An Empirical Study of Adoption of Software Testing in Open Source Projects

Kochaar Pavneet Singh, Tegawendé F. Bissyandé, David Lo, Lingxiao Jiang

An Empirical Study of Adoption of Software Testing in Open Source Projects

Pavneet Singh Kochhar[1], Tegawendé F. Bissyandé[2], David Lo[1], and Lingxiao Jiang[1]
[1]*Singapore Management University, Singapore*
[2]*LaBRI, University of Bordeaux, France*
kochharps.2012@smu.edu.sg, bissyand@labri.fr, davidlo@smu.edu.sg, lxjiang@smu.edu.sg

***Abstract*—Testing is an indispensable part of software development efforts. It helps to improve the quality of software systems by finding bugs and errors during development and deployment. Huge amount of resources are spent on testing efforts. However, to what extent are they used in practice? In this study, we investigate the adoption of testing in open source projects. We study more than 20,000 non-trivial software projects and explore the correlation of test cases with various project development characteristics including: project size, development team size, number of bugs, number of bug reporters, and the programming languages of these projects.**

***Keywords*-Empirical study, Software testing, Adequacy, Test cases**

I. Introduction

Software testing is an important part of software development life-cycle. Despite availability of various tools to ensure quality of software, most of the software products suffer from insufficient testing. The impact of inadequate testing are increased failures which leads to poor quality software, higher software development costs and delays the time to market the product. A study conducted by the National Institute of Standards and Technology reported that inadequate software testing costs the U.S economy $59.5 billions annually, i.e., about 0.6% of its GDP [33]. The number of bugs overwhelms the number of developers working in the project. A triager from Mozilla project admitted that the project receives almost 300 bugs everyday that need triaging [1]. These figures reinforces the fact that software testing is paramount for developing bug free software.

Software testing is used to verify that the program or system under test produces the desired output based on the set of inputs and execution environment, which are specified in the requirements document. As the complexity of software increases, detecting all software bugs is practically impossible, thus, making complete testing infeasible. In the past, several studies have explored different software testing strategies and techniques to address these challenges and propose methods to perform exhaustive testing of software [2], [6], [11], [12].

Although a large body of research about software testing has been built, software programs continue to suffer from numerous defects. Consequently, is software testing really popular in development projects? Does it noticeably impact the quality of software code? What kind of projects are more likely to include tests? These are some of the important questions which can increase our understanding of the unexplored areas of software testing and its impact on software evolution. Our goal in this paper is indeed to fill a research gap in the importance of software testing through a large-scale empirical evaluation.

In this study, we analyse a large number of open source projects from the GitHub hosting site. GitHub platform holds millions of software projects including important projects such as Linux and Ruby on Rails. GitHub provides various features which makes it an important platform for storing open source projects. GitHub also provides an in-house issue tracking system where users record issues and classify them as bugs, feature requests, and other self-defined categories. We investigate in this study different characteristics of software development that are related to testing: e.g., numbers of developers in projects that include test cases. We also study how the presence/absence of test cases can affect the quality of software in terms of the number of reported bugs. Finally, we investigate the programming languages in relation to the projects with test cases.

The contributions of this paper are as follows:

1) We are the first to perform a large-scale empirical study on the adoption of software testing in practice. Our study involves the analysis of more than 20,000 software projects of sizes ranging from 500 to 17 millions LOC.
2) We examine the relationships between the number of test cases and various project characteristics. These include: the size of the projects, the number of the developers in the projects, the number of bugs, the number of bug reporters, and the programming languages that the projects are written in.
3) We employ a number of statistical tests to confirm if the findings of our experiments are statistically significant.

This empirical study is an extension of our previous preliminary work (described in a 4-page ERA (early research achievement) track paper) [19]. For this project, we curate the previously collected data to filter toy projects (i.e. projects of small sizes). Further, we examine three more research questions in addition to the two questions discussed in the previous empirical study.

The structure of this paper is as follows. In Section II, we describe preliminary materials about test cases and GitHub. Our empirical study methodology is presented in Section III. Next, we present the result of our experiments that answer a number of research questions in Section IV. We highlight

several threats to validity in Section V. Related work is described in Section VI. Section VII concludes and describes future work.

II. Preliminaries

In this section, we mainly discuss the importance of testing in software development and briefly describe GitHub, the platform where we have collected the dataset of projects for our study.

A. Software Testing and Test Cases

Software development produces programs that are often buggy or incomplete with respect to some features. Software complexity, (and therefore that of bugs) grows to the limits of our ability to manage that complexity [3]. To verify that software is compliant with its requirements, developers often resort to software testing. As developers go on addressing previously discovered bugs while simultaneously adding new features, they produce complex software containing subtler bugs that are more challenging to detect and to handle. This in turn makes thorough testing tedious but necessary. Actually, studies have shown that software testing accounts for 40-70% of the time and cost of the complete software development process [25]. Unfortunately, although software testing is a challenging, time consuming and expensive exercise, it has become a common part of the development process, as releasing software with inadequate testing may lead to even higher costs [33].

Testing is an activity which is managed through the building of a set of test cases. A test case consists of a set of inputs, preconditions, as well as the associated expected results to be compared with the outputs of the program under testing. Together with test scripts, test cases are constructed to check whether a given software performs in conformance with its requirements. Test cases can further provide the following information such as bug count, help managers in decision making, examine quality of product, how to get these bugs fixed etc. Tests are useful when they produce significant results, provide insight into a product or application, easier to evaluate and reveal valuable information.

Exhaustive testing requires, for each functional requirement, both a positive test, which evaluates that an application produces the expected result, and a negative test that evaluates the outcome when the application is run under conditions outside what is defined. Such tests are included in test suites which also contain detailed descriptions of test cases and are progressively constructed by project teams. It is to be noted however that the presence of a large number of test cases does not guarantee that a software program is free from defects. Developers therefore regularly seek new ways to test their code more effectively. To this end, they often rely on various approaches, such as test case prioritization and test case reduction techniques that have been proposed to reduce time and cost of software testing [4], [16], [18], [20], [21]. In this study we focus on examining how the presence/absence of test cases in a software project correlate with various characteristics of software development.

B. GitHub

GitHub is a web-based project hosting platform which was launched in 2008 and has become one of the premier open source development sites hosting more than 3,000,000 projects. GitHub implements the concept of social coding to create a developer-friendly environment where developers are enabled to network, collaborate and promote their projects.

GitHub, an open source coding repository site allows developers to create and manage projects. It provides features such as followers, feeds and network graph for developers to monitor their repository. GitHub provides an open source wiki engine gollum which is backed by Git and provides plethora of text formats.

Most of GitHub project repositories are publicly accessible and can be retrieved through an extensive set of REST APIs [9]. GitHub hosts diverse types of projects from various application domains, from gaming software, web applets, to operating systems. The code in the projects are also written in a myriad of programming languages by development teams ranging from 1 to several thousand developers. This variety of project instances makes GitHub an appealing source to collect data for empirical studies.

Although our study is on the test cases in software projects, we also collect information about other development artifacts in order to estimate the correlations between test cases and a number of software characteristics. We have thus collected huge amounts of data from GitHub through its API and processed raw program code data to extract information such as numbers of lines of code and test cases, which are not directly available through the GitHub API.

III. Methodology

For our empirical study, we analyse projects downloaded using the GitHub API. GitHub does not follow a distinct ordering scheme to download the projects. Thus, the results vary every time with a new request. To ensure that most of the projects are non toy projects in our dataset, we filtered the data and selected the projects which have more than 500 lines of code (LOC). We have in total 20,817 projects of sizes 500 to 17 millions LOC. These include well-known projects such as Ruby on Rails and jQuery.

A. Collecting the dataset

a) Lines of code: GitHub uses the git software configuration management system (SCM) to store software revisions. We cloned the git repositories of the projects and used the SLOCCount[1] utility to count the lines of code of the latest revisions of these projects. Figure 1 shows the lines of code of different projects. We observe that 40% of our projects

[1] http://dwheeler.com/sloccount

have LOC between 1,000 and 5,000. Around 27% of the projects lie between 500 to 1,000 LOC , while more than 23% of the projects have more than 10,000 lines of code. Also, over 15,000 projects (total 20,817 projects) have more than 1,000 LOC.

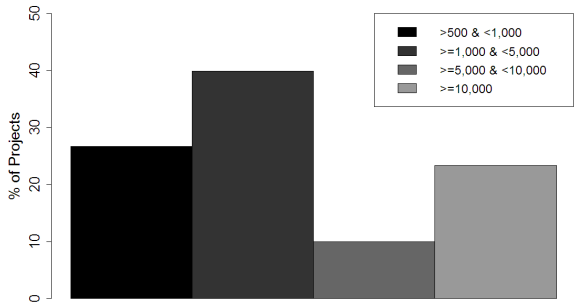

Figure 1. Distribution of Projects in Terms of Total Lines of Code

b) Test Cases: Test cases are an important part of a project as they help developers confirm whether their code meets the requirement laid down for the software. Collection of test cases for large number of projects is an arduous task as different languages follow different naming conventions. We perform a *lightweight* identification of test cases that can scale to thousands of projects. We notice that most test cases contain the word "test" as part of their file names. Thus we select files whose name contains the word "test". For each project, we then count the number of such files which are treated as the number of test cases. We then investigate the relationships between the number of test cases and various project characteristics.

c) Issues & Bugs: GitHub has its own issue tracking system which provides issue trackers for each hosted project where reporters can file issue tickets, and label them with different tags. We collect all the issues (open and close) reported through the in-house tracker. We further find information such as reporter's identity and different labels used to report the issues. In our dataset, issues are labelled as enhancement, bug, feature requests, error, fixed etc. Further, we find the issues labelled as bugs, errors or defects because they most likely represent the actual bugs in the project. We also calculate the number of bug reporters in a project, i.e., people who reported issues for the project.

d) Developer contributions: Git records store contributors name and email for each revision of the repository. There are two types of contributors: committers and revision authors. Committers have access to main repository and commit the code contributions from revision authors. These revision authors are the end contributors of the code. We calculate the number of developers (i.e., revision authors) for each project and examine the impact of the number of developers on the presence of test cases.

B. Research questions

We examine five research questions which pertains to the importance of software testing in software development. We collect several software metrics to investigate correlations between them, which can contribute towards improvement of software testing process and overall software development. We are thus interested in analyzing the following research questions:

RQ1: *How many projects have test cases?* Testing is a crucial activity in the life-cycle of software development process. Testing is used to detect the conditions under which a program may fail and provides directions to rectify that problem. Investigating test cases in a project is important as we wish to know whether projects are properly tested or not. Although presence of test cases does not ensure that project is bug free, but it can help developers analyse the defects and provide motivation to remove those bugs.

In this research question, we examine the prevalence of test cases in open source projects. We analyse the projects containing test cases to investigate whether test cases commensurate with the lines of code of the project.

RQ2: *Does the number of developers affect the number of test cases present in a project?* Developers are the people who are main contributors of the project. They analyse requirements, prepare documents, write code and finally test the code. Usually, developers write unit test cases to test their individual modules or functions as they have better knowledge about the product or application they are developing. They are the best people to write white box tests as they can develop multiple test cases to extensively test the application. Our dataset consist of both small and big projects where numbers of developers vary from as small as 1 to several thousands collectively working on the project.

Thus, we investigate the correlation between the number of developers working on a project and the number of test cases available for the project.

RQ3: *Does the presence of test cases correlate with the number of bugs?* A bug manifests itself as an error, failure or fault which can seriously affect the functionality of a program. The main objective of running test cases is to detect bugs in the application and find ways to fix it. Test cases can help us to find as many bugs as possible, thus, improving the efficacy of testing. Test cases can be created by analysing the bugs which can be further used to create regression test suite.

In this question, we investigate the correlation between the bug count and the number of test cases. We wish to examine whether presence of test cases has an effect on the bug count.

RQ4: *Does the presence of test cases encourage bug reporting?* Bug reports are the documents which contain details about the bugs in the program. Bug reports increases the chances of removing bugs from the software. Bug reports are also called as fault reports, problem reports, change

requests etc. When a developer or tester runs test cases and find bugs, they can log this information in a bug report. Bug reports and test results can be used to analyse the quality of software.

In this research question, we examine, indirectly, whether the presence of test cases persuades users to run these test cases and report bugs. To this end, we determine the correlation between number of test cases and number of bug reporters, i.e., people who report bugs.

RQ5: *Which programming languages appear to have more test cases?* Our dataset consists of 20,817 projects written in different languages. Some people prefer writing code using their favourite language. Although we randomly selected our projects, we still want to determine if people prefer writing test cases in some particular programming language. Some of the programming languages provide unit test framework which supports writing and running of test cases. So, we investigate whether number of test cases depends upon the popularity of programming languages.

C. Statistical measurements

To the best of our knowledge, this is the first study which explores relationship of test cases with different characteristics of the project on such a large scale. We use common metrics in statistical analysis to confirm the existence of a correlation among the data and for examining the statistical significance of our figures.

a) The Mann-Whitney-Wilcoxon (MWW) test: The MWW test is a non-parametric statistical hypothesis test to assess the statistical significance of the difference between the distributions in two datasets [22]. As this test does not assume any specific distribution, we use it for our project as we collected data from different open source projects which might not be normally distributed. Given two independent samples x and y, of size n1 and n2 respectively, the MWW test allows us to evaluate whether these distributions are identical. The test first combines and arranges the data points of the two samples in ascending order of their values. Data points with identical values are assigned a rank equal to the average position of those scores in the ordered sequence. Second, the algorithm sums the ranks of data points in the first sample (x). Let us denote this sum as T. The formula for computing the Mann-Whitney U for x is :

$$U = n_1 n_2 + \frac{n_1(n_1+1)}{2} - T$$

The U value calculated above is used to determine the p-value. Given a significance level α = 0.05, if p-value $<$ α, then the test rejects the null hypothesis. This implies that at the significance level of α = 0.05, the two datasets have different distributions.

b) Spearman's rho: Spearman's rho (ρ), also known as Spearman's rank correlation coefficient, is a non-parametric measure used to assess statistical dependence between two variables X and Y using a monotonic function. This measure can be used when data is not normally distributed. Thus, making it a good fit for the datasets that we investigate in this study. The values of ρ are limited to the interval [-1; 1]. A perfect Spearman correlation of -1 or +1 occurs when each variable is a perfect monotone function of the other. The closer to 0 ρ is, the more independent the variables are. Equation 2 states the formula for finding this coefficient.

$$\rho = \frac{\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \overline{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \overline{y})^2}}$$

In this equation, x_i and y_i represent the ranks of elements Xi and Yi in X and Y respectively, while $\overline{x}$ and $\overline{y}$ represent the averages of the ranks.

IV. Empirical Evaluation

In this section, we examine the research questions and report the results of our empirical study.

A. RQ1: Popularity of Test Cases

To answer this research question, we tabulate the number of test cases in the projects. Table I shows the distribution of test cases in the projects. After curation, our dataset includes 20,817 projects of significant size, out of which 7,982 projects do not contain test cases, which represents 38.34% of the total projects. The remaining 61.65% of the projects contain one or more test cases. In total, we have 1,875,409 test cases from 12,835 projects in our dataset. We examine how presence/absence of test cases correlate with other characteristics of the projects such as lines of code (LOC).

Table I
Test Cases Distribution

Projects	# of Projects	% of Projects
Without Test Cases	7,982	38.34%
With Test Cases	12,835	61.65%

Table II details the prevalence of test cases: 84.87% of the projects have less than 100 test cases. 10.7% of the projects have between 100 and 500 test cases, whereas less than 4.5% of the projects have more than 500 test cases. Only 17 projects have more than 10,000 test cases.

Table II
Prevalence of Test Cases

# of Test Cases	# of Projects	% of Projects with Test Cases
1-9	6,195	48.26%
10-49	3,769	29.36%
50-99	931	7.25%
100-249	964	7.51%
250-499	410	3.19%
500-999	303	2.36%
1000-4999	219	1.70%
5000-9999	27	0.21%
$>$ 10000	17	0.13%

We believe that bigger projects have higher test cases due to large number of functionalities that needs to be tested to produce a high quality software. So, we examine the correlation between the number of test cases in a project to the corresponding number of lines of code.

Figure 2. Test Cases and Lines of Code

Figure 2 [2] shows the distribution of project sizes (in terms of LOC) for projects with and without test cases. We observe that projects with test cases have an average of 107,096 LOC (median=3549) whereas average of projects without test cases is 5,605 LOC (median=1353). We compare the LOC numbers of the set of projects with test cases and that of those without test cases using Mann-Whitney-Wilcoxon (MWW) test. Our results show that the difference between these two sets is statistically significant with p-value $< 2.2\ e^{-16}$ [3]. Thus, we can conclude that projects with test cases are bigger in size than the projects without test cases.

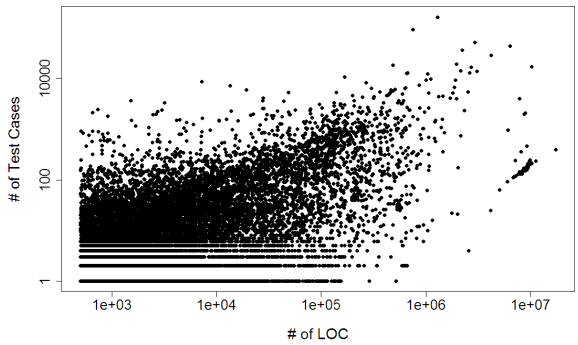

Figure 3. Correlation between Test Cases and Lines of Code

To verify that projects with test cases have higher LOC, we analyse the correlation between the number of LOC and the number of test cases. Figure 3 shows the scatter plot between the number of LOC and the number of test cases. The graph shows that there is positive correlation between these two metrics. To confirm this correlation, we use Spearman's rho which gave a value of 0.427 with p-value $< 2.2\ e^{-16}$ [4]. The result validates that there is a positive correlation between the number of test cases and the number of LOC.

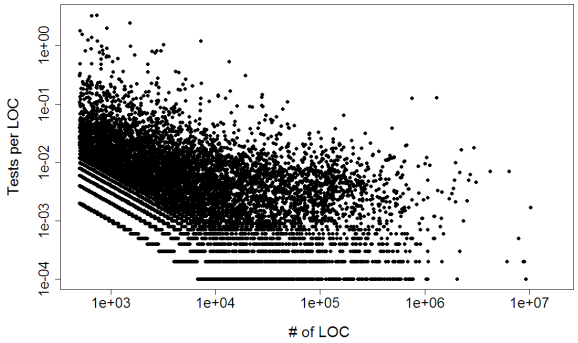

Figure 4. Correlation between Test Cases per LOC and Lines of Code

Although correlation between the number of test cases and the number of LOC is positive, we wish to examine the correlation between the number of lines of code and the number of test cases per LOC. Here, we only consider projects with test cases and divide the number of test cases by the corresponding LOC of that project. Figure 4 depicts the correlation between these two variables. We can observe that with an increase in the number of LOC, we see a decrease in the number of tests per LOC. The Spearman's rho for the distribution is -0.451 with p-value $< 2.2\ e^{-16}$, which confirms that there is a negative correlation between the lines of code and the number of test cases per LOC.

Eighty five percent of the projects have less than 100 test cases. Projects with test cases are bigger in size than projects without test cases. However, the number of test cases per LOC decreases with increasing LOC.

B. RQ2: Developers and Test Cases

Developers form an important part of the project as they contribute by writing/modifying code, developing test cases, running them and solving bugs logged in bug tracking system. So, finding a correlation correlation between the numbers of developers and the numbers of test cases is important to understand the impact of these developers on the presence of test cases. Our dataset consists of 20,817 projects which contain a total of 2,916,105 developers who have contributed to the code bases of the projects. The projects with test cases have 2,861,031 developers whereas the projects without test cases have 55,074 developers. Thus, projects with test cases have a higher numbers of developers.

[2]The line in the middle of the box represents the median. The upper part of the box represents the upper quartile, while the lower part of the box represents the lower quartile. The lines on top and below the box are referred to as whiskers. Data points above and below these whiskers are regarded as outliers – data points which are significantly different from the majority of the data points.

[3]Here, lines of code is the dependent variable and the presence/absence of test cases is the independent variable. The null hypothesis is: there is no difference in the size of projects with test cases and those without test cases. The alternative hypothesis is: projects with test cases have more LOC than those without test cases. We consider a significance level α=0.05. For this α value, if the p-value < 0.05, we reject the null hypothesis.

[4]Null hypothesis (rho is zero) is rejected

We can observe from Figure 5 that projects with test cases have more developers. We used MWW test between the set of numbers of developers of projects with test cases and those for projects without test cases which gave p-value $< 2.2e^{-16}$ [5]. The results signify that the difference between these two sets is statistically significant.

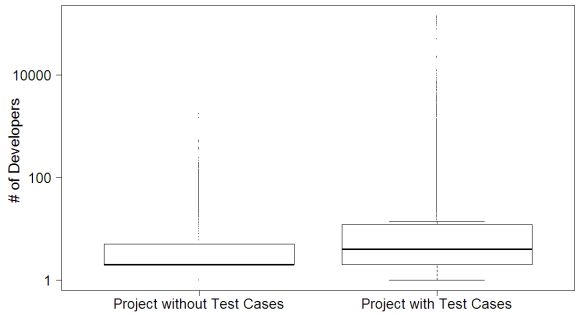

Figure 5. Number of Developers in Projects with/without Test Cases

We wish to examine whether increase in the number of developers leads to an increase in the number of test cases in that project. We use scatter plot (Figure 6) to examine the correlation between the numbers of developers and the numbers of test cases. We calculated Spearman's rho to confirm the correlation between these two variables which gave a value of 0.207 (p-value $< 2.2\ e^{-16}$). This suggests that there is a weak positive correlation between the number of developers and test cases.

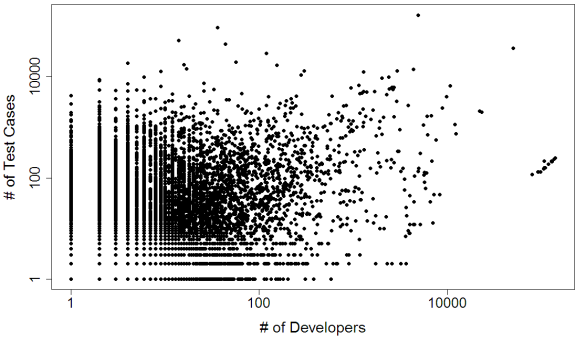

Figure 6. Test Cases and Number of Developers

We further investigate the average number of test cases contributed by each developer. For each project, we divide the total number of test cases by the corresponding number of developers in that project. Figure 7 depicts the correlation between the numbers of developers and the numbers of test cases per developer. We use Spearman's rho to find the correlation between these two variables. The Spearman's value is -0.444 with p-value $< 2.2\ e^{-16}$. Thus, the correlation between the number of developers and the number of test cases per developer is negative. As only some of the developers write test cases, we observe a decrease in the test count per developer with an increase in the number of developers.

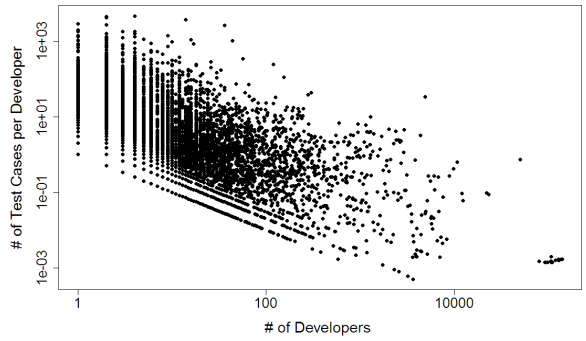

Figure 7. Correlation between # of Test Cases per Developer and # of Developers

The number of test cases increases when there are more developers in a project. However, the number of test cases per developer decreases for the projects with more developers.

C. RQ3: Test Cases and Bug Counts

In this research question we examine whether the number of bugs is correlated with the number of test cases present within a project. First, we identify the issue reports present in our dataset. GitHub provides an issue tracking system which lets users file issue tickets, tag them according to the issue and label them as the state of the issue changes. It also allows the project development team to either enable or disable the issue tracking system. Users can tag issues and categorize them. However, user-supplied tags can create a problem for developers as there can be typographical errors while tagging. Since tags are not predetermined by GitHub, a tag can be reported in different forms. For example, a bug can be tagged as defect, type:bug, bugfix, etc. Table III depicts several representations of tags which we count as bugs for our project.

Table III
TAGS REPRESENTING BUGS

bug	bug; T bug; Bug Confirmed; bugs; starter bug; bug fix etc.
defect	defect; Type-Defect; minor defect
error	error; Wow error; build error; error page; user error etc.

Since errors can be represented by any combination of these tags, we use these tags to account for all the bugs. In total, we have 1,081 projects which contain 24,703 bugs as represented by the tags mentioned above. These projects

[5]Here, number of developers is the dependent variable and the presence/absence of test cases is the independent variable. The null hypothesis is: there is no difference in the number of developers of projects with test cases and those without test cases. The alternative hypothesis is: projects with test cases have more developers than those without test cases. We consider a significance level α=0.05. For this α value, if the p-value $<$ 0.05, we reject the null hypothesis.

contain 83,576 test cases written by the project development teams.

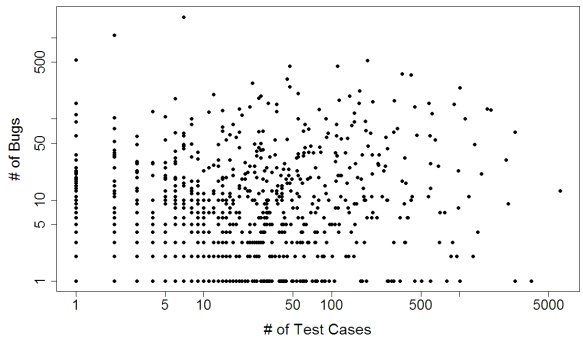

Figure 8. Correlation between # of Test Cases and # of Bugs

Our aim is to study and see that with increase in the number of test cases, bug count increases. Figure 8 shows a scatter plot to explore the correlation between the number of bugs and the number of test cases. Here, we can see that as the number of test cases increases, we see an increase in the number of bugs. We calculated the Spearman's correlation which yields rho value 0.181 (p-value $= 1.78\ e^{-09}$), suggesting a weak correlation between the number of test cases and the number of bugs.

Projects having higher numbers of test cases observe an increase in the number of bugs, although the correlation is weak between them.

D. RQ4: Test Cases and Bug Reporters

We wish to know if the presence/absence of test cases affects bug reporting. We examine the relationship between the number of test cases and the number of bug reporters. Bug reporters are the people who report or log bugs related to a particular application or software. Based on the user names, we collected the data about people who have reported issues in the project. As not all the projects contain issues, we identified 6,230 projects in which users logged issues. These issues were filed by 274,276 reporters.

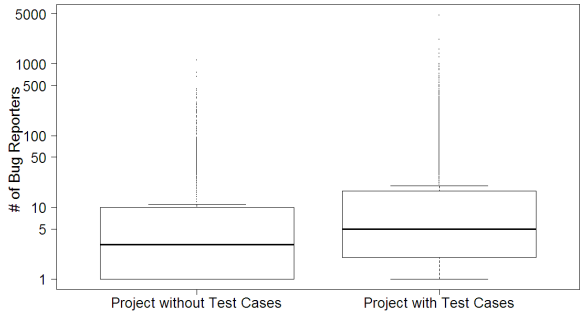

Figure 9. Test Cases and Bug Reporters

We can observe from the Figure 9 that projects with test cases have higher number of bug reporters (median=5) as compared to projects without test cases (median=3). We performed the MWW test and found that the difference between the set of bug reporters in projects without test cases and those of projects with test cases is statistically significant (p-value $< 2.2\ e^{-16}$). We can infer that if test cases are present, it can persuade users to run these test cases and if they found bugs, they can log them in issue tracking systems.

Figure 10 shows the scatter plot of the numbers of bug reporters and the numbers of test cases. We computed Spearman's rho for the distribution which yielded the value 0.171 (p-value $< 2.2\ e^{-16}$), suggesting a weak dependence between the number of test cases and the number of bug reporters.

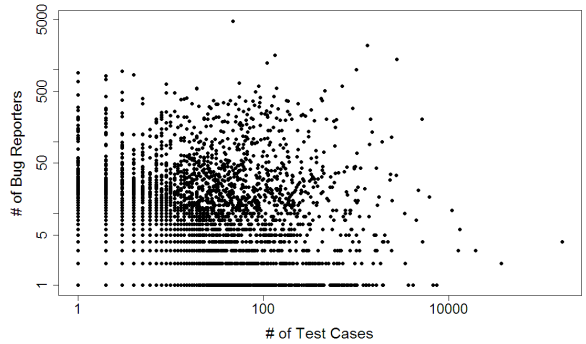

Figure 10. Correlation between # of Bug Reporters and # of Test Cases

There is weak correlation between the number of test cases and the number of bug reporters.

E. RQ5: Test Cases and Programming Languages

With this research question, we attempt to establish whether projects written in common languages such as C#, Java, PHP or JavaScript, contain more number of test cases than other languages. We first compute the number of test cases present in projects depending on the programming language that is used. We then select projects developed in the top ten languages with the highest number of test cases.

Figure 11 shows the number of projects of the corresponding top ten languages in our dataset. Out of 20,817 projects in our dataset, 19,327 projects use one of these top ten languages. During the analysis, we find out that Java has 3,112 number of projects and also the highest count among all the projects. Our dataset contains 3,016, 2,902 and 2,536 projects written in ruby, PHP and Python respectively. Perl has the lowest number of projects among the projects written in the computed top ten languages. C++ has the highest number of test cases being 648,773 present in 1,920 projects. Then, we have projects written in ANSI C, PHP and Java having respective count of 286,009, 255,553 and 196,703 test cases. Perl has lowest number of test cases, i.e., 7,690 present in 630 projects.

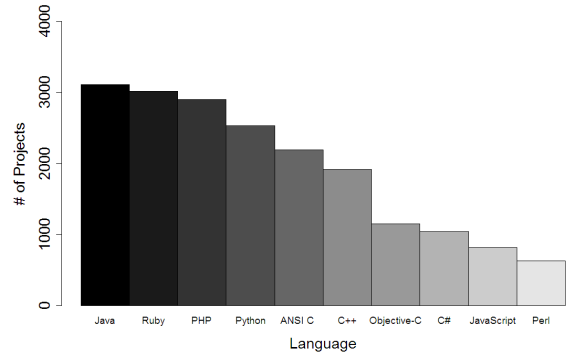

Figure 11. Count of Projects and Different Languages

Figure 12 shows the distribution of the number of tests of top-10 languages that are used in the projects of our dataset. We observe that median values of some of the pairs such as C# and Ruby, Python and Java, ANSI C and PHP, Objective-C and Perl are almost comparable to each other. JavaScript has a median value of 4 test cases, 1 less than the median value of C# and Ruby.

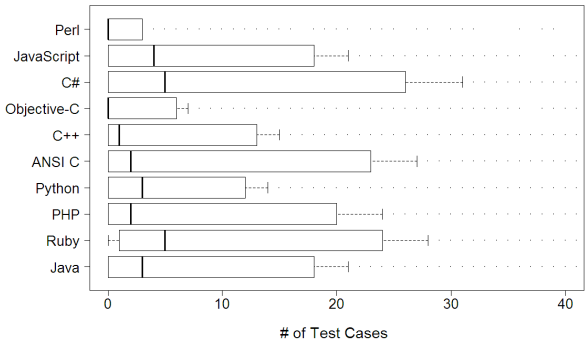

Figure 12. Prevalence of Test Cases for Common Languages

As most of the projects have lower number of test cases, we can observe that median line is gravitating towards the left, i.e, data is skewed towards the right. The rest of the projects having higher number of test cases are considered as outliers as they are small in number and does not have a significant impact on the box plots [6]. Thus, we can observe a big difference in the mean and median values for all the languages.

We further analyze the number of test cases per project. Table IV depicts the mean number of test cases per project for each language. We observe that C++ has the highest value, i.e., 337.90, whereas Perl has the lowest value among all the top ten languages. JavaScript projects has higher number of test cases per project than Python and Objective-C projects. Although the numbers of projects written in C++, ANSI C and PHP are less as compared to the numbers of projects written in Java and Ruby, they have higher mean numbers of test cases per project.

[6]https://github.com/isis-project/WebKit having 166488 test cases and https://github.com/chrispilot2293/CM9 having 44871 test cases

Table IV
DISTRIBUTION OF TEST CASES PER PROJECT

Language	# of Projects	# of Test Cases	Test Cases/Project
C++	1,920	648,773	337.90
ANSI C	2,197	286,009	130.18
PHP	2,902	255,553	88.06
C#	1,042	81,334	78.05
Java	3,112	196,703	63.20
Ruby	3,016	173,864	57.64
JavaScript	819	39,070	47.70
Python	2,536	103,600	40.85
Objective-C	1,153	21,343	18.51
Perl	630	7,690	12.20

Projects written in popular languages, such as C++, ANSI C, and PHP, have higher mean numbers of test cases per project.

V. THREATS TO VALIDITY

We now describe some threats to validity that we have identified in the course of this study.

External validity is related to the generalizability of our results. Although our dataset consists of over twenty thousand projects, the results may not represent all real world projects. Also, our study is conducted on GitHub, which is one of the biggest repository for open source projects. So, the results may differ for closed source projects. To the best of our knowledge, GitHub hosts projects from myriad of areas and we selected the projects randomly.

Threats to internal validity refers to whether an experimental condition makes a difference or not. Data quality is one of biggest threat here. We have tried to ensure quality of our dataset by examining that we take the proper count of number of test cases for all projects. We use heuristics to detect test files, i.e., we consider the files whose name contains test. This might not identify all test files whose name does not have word test and conversely, detect some files whose name contain the word test but actually are not test files. In order to scale to a large number of projects, we need to take these heuristics. We have manually checked and counted the test cases for some of the projects to validate our results. For counting bugs we had to take into account all of the labels that can be marked as bugs such as defects, error, bugfix etc. Since GitHub does not provide set of labels to be marked as bugs or defects, users are free to mark labels in any fashion suitable to them. Different labels were identified through a painstaking review of projects and characterizing labels into different categories. Still there is possibility that we may have wrongly identified some of the labels as bugs which may not be actual bugs. For multi-language projects, we consider only the dominant language, i.e., the language with the highest number of lines of code in a project.

VI. Related Work

In the following section, we highlight studies on open source software projects, empirical studies on testing, and other large scale studies.

A. Studies on Open Source Software Projects

Open source software projects has received enormous attention from industry as well as the research community in the last several years. Pham et al. discuss several strategies to understand the testing culture on social coding sites such as GitHub. They also present some guidelines which can be used by developers and managers to influence the testing behavior in their projects [26]. Dabbish et al. examine open social software repository namely GitHub, to understand the value of transparency for large-scale distributed collaborations and communities of practice [10]. Crowston et al. analyse the social structure of Free/Libre and Open Source Software (FLOSS) by examining the communication patterns used in bug tracking systems [7]. Their results shows that FLOSS projects cannot be characterised into particular pattern of communications centralization or decentralization.

Roberts et al. develop a model to understand motivations, participation, and performance of open source software developers [28]. They validate their model using data collected from Apache projects. Bird examine the relationship between developers in large open source projects [5]. They use source code repository histories, communication and coordination data from mailing lists, and bug tracking databases to understand the relationship between participants social and development behavior and the social structure that exists between them. Crowston et al. use coordination theory to analyse free/libre open source software (FLOSS) development projects [9]. They also compare these results with the existing literature on coordination in proprietary software development and found several similarities and differences in the coordination mechanisms used in the project. Sowe et al. investigate knowledge sharing activities between the knowledge providers and knowledge seekers using the Developer and User mailing lists of Free/Open Source Software (F/OSS) project, namely Debian [29]. Crowston et al. analyze the structure and coordination practices used by development teams during bug fixing practices in free/libre open source software (FLOSS) [8].

Raja et al. use data and text mining to build and validate a model to analyse the effect of project type, end user activity, process quality, team size and project popularity on the defect density of operational open source software (OSS) projects [27]. Michlmayr et al. perform exploratory interviews on open source developers to gain an insight into quality practices and quality problems particular to free software projects [24]. Krogh et al. develop an inductive theory of the open source software (OSS) innovation process to analyse why new people join existing developer community [35]. Zanetti et al. use data driven approach to get insight into sustainability of software development by analysing large number of open source software projects [37].

We perform large scale study on more than twenty thousand open source projects hosted on GitHub.

B. Empirical Study on Testing

There have been a lot of studies that assess various aspect of testing. We just highlight some of them here. Greiler et al. conduct a qualitative study of test practices followed by a community of people working on plug-in based applications [14]. Rehman et al. discuss several software component testing issues and classify set of testing techniques used when a component is integrated with its target system [34]. Memon et al. present their analysis to improve the current testing techniques and strategies to create new collaborative development and testing processes where developers can share tools and information repositories [23].

Zaidman et al. study the co-evolution between production code and test code on two open source and one industrial project [36]. Fraser et al. use search-based software testing for test data generation for open source projects [13]. They perform case study on 100 Java projects selected from SourceForge and give directions for future research. Cecato et al. perform an empirical study to analyse the impact of automatically generated test cases on accuracy and efficiency of debugging.

Stamelos et al. conduct an empirical study on open source projects to understand the implications of structural quality and the probable benefits of such analysis on software development [30]. They perform the experiment to measure the quality characteristics of 100 applications written for Linux. Kamei et al. perform an empirical study to identify the real-time software changes that can have a high probability of introducing a defect [17]. They evaluate a ”Just-InTime (JIT) Quality Assurance” approach on 6 open source and 5 commercial projects.

In this work, we consider a separate research problem namely on the adoption rate of testing in practice. We investigate a large number of projects from GitHub.

C. Large Scale Studies

Aside from our work, there have been many past studies that also perform large scale studies on hundreds and even thousands of projects. Gruska et al. evaluates a lightweight anomaly detection technique on a collection of 6,000 projects [15]. They extract formal rules in the form of computational tree logic expressions from code and detect for violations of these expressions. Surian et al. analyze a snapshot of projects in SourceForge.Net [32]. They find several patterns of how developers collaborate with one another in SourceForge.Net. Surian et al. also analyze projects in SourceForge.Net to build a system to effectively recommend developers to one another [31]. They perform random walk with restart over a graph containing links

between developers and projects to realize their proposed solution.

VII. Conclusion and Future Work

Software testing is used to ensure that the software produced is complete, correct, secure and of higher quality. Test cases are used to confirm that software meets all these criteria. We conduct a large scale empirical study to analyse the prevalence of test cases in open source projects and the correlations between the test cases and other important software metrics. We plot graphs to depict these correlations and use statistical analysis techniques to confirm the relationships depicted by the graphs.

Our analysis shows the following results:

1) Projects with test cases have more LOC than those without test cases. As projects grow in size the number of test cases per LOC decreases.
2) Projects with more number of developers have more test cases. However as the number of developers grow, the number of test cases per developer decreases.
3) There is weak positive relationship between number of test cases and the number of bugs.
4) Number of test cases has a weak correlation with the number of bug reporters.
5) Projects written in popular languages, such as C++, ANSI C, and PHP, have higher mean numbers of test cases per project as compared to projects in other languages.

In this work, we only consider around 20,000 projects. We plan to increase the number of projects in a future work. We also plan to investigate more project characteristics and analyse their correlation with the number of test cases. Further, we intend to perform qualitative analysis of test cases to understand how well test suites cover the code.

References

[1] J. Anvik, L. Hiew, and G. C. Murphy, "Who should fix this bug?" in *ICSE*, 2006.

[2] V. Basili and R. Selby, "Comparing the effectiveness of software testing strategies," *Software Engineering, IEEE Transactions on*, 1987.

[3] B. Beizer, *Software Testing Techniques*, 2nd ed. Boston: International Thomson Computer Press, 1990.

[4] D. Binkley, "Using semantic differencing to reduce the cost of regression testing," in *ICSM*, 1992.

[5] C. Bird, "Sociotechnical coordination and collaboration in open source software," in *ICSM*, 2011.

[6] L. Briand and Y. Labiche, "Empirical studies of software testing techniques: Challenges, practical strategies, and future research," *ACM SIGSOFT Software Engineering Notes*, 2004.

[7] K. Crowston and J. Howison, "The social structure of free and open source software development," *First Monday*, 2005.

[8] K. Crowston and B. Scozzi, "Bug fixing practices within free/libre open source software development teams," *Journal of Database Management*, 2008.

[9] K. Crowston, K. Wei, Q. Li, U. Y. Eseryel, and J. Howison, "Coordination of free/libre open source software development," in *ICIS*, 2005.

[10] L. Dabbish, C. Stuart, J. Tsay, and J. Herbsleb, "Social coding in github: Transparency and collaboration in an open software repository," in *CSCW*, 2012.

[11] P. Frankl and O. Iakounenko, "Further empirical studies of test effectiveness," in *ACM SIGSOFT Software Engineering Notes*, 1998.

[12] P. Frankl and S. Weiss, "An experimental comparison of the effectiveness of branch testing and data flow testing," *IEEE Transactions on Software Engineering*, 1993.

[13] G. Fraser and A. Arcuri, "Sound empirical evidence in software testing," in *ICSE*, 2012.

[14] M. Greiler, A. van Deursen, and M.-A. D. Storey, "Test confessions: A study of testing practices for plug-in systems," in *ICSE*, 2012.

[15] N. Gruska, A. Wasylkowski, and A. Zeller, "Learning from 6, 000 projects: lightweight cross-project anomaly detection," in *ISSTA*, 2010.

[16] D. Jeffrey and N. Gupta, "Test case prioritization using relevant slices," in *COMPSAC*, 2006.

[17] Y. Kamei, E. Shihab, B. Adams, A. E. Hassan, A. Mockus, A. Sinha, and N. Ubayashi, "A large-scale empirical study of just-in-time quality assurance," *IEEE Transactions on Software Engineering*, 2012.

[18] C. Kaner, "Exploratory testing," in *Quality Assurance Institute Worldwide Annual Software Testing Conference*, 2006.

[19] P. S. Kochhar, T. F. Bissyande, D. Lo, and L. Jiang, "Adoption of software testing in open source projects - a preliminary study on 50,000 projects," in *CSMR*, 2013.

[20] B. Korel and J. Laski, "Algorithmic software fault localization," in *HICSS*, 1991.

[21] D. Leon and A. Podgurski, "A comparison of coverage-based and distribution-based techniques for filtering and prioritizing test cases," in *ISSRE*, 2003.

[22] H. B. Mann and D. R. Whitney, "On a test of whether one of two random variables is stochastically larger than the other," *The Annals of Mathematical Statistics*, 1947.

[23] A. Memon, A. Porter, and A. Sussman, "Community-based, collaborative testing and analysis," in *FoSER*, 2010.

[24] M. Michlmayr, F. Hunt, and D. Probert, "Quality practices and problems in free software projects," in *OSS*, 2005.

[25] J. Offutt, J. Pan, and J. Voas, "Procedures for reducing the size of coverage-based test sets," in *ICST*, 1995.

[26] R. Pham, L. Singer, O. Liskin, F. F. Filho, and K. Schneider, "Creating a shared understanding of testing culture on a social coding site," in *ICSE(to appear)*, 2013.

[27] U. Raja and M. J. Tretter, "Antecedents of open source software defects: A data mining approach to model formulation, validation and testing," *Information Technology and Management*, 2009.

[28] J. A. Roberts, I.-H. Hann, and S. Slaughter, "Understanding the motivations, participation, and performance of open source software developers: A longitudinal study of the apache projects," *Management Science*, 2006.

[29] S. K. Sowe, I. Stamelos, and L. Angelis, "Understanding knowledge sharing activities in free/open source software projects," *Journal of Systems and Software*, 2007.

[30] I. Stamelos, L. Angelis, A. Oikonomou, and G. L. Bleris, "Code quality analysis in open source software development," *Information Systems Journal*, 2002.

[31] D. Surian, N. Liu, D. Lo, H. Tong, E.-P. Lim, and C. Faloutsos, "Recommending people in developers' collaboration network," in *WCRE*, 2011.

[32] D. Surian, D. Lo, and E.-P. Lim, "Mining collaboration patterns from a large developer network," in *WCRE*, 2010.

[33] G. Tassey, "The economic impacts of inadequate infrastructure for software testing," *National Institute of Standards and Technology, RTI Project*, 2002.

[34] M. J. ur Rehman, F. Jabeen, A. Bertolino, and A. Polini, "Testing software components for integration: a survey of issues and techniques," *Software Testing, Verification & Reliability*, 2007.

[35] G. von Krogh, S. Spaeth, and K. R. Lakhani, "Community, joining, and specialization in open source software innovation: A case study," *Research Policy*, 2003.

[36] A. Zaidman, B. V. Rompaey, A. van Deursen, and S. Demeyer, "Studying the co-evolution of production and test code in open source and industrial developer test processes through repository mining," *Empirical Software Engineering*, 2011.

[37] M. S. Zanetti, "The co-evolution of socio-technical structures in sustainable software development: Lessons from the open source software communities," in *ICSE*, 2012.