# Test-Related Factors and Post-release Defects: An Empirical Study

Fabiano Pecorelli
University of Salerno
Italy
fpecorelli@unisa.it

## ABSTRACT

Testing is a very important activity whose purpose is to ensure software quality. Recent studies have studied the effects of test-related factors (e.g., code coverage) on software code quality, showing that they have good predictive power on post-release defects. Despite these studies demonstrated the existence of a relation between test-related factors and software code quality, they considered different factors separately. That led us to conduct an additional empirical study in which we considered these factors all together. The key findings of the study show that, while post-release defects are strongly related to process and code metrics of the production classes, test-related factors have a limited prediction impact.

## CCS CONCEPTS

• **Software and its engineering → Software testing and debugging**.

## KEYWORDS

Software Testing, Post-release Defects, Empirical Study

**ACM Reference Format:**
Fabiano Pecorelli. 2019. Test-Related Factors and Post-release Defects: An Empirical Study. In *Proceedings of the 27th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE '19), August 26–30, 2019, Tallinn, Estonia.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3338906.3342500

## 1 RESEARCH PROBLEM AND MOTIVATION

Software systems are constantly subject to continuous changes. In order to check whether a change introduces one or more defects [13], developers rely on *software testing* that represent an effective defense weapon against the introduction of defects [8].

Many past studies investigated the properties that make test code more effective [1, 2, 9, 10] and demonstrated that test code quality strongly impacts the number of post-release defects contained in the exercised classes [2, 6].

However, we identified some common limitations in the previous work: first, all these studies analyzed the impact of several test-related factors in isolation, without controlling for other test-related factors, neither for additional control variables related to production code (e.g., product or process metrics [11], [12]). So, it is still unclear how putting all these findings together in a single statistical model will affect the previously identified results.

The goal of the study is to address these limitations by building a more reliable statistical model that is able to describe the real effects of test-related factors on post-release defects.


*ESEC/FSE '19, August 26–30, 2019, Tallinn, Estonia*

ACM ISBN 978-1-4503-5572-8/19/08.
https://doi.org/10.1145/3338906.3342500

## 2 BACKGROUND AND RELATED WORK

In the last decade, a number of researchers focused their studies on assessing the defect-proneness of source code considering test-related factors to describe the number of future defects in a system.

Nagappan et al. [11] used the STREW-J metric suite [9] to find a relation between in-process testing metrics and software quality. The results, later confirmed by Rafique and Misic [12] showed a significant influence of the effort invested into testing on code quality. Other studies found an influence of test effort and test-driven development on product quality [10, 19] based on other testing metrics such as code coverage [1, 2, 10] and other static metrics (e.g., number of assertions) [11]. Kudrjavets et al. [7] showed the existence of a high correlation between assertion density and defect-proneness of production code, while Chen and Wong [2] used code coverage for software failures prediction and showed that this metric influences code quality. Later, Cai and Lyu [1] confirmed this result. Nevertheless, a recent work by Kochhar et al. [6] contradicted those findings, reporting that coverage has an insignificant correlation with the number of post-release defects. Finally, Spadini et al. [17] studied the relation between test smells and post-release defects, finding that production classes exercised by smelly test suites are more defect-prone.

## 3 APPROACH AND UNIQUENESS

Many Software Engineering research papers investigated the relation between testing and software quality. However, all these studies only considered the effects of a limited set of test-related factors, without combining them and without considering the effects of some other control variables (e.g., size of the production class). The goal of this study is to assess the impact of these factors on the quality of software code when considered all together. The details of the statistical model are reported below.

**Dependent variable** Since many previous studies on this topic rely on post-release defects to describe software code quality, we used it as our dependent variable. We calculate it by performing the following steps:

- For each system, we identified the bug-fixing commits analyzing the commit messages as proposed by Fischer et al. [3];

**Table 1: Set of variables used in the experiment**

| Group | Name | Description |
|---|---|---|
| Independent Variables | T-LOC | Number of lines of code of the Test Class |
| | T-WMC | Weighted Method Count of the Test Class |
| | T-EC | Efferent coupling of the Test class |
| | Assertion Density | Percentage of assertion statements in the test code (i.e., number of assertions / T_LOC) |
| | Assertion Roulette | A value indicating if a test contains several assertions with no explanation |
| | Eager Test | A value indicating if a test method tests more methods of the production target |
| | Resource Optimism | A value indicating if a test makes optimistic assumptions on the existence of external resources |
| | Line Coverage | Number of statement in production class that are covered by the test |
| | Mutation Coverage | Number of mutated statement in production class that are covered by the test |
| Control Variables | LOC | Number of lines of code of the Production Class |
| | WMC | Weighted Method Count of the Production Class |
| | EC | Efferent coupling of the Production class |
| | pre-release changes | Number of changes affecting a class prior to the selected release |
| | pre-release defects | Number of defects affecting a class prior to the selected release |

- For each bug-fixing commit we used a function included in PyDriller [16], named `get_commits_last_modified_lines`, that exploits the SZZ algorithm to determine the set of defect-inducing commits.
- For each release of a system, we calculated post-release defects as the number of defect-inducing commits in the period between the selected release and its subsequent release.

**Control Variables** Besides considering test-related factors, we also included a set of well-known metrics as control factors (see Table 1) in order to avoid a biased interpretation of the results.

**Independent Variables** In order to select the independent variables, we analyzed a huge set of previous studies in literature in order to search for test related factors influencing the quality of software code. As a result, we obtained a large set of variables describing both static qualities (e.g., assertion density), the dynamic behavior (e.g., line coverage), and the presence of test smell (e.g., Eager Test). Table 1 reports the complete list of variables.

**Statistical Modelling and Data Analysis** After collecting data for all the considered projects, we built a statistical model relating the independent and control variables to the post-release defects relying on a Multiple Linear Regression model [14]. To verify its assumption, we executed the Shapiro-Wilk normality test [15] to verify the normality of the distribution and a hierarchical clustering based on the Spearman's rank correlation coefficient [18], removing variables having a correlation higher than 0.6 with any other variable, to deal with multicollinearity.

## 4 RESULTS AND CONTRIBUTIONS

We performed a preliminary evaluation on three software systems belonging to the Apache family: COMMONS-IO, COMMONS-LANG, and COMMONS-MATH.

**Table 2: Results table - The impact of test-related factors on the number of post-release defects.**

| | Estimate | S.E. | Sig. |
|---|---|---|---|
| Intercept | -0.1082 | 0.1047 | |
| Line Coverage | 0.1185 | 0.0947 | |
| T-LOC | -0.0000 | 0.0002 | |
| Assertion Density | 0.1567 | 0.1869 | |
| Assertion Roulette | -0.1475 | 0.0792 | . |
| Eager Test | -0.1123 | 0.0565 | * |
| ResourceOptimism | 0.0893 | 0.5284 | |
| LOC | 0.0008 | 0.0001 | *** |
| Pre-Release Changes | 0.0022 | 0.0057 | |
| Pre-Release Defects | 0.0642 | 0.0106 | *** |
| Multiple R-squared: 0.369; Adjusted R-squared: 0.355 | | | |

significance codes: '***'p <0.001, '**'p <0.01, '*'p <0.05, '.'p <0.1

The preliminary results (see Table 2) show that software code quality is not strongly influenced by test-related factors.

The variables that highly affect software quality are LOC and PRE-RELEASE DEFECTS that confirm the findings of previous studies. Just as large classes are more defect-prone than the other [20], the past fault history also affects the number of future defects [4, 5].

The only two significant factors related to tests are ASSERTION ROULETTE and EAGER TEST. The former is a test smell describing test classes having a large number of assertions without documentation, while the latter describes a test method exercising more than one production method. Despite the presence of test smells is an indicator of poor quality of test code, in this case, it increases the quality of production code. Indeed in both cases, the Estimate value is negative, which means that the higher the number of Assertion Density and Eager Test, the lower the number of Post-release defects. In the first case, it is correct to think that a high number of assertions (documented or not) makes tests more robust. Also in the case of Eager Test, having test method exercising more than one method can help the early identification of defects and consequently reduce the number of future defects.

The main surprising outcome of our study is that most of the considered test-related factors are not able to explain post-release defects. Despite this could seem strange, a possible reason is that, while good quality tests are likely to accurately identify defects present in the same snapshot of the production code, they do not necessarily predict the future defects affecting the class under test.

The major contributions provided by this paper are the following:

- The inclusion of control variables related to the structure (e.g., LOC) or to the past history (e.g., Pre-Release-Defects) of production code overshadows the effects of test-related factors on the prediction of post-release defects.
- Test-related factors commonly known and used in literature could be not appropriate enough to catch the defect-proneness of the exercised code. Proposing new metrics able to describe the relation between tests and software quality could be an interesting cue for future works.

# REFERENCES


[1] Xia Cai and Michael R Lyu. 2007. Software reliability modeling with test coverage: Experimentation and measurement with a fault-tolerant software project. In *The 18th IEEE International Symposium on Software Reliability (ISSRE'07)*. IEEE, 17–26.

[2] M-H Chen, Michael R Lyu, and W Eric Wong. 2001. Effect of code coverage on software reliability measurement. *IEEE Transactions on reliability* 50, 2 (2001), 165–170.

[3] Michael Fischer, Martin Pinzger, and Harald Gall. 2003. Populating a release history database from version control and bug tracking systems. In *Software Maintenance, 2003. ICSM 2003. Proceedings. International Conference on*. IEEE, 23–32.

[4] Todd L Graves, Alan F Karr, James S Marron, and Harvey Siy. 2000. Predicting fault incidence using software change history. *IEEE Transactions on software engineering* 26, 7 (2000), 653–661.

[5] Sunghun Kim, Thomas Zimmermann, E James Whitehead Jr, and Andreas Zeller. 2007. Predicting faults from cached history. In *Proceedings of the 29th international conference on Software Engineering*. IEEE Computer Society, 489–498.

[6] Pavneet Singh Kochhar, David Lo, Julia Lawall, and Nachiappan Nagappan. 2017. Code coverage and postrelease defects: A large-scale study on open source projects. *IEEE Transactions on Reliability* 66, 4 (2017), 1213–1228.

[7] Gunnar Kudrjavets, Nachiappan Nagappan, and Thomas Ball. 2006. Assessing the relationship between software assertions and faults: An empirical investigation. In *2006 17th International Symposium on Software Reliability Engineering*. IEEE, 204–212.

[8] Meir M Lehman, Juan F Ramil, Paul D Wernick, Dewayne E Perry, and Wladyslaw M Turski. 1997. Metrics and laws of software evolution-the nineties view. In *Proceedings Fourth International Software Metrics Symposium*. IEEE, 20–32.

[9] Nachiappan Nagappan et al. 2005. A software testing and reliability early warning (strew) metric suite. (2005).

[10] Nachiappan Nagappan, E Michael Maximilien, Thirumalesh Bhat, and Laurie Williams. 2008. Realizing quality improvement through test driven development: results and experiences of four industrial teams. *Empirical Software Engineering* 13, 3 (2008), 289–302.

[11] Nachiappan Nagappan, Laurie Williams, Mladen Vouk, and Jason Osborne. 2005. Early estimation of software quality using in-process testing metrics: a controlled case study. In *ACM SIGSOFT Software Engineering Notes*, Vol. 30. ACM, 1–7.

[12] Yahya Rafique and Vojislav B Mišić. 2013. The effects of test-driven development on external quality and productivity: A meta-analysis. *IEEE Transactions on Software Engineering* 39, 6 (2013), 835–856.

[13] Gregorio Robles, Juan Jose Amor, Jesus M Gonzalez-Barahona, and Israel Herraiz. 2005. Evolution and growth in large libre software projects. In *Eighth International Workshop on Principles of Software Evolution (IWPSE'05)*. IEEE, 165–174.

[14] George AF Seber and Alan J Lee. 2012. *Linear regression analysis*. Vol. 329. John Wiley & Sons.

[15] Samuel Sanford Shapiro and Martin B Wilk. 1965. An analysis of variance test for normality (complete samples). *Biometrika* 52, 3/4 (1965), 591–611.

[16] Davide Spadini, Maurício Aniche, and Alberto Bacchelli. 2018. *PyDriller: Python Framework for Mining Software Repositories*. https://doi.org/10.1145/3236024.3264598

[17] Davide Spadini, Fabio Palomba, Andy Zaidman, Magiel Bruntink, and Alberto Bacchelli. 2018. On the relation of test smells to software code quality. In *2018 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, 1–12.

[18] Charles Spearman. 1904. The proof and measurement of association between two things. *American journal of Psychology* 15, 1 (1904), 72–101.

[19] Jaymie Strecker and Atif M Memon. 2012. Accounting for defect characteristics in evaluations of testing techniques. *ACM Transactions on Software Engineering and Methodology (TOSEM)* 21, 3 (2012), 17.

[20] Hongyu Zhang. 2009. An investigation of the relationships between lines of code and defects. In *2009 IEEE International Conference on Software Maintenance*. IEEE, 274–283.