An Improvement to Test Case Failure Prediction in the Context of Test Case Prioritization

Conference Paper · October 2018

DOI: 10.1145/3273934.3273944

Francis Palma
Ryerson University
22 PUBLICATIONS **296** CITATIONS
SEE PROFILE

Tamer Abdou
Ryerson University
15 PUBLICATIONS **15** CITATIONS
SEE PROFILE

Ayse Bener
Ryerson University
184 PUBLICATIONS **2,679** CITATIONS
SEE PROFILE

An Improvement to Test Case Failure Prediction in the Context of Test Case Prioritization

Francis Palma
Data Science Lab
Ryerson University
Toronto, ON, Canada
francis.palma@ryerson.ca

Tamer Abdou
Data Science Lab
Ryerson University
Toronto, ON, Canada
tamer.abdou@ryerson.ca

Ayse Bener
Data Science Lab
Ryerson University
Toronto, ON, Canada
ayse.bener@ryerson.ca

John Maidens
Data Science Lab
Ryerson University
Toronto, ON, Canada
maidens@ryerson.ca

Stella Liu
IBM Canada Lab
Toronto, ON, Canada
stella.liu@ibm.com

ABSTRACT

Aim: In this study, we aim to re-evaluate research questions on the ability of a logistic regression model proposed in a previous work to predict and prioritize the failing test cases based on some test quality metrics. **Background:** The process of prioritizing test cases aims to come up with a ranked test suite where test cases meeting certain criteria are prioritized. One criterion may be the ability of test cases to find faults that can be predicted *a priori*. Ranking test cases and executing the top-ranked test cases is particularly beneficial when projects have tight schedules and budgets. **Method:** We performed the comparison by first rebuilding the predictive models using the features from the original study and then we extended the original work to improve the predictive models using new features by combining with the existing ones. **Results:** The results of our study, using a dataset of five open-source systems, confirm that the findings from the original study hold and that our predictive models with new features outperform the original models in predicting and prioritizing the failing test cases. **Conclusions:** We plan to apply this method to a large-scale dataset from a large commercial enterprise project, to better demonstrate the improvement that our modified features provide and to explore the model's performance at scale.

KEYWORDS

Test Case Prioritization, Prediction, Logistic Regression Model, Machine Learning, Test Quality Metrics.

ACM Reference Format:
Francis Palma, Tamer Abdou, Ayse Bener, John Maidens, and Stella Liu. 2018. An Improvement to Test Case Failure Prediction in the Context of Test Case Prioritization. In *The 14th International Conference on Predictive Models and Data Analytics in Software Engineering (PROMISE'18), October 10, 2018, Oulu, Finland.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3273934.3273944

1 INTRODUCTION

The aim of this study is to re-examine the research questions from a previous study on studying test case failure prediction for test case prioritization by Noor and Hemmati [11]. The goals of the original study in [11] included: (1) to examine if combining the traditional test quality metrics using a regression model (*i.e.*, logistic regression model) can improve test prioritization compared to ranking the test cases only using the individual traditional test quality metrics and (2) to examine if it is possible to further improve the test prioritization results by adding similarity-based test quality metrics into the model along with traditional test quality metrics, *i.e.*, to determine if models with similarity-based test quality metrics in combination with traditional test quality metrics outperform the models with solely traditional test quality metrics.

We believe the benefit of this study for practitioners and researchers is two-fold. First, we plan to re-examine the results of the original study, *i.e.*, to determine if they are valid. This will help the researchers to understand the significance and limitations of the study. Second, this study will attempt to reach closer towards the goal in the original study relying on the same dataset, *i.e.*, to show that it is possible to improve the test case prioritization further.

The original study [11] investigated the following two research questions:

- RQ1: *Can combining the traditional test quality metrics using a regression model improve test case prioritization compared to ranking the test cases using the individual metrics?*
- RQ2: *Can the test case prioritization results be improved by adding similarity-based quality metrics into the traditional model of RQ1?*

In addition, in this study, we are interested to determine if we can further improve the test case prioritization, and, thus, we introduce and investigate the following research question:

- RQ3: *Can we further improve test case prioritization results by reusing existing quality metrics into the similarity metrics-based model of RQ2?*

For the purpose of this study, we rely on the same approach to analyze data from five open-source projects. We also compare our findings with those in the original study.

The remainder of the paper is organized as follows: Section 2 summarizes the method and findings from the original study while Section 3 discusses our evaluation method. Section 4 presents detailed discussion of our findings and the comparison with the original study. Finally, Section 5 concludes the paper and highlights future plans.

2 INFORMATION ON THE ORIGINAL STUDY

The original study [11] showed the usefulness of eight traditional metrics and similarity metrics in prioritizing test cases. The test cases are ranked in the order of their fault-proneness after the probability of test cases being fault-prone is predicted using a logistic regression model. According to the proposed model, a higher rank is assigned for a test case with higher predicted probability among a set of candidate test cases. Consequently, the higher ranked test cases are executed earlier. The authors in the original study evaluated their models using five open-source Java projects as shown in Table 1.

2.1 Motivation of the Original Study

The effectiveness of a test suite can be measured by the ability to detect faults as early as possible. Accordingly, test case prioritization has been receiving more research attention and is an important step in ensuring software quality, especially when software development projects suffer from limited resources, *i.e.*, personnel, budget, and time. Since the ultimate goal of test cases is to find faults, an approach to predict failure probability of test cases beforehand may be useful to prioritize test cases.

Instead of starting from the scratch, we re-investigated an existing fault failure prediction approach from the literature. We chose the study by Noor and Hemmati [11] mainly because its predictive model considers real failure history. Additionally, their research questions can serve as the basis of our future research goal of improving the prioritization of test cases. Thus, the purpose of this study is two-folds: (1) this study provides more insights for our ongoing research project and (2) this study will validate the original study on the same dataset.

2.2 Traditional Test Quality Metrics

Four traditional test quality metrics were considered in the original study for the test case prioritization.

- *Traditional Historical Fault Detection Metric (TM)* assesses a test case as effective if the test case already detected fault in previous versions of a system [7]. The TM metric is measured by counting the number of versions for which it failed previously. A higher TM value ranks a test case higher.
- *Method Coverage (MC)* measures method coverage by a test case, *i.e.*, it is the ratio of the number of methods/procedures called by a test case to the total number of methods in the source code [5]. A higher MC value ranks a test case higher.
- *Changed Method Coverage (CMC)* is the ratio between the number of changed methods from the previous version and the total number of methods in the source code. The CMC metric is useful when the goal is to focus on changed parts of the source code. A test case with higher MC value is ranked higher.
- *Size of Tests (ST)* refers to either the lines of code or the total number of *assertions* in a test case. Both our study and the original study consider lines of code as the ST measure. A test case with higher ST value is ranked higher.

2.3 Similarity-based Test Quality Metrics

Four similarity-based test quality metrics were considered in the original study. The authors defined similarity-based test quality metrics based on the execution traces of test cases [10]. For example, two test Cases are considered to be similar, if their execution traces contain the same sequence of method calls. The input to the similarity function is the execution traces, and the output is a value that determines how similar a test case is to prior failed test cases.

- *Basic Counting (BC)* of a test case is the number of unique method calls in the test trace from the current release that also appear in the previous failing sequences for that test case. The BC is normalized between 0 to 1, which is the ratio between the total number of common and unique method calls and the total number of unique method calls from the entire history [10]. A test case with higher BC value is ranked higher.
- *Edit Distance (ED)* a common implementation of edit distance is used, *i.e.*, Levenshtein distance [3]. A lower ED value refers to higher similarity, thus, test cases with lower ED values are ranked higher.
- *Hamming Distance (HD)* is the minimum number of edit operations (insertions, deletions, and substitutions) required to convert a sequence to another one [3]. However, hamming is applicable only on sequences of equal length. A low hamming distance value for a test case puts it to higher rank.
- *Improved Basic Counting (IBC)* is the combination of BC and HD, where by default the BC value is used, unless the BC is too low, in which case the HD value is used. A test case with higher IBC value is ranked higher.

2.4 Building a Regression Model

The original study combined four similarity-based test quality metrics with the traditional test quality metrics into a regression model and prioritized the test cases based on the predicted failure probabilities. The model building was done in two phases: first with only the four traditional metrics, *e.g.*, TM, MC, CMC, and ST, and then with eight test quality metrics—four similarity-based metrics along with the four traditional metrics. To predict the failure probability of a test from a certain version, the model is trained with all previous versions in a cumulative manner. Since, the dependent variable of the model is either pass or fail, the model used was a logistic regression model.

2.5 Subjects Under Study

The original study used five open-source Java projects for the experiment as outlined in Table 1 [11]. The projects are publicly available as a part of the *defects4j* database[1]. The database provides 357 faults and 20,109 JUnit test cases from those five different open-source Java projects [6].

Table 1: Details on the Projects used in the Study.

Project Names	#Versions	#Studied Versions	#TCs	#Versions with #TCs ≥ 5
JFreeChart	26	24	2,205	16
Closure Compiler	133	95	7,927	85
Commons Math	106	43	3,602	36
Commons Lang	65	60	2,245	41
Joda Time	27	25	4,130	16

2.6 Evaluation Metric

Since fault-revealing test cases should be prioritized higher and executed earlier, once a prioritized list of test cases is obtained, the index of the first test case that fails is considered to calculate the percentage of test cases to be executed to find the first fault. Thus for evaluating the ranking, we will use the rank of the first failing test case normalized to a value between 0 and 1 representing the percentage of test cases required to run to reveal the first fault. The ranks for all the versions of a project will be plotted using a boxplot. In these plots, a lower median represents better prioritization of test cases for a project.

To compare two distributions, *i.e.*, prioritized test cases by two alternative models, the original study performed a non-parametric statistical significance test (U-test) [9]. In addition to measuring the statistical significance, it is also important to know the the magnitude of the differences, *e.g.*, the effect size measures [1]. In the original study, the authors used the Vargha and Delaney's Â12 statistics [4, 9].

2.7 Level of Interaction with Original Authors

We report the level of interaction with the authors of the original study as per guidelines by Carver [2].

We had a couple of interactions through email with the first author to understand the approach and in particular to access the original dataset they used for their experiments. Therefore, the level of interaction with the first author in this study was not limited to reading and trying to comprehend the method and results, rather the first author clarified points about their test case ranking mechanism for the tied candidates as well as 'how they find the ranking first failing test case. In summary, the first author guided us as an external expert/consultant. After our interactions with the first authors we made some changes to the prioritization mechanism from the original study, which we discuss in the following section.

2.8 Changes to Original Study

In this section, we explain the deviation of our study from the original study in [11].

Calculation of IBC Metric: In the original study, the authors calculated the IBC metric as the default BC metric. In case the BC value is too low the authors then take the HD value as the IBC metric. But if the HD value is also very low, they consider the BC metric as the value of IBC. To make it simpler we calculate the IBC as the max(BC, HD), and, hence it is straightforward and simple to implement.

30 Sets of Repetitions: When multiple test cases are tied and ranked equal against a measure for a project version, in the original study, the authors performed the ranking 30 times and each time recorded the position of the first failing test case. At the end, they took the median of the positions of the first failing test cases as its final position. This was done by using the 'random' function available in R programming language. However, in this study, we believe performing a random ranking of test cases is impractical without the proper knowledge of the context, and thus, we keep the original ranking as provided by the model and avoid repetitive prioritizations of the same set of test cases multiple times.

2.9 Summary on the Findings from the Original Study

To answer RQ1, in the original study [11] the authors compared the ranks of the first failing test cases using each traditional metric (*e.g.*, TM, CM, CMC, ST) and a combination of these four, *a.k.a.*, the Traditional Model. However, the traditional metrics among ST, MC, CMC and TM do not outperform while compared to each other. Although, the prioritization produced by the model that combines the four traditional metrics is more consistent than the model based on the individual metrics.

To answer RQ2, the original study investigated how the model performs when the similarity-based metrics are integrated, *a.k.a*, the Proposed Model, compared to only the traditional test quality metrics. The findings on RQ2 suggested that, in general, the ranks predicted by the Proposed Model are better than the ranks predicted by the Traditional Model, *i.e.*, the rank of the first failing test case is higher in the Proposed Model than in the Traditional Model.

3 INFORMATION ON THIS STUDY

In this study, we analyze five open-source systems namely Commons Lang, JFreeChart, Commons Math, Joda Time, and Closure Compiler details of which are shown in Table 1. Figure 1 shows an overview of our data collection, model building and comparison, and model improvement. The remainder of this section elaborates each step in brief from Figure 1.

Step 1: Data Collection - The first step involves the data collection and preparation. Traditional metrics like Size of Test Case (ST), Method Coverage (MC), Changed Method Coverage (CMC), and Traditional historical fault-based Metric (TM) are directly extracted from the projects. ST is the number of statements in a test method and is derived from Java Abstract Syntax Tree (AST) parser using the Eclipse JDT API [13]. The MC is calculated in the original study from the execution traces produced by Daikon and AspectJ [10]. For calculating the CMC, first a list of method names is identified in the current version that were modified from the previous version, after counting the change that appears in the current execution trace, the CMC is achieved by dividing the total number of changed

[1]https://github.com/rjust/defects4j

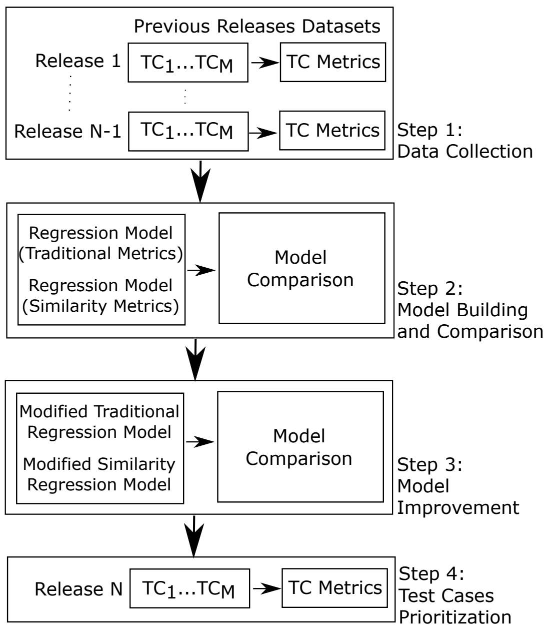

Previous Releases Datasets
Release 1
TC1...TCM
TC Metrics
Release N-1
TC1...TCM
TC Metrics
Step 1: Data Collection
Regression Model (Traditional Metrics)
Regression Model (Similarity Metrics)
Model Comparison
Step 2: Model Building and Comparison
Modified Traditional Regression Model
Modified Similarity Regression Model
Model Comparison
Step 3: Model Improvement
Release N
TC1...TCM
TC Metrics
Step 4: Test Cases Prioritization

Figure 1: An Overview of our Approach to Study the Test Case Failure Prediction.

methods by the total number of unique method calls from the test execution trace and is normalized between 0 and 1.

As an additional step, we also removed the versions for each project that has less than five new or modified test cases. In the original paper, the authors argued that prioritizing a small test suite would not be so practical, and, they ignore any project version during the training of the model. We also ignore the project versions with smaller size of test suite by manually removing them from the original dataset before the training phase.

Step 2: Model Building and Comparison - The second step involves building our models and training them using the test quality attributes for all the versions for each project. In this step, we build the models for both the traditional test quality-based and similarity-based metrics. While finding the model fit, we did not allow the pair-wise interaction between variables. At the end, we evaluate and compare the combined quality and similarity metrics-based models with the individual metric-based models on the test failure probability.

Step 3: Model Improvement - In this step, we attempt to further improve the models. One way we adopted is by adding more features to the model because the number of features in the original model is a few, *i.e.*, only eight metrics. Thus, the fourth step builds our modified models and again train them using the test quality attributes for all the versions for each project. Also, in this step, we build the models for both the traditional test quality-based and similarity-based metrics. At the end, we evaluate and compare with the models in Step 2 the test failure probability.

Step 4: Test Case Prioritization - In this last step, we prioritize the test cases based on their failure probability. The *defects4j* explicitly mentions which test cases fails in a certain version of a project. From the ranked test cases, we find the position of the first failing test case, and compute the percentage of the test cases that we need to execute to find that first fault. This performance measure is also related to the testing resources required in a project, *i.e.*, the more earlier a test case find faults the less resources we require to spend and a system of higher quality can be delivered.

In the following sections, we discuss our variable selection and analysis method.

3.1 Variable Selection

As our independent variables we have a set of eight metrics four of which are traditional test quality metrics and others are similarity-based metrics as discussed in Sections 2.2 and 2.3. The traditional test quality metrics include: Traditional Historical Fault detection metric (TM), Method Coverage (MC), Changed Method Coverage (CMC), and Size of Tests (ST). The similarity-based metrics include: Basic Counting (BC), Edit Distance (ED), Hamming Distance (HD), and Improved Basic Counting (IBC).

In addition to these eight test quality metrics from the original study, we also propose two new quality metrics by reusing three of the metrics from the original study.

- *Weighted Method Coverage (WMC)* is the ratio of method coverage and size of the test case where a test case is assessed as effective if the it has higher WMC value. We define WMC as the ratio of MC and ST, *i.e.*, WMC = $\frac{MC}{ST}$. The WMC metric is normalized between 0 and 1.
- *Weighted Changed Method Coverage (WCMC)* is the ratio of changed method coverage and size of the test case where a test case is assessed as effective if the it has higher WCMC value. We define WCMC as the ratio of CMC and ST, *i.e.*, WCMC = $\frac{CMC}{ST}$. The WCMC metric is normalized between 0 and 1.

Our dependent variable is the status of a test case whether it fails/passes during the execution. This status can be predicted with a probability value based on which the test cases can be prioritized.

Table 2: Information Gain Ratio Across the Five Projects.

Variable	Closure Compiler	Commons Lang	Commons Math	JFree Chart	Joda Time
ST	0.02	0.13	0.04	0.09	0.17
MC	0.24	0.07	0.05	0.12	0.21
BC	0.19	0.08	0.09	0	0.27
HD	0.23	0.09	0.08	0.18	0.23
ED	0.46	0.32	0.28	0.43	0.64
CMC	0.32	0.14	0.07	0.10	0.22
TM	0	0	0	0	0
IBC	0.19	0.08	0.09	0	0.27
WMC	0.16	0.06	0.03	0.11	0.27
WCMC	0.35	0.11	0.04	0.09	0.20

To rank the set of variables and validate the newly proposed metrics WMC and WCMC, the information gain ratio (IGR) has been calculated for each variable across the five projects in our dataset. The IGR is one of the commonly used filter-based feature ranking techniques that is based on the entropy concept from information theory [16]. The IGR for each project is calculated using Equation 1 with the help of *FSelector* R-Package [12]. The newly proposed metrics scored values within the range of other variables in each project, except for JFreeChart WCMC metric came the last by 9%, as shown in Table 2.

$$IGR(Class, Variable) = \frac{Entropy(Class) - Entropy(Class|Variable)}{Entropy(Variable)} \tag{1}$$

According to the information gain ratios, we decided to scope out all the variables which are below the first quartile, to reduce the dimensionality of the dataset under study. To investigate the multicollinearity among the independent variables, we calculated the Variance Inflation Factor (VIF) [8] for each variable. We observed a number of high correlation among some of the independent variables for two projects, namely Closure Compiler and JFreeChart. We removed those variables that scored more than 10 [8] and reported the results of the regression models in Table 3. The ♣ sign indicates a variable has a VIF higher than 10, and was excluded from the initial regression model. For instance, in the Closure Compiler dataset, since the VIF value for CMC was more than 10, this variable was removed and the model was deployed once again without it, until all the VIF scores for all the variables are less than 10. The Odds Ratios (ORs), the 95% intervals, and the AUC values have been reported, as shown in Table 3. For instance, in the JFree Chart dataset, the odds ratio corresponding to the WMC is 2.21, with 95% confidence interval equals to (0.002, 0.09). This implies that if we fix the other variables in this model, increasing the WMC by 1 unit will increase the odds of having a defective test case by 2.21 units. Noting that, MC has been removed initially because of collinearity and WCMC metric has shown as insignificant variable to explain the odds ratio for our dependent variable.

3.2 Analysis Method

We apply the Wilcoxon rank sum and Kruskal-Wallis tests [14] to compare two distributions of the results to see if one outperforms the other using a 95% confidence level (*i.e.*, p-value<0.05). The Wilcoxon rank sum test is a non-parametric statistical test to assess if two distributions are different. The Kruskal-Wallis test is an extension of the Wilcoxon rank sum test applicable to more than two distributions. We choose the Wilcoxon rank sum and Kruskal-Wallis tests because samples are independent of one another and normally distributed. For any comparison exhibiting a statistically significant difference, we further compute the Vargha and Delaney's Â12 statistics to quantify the importance of the difference. The Vargha and Delaney Â12 measure [15] explains the effect size measure between two treatment groups.

3.3 Replication Package

All the data used in this study are publicly available on github[2]. In particular, we provide: (i) the original dataset, (ii) the predicted rank data based on the original models, (iii) the predicted rank data based on the modified models, and (iv) the R sources and all the generated figures.

[2]https://github.com/franpalma/pred-rep/

4 COMPARISON OF RESULTS

We compare our findings with the original study in three phases. First, we build models with the individual traditional test quality metrics (ST, MC, CMC, TM) and compare them with the model comprised of all these four metrics. In this phase we also check if we can further improve the traditional metrics-based model. This answers our RQ1.

Second, we compare the previously built traditional test quality metrics-based model with the similarity metrics-based model that is comprised of four similarity metrics (BC, ED, HD, and IBC) in addition to those four traditional metrics. Here, we also look if we can further improve the similarity metrics-based model. This will answer the RQ2.

Third, we make a global comparison among the original models, the models reproduced in this study, and the modified models, which will answer the RQ3.

4.1 RQ1: Combining the Traditional Test Quality Metrics using a Regression Model

Below, we can summarize the model with the traditional metrics for Closure Compiler as:

We build our logistic regression model with the *glm()* function in R (with the family = *binomial()*), with the 'execution status' of the test case as the dependent variable and three other metrics, *e.g.*, CMC, MC, and ST as the independent variables. The coefficients of the models are:

As we observe, for this particular model, the CMC and MC metrics were more significant than the ST metric. The TM metric was absent for the Closure Compiler project, and, thus, we ignore TM in fitting the model for this particular project. However, this might be not be the case for other projects.

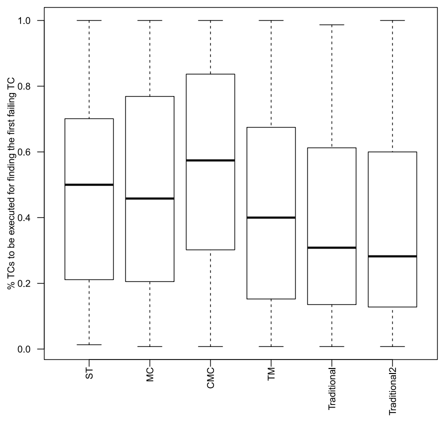
Figure 2 shows the comparison among the ranks of the first failing test cases provided by the models using the traditional metrics TM, MC, CMC, TM, combination of these four (Traditional), and a modified version of the combined model (Traditional2). As shown in Figure 2, none of the individual traditional quality metrics outperforms others for all the project, *i.e.*, individual traditional quality metrics are not consistent in terms of ranking across the projects. This finding also aligns with the original study [11].

However, considering the combined model 'Traditional' in Figure 2, it shows that the model is outperforming all the individual metrics for all the projects with the p-value less than 0.05, as confirmed by a non-parametric U test, *i.e.*, Wilcoxon rank sum test. The difference between the means between each traditional metric and the model with combined metrics is significant, as shown in Table 8.

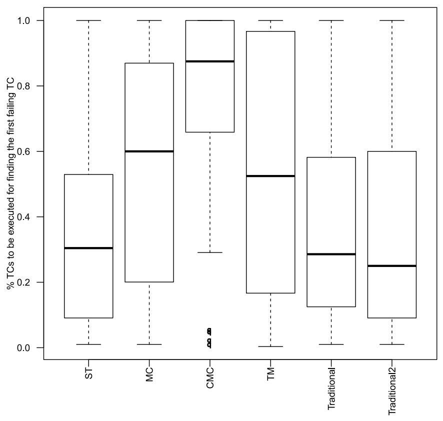
Then, we seek to improve further the ranking performance of the combined model and as discussed in Section 3, we introduced two new features, which slightly improved the ranking for Closure Compiler, Commons Lang, Commons Math, and JFreeChart. In particular, for the Closure Compiler, the ranking with the combination of traditional metrics was 0.308, which then improved to 0.282. These improvement values for Commons Lang, Commons Math, and JFreeChart are: 0.286 to 0.25, 0.131 to 0.128, and 0.235 to 0.205, respectively. It is important to note that whenever we say ranking of test cases as 0.308, it means we need to execute 30.8% of test cases to find the first failing test cases. Thus, for the Closure

Table 3: Analysis of the Logistic Regression Models for the Five Projects with p-value<0.05 and VIF<10. ORs: Odds Ratios.

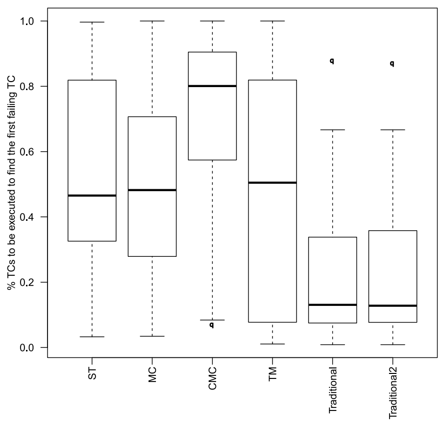
	Closure Compiler			Commons Lang			Commons-Math			JFree Chart			Joda-Time		
Variable	ORs	2.50%	97.50%	ORs	2.50%	97.50%	ORs	2.50%	97.50%	ORs	2.50%	97.50%	ORs	2.50%	97.50%
ST				n.s											
MC	1.63	1.25	2.11							♣			6.60	3.31	14.23
BC				1.60	1.22	2.07							n.s		
HD				n.s			n.s						0.33	0.12	0.84
ED	n.s			0.71	0.53	0.95	1.30	0.99	1.65	0.24	0.11	0.48			
CMC	♣			0.3	0.19	0.44							n.s		
TM															
IBC	1.78	1.44	2.24												
WMC										2.21	1.64	3.03			
WCMC	0.31	0.19	0.46				0.01	0	0.08	n.s					
Intercept	0.05	0.04	0.06	0.01	0.01	0.02	0	0	0.01	0.03	0.02	0.04	0.12	0.07	0.18
AUC	0.72			0.74			0.75			0.75			0.77		



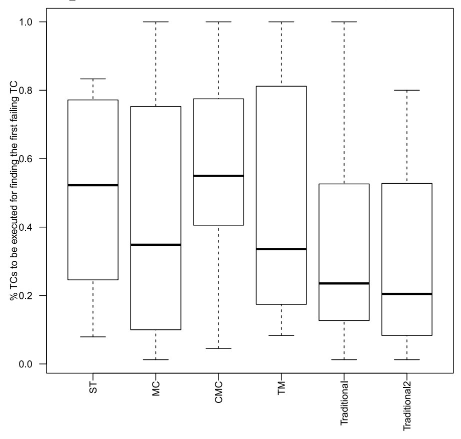
(a) Closure Compiler.



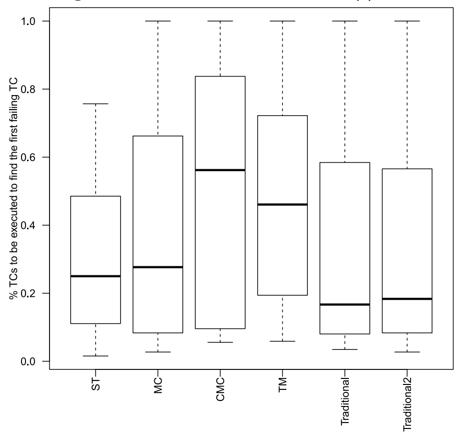
(b) Commons Lang.



(c) Commons Math.



(d) JFreeChart.



(e) Joda Time.

Figure 2: The Comparison among the Ranks of the First Failing Test Cases using TM Model, CMC Model, MC Model, ST Model, the Traditional Metrics-based Regression Model, and Modified Traditional Metrics-based Regression Model for each Version of the Project.

Compiler project, we improved by 0.026, we need to execute 2.6% fewer test cases to find the first failing test case.

However, the ranking performance for Joda Time slightly degraded, which can be due to it unique project nature and requires further investigation.

	Estimate	Std. Error	z value	Pr(>|z|)	
(Intercept)	-2.543e+00	1.129e-01	-22.530	< 2e-16	***
CMC	-1.612e+02	2.699e+01	-5.973	2.32e-09	***
MC	1.152e-03	1.476e-04	7.806	5.92e-15	***
ST	-3.627e-02	1.520e-02	-2.386	0.017	*

Although, it is visible from Figure 2 and Table 8 that we improved the prioritization of test cases for four projects, the statistical significance of the differences and the effect size measured with Vargha and Delaney A measure do not show promising values in Table 4. The p-value is between 0.5252 and 0.9699 with the VD.A effect size measure between 0.45 and 0.56 as negligible.

Table 4: Wilcoxon Rank Sum Test and Vargha-Delaney Â12 Measure for the Traditional Metrics-based Models.

Projects	Treatment Groups	p-value	VD.A Measure
Closure Compiler	Traditional2 ~Traditional	0.774	0.4871032 (negligible)
Commons Lang	Traditional2 ~Traditional	0.5252	0.458953 (negligible)
Commons Math	Traditional2 ~Traditional	0.973	0.4972994 (negligible)
JFreeChart	Traditional2 ~Traditional	0.692	0.4570312 (negligible)
Joda Time	Traditional2 ~Traditional	0.9699	0.4941406 (negligible)

Summary on RQ1: The findings in the original study hold. The combined traditional test quality metrics-based model improves the test case ranking at a scale of statistical significance. The prioritization is further improved after the combined traditional test quality metrics-based model is enhanced with new features but the improvement is not at statistical significance. Out of five projects under study, four projects had better ranking using the model – we had an average improvement of 2.3%.

4.2 RQ2: Adding Similarity-based Metrics to the Traditional Model

To further improve the traditional test quality metrics-based model, we enriched the model with new similarity-based metrics (BC, ED, HD, and IBC). Similarity-based metrics are calculated form the execution trace history of test cases, and, thus, provide more relevant information on the failing probability for a test case [11].

Figure 3 shows the direct comparison of three models in this study. First, we obtained the traditional metrics-based model in RQ1 and boxplot the prioritization results (named as 'Traditional' in Figure 3), then, we boxplot the prioritization results using the model enriched with similarity-based metrics (named as 'Similarity' in Figure 3), finally, we boxplot the prioritization results using the modified model enriched with similarity-based metrics and two new derived metrics (named as 'Similarity2' in Figure 3).

While we boxplot the Similarity model, as shown in Figure 3, the rankings of test case prioritization improved for Commons Lang and JFreeChart, however, the rankings for other three projects are very close to the traditional metrics-based model. The median of rankings of the first failing test cases for Commons Lang using Traditional and Similarity metrics are 0.28571 and 0.16667, respectively. Moreover, the median of rankings of the first failing test cases for JFreeChart for all the versions using Traditional and Similarity metrics are 0.23535 and 0.13068, respectively. Therefore, for Commons Lang and JFreeChart projects the rankings of test cases were improved by 12.04% and 10.47%, respectively.

After we modified the similarity-based model by adding new measures derived from existing metrics, the ranking performance Closure Compiler, Commons Lang, and Joda Time improved. The medians two models (Similarity, Similarity2) for these three projects are (0.38431, 0.3739), (0.16667, 0.14286), (0.20417, 0.14583), respectively. Thus, by adding new measures derived from existing metrics, we improved the rankings for these three projects by 1.04%, 2.38%, and 5.83%, respectively.

Table 5: Wilcoxon Rank Sum Test and Vargha-Delaney Â12 Measure for the Similarity Metrics- and the Traditional Metrics-based Models.

Projects	Treatment Groups	p-value	VD.A Measure
Closure Compiler	Similarity ~Traditional	0.2628	0.4499008 (negligible)
Commons Lang	Similarity ~Traditional	0.3757	0.5571089 (negligible)
Commons Math	Similarity ~Traditional	0.5657	0.4602623 (negligible)
JFreeChart	Similarity ~Traditional	0.6508	0.5488281 (negligible)
Joda Time	Similarity ~Traditional	0.9548	0.4921875 (negligible)

Table 6: Wilcoxon Rank Sum Test and Vargha-Delaney Â12 Measure for the Similarity Metrics- and the Similarity Metrics-based Models.

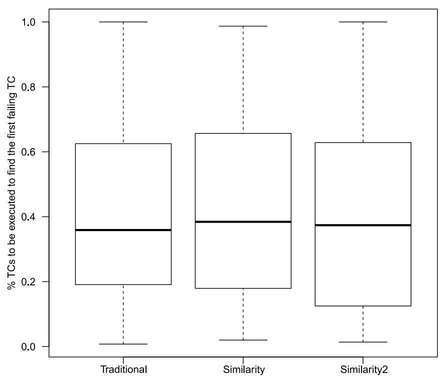
Projects	Treatment Groups	p-value	VD.A Measure
Closure Compiler	Similarity2 ~Similarity	0.7247	0.4841978 (negligible)
Commons Lang	Similarity2 ~Similarity	0.8311	0.4860202 (negligible)
Commons Math	Similarity2 ~Similarity	0.9596	0.496142 (negligible)
JFreeChart	Similarity2 ~Similarity	0.8504	0.5214844 (negligible)
Joda Time	Similarity2 ~Similarity	0.5448	0.4355469 (negligible)

Table 7: Wilcoxon Rank Sum Test and Vargha-Delaney Â12 Measure for the Modified Similarity Metrics- and the Traditional Metrics-based Models.

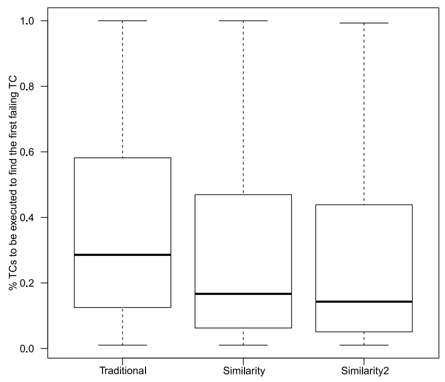
Projects	Treatment Groups	p-value	VD.A Measure
Closure Compiler	Similarity2 ~Traditional	0.6535	0.5201247 (negligible)
Commons Lang	Similarity2 ~Traditional	0.09052	0.3911362 (small)
Commons Math	Similarity2 ~Traditional	0.6123	0.535108 (negligible)
JFreeChart	Similarity2 ~Traditional	0.3654	0.4042969 (small)
Joda Time	Similarity2 ~Traditional	0.5833	0.4414062 (negligible)

Table 5, Table 6, and Table 7 show the significances of the difference and effect sizes between various ranking groups. In particular, from Table 7, it is visible that the rankings provided by the modified similarity metrics-based model outperforms the model based on the traditional test quality metrics. In particular, for the Commons Lang project, we achieved a p-value of 0.09 and a 'small' effect size for VD.A measure. We also showed in Figure 3 that the Similarity2 (modified similarity metrics-based) model outperforms Similarity (similarity metrics-based) model at smaller scale for all the projects except for JFreeChart.

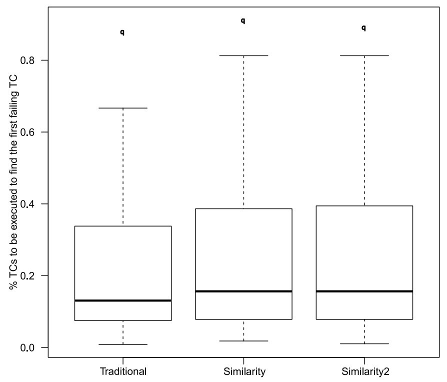
Summary on RQ2: The findings in the original study partially holds. In the original study, the similarity-based model outperformed the traditional quality metrics-based model for all the projects



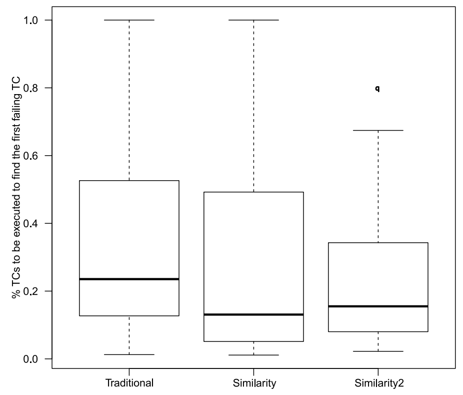
(a) Closure Compiler.



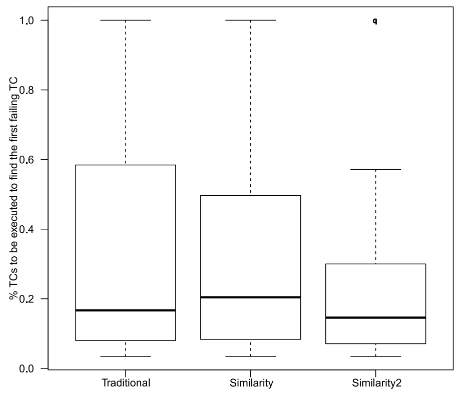
(b) Commons Lang.



(c) Commons Math.



(d) JFreeChart.



(e) Joda Time.

Figure 3: The Comparison among the Ranks of the First Failing Test Cases using the Traditional Metrics-based Model, the Similarity Metrics-based Model, and Modified Similarity Metrics-based Model for each Version of the Project.

except the Closure Compiler. In this study, in addition to the Closure Compiler project, the similarity-based model also did not perform better than the traditional model for the Joda Time project. However, the prioritization can be further improved after enriching the similarity-based model by introducing new measures.

4.3 RQ3: Enrich the Model by Reusing Existing Measures

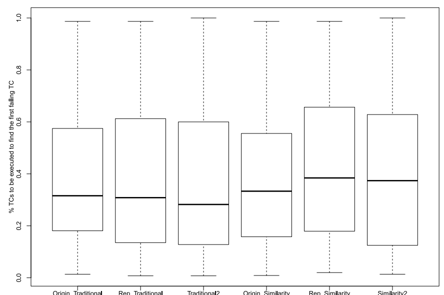
Since, the number of metrics in the original study is low, *i.e.*, only eight metrics, and some of the metrics are correlated and derived from others, improving the model further is a challenge. One possible way was to introduce new measures from the existing ones but still meaningful for the model. In answering the RQ3, we compare the results from the original study, the results with our models, and the results from the modified models and boxplot them in Figure 4.

From the Figure 4, a general observable trend is that for each project, the modified traditional metrics-based model and the modified similarity metrics-based model are outperforming the traditional metrics-based model and similarity metrics-based model, respectively. However, exceptions include Closure Compiler, where the modified similarity-based model did not outperform the similarity-based model from the original study. Also, for the Commons Lang project, the modified traditional metrics-based model did not outperform the traditional metrics-based model in the original study.

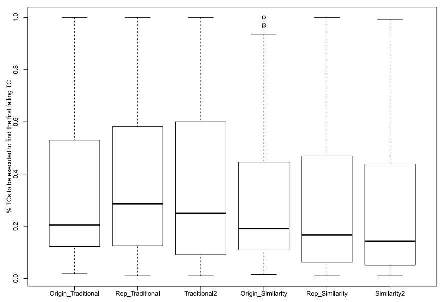
Table 8 provides a global picture of the rankings provided by individual metrics-based models, our reproduced models, and modified models, where the Traditional models include only traditional measures ST, MC, CMC, and TM, and the similarity models include similarity metrics BC, ED, HD, and IBC in addition to these traditional metrics. On the comparison of the rankings with the original study, in general, we achieved consistent prioritizations, *i.e.*, values close to the original studies.

For individual projects, only for the Joda Time project, the modified Traditional model (M_Traditional) underperforms the reproduced model, however, it outperforms the ranking from the original study. As for the modified Similarity model (M_Similarity), only the JFreeChart project underperforms than the reproduced similarity-based model. It is important to mention that on an average, for all the projects, we observe an improved prioritization for the modified Similarity (M_Similarity) model than the original Traditional (O_Traditional) in terms of median of the ranking of first failing test case.

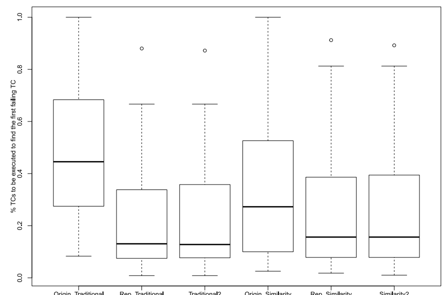
In Figure 4, although we observe the differences in the medians among the three models: modified similarity-based model (Similarity2), reproduced similarity-based model (Rep_Similarity), and



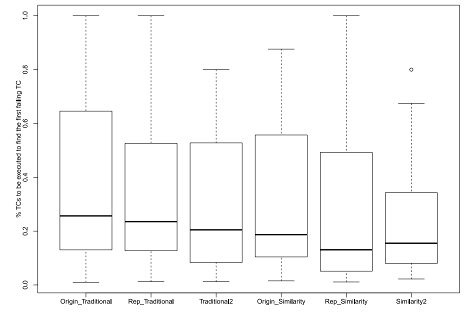
(a) Closure Compiler.



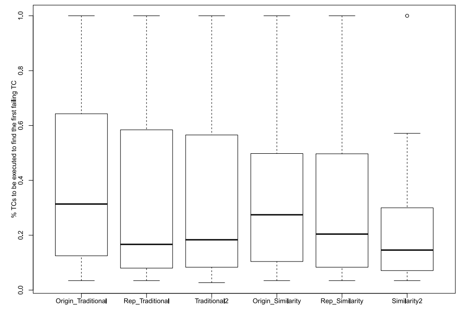
(b) Commons Lang.



(c) Commons Math.



(d) JFreeChart.



(e) Joda Time.

Figure 4: The Comparison of Ranks of the First Failing Test Cases using Original Traditional Metrics-based Model, Reproduced Traditional Metrics-based Model, Modified Traditional Metrics-based Model, Original Similarity Metrics-based Model, Reproduced Similarity Metrics-based Model, Modified Similarity Metrics-based Model for each Version of the Project.

Table 8: The Comparison of Ranks of the First Failing Test Cases using ST, MC, CMC, TM, Traditional, Reproduced Traditional, Modified Traditional, Similarity, Reproduced Similarity, and Modified Similarity Models. 'O_' refers to 'Original', 'R_' refers to 'Reproduced', and 'M_' refers to Modified.

Projects	O_ST	R_ST	O_MC	R_MC	O_CMC	R_CMC	O_TM	R_TM	O_Traditional	R_Traditional	M_Traditional	O_Similarity	R_Similarity	M_Similarity
Closure Compiler	0.468	0.5	0.443	0.458	0.391	0.574	-	-	0.316	0.308	0.282	0.333	0.384	0.374
Commons Lang	0.265	0.304	0.59	0.6	0.354	0.875	0.558	0.525	0.257	0.286	0.25	0.187	0.167	0.143
Commons Math	0.5	0.465	0.5	0.482	0.472	0.801	0.5	0.505	0.205	0.131	0.128	0.191	0.156	0.156
JFreeChart	0.591	0.522	0.364	0.348	0.5	0.55	-	-	0.446	0.235	0.205	0.273	0.131	0.155
Joda Time	0.25	0.25	0.313	0.277	0.479	0.562	-	-	0.314	0.167	0.183	0.275	0.204	0.146
Average	0.415	0.408	0.442	0.433	0.439	0.672	0.529	0.515	0.307	0.225	0.21	0.252	0.208	0.195

reproduced traditional metrics-based model (Rep_Traditional), the statistical significance measured by the Kruskal–Wallis Test as reported in Table 9 is not at the significance level.

Summary on RQ3: The ranking using the modified similarity-based ranking after integrating two new measures into the model improved for all the projects except for the JFreeChart project. However, for the JFreeChart project, although the modified similarity-based model could not outperform the reproduced similarity-based model, it still could outperform the reproduced traditional metrics-based model and the traditional model from the original study.

Nevertheless, we observed a general trend of improvement for all the projects by the similarity-based models than the traditional metrics-based models. In fact, the prioritization can be further improved after enriching our modified similarity-based model by introducing new measures independent of existing ones.

Table 9: Kruskal–Wallis Test for Reproduced Similarity Metrics-based, Modified Similarity Metrics-based, and Reproduced Traditional Metrics-based Models.

Projects	Treatment Groups	p-value
Closure Compiler	Similarity2 ~Rep_Similarity ~Rep_Traditional	0.8687
Commons Lang	Similarity2 ~Rep_Similarity ~Rep_Traditional	0.1642
Commons Math	Similarity2 ~Rep_Similarity ~Rep_Traditional	0.8296
JFreeChart	Similarity2 ~Rep_Similarity ~Rep_Traditional	0.5012
Joda Time	Similarity2 ~Rep_Similarity ~Rep_Traditional	0.7871

4.4 Threats to Validity

In this section, we discuss the threats to the validity of this study: *Internal validity:* The main threat to the internal validity is statistical validity. To minimize the threats to the internal validity, we made sure not to violate assumptions of the significance test. As in to the original study, we relied on the non-parametric Wilcoxon rank sum test. Also, to minimize the threats to the internal validity, we followed each step in the original study for prioritizing test cases to the best of our knowledge. However, in case of confusion, we contacted the first author and received feedback. We did not attempt to minimize the threats to the *external validity* because our focus was improving the prioritization, and we could best do it by applying the modified models on the same dataset. However, we are aware that this study should be extended to other open-source systems to further minimize the threats to the external validity. The *conclusion validity* threats concern the relation between the treatment and the outcome. In our study, we ensured we did not violate the basic assumptions of the statistical tests we performed. In particular, we used non-parametric tests like the Wilcoxon rank sum test (for two treatment groups) and the Kruskal Wallis test (for more than two treatment groups) to test the significance of our ranking results.

5 CONCLUSION

In the context of software testing, test case prioritization (TCP) provides a ranked test suite assessed based on some quality measures. In the original study [11], the authors proposed to measure the defect-proneness of test cases by relying on four traditional metrics, *e.g.*, Traditional Historical Fault detection metric (TM), Method Coverage (MC), Changed Method Coverage (CMC), Size of Tests (ST); and also four similarity-based metrics, *e.g.*, Basic Counting (BC), Edit Distance (ED), Hamming Distance (HD), Improved Basic Counting (IBC).

We conducted an empirical evaluation of the research questions presented in the original study [11] on predicting test case failure to prioritize them. In particular, we analyzed and highlighted conclusions about the ranking performance among the different logistic regression models built on various traditional- and similarity-based test quality metrics. We used the exact dataset as in the original study [11]. However, with the same dataset we attempted to improve the predictive models by combining existing features. Our findings on the research questions are as follows:

- ***RQ1:** Can combining the traditional test quality metrics using a regression model improve test case prioritization compared to ranking the test cases using the individual metrics?*
  As the findings in the original study hold, the combined traditional test quality metrics-based model can improve the test case ranking than individual metrics at a scale of statistical significance.
- ***RQ2:** Can the test case prioritization results be improved by adding similarity-based quality metrics into the traditional model of RQ1?*
  The findings in the original study hold but partially, as in the original study, the similarity-based model outperformed the traditional quality metrics-based model for all the projects except one project (*i.e.*, Closure Compiler). In this study, in addition to the Closure Compiler project, the similarity-based model also did not perform any better than the traditional model for the Joda Time project.
- ***RQ3:** Can we further improve test case prioritization results by reusing existing quality metrics into the similarity metrics-based model of RQ2?*
  The ranking using the modified similarity-based ranking after integrating two new measures into the model improved for all the projects except for the JFreeChart project. We observed a general trend of improvement for all the projects by the similarity-based models than the traditional metrics-based models.

Although this study provides clues on why rankings vary by different predictive models, further validations are required with more open-source systems to generalize the findings from this study. Moreover, this prioritization method should be extended for projects with multiple faults. As such, we also plan to use the APFD (Average Percentage Faults Detected) metrics to evaluate the proposed approach. One possible way to improve the prediction, and, thus, the ranking of test cases is to introduce more independent quality measures.

ACKNOWLEDGMENTS

This study is supported in part by IBM Canada Lab and Collaborative Research and Development Grant (CRDPJ 499518-16) from NSERC, Canada.

REFERENCES

[1] Arcuri, A., and Briand, L. A Practical Guide for Using Statistical Tests to Assess Randomized Algorithms in Software Engineering. In *Proceedings of the 33rd International Conference on Software Engineering* (New York, NY, USA, 2011), ICSE'11, ACM, pp. 1–10.

[2] Carver, J. C. Towards Reporting Guidelines for Experimental Replications: A Proposal. In *1st International Workshop on Replication in Empirical Software Engineering* (2010), Citeseer, pp. 2–5.

[3] Dong, G., and Pei, J. *Sequence Data Mining.* Advances in Database Systems. Springer US, 2007.

[4] Goulden, K. J. Effect sizes for research: a Broad Practical Approach. *Journal of Developmental & Behavioral Pediatrics 27*, 5 (2006), 419–420.

[5] Graham, D., Veenendaal, E. V., and Evans, I. *Foundations of Software Testing.* Cengage Learning, 2008.

[6] Just, R., Jalali, D., and Ernst, M. D. Defects4J: a database of existing faults to enable controlled testing studies for Java programs. In *Proceedings of the 2014 International Symposium on Software Testing and Analysis - ISSTA 2014* (San Jose, CA, USA, 2014), pp. 437–440.

[7] Kim, S., Zimmermann, T., Jr., E. J. W., and Zeller, A. Predicting Faults from Cached History. In *29th International Conference on Software Engineering (ICSE'07)* (May 2007), pp. 489–498.

[8] Kutner, M., Nachtsheim, C., Neter, J., and Li, W. Building the Regression Model II: Diagnostics. In *Applied Linear Statistical Models*, McGraw-Hill, Ed., 5 ed. 2005, ch. 10, pp. 384–420.

[9] Mann, H. B., and Whitney, D. R. On a Test of Whether one of Two Random Variables is Stochastically Larger than the Other. *The Annals of Mathematical Statistics 18*, 1 (1947), 50–60.

[10] Noor, T. B., and Hemmati, H. A Similarity-based Approach for Test Case Prioritization using Historical Failure Data. In *2015 IEEE 26th International Symposium on Software Reliability Engineering (ISSRE)* (November 2015), pp. 58–68.

[11] Noor, T. B., and Hemmati, H. Studying Test Case Failure Prediction for Test Case Prioritization. In *Proceedings of the 13th International Conference on Predictive Models and Data Analytics in Software Engineering* (2017), ACM, pp. 2–11.

[12] Romanski, P., and Kotthoff, L. *FSelector: Selecting Attributes*, 2018. R package version 0.31.

[13] Shavor, S., D'Anjou, J., Fairbrother, S., Kehn, D., Kellerman, J., and McCarthy, P. *The Java developer's guide to Eclipse.* Addison-Wesley Longman Publishing Co., Inc., 2003.

[14] Sheskin, D. J. *Handbook of Parametric and Non-parametric Statistical Procedures.* crc Press, 2003.

[15] Vargha, A., and Delaney, H. D. A critique and improvement of the "cl" common language effect size statistics of mcgraw and wong. *Journal of Educational and Behavioral Statistics 25*, 2 (2000), 101–132.

[16] Witten, I., Frank, E., Hall, M., and Pal, C. Algorithms: The Basic Methods. In *Data Mining: Practical machine learning tools and techniques*, 4 ed. Morgan Kaufmann, 2017, ch. 4, pp. 91–160.