.

4) Construction granularity: As method-level test construction may greatly break the coupling effect of original test suites, in this section we also investigate class-level test suite

[8] Due to space-limit, we only present the results of statement coverage here.

TABLE VI
CORRELATIONS BETWEEN THE TWO COVERAGE CRITERIA AND KILLING RATIOS FOR CLASS-LEVEL TEST SUITE CONSTRUCTION.

project	Statement Coverage	Assertion Coverage
aste	0.9915 / 0.9196	0.9306 / 0.7852
jfree	0.9960 / 0.9424	0.9909 / 0.9433
jlib	0.9940 / 0.9321	0.9431 / 0.8165
lamb	0.9908 / 0.9456	0.9848 / 0.9265
lang	0.9964 / 0.9496	0.9963 / 0.9531

construction. The only difference between the two levels of test suite construction lies in the granularity of test selection, i.e., whether each single test method is randomly selected, or all the test methods within each test class are selected as a whole during each step in test suite construction. To our knowledge, no previous work has ever compared class-level and method-level test suite construction in coverage criteria studies.

The results are shown in Table VI. When performing class-level test suite construction, we found that the correlation results are a bit closer to original test suites (see Table II). The reason may be that method-level is too fine-grained for simulating real test suite construction, as in practice developers are more likely to add an entire test class including multiple test methods rather than adding separate test methods into different test classes. Based on all the observations above, we get the following finding:

> **Finding-5:** Class-level-constructed pseudo test suites has a weaker correlation with mutant killing ratio than method-level-constructed test suites.

5) Construction basis: Another interesting finding from Figure 2, Figure 3, and Figure 5, is that statement coverage is usually higher than killing ratio, while assertion coverage is usually lower than killing ratio. This may be also due to that statement coverage is rather easy to achieve, and tests may easily increase statement coverage but fail to find additional faults. On the contrary, it is hard to achieve assertion coverage, and there are many fault-revealing tests that may not increase assertion coverage. Although coverage and mutant killing ratio are different physical meanings, such comparison results may provide implications when trying to use coverage criteria to replace mutant killing ratios.

The findings on statement and assertion coverage inspire us to further investigate their relationship in the setting of coverage-driven test augmentation: statement-coverage-driven test augmentation (adding a randomly-selected test method only when it can increase statement coverage) and assertion-coverage-driven test augmentation (adding a randomly-selected test method only when it can increase assertion coverage).

The results are shown in Figure 5, where we augment test methods one by one (until no more test methods can increase statement coverage or assertion coverage). From these charts, we can see that the killing ratio increases steadily with both test augmentations, but faster in the statement-coverage-driven case. For example for project *jlib*, when choosing 100 test methods, statement-coverage-driven test augmentation can achieve a killing ratio around 0.3500, while for assertion-coverage-driven test augmentation, the value is below 0.3000. Additionally, after finishing the test augmentation process (i.e., when no other tests can still increase the statement/assertion coverage), statement-coverage-driven test augmentation always achieves higher killing ratio. The reason is that some tests do not increase statement/assertion coverage but can still find faults. Assertion coverage, being more difficult to increase, suffers more from this fact.

Lastly, we would like to revisit the overall correlation between statement coverage and assertion coverage using the results observed in the previous sections. We compare these two criteria from three aspects: their correlation with test suite effectiveness, their values, and their performance in driving test generation. From our previous results, the correlation of statement coverage is slightly, but consistently, stronger than that of assertion coverage. For example, in the direct correlation analysis, the correlation result is 0.8637 for statement coverage, while is 0.6777 for assertion coverage. Such observation indicates the superiority of statement coverage in measuring test-suite effectiveness. Therefore, we get the following finding:

> **Finding-6:** Statement coverage is better than assertion coverage in indicating test suite effectiveness with both pseudo test suites and original test suites. Statement-coverage-driven test augmentation is also better than assertion-coverage-driven test augmentation in detecting more faults.

There are two possible reasons for the better performance of statement coverage. First, a test method may reveal a fault through either assertion violations or exceptions, while assertion coverage only checks the former. Second, increasing assertion coverage can be harder to achieve than statement coverage in practice. For example, when some valuable tests that revealing additional faults are augmented, assertion coverage may have a lower probability to increase than statement coverage.

D. Threats to Validity

Threats to internal validity mainly lie in the implementation of our code analysis tools. To reduce these threats, we use mature tools and frameworks in our implementation, such as PIT/Major, JavaSlicer, and ASM. We also carefully review our code and scripts.

Threats to external validity mainly lie in the subjects, tests, and faults that we used. To reduce the threat related to the subjects and tests, we use 123 real-world Java projects. To reduce the threat to the faults used in our evaluation, we use the widely used PIT and Major mutation testing tools and generate a large number of mutation faults. Further reduction of the threats to external validity requires evaluation of more projects with real faults.

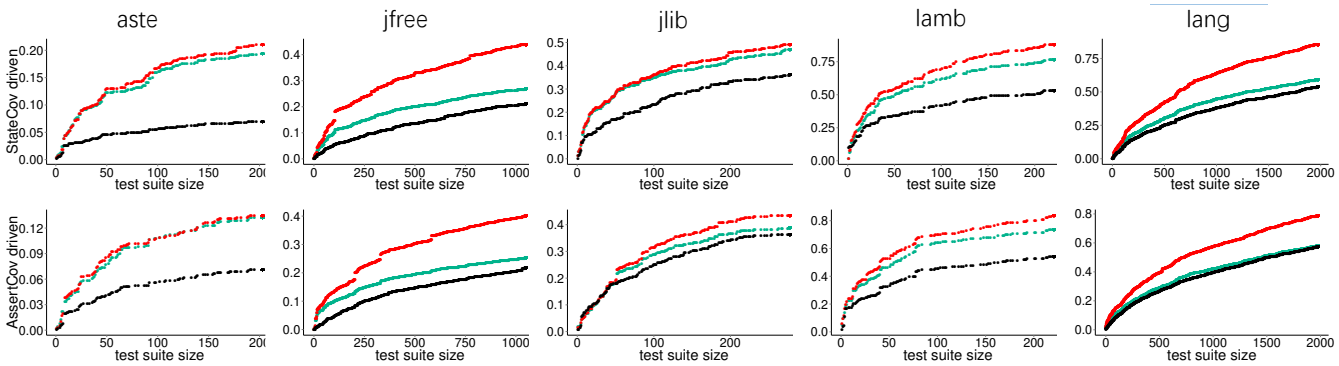

Fig. 5. The increase of killing ratio when augmenting tests one by one only when the statement coverage (shown in the upper figures) or assertion coverage (shown in the lower figures) increases. Green/red/black points represent killing ratio/statement coverage/assertion coverage.

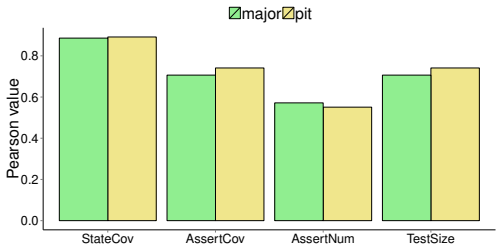

Fig. 6. Comparison between *PIT* and *Major* on original test suites. From this figure, the four variables have similar correlations with the mutant killing ratio indicated through different mutation tools.

Threats to construct validity lie in how the experimental results are measured. In this experimental study, we use Pearson's r, and adjusted R^2 for correlation analysis, which are widely used in previous work on test effectiveness [26], [21], [22]. The threat is further reduced by using $Kendall\ \tau_b$.

V. Extended Analysis and Discussion

A. Extended Analysis

Except for the results mentioned in Section IV, there are some other interesting while relatively minor aspects for us to explore. These aspects are also studied by various previous work and are important for evaluating test suites' fault-revealing ability. In particular, we introduce whether **mutation tool** would impact the correlation results in Section V-A1, the comparison results between **statement coverage** and **branch coverage** in Section V-A2, and the performance of **assertion number/test-suite size** in Section V-A3. We omit the tables and plots when presenting most of the results while describing the results inside the text to save space.

1) Comparison between Different Mutation Tools: The mutant killing ratio may be affected by mutation tools that are used to generate faults [29]. To investigate whether mutation tool is a threat to our correlation results, except for PIT, we use another widely used tool–*Major* [45]–to provide the mutant killing ratio for representing test suites' fault-revealing ability. We then repeat our experiments and check if different mutation tools would yield noticeably different correlation results.

The *Major* tool only supports the Ant build system. After removing the projects that failed to be built with Ant or to be handled by *Major*, 63 subjects are left. To fairly compare *PIT* and *Major*, we conduct our analysis with these 63 projects on the mutant killing ratio provided by both tools. We only present the results for original test suites to save space.

The results are listed in Figure 6. The yellow/blue bars show the correlation results (*Pearson* values) between the total mutant killing ratio provided by the *PIT/ Major* tool and the four variables: statement coverage, assertion coverage, assertion number, and test-suite size. As we can see, the heights of the two types of bars are almost the same, indicating that the results from the two are consistent, with minor differences, and thus the preceding conclusions based on *PIT* may be generalized to *Major*. On the other hand, such consistency also laterally reflects that using mutant killing ratio to indicate test suites' fault-revealing ability is stable and reliable.

2) Comparison between Statement and Branch Coverage: Although the focus of this paper is on statement coverage and assertion coverage, we are also aware that previous work has conflicting conclusions towards the comparison results of statement coverage and branch coverage [22], [23]. In this section, using original test suites, we revisit this comparison between statement coverage and branch coverage[9], by calculating the correlations between these two criteria and killing ratio on the 123 projects[10].

The resulting correlation is as follows: on original test suites, statement coverage has a $Pearson$ of 0.8770 and $Kendall\ \tau_b$ of 0.6540, and branch coverage has a $Pearson$ of 0.8720 and $Kendall\ \tau_b$ of 0.6320. This observation indicates that statement coverage outperforms branch coverage by a small margin. The result agrees with that of Gopinath et.al [23] (also with original test suites), further demonstrating the validity of our study.

3) Performance of Assertion Number, and Test-suite Size: Assertion number and test-suite number are also widely studied as criteria to indicate the effectiveness of test suites [26], [21], [57]. For example, Zhang et al. [26] found that assertion number is also highly correlated with test suites' fault-revealing ability using pseudo test suites.

In our paper, we find that when using pseudo test suites, the assertion number gets above 0.9700 $Pearson$ in direct analysis, while only 0.1970 $Pearson$ in partial analysis; when

[9] We use the branch coverage reported by Cobertura.

[10] Without needing JavaSlicer for measuring assertion coverage, we can now use the complete test suite of each project.

using original test suites, the assertion number gets around 0.5600 $Pearson$ in direct analysis and 0.1850 $Pearson$ in partial analysis. The correlation results are roughly the same for test-suite size. These findings again indicate that the types of test suites, as well as analysis approaches, could have a significant impact on the correlation results, not only for coverage criteria but also for assertion number and test-suite size. When comparing coverage criteria and these two criteria, assertion number and test-suite size have significantly weaker correlations than statement coverage and assertion coverage. This observation matches the existing knowledge that assertion number and test-suite size are coarse-grained. They are not able to reliably indicate test suites' fault-revealing ability as statement coverage or assertion coverage does.

B. Related Work

Our work is related to several areas such as test generation, test prioritization, and fault localization. Due to space reasons, we discuss and analyze the most related work that has similar or different conclusions with ours in Section V-B. Noticing several related works have different conclusions on another widely used coverage criteria: branch coverage, we also discuss the results of the comparison between statement coverage and branch coverage in Section V-A2.

Our study found that in direct correlation analysis, statement coverage and assertion coverage have very high effectiveness in indicating the fault-revealing ability of pseudo test suites. Such perfect correlation (without any variable controlling) is consistent with the work of Zhang et al. [26], Andrews et al. [58], and Inozemtseva and Holmes [21], which all present correlation values very close to 1.

Using only pseudo test suites, the work of Inozemtseva and Holmes [21] found that there is moderate to very strong correlation between fault-revealing ability and test-suite size, while our results indicate that the strong correlation may be aroused by the threat of pseudo test suites and direct correlation analysis. For example, when using original test suites and partial correlation analysis, test-suite size has only low correlation degree (0.2500) with the fault-revealing ability of test suites. On the other hand, the paper found that statement coverage has low to moderate correlation with the normalised test-suite effectiveness (the ratio of the killed-mutant number to covered-non-equivalent-mutant number) when the test suite size is fixed. In our study, we do not use normalised test-suite effectiveness, because the number of covered mutants may be affected by code coverage in a great deal, which may cause a bias towards the correlation results. Also, when constructing test suites, we avoid using only very small-size test suites (such as 3 methods, 10 methods, and 30 methods used in prior work [21]), because the test suites with such small size tend to have great diversity in code coverage, which would contribute to the weak correlation between code coverage and fault-revealing ability.

On the effectiveness of code coverage using original test suites, we conclude that although the effectiveness of statement coverage decreases in a great deal comparing to pseudo test suites, it still has over 0.8000 $Pearson$ value. Such a conclusion is consistent with the work of Gopinath et al. [23], which also used original test suites and found that statement coverage has relatively good effectiveness.

On the superiority of statement coverage, Zhang et al. [26] found that statement coverage is less effective than assertion coverage in indicating the test suites' fault-revealing ability. In particular, they controlled assertion coverage to observe statement coverage on pseudo test suites, and found that the correlation of statement coverage decreases as a result. In our work, we also look at the opposite controlling configuration: to control statement coverage to observe assertion coverage. We found that the correlation of assertion coverage also decreases significantly in this case. On the other hand, we use both pseudo test suites and original test suites on a large number of projects. This complete picture leads us to the conclusion that statement coverage is better than assertion coverage when other criteria are controlled.

On the decrease of correlation when controlling other variables, consistent with our conclusions, Briand and Pfahl [59], Namin et al. [60], and Inozemtseva and Holmes [21] also found that the correlation between coverage and test-suite effectiveness is lower when controlling test-suite size.

On the influence of test-pool size, our finding is consistent with the work of Namin et al. [60], which found that the effectiveness of coverage is dependent on the software under study. Cai et al. [61] also found that different kinds of test cases complicate the relationship between code coverage and fault detection.

VI. Conclusions

In this paper, we conducted a study with 123 real-world java projects on the threats in test suites selection when used for correlating coverage and mutant killing ratio. The findings are revealing. We are able to conclude that the use of pseudo test suites does result in inflated correlation between coverage and mutant killing ratio compared to the case of using the original test suites cross projects. Equipped with this new understanding, we compared the effectiveness of statement coverage and assertion coverage and discovered that, contrary to previously reported, statement coverage is actually more effective than assertion coverage in evaluating tests suites, showing the significance of the difference in test suite selection. In addition, we also found that variation of test-suite size is important for the correlations study, while the number of test suites matters less.

Acknowledgments

This work is supported by the National 973 Program of China with No. 2015CB352201, the Natural Science Foundation of China with No. 61872008 and 61861130363. It is also partially supported by National Science Foundation grants CCF-1566589 and CCF-1763906, the Royal Society Newton Advanced Fellowship: NAF\R1\180142, the Royal Society International Exchanges Cost Share: IE150982, and the ERC advanced grant with No. 741278.

REFERENCES

[1] Matthew J Rutherford, Antonio Carzaniga, and Alexander L Wolf. Evaluating test suites and adequacy criteria using simulation-based models of distributed systems. *TSE*, 34(4):452–470, 2008.

[2] Myra B Cohen, Matthew B Dwyer, and Jiangfan Shi. Coverage and adequacy in software product line testing. In *Proc: ISSTA workshop*, pages 53–63. ACM, 2006.

[3] Zhiyi Zhang, Zhenyu Chen, Ruizhi Gao, Eric Wong, and Baowen Xu. An empirical study on constraint optimization techniques for test generation. *Science China Information Sciences*, 60(1):012105, 2017.

[4] Gordon Fraser and Andrea Arcuri. Evosuite: automatic test suite generation for object-oriented software. In *Proceedings of the 19th ACM SIGSOFT symposium and the 13th European conference on Foundations of software engineering*, pages 416–419. ACM, 2011.

[5] Carlos Pacheco, Shuvendu K Lahiri, Michael D Ernst, and Thomas Ball. Feedback-directed random test generation. In *Proceedings of the 29th international conference on Software Engineering*, pages 75–84. IEEE Computer Society, 2007.

[6] Cristian Cadar and Koushik Sen. Symbolic execution for software testing: three decades later. *Communications of the ACM*, 56(2):82–90, 2013.

[7] Christopher Henard, Mike Papadakis, Mark Harman, Yue Jia, and Yves Le Traon. Comparing white-box and black-box test prioritization. In *Proc. ICSE*, pages 523–534. ACM, 2016.

[8] Gregg Rothermel, Roland H. Untch, Chengyun Chu, and Mary Jean Harrold. Prioritizing test cases for regression testing. *IEEE Transactions on software engineering*, 27(10):929–948, 2001.

[9] Ripon K Saha, Lingming Zhang, Sarfraz Khurshid, and Dewayne E Perry. An information retrieval approach for regression test prioritization based on program changes. In *Proceedings of the 37th International Conference on Software Engineering-Volume 1*, pages 268–279. IEEE Press, 2015.

[10] Dawei Qi, Abhik Roychoudhury, and Zhenkai Liang. Test generation to expose changes in evolving programs. In *Proc. ASE*, pages 397–406. ACM, 2010.

[11] Jie Zhang, Yiling Lou, Lingming Zhang, Dan Hao, Lu Zhang, and Hong Mei. Isomorphic regression testing: executing uncovered branches without test augmentation. In *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, pages 883–894. ACM, 2016.

[12] Geoff Birch, Bernd Fischer, and Michael R Poppleton. Fast model-based fault localisation with test suites. In *Proc. ICTP*, pages 38–57. Springer, 2015.

[13] Tezcan Dilshener, Michel Wermelinger, and Yijun Yu. Locating bugs without looking back. In *Proc. MSR*, pages 286–290. ACM, 2016.

[14] Tsutomu Kumazawa and Tetsuo Tamai. Counterexample-based error localization of behavior models. In *NASA Formal Methods Symposium*, pages 222–236. Springer, 2011.

[15] Xia Li and Lingming Zhang. Transforming programs and tests in tandem for fault localization. *Proceedings of the ACM on Programming Languages*, 1(OOPSLA):92, 2017.

[16] Mengshi Zhang, Xia Li, Lingming Zhang, and Sarfraz Khurshid. Boosting spectrum-based fault localization using pagerank. In *Proceedings of the 26th ACM SIGSOFT International Symposium on Software Testing and Analysis*, pages 261–272. ACM, 2017.

[17] Phyllis G Frankl and Stewart N Weiss. An experimental comparison of the effectiveness of branch testing and data flow testing. *TSE*, 19(8):774–787, 1993.

[18] W Eric Wong, Joseph R Horgan, Saul London, and Aditya P Mathur. Effect of test set size and block coverage on the fault detection effectiveness. In *Proc. ISSRE*, pages 230–238, 1994.

[19] Phyllis G Frankl and Oleg Iakounenko. Further empirical studies of test effectiveness. *ACM SIGSOFT Software Engineering Notes*, 23(6):153–162, 1998.

[20] Phyllis G Frankl, Stewart N Weiss, and Cang Hu. All-uses vs mutation testing: an experimental comparison of effectiveness. *Journal of Systems and Software*, 38(3):235–253, 1997.

[21] Laura Inozemtseva and Reid Holmes. Coverage is not strongly correlated with test suite effectiveness. In *Proc. of ICSE*, pages 435–445, 2014.

[22] Milos Gligoric, Alex Groce, Chaoqiang Zhang, Rohan Sharma, Mohammad Amin Alipour, and Darko Marinov. Comparing non-adequate test suites using coverage criteria. In *Proc. ISSTA*, pages 302–313. ACM, 2013.

[23] Rahul Gopinath, Carlos Jensen, and Alex Groce. Code coverage for suite evaluation by developers. In *Proc. ICSE*, pages 72–82. ACM, 2014.

[24] Jie Zhang, Junjie Chen, Dan Hao, Yingfei Xiong, Bing Xie, Lu Zhang, and Hong Mei. Search-based inference of polynomial metamorphic relations. In *Proceedings of the 29th ACM/IEEE international conference on Automated software engineering*, pages 701–712. ACM, 2014.

[25] David Schuler and Andreas Zeller. Assessing oracle quality with checked coverage. In *Proc. ICST*, pages 90–99. IEEE, 2011.

[26] Yucheng Zhang and Ali Mesbah. Assertions are strongly correlated with test suite effectiveness. In *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering, ESEC/FSE*, pages 214–224, 2015.

[27] David Schuler, Valentin Dallmeier, and Andreas Zeller. Efficient mutation testing by checking invariant violations. In *Proc. of ISSTA*, pages 69–80, 2009.

[28] James H Andrews, Lionel C Briand, and Yvan Labiche. Is mutation an appropriate tool for testing experiments?[software testing]. In *Proc. of ICSE*, pages 402–411, 2005.

[29] Mike Papadakis, Marinos Kintis, Jie Zhang, Yue Jia, Yves Le Traon, and Mark Harman. Mutation testing advances: an analysis and survey. *Advances in Computers*, 2017.

[30] Jie Zhang, Ziyi Wang, Lingming Zhang, Dan Hao, Lei Zang, Shiyang Cheng, and Lu Zhang. Predictive mutation testing. In *Proc. ISSTA*, pages 342–353. ACM, 2016.

[31] Jie Zhang, Muyao Zhu, Dan Hao, and Lu Zhang. An empirical study on the scalability of selective mutation testing. In *Software Reliability Engineering (ISSRE), 2014 IEEE 25th International Symposium on*, pages 277–287. IEEE, 2014.

[32] Jie Zhang, Shi Han, Dan Hao, Lu Zhang, and Dongmei Zhang. Automated refactoring of nested-if formulae in spreadsheets. In *Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 833–838. ACM, 2018.

[33] Partial Correlation. http://www.statisticssolutions.com/conduct-interpret-partial-correlation/.

[34] DBLP. http://dblp.uni-trier.de.

[35] homepage. https://github.com/researchlover/test-threat.

[36] JfreeChart. http://www.jfree.org/jfreechart/.

[37] Apache Commons Lang. http://commons.apache.org/proper/commons-lang/.

[38] Urban Airship Java Library. http://www.urbanairship.com/reference/libraries/java/.

[39] lambdaj. https://code.google.com/p/lambdaj/.

[40] Asterisk-Java. https://blogs.reucon.com/asterisk-java/.

[41] JavaSlicer. https://www.st.cs.uni-saarland.de/javaslicer/.

[42] Cobertura. http://cobertura.github.io/cobertura/.

[43] Jifeng Xuan and Martin Monperrus. Test case purification for improving fault localization. In *Proc. FSE*, pages 52–63. ACM, 2014.

[44] Jie Zhang, Lingming Zhang, Mark Harman, Dan Hao, Yue Jia, and Lu Zhang. Predictive mutation testing. *IEEE Transactions on Software Engineering*, 2018.

[45] René Just, Darioush Jalali, Laura Inozemtseva, Michael D Ernst, Reid Holmes, and Gordon Fraser. Are mutants a valid substitute for real faults in software testing. In *Proc. of FSE*, pages 654–665, 2014.

[46] PIT. http://pitest.org/.

[47] Yafeng Lu, Yiling Lou, Shiyang Cheng, Lingming Zhang, Dan Hao, Yangfan Zhou, and Lu Zhang. How does regression test prioritization perform in real-world software evolution? In *Software Engineering (ICSE), 2016 IEEE/ACM 38th International Conference on*, pages 535–546. IEEE, 2016.

[48] Lingchao Chen and Lingming Zhang. Speeding up mutation testing via regression test selection: An extensive study. In *Software Testing, Verification and Validation (ICST), 2018 IEEE 11th International Conference on*, pages 58–69. IEEE, 2018.

[49] August Shi, Alex Gyori, Milos Gligoric, Andrey Zaytsev, and Darko Marinov. Balancing trade-offs in test-suite reduction. In *Proc. of FSE*, pages 246–256, 2014.

[50] Duncan Cramer. A cautionary tale of two statistics: partial correlation and standardized partial regression. *The Journal of psychology*, 137(5):507–511, 2003.

[51] Peter Fransson and Guillaume Marrelec. The precuneus/posterior cingulate cortex plays a pivotal role in the default mode network: Evidence from a partial correlation network analysis. *Neuroimage*, 42(3):1178–1184, 2008.

[52] Jie Peng, Pei Wang, Nengfeng Zhou, and Ji Zhu. Partial correlation estimation by joint sparse regression models. *Journal of the American Statistical Association*, 2012.

[53] Pearson. http://www.statisticssolutions.com/pearsons-correlation-coefficient/.

[54] Pearson. http://www.statisticshowto.com/what-is-the-pearson-correlation-coefficient/.

[55] Andy Field. *Discovering statistics using IBM SPSS statistics*. Sage, 2013.

[56] Ronald L. Wasserstein and Nicole A. Lazar. The asa's statement on p-values: Context, process, and purpose. *The American Statistician*, 70(2):129–133, 2016.

[57] Pavneet Singh Kochhar, Ferdian Thung, and David Lo. Code coverage and test suite effectiveness: Empirical study with real bugs in large systems. In *Proc. SANER*, pages 560–564, 2015.

[58] James H Andrews, Lionel C Briand, Yvan Labiche, and Akbar Siami Namin. Using mutation analysis for assessing and comparing testing coverage criteria. *TSE*, 32(8):608–624, 2006.

[59] Lionel Briand and Dietmar Pfahl. Using simulation for assessing the real impact of test coverage on defect coverage. In *Proc. ISSRE*, pages 148–157. IEEE, 1999.

[60] Akbar Siami Namin and James H Andrews. The influence of size and coverage on test suite effectiveness. In *Proc. ISSTA*, pages 57–68. ACM, 2009.

[61] Xia Cai and Michael R Lyu. The effect of code coverage on fault detection under different testing profiles. *ACM SIGSOFT software engineering notes*, 30(4):1–7, 2005.