Do open source software projects conduct tests enough?

Ryohei Takasawa[1], Kazunori Sakamoto[2], Akinori Ihara[3], Hironori Washizaki[1], Yoshiaki Fukazawa[1]

[1] Waseda University, 3-4-1 Okubo, Shinjuku-ku, Tokyo, Japan
[2] National Institute of Informatics, 2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo, Japan
[3] Nara Institute of Science and Technology, 8916-5 Takayama, Ikoma, Nara, Japan

Abstract. Do open source software projects provide and maintain tests? What metrics are correlated with the test success? This paper answers these questions by executing tests of 452 open source software projects in GitHub and measuring 13 metrics from 77 projects. Only 117 projects passed all test cases. Additionally, the results are correlated with the comment density, public documented API density, and test coverage.

1 Introduction

The number of Open Source Software (OSS) projects is increasing [1]. Because the source code in OSS is more readily available than commercial software, empirical studies on OSS are being actively conducted. For example, Hars et al. analyzed the reasons why developers participate in OSS projects [2]. Schryen et al. compared OSS and Closed Source Software (CSS) in terms of vulnerability [3], while other studies have examined OSS from various viewpoints. However, little is known about the test activities on OSS.

It is unclear what percentage of OSS projects pass all their test cases and then how carefully we should treat OSS projects in general. Moreover, executing tests is a simple and good way to evaluate OSS projects, but such tasks are time consuming. It is also unclear whether there are useful metrics to estimate test quality such as test results (passed or failed) and test coverage without execution.

Thus, we conducted large-scale analysis of OSS projects by collecting 791 Maven projects from GitHub, executing tests of 452 projects and measuring 13 metrics from 77 projects. As a result, only 117 projects passed all their test cases and comment metrics (comment density and public documented API density) and test coverage are useful metric to estimate test results.

The research questions (RQs) of this paper are following.

RQ1 What percentage of OSS projects pass all their test cases?
RQ2 Which metrics are useful to estimate whether OSS projects are well tested without execution?

The contributions of this paper are following.

- We show only 14.8% OSS projects passed all their test and we recommend users and contributors to treat OSS projects carefully.
- We find comment metrics are useful to evaluate OSS projects quickly without execution.

2 Experimental Setup

We collected projects in GitHub[4], a famous project hosting service. We targeted projects that use Maven[5], which is the most poplular project management tool for Java, because it automates projects' build, reporting and documentation, including testing and measuring. We used Maven to test projects and measure metrics through SonarQube[6], a metrics management platform.

We collected all the found projects with a search query indicating that target projects contain the `pom.xml` file which is a build file for Maven. We limited the size of the `pom.xml` file is 8,798-8,894 or 9,001-9,189 or 9,501-9,562 bytes in order to avoid the limitation of the search result of GitHub, which shows 1,000 projects at a maximum.

We chose 13 metrics[7] : lines of code, number of statements, number of files, comment density, lines of comments, cyclomatic complexity per files, total cyclomatic complexity, line coverage, branch coverage, public documented API density, duplicated lines density, violations, violations per file.

Lines of code, number of statements and number of files are a popular index of the scale of a project. In this paper, lines of code exclude comments. Comment density is defined as the ratio of the lines of comments from the lines of code and comments. Line coverage and branch coverage are the ratio of the executed program elements (line or branch) in testing. Public documented API density is defined as the ratio of public APIs with document from all public APIs which are public classes, interfaces, methods, constructors, annotations and fields. Duplicated lines density is defined as the ratio of duplicated lines from all lines. Violations is defined as the number of issues found in static code analysis.

3 Experimental Results

Although 791 projects were gathered from GitHub, the metrics in some projects could not be measured due to build failures, especially failures caused by omissions of the dependency on the configuration file. Therefore, the experiment included 452 projects (57.1%) that were built without errors.

Next we executed tests and collected the metrics from these 452 projects. Test cases were not run in 276 projects (34.9%). In 59 projects (7.46%), tests cases were run but failed. Only 117 (14.8%) passed the tests.

[4] http://github.com/
[5] http://maven.apache.com
[6] http://www.sonarqube.org/
[7] http://docs.codehaus.org/display/SONAR/Metric+definitions

In order to measure meaningfully, we targeted projects which have 10 or more test cases and 1 or more files. Then we analyzed the averages of the metrics values for the 51 projects that successfully passed all the test case runs (successful projects) and the 27 projects that failed some test cases (failed projects).

Table 1 shows the results. Although successful projects had higher values of most metrics than failed projects, only 4 metrics listed in Table 1 had statistically significant difference in the average values ($p < 0.05$).

Table 1. Results of metrics

	Comment Density		Documented API Density		Line Coverage		Branch Coverage	
	Success	Failure	Success	Failure	Success	Failure	Success	Failure
average	18.9%	10.8%	52.4%	35.0%	50.5%	31.0%	43.6%	23.2%
dispersion	0.0102	0.0100	0.0957	0.103	0.107	0.0704	0.0918	0.0727
p value	0.00121 (< 0.05)		0.0223 (< 0.05)		0.0170 (< 0.05)		0.00769 (< 0.05)	

4 Discussion

Surprisingly, 14.8% of projects have test cases and pass all of them (RQ1). Pham et al. found that projects with test cases have more contributions from outside developers than those without test cases [4]. Speaking in terms of psychological aspects [5], not fixing failed test cases can be harmful because their presence may cause developers to ignore future failed test cases, increasing the number of failed test cases.

Thus, a method to aid developers in writing test cases should improve OSS projects. We recommend that users of OSS projects pay attention to the existence of test cases and the OSS quality. Because most OSS projects lack the ability to test the contributions from outside developers, we also recommend that contributors write test cases to assure the quality of their contributions.

Although reviewing and executing existing test cases is a simple and effective way to evaluate OSS quality, it is a costly task. Arafat et al. mentioned that successful OSS projects are consistently well documented and commented [6]. Our results show that the comment metrics and test coverage affect the test results. Because the comment metrics can be measured without execution, it is useful to evaluate OSS projects quickly (RQ2).

5 Threats to Validity

The experiment only used Maven projects, which may cause limitations. However, measuring metrics or running test code is difficult without using tools. Consequently, this issue is unavoidable when studying metrics or testing.

There are other repository hosting services besides GitHub (e.g., SourceForge[8] or Google Code[9]). The difference of the service may affect the experimental results, thus, we will conduct more experiments on other services.

6 Conclusion

Herein we mined OSS projects in GitHub, and we found that most of projects do not have test code. Furthermore, we found a correlation between the testing results and metric values.

The answers (As) to the research questions are as follows:

A1 Only 14.8% of the projects passed their test cases. Thus, users should pay attention to the quality of OSS projects and contributors should write their own test cases.

A2 The comment metrics and test coverage are correlated with the test results, thus, the comment metrics can be used as lightweight metrics to evaluate OSS projects without execution.

We plan to publish our data set and create a platform to search specific characteristics of OSS projects. For example, users can search projects by the percentage of successful test cases. This platform may make it easier to conduct studies on OSS projects.

References

[1] Amit Deshpande, et al.: The Total Growth of Open Source, IFIP Advances in Information and Communication Technology, Vol. 275, 197–209 (2008).

[2] Alexander Hars, et al.: Working for Free? -Morivations of Participating in Open Source Projects, International Journal of Electronic commerce, Vol. 6, 25–69 (2002).

[3] Guido Schryen, et al.: Open source vs.closed source software: towards measurring security, In Proc. of the 24th Annual SCM Symposium on Applied Computing, 2016–2013 (2009).

[4] Raphael Pham, et al.: Creating a Shared Understanding of Testing Culture on a Social Coding Site. In Proc. of the 35th International Conference on Software Engineering, 112–121 (2013).

[5] Wilson, James Q., and George L. Kelling: Broken windows. Atlantic monthly 249.3 29–38 (1982).

[6] Oliver Arafat, et al. The Commenting Practice of Open Source. In Proc. of 24th ACM SIGPLAN conference companion on Object oriented programming systems languages and applications, 857–864 (2009).

[8] http://sourceforge.net/

[9] http://code.google.com/