Mining Peer Code Review System for Computing Effort and Contribution Metrics for Patch Reviewers

Rahul Mishra, Ashish Sureka
Indraprastha Institute of Information Technology, Delhi (IIITD)
New Delhi
{rahulm, ashish}@iiitd.ac.in

Abstract—Peer code review is a software quality assurance activity followed in several open-source and closed-source software projects. Rietveld and Gerrit are the most popular peer code review systems used by open-source software projects. Despite the popularity and usefulness of these systems, they do not record or maintain the cost and effort information for a submitted patch review activity. Currently there are no formal or standard metrics available to measure effort and contribution of a patch review activity. We hypothesize that the size and complexity of modified files and patches are significant indicators of effort and contribution of patch reviewers in a patch review process. We propose a metric for computing the effort and contribution of a patch reviewer based on modified file size, patch size and program complexity variables. We conduct a survey of developers involved in peer code review activity to test our hypothesis of causal relationship between proposed indicators and effort. We employ the proposed model and conduct an empirical analysis using the proposed metrics on open-source Google Android project.


I. RESEARCH MOTIVATION AND AIM

Measuring the scale and complexity of a Software Engineering (SE) activity is required for computing the contribution and performance of developers and also for effort and cost measurement models [1][3][5][8][9]. However several widely used software engineering tools, supporting various developer activities, do not explicitly save and maintain the task size and complexity data. Issue Tracking Systems (ITS) such as Bugzilla and Peer Code Review (PCR) systems such as Rietveld1 and Gerrit2 are applications used for coordinating and managing bug resolution and peer code review activities respectively. However, such systems do not record or maintain the size and complexity of the reported issues. For example, patches submitted for code inspection to a peer code review system can vary widely in terms of size (number of files and code churn) and complexity (program complexity). Several variables such as developer expertise, time taken to accomplish a task and size and complexity of a task is required to develop effort estimation model. Similarly, size and complexity of a task is required to objectively and accurately measure the contribution and performance of developers.

The research motivation and aim of the work presented in this paper is to investigate metrics for measuring a code reviewer’s effort and contribution based on the size and complexity of the modified files and the submitted patch. Our aim is to conduct a survey of practitioners involved in peer code review activity to gather information and test our hypothesized causal relationship. Peer code review systems for open source software projects are software repositories archiving code inspection data. The objective of our study is to conduct a case-study using publicly available dataset from open-source Google Android project wherein we compute effort and contribution values for submitted issues (on a sample dataset) using the proposed metrics and present our insights. We adopt the quantitative research method to test our hypothesis and answer specific research questions by conducting a survey of experienced practitioners. We propose a metric after testing our hypothesis and then employ the proposed model on a real-world dataset for computing effort and contribution of patch reviewers.

II. RELATED WORK AND RESEARCH CONTRIBUTIONS

Effort and contribution measurement of bug fixers and testers in software quality assurance is an area that has attracted several researcher’s attention. Kanij et al. mention that currently no standard methods for assessing the performance of software testers (or standard assessment criteria) are present. They conduct a survey of industry professionals to list important factors for assessing the performance of software testers [6]. A recent study presents insights from industrial practice in the area of tester performance appraisal by surveying nearly 20 software development project managers and proposes a Performance Appraisal Form (PAF) for software testers [5]. Kaner et al. presents a multidimensional, multi-sourced and qualitative approach to evaluate individual testers and argue that just bug counts should not be used to measure productivity, efficiency and skills of software testers [2][4].

Rastogi et al. present several metrics to measure contribution and performance of bug fixers, bug reporters and triagers by mining activity data from issue tracking system [9]. Nagwani et al. present a team member ranking approach by mining attributes (such as severity and priority of bugs, number of bugs fixed and comments made by developers) available in software bug repositories [8]. Gousios propose a model by combining traditional contribution metrics with data mined from software repositories for the purpose of accurately measuring developer contributions [3]. Ahsan et al. present an approach of mining effort data from the history of the developer’s bug fix activities and present an approach to build an effort estimation model for Open Source Software [1]. In context to existing literature, the work presented in this paper makes the following novel contributions:

1http://code.google.com/p/rietveld/
2http://code.google.com/p/gerrit/
TABLE I: Survey Results of 44 Google Android Project Members Engaged in Peer Code Review Activity

<table>
<thead>
<tr>
<th>Q1: How many years of experience do you have with peer code review systems, code inspection, patch submission and review?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Less than 1 year</td>
</tr>
<tr>
<td>Between 1 and 3 years</td>
</tr>
<tr>
<td>Between 3 and 5 years</td>
</tr>
<tr>
<td>More than 5 years</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Q2: Patch review and code inspection requires program comprehension and understanding. Please evaluate the following size and complexity measures in terms of their suitability in computing the effort required to review a patch and contribution made by a patch reviewer.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of issues reviewed</td>
</tr>
<tr>
<td>Code churn which is the sum of the number of lines added and deleted in a patch</td>
</tr>
<tr>
<td>Number of classes in the modified source code files</td>
</tr>
<tr>
<td>Number of functions in the modified source code files</td>
</tr>
<tr>
<td>Number of non-commenting source statements in the modified source code files</td>
</tr>
<tr>
<td>Average non-commenting source statement per function in the modified source code files</td>
</tr>
<tr>
<td>Average cyclomatic complexity per function in the modified source code files</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Q3: More complex patches take more time to get accepted in general.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Completely agree</td>
</tr>
<tr>
<td>Somewhat agree</td>
</tr>
<tr>
<td>Neither disagree nor agree</td>
</tr>
<tr>
<td>Somewhat disagree</td>
</tr>
<tr>
<td>Completely disagree</td>
</tr>
</tbody>
</table>

1) A metric for computing the effort and contribution of a patch reviewer based on modified files and patch size and program complexity variables. While there has been work done in the area of effort and contribution metrics for bug fixers and software testers, to the best of our knowledge this paper is the first study focusing on mining peer code review systems for measuring effort and contribution of patch reviewers.

2) A survey of practitioners and developers of open-source software projects on the topic of effort and contribution metrics for patch reviewers.

3) A case-study and empirical analysis consisting of employing the proposed model on open-source Google Android project and sharing our insights and perspectives.

III. SURVEY OF PATCH REVIEWERS

The research work presented in this paper is motivated by the need to investigate and to come up with solutions to automate the contribution and effort assessment of Software Maintenance Professionals. We believe that inputs from practitioners are needed to inform our research. Our purpose of conducting a survey is both information gathering and theory testing and building. We conducted a survey (short questionnaire based online survey) of Google Android project members who are engaged in patch submission and review activities. We got 44 responses from experienced Android developers, actively involved in peer code review process of Android project.

Table I shows the results of our survey (questions consists of collecting information about respondents and respondent's opinions). We received responses from developers with less than 1 year of experience to more than 5 years of experience. 43.18% of developers have more than 5 years of experience. Table I reveals responses of developers on size and complexity measures in terms of their suitability in computing the effort required to review a patch and contribution made by a patch reviewer (a narrow and specific close-ended question). Responses to each item in Question 2 is on a ordinal scale of 1 (low importance) to 5 (highly important). Table I reveals average value of 44 responses and indicates that the size of the code churn and average cyclomatic complexity per function in the modified source code files are the two most important factors in measuring effort. We conclude based on our hypothesis and theory testing survey that all three factors of size of code churn, size of modified source code files and program complexity of the code are important attributes in measuring the effort and contribution of a patch reviewer.

IV. EFFORT AND CONTRIBUTION METRICS

We define following six variables to measure the size and complexity of a patch review task:

1) CHN: Code churn which is the sum of the number of lines added and deleted in a patch
2) NCL: Number of classes in the modified source code files
3) NFL: Number of functions in the modified source code files
4) NCS: Number of non-commenting source statements in the modified source code files
5) ANF: Average non-commenting source statement per function in the modified source code files
6) ACF: Average cyclomatic complexity per function in the modified source code files

CHN is an attribute of a patch. NCL, NFL, NCS, ANF and ACF are attributes of the modified source code files and not the patch. We propose variables CHN, NCL, NFL, NCS, ANF and ACF as these variables are concrete concepts of interest which can be accurately, reliably and consistency measured. Factors like reviewer’s prior work experience, skills, familiarity with the code and cognitive abilities also influence the cost and effort. However these are abstract concepts which cannot be accurately and consistently measured and hence is a limitation and threat to validity of our proposed metrics. In this study, we start with an a priori theory and an alternative hypothesis (causal relationship between CHN, NCL, NFL, NCS, ANF and ACF and effort or contribution) and test our theory using a survey-based qualitative research method. We hypothesize that both the size of the difference or change as well as...
the size of the modified source code is important to measure effort required for code inspection as understanding both the modified lines of code as well as it’s context is required. The survey results supports our actual desired conclusion and the alternative hypothesis (we take an inductive approach from theory, hypothesis, data collection through survey and confirmation).

A low CHN and a high NCS means that the number of non-commenting source statements in the modified source file is high but the change is small. ACF is an attribute of program complexity and not size. Two functions can have the same size (same number of non-commenting source statements) but different program complexity. A higher value of ACF means more effort in terms of program readability and comprehension. We propose the following two alternate metrics to compute the effort required to review a patch.

\[ \text{EFF} = \left( \frac{\text{ACF}}{\text{ANF}} \right)^{\alpha} \left( W_{\text{CHN}} \times \text{CHN} + W_{\text{NCS}} \times \text{NCS} \right) \] (1)

\[ \text{EFF} = \left( \frac{\text{ACF}}{\text{ANF}} \right)^{\alpha} \left( W_{\text{CHN}} \times \text{CHN} + W_{\text{NFL}} \times \text{NFL} \right) \] (2)

\( \text{EFF} \) measures the effort required to comprehend and review a patch. Before computing the effort, the variables in Equations 1 and 2 are filtered by removing outliers and then normalized in-order to make the variables (belonging to different scale) comparable. As shown in Equations 1 and 2, \( \text{EFF} \) is a function of multiple factors and one of the factors is program complexity per line of code. Program complexity per line of code is estimated by taking the ratio of \( \text{ACF} \) and \( \text{ANF} \). The ratio of \( \text{ACF} \) and \( \text{ANF} \) is raised to the power \( \alpha \) which can be set to increase or decrease the impact of program complexity per line on the effort metric. In Expressions 1 and 2, program complexity per line of code is the base and \( \alpha \) is the exponent which influences the rate of growth and decay of \( \text{EFF} \). Exponent \( \alpha \) is a positive whole number and greater than or equal to 1 (\( \alpha \geq 1 \)). The program complexity per line of code is always greater than 0. Hence, \( \left( \frac{\text{ACF}}{\text{ANF}} \right)^{\alpha} \) is a function which represent an exponential growth. We use the nonlinear exponential function to model the relationship between the problem complexity per line of code and effort so that a constant change in \( \left( \frac{\text{ACF}}{\text{ANF}} \right) \) gives the same proportional change in the \( \text{EFF} \) variable.

As shown in Equations 1 and 2, the effort metric is a function of the size of the code churn (CHN) multiplied by the program complexity per line and the size of the modified source code files multiplied by the program complexity per line. In Equation 1, we define two weights \( W_{\text{CHN}} \) and \( W_{\text{NCS}} \) to tune the impact of CHN and NCS respectively on the computed effort. In Equation 1, we use NCS as a measure for the size of the modified source code whereas in Equation 2 we use NFL as a measure for the size of the source code. NFL and NCS are correlated and hence we propose two alternate metrics (Equation 1 and 2) to compute the review effort for a submitted patch. We define coefficients \( W_{\text{CHN}} \) and \( W_{\text{NCS}} \) are real numbers greater than 0. NCL is defined but not used in the equations as it serves a similar purpose as NCS and NFL.

V. Empirical Analysis on Google Android Dataset

We perform a case-study on Google Android project peer code review dataset. Google Android is an open-source project which uses Gerrit peer code review system\(^3\). We download 1934 issues from issue id 20107 to 38729 (from January 2011 to July 2012) containing Java source files. We extract java source files and code churn attributes using REST API\(^4\) and Android peer code review dataset shared (publicly available) by Mukadam et al [7]. We use JavaNCSS\(^5\) which is a source measurement suite for Java for computing the non-commenting source statements and the cyclomatic complexity number (McCabe metric). Data collection is one of the most important stage in conducting qualitative research and the quality of result obtained depends both on research design and data gathered. Hence, we chose Google Android for conducting our case-study as it is a large, long-lived, well-known project and its peer code review dataset is publicly available. Also, Google Android has been widely used in the past for conducting Software Engineering research by several researchers. Google Android online documentation provides a full process\(^6\) of submitting a patch to the AOSP (Android Open-Source Software Project), including reviewing and tracking changes with Gerrit.

As an academic, we believe and encourage academic code or software sharing in the interest of improving openness and research reproducibility. We release our code and dataset in public domain so that other researchers can validate our scientific claims and use our tool for comparison or benchmarking purposes (and also reusability and extension). Our code and dataset is hosted\(^7\) on GitHub which is a popular web-based hosting service for software development projects. We select GPL license (restrictive license) so that our code can never be closed-sourced.

We compute descriptive statistics consisting of five-number summary (minimum, first quartile, median, third quartile, maximum) for six variables: \( \text{ANF} \), \( \text{ACF} \), \( \text{NCS} \), \( \text{NFL} \), \( \text{NCL} \) and CHN. Table II shows the descriptive statistics for six variables in the dataset consisting of 1934 issues. Table II reveals wide variability and dispersion in the dataset. Results in Table II shows that the average cyclomatic complexity per function varies from a minimum of 1 to a maximum of 51 having a median value of 3.355. The first quartile and third quartile for code churn variable is 7 and 174 respectively which indicates that 25\% of issues have a code churn of less than 7 lines whereas 75\% of issues have a code churn of more than 174 lines. We notice that the number of non-commenting source code statements in the modified source code files also has a wide dispersion. As shown in Table II, 25\% of issues have \( \text{NCS} \) value of less than 185 whereas 25\% of issues have more than 1184 lines in the modified source code files. The descriptive statistics for the size and complexity variables presented in Table II supports the argument that two issues can have substantial difference between them in terms of size and complexity. The data in Table II indicates that the effort

\(^3\)https://android-review.googlesource.com
\(^4\)https://gerrit-review.googlesource.com/Documentation/rest-api.html
\(^5\)http://www.kclee.de/clemens/javacovanns/
\(^6\)https://source.android.com/source/submit-patches.html
\(^7\)https://github.com/ashishsureka/PCR
TABLE II: Descriptive Statistics for Size and Complexity variables

<table>
<thead>
<tr>
<th></th>
<th>ANF</th>
<th>ACF</th>
<th>NCS</th>
<th>NFL</th>
<th>NCL</th>
<th>CHN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Min</td>
<td>1</td>
<td>1</td>
<td>6</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Q1</td>
<td>7.085</td>
<td>2.18</td>
<td>185</td>
<td>16</td>
<td>2</td>
<td>7</td>
</tr>
<tr>
<td>Q2</td>
<td>9.065</td>
<td>3.355</td>
<td>510</td>
<td>49</td>
<td>6</td>
<td>31</td>
</tr>
<tr>
<td>Q3</td>
<td>12.185</td>
<td>4.4</td>
<td>1184</td>
<td>111</td>
<td>18</td>
<td>174</td>
</tr>
<tr>
<td>Max</td>
<td>156</td>
<td>51</td>
<td>26848</td>
<td>2240</td>
<td>349</td>
<td>44361</td>
</tr>
</tbody>
</table>

required to comprehend a patch and the contribution made by reviewers by closing an issue can vary significantly due to wide variability in the size and complexity of a patch.

We compute the effort incurred on each of the 1934 issues using Equation 1. We set the value of $\alpha$ as 1 and the values of $W_{CHN}$ and $W_{NCS}$ as 0.5. Figure 1 shows the distribution of the effort. In Figure 1, we apply the Box-Cox power transformation to the effort variable to deal with skewness and make it easy to interpret and visualize the distribution. The $\lambda$ value for Box-Cox power transformation is 0.2 and is applied to transform the non-normal data to a distribution that is approximately normal. Figure 1 reveals variations in effort across several reviews (the y-axis represents the number of reviews and x-axis represents effort value using Box-Cox power transformation). Figure 1 reveals that the dependent variable $EFF$ (a continuous variable as it can take any value within a continuum depending on the variables and coefficients in Equations 1 and 2) has the shape of a normal distribution. We observe that the distribution of the random variable $EFF$ is nearly symmetric about its mean and the skewness is close to 0. We notice some points (x-axis value between $-4.5$ and $-5$) in the dataset and distribution to be appearing as outliers as these points are distant from other observations.

![Effort distribution for all issues](image1)

**Fig. 1: Effort distribution for all issues**

Figure 2 shows a box-plot (displaying distribution) for the effort value or contribution score for 12 reviewers who were assigned 7 patch review tasks each. The box-plot displays the full range of variation from minimum to maximum and indicates that there is a significant variance in the effort score of reviewers who reviewed the same number of issues. We use the experimental result and box-plot in Figure 2 to convey that multiple developers reviewing the same number of issues can have wide variation in terms of total effort spent in reviewing the issues. Figure 3 shows the box plot for the effort score of all reviewers in the experimental dataset. There are 433 reviewers in the dataset (one issue can be assigned and reviewed by multiple reviewers). The average number of reviews per reviewer is 14. Figure 3 reveals that the data distribution is asymmetric and is right skewed as it is more spread on the right side. We observe few extreme values and outliers which shows that few developers are long-term, active contributors and core members.

![Box-plot for the contribution score for 12 reviewers](image2)

**Fig. 2: Box-plot for the contribution score for 12 reviewers**

![Box-plot for reviewer score](image3)

**Fig. 3: Box-plot for reviewer score**

VI. CONCLUSIONS

Peer code review systems are applications supporting code inspection, patch submission and patch review activity. We present an application of mining historical data in peer code review systems for reviewer effort and contribution assessment. We define six variables across three dimensions to measure the size and complexity of a patch review task: size of the code churn, size of the modified source code files and program complexity of the modified source code files. We define a metric to measure the effort incurred in a patch review task and conduct a survey of Google Android project developers to validate the suitability of proposed effort variables. We perform an empirical analysis on Google Android peer code review dataset and apply the proposed metrics to compute reviewer effort and contribution. Significant and need to be included in assessment metric. Survey results reveal that code churn
(which is the sum of the number of lines added and deleted in a patch and average cyclomatic complexity per function in the modified source code files are the first and second most important variable influencing the code review effort. Experimental results reveal that there is wide variance in the size and program complexity of modified files and patch across issues resulting in significant variability in-terms of the effort required to close the issue.

REFERENCES


