Who Does What during a Code Review? Datasets of OSS Peer Review Repositories

Kazuki Hamasaki, Raula Gaikovina Kula, Norihiro Yoshida,
A. E. Camargo Cruz, Kenji Fujiwara, and Hajimu Iida
NAIST, JAPAN
{kazuki-h, raula-k, yoshida, camargo, kenji-f}@is.naist.jp, iida@itc.naist.jp

***Abstract*—We present four datasets that are focused on the general roles of OSS peer review members. With data mined from both an integrated peer review system and code source repositories, our rich datasets comprise of peer review data that was automatically recorded. Using the Android project as a case study, we describe our extraction methodology, the datasets and their application used for three separate studies. Our datasets are available online at http://sdlab.naist.jp/reviewmining/**

***Index Terms*—Open Source Software; Quality Assurance; Peer Review Repository Mining;**

I. Introduction

Code reviews serve as an effective software quality assurance activity [1], [2], assuming adequate experience, skills and knowledge are key attributes. This becomes more essential in an OSS setting, where developers are distributed and both, project managers and developers themselves, are usually unaware of their social standing within the review research community. Software repository mining tools and techniques could provide insights on the review members and their activities.

The most common peril associated with mining repositories has been the linkages and manual/semi-automatic matching of multiple data repositories [3], such as a source code repository, mailing lists or bug tracking systems. The result is a smaller analysis dataset, which is also prone to errors. Concerning peer review process, to the best of our knowledge, there is a current lack of review datasets that are freely available. The JavaScript Object Notation (JSON) data structure is used to capture the review process. Due to shortfalls in handling complex structures and type definitions, the simple use of JSON datasets is insufficient.

We extracted four datasets fully automatically from the Gerrit Code Review System[1] and Rietveld[2], providing reliable peer review activity data of Android Open Source Project, OpenStack, Qt and Chromium. Our datasets capture from when the patch is submitted, till its successful merge into the system. In this paper, using the Android Gerrit Code Review System as a case study, we present our extraction methodology and resulting datasets. At first, we present a raw dataset, which captures all data from both the peer review and related source code activity logs. And then, our preprocessed refined datasets, that are more robust and easier to use. Additionally, we demonstrate the application of our datasets for three separate studies on peer review activities.

Our datasets provide general information of peer review activity and focuses on the human aspects of the peer review process. Specifically, analysis of the different roles and activities for OSS peer review members. The datasets were utilized in three different aspects of peer review activities:

- **Self Reviews.** We investigated the impact of code changes that were submitted, reviewed and committed into the source code by the same person.
- **Member Profiling and Expert type Classification**. Based on the peer review activities, we provide a visual analysis of member's performance compared to the rest of the community.
- **Social Networks of the Peer Review Community**. Using social network measures, we analyse the evolution and importance of the different roles of members in the OSS community.

II. Android Project: Gerrit Code Review

The Android Open Source Project (AOSP) is a linux-based operating system for mobile devices such as smartphones and tablet computers, developed by Google in conjunction with the Open Handset Alliance[3]. The first Android-powered smart phones were sold in Q1 2009, and has since grown to become the biggest smart phone operating system.

AOSP uses the GIT source code management system in conjunction with gerrit, a web-based collaborative code review tool. Gerrit automatically records and tracks all merges into the source code, including details related to the code patch and the peer review process activities. AOSP currently has a public and private branch for members to contribute patches. This dataset is solely focused on the public branch[4].

In this paper, a *patch set* refers to the the code change submitted for review. A *review* refers to a code review. *Members* are individuals that are registered into the review system.

From the AOSP documentation[5] and pre-analysis of the dataset, we identified the following members roles in a code review:

[1]http://code.google.com/p/gerrit/
[2]https://code.google.com/p/rietveld/
[3]http://source.android.com/
[4]accessible at https://android-review.googlesource.com/
[5]http://source.android.com/source/roles.html

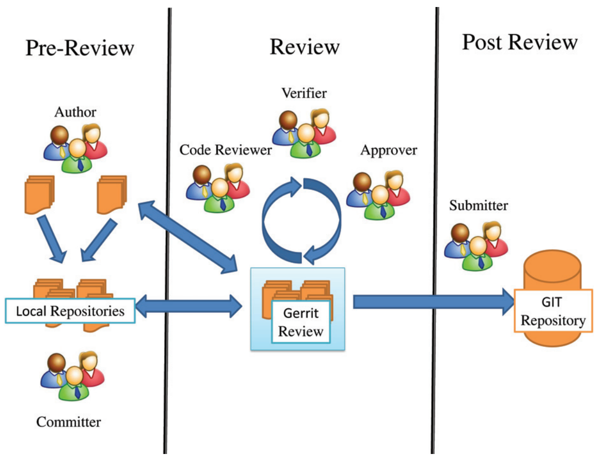

Fig. 1. Simplified Version of the Code Review Process.

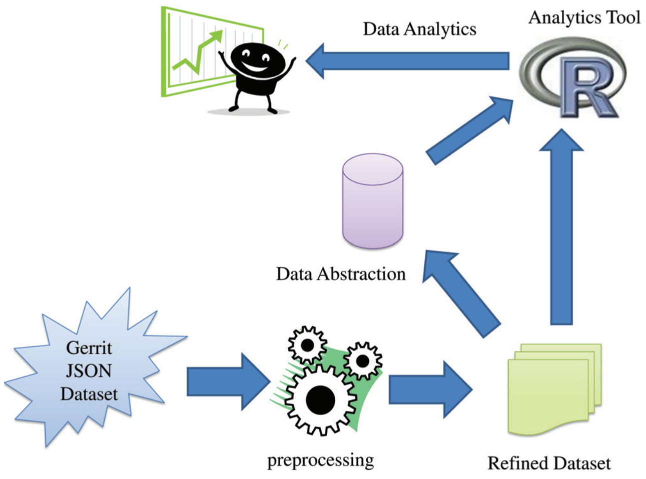

Fig. 2. Overview of the Mining Methodology

- **Author.** Member that writes a patch. It is usually from a local copy of the source code.
- **Committer**[6]**.** Member that submits either a patch or set of patches for review. Note that committers can be authors as well.
- **Code Reviewer.** Members invited to review the patches. The importance difference is that code reviewers cannot verify or approve reviews.
- **Verifier.** Members that tests patches before they are approved for merging into the source code. Developers who have submitted a significant amount of high-quality code are invited as verifiers.
- **Approver.** Members that decide whether or not the patch is acceptable for the source code.
- **Submitter**[6]**.** Once the code is approved, the submitter is responsible for merging the patch into the source code.

Figure 1 illustrates a simplified version of these different member roles within the AOSP peer review process. The interaction and multiple roles that could be inherited by a single member, in reality, is much more complex.

III. Mining Methodology

Figure 2 presents an overview of the methodology used for our datasets. To conserve the structure and integrity of the raw data from the online repositories, the peer review data was extracted in JavaScript Object Notation (JSON) format which is compatible with Gerrit Code Review. Then, this raw dataset was pre-processed to extract only relevant data. The output were two datasets in CSV format and are referred to as our refined datasets.

After initial analysis, we found that there was four main reasons for the preprocessing of the JSON dataset.

- **Structure Complexity.** As seen in Figure 1, the complexity of the roles within the groups would require a much sophisticated data structure than the raw JSON dataset.
- **Size.** Due to the size of the data collected, we needed to abstract and filter only information related to the motivations of our study.
- **Robustness.** Our dataset should be easily imported and used into analytical tools or systems.

In this paper, therefore, we present two post-processed refined datasets that provide concise and reliable/robust information for peer review activity. For the processing, we developed a simple Java tool Mirai, which extracts the needed attributes from the JSON dataset and yields the data in CSV format[7].

TABLE I
Data Summary

Duration	# of Reviews	# of Members
2008/10/21 - 2012/01/28	11,633	1,204

IV. Datasets

Our datasets are available online[7]. Table I presents an overview of the data analyzed. Next, we provide a detailed description of both datasets.

A. JSON Raw Dataset

For each review, the JSON dataset has a unique *ChangeID* for a review. The JSON dataset comprises of the following features: The review information with reviewers comments, the patch updates, and if approved, details of its merge into the source code. Due to space limitations the full scheme is accessible online[8]. The JSON format provides a flexible parsing mechanism that is independent of specific data schema, and also both comments and patch sets can grow according to the number of the resubmissions of the review. There can be

[6]Note that submitter is usually a term for submitting reviews and not committing source code

[7]http://sdlab.naist.jp/reviewmining/

[8]http://sdlab.naist.jp/reviewmining/datastructure.html

TABLE II
REFINEDREVIEWS

Features	Attribute	Description	Data Type	Rationale
Review Info	ChangeID	Unique ID for review	String	ID for review
	Owner	Owner of the review	String	Owner ID
	Created	When review was submitted	Date/Time	Duration
	Updated	Latest update of the review	Date/Time	Duration
	Subject	Subject of the review	String	Patch Topic
	Link	Weblink of the review	String	Weblink reference
	Branch	Location of the patch	String	Location
	Project	Project of the patch	String	Location
	Status	Status of the review	Open/Merged/Abandoned	State of the review
	#Comments	# of comments posted	Int	Review complexity
Reviewers Related Info	Approvers	# of approvers assigned	Int	Review complexity
	Verifiers	# of verifiers assigned	Int	Review complexity
	AssignedReviewers	# of code reviewers assigned	Int	Review complexity
	CodeReviewers	# of actual reviewers (no verifiers/approvers)	Int	Actual review activity
	RealReviews	# of actual reviewers (with verifiers/approvers)	Int	Actual review activity
	SelfReviews	Owner/approver are same person	Boolean	Self reviews
Patch Info	#PatchSets	# of patch resubmissions	Int	Review complexity
	#FilesInPatch	# of files affected by the patch	Int	Review size
SourceCode Info	InlineComments	# of Lines of Comments	Int	Review importance
	AddFiles	# of new files	Int	Review size
	ModFiles	# of modified files	Int	Review size
	DelFiles	# of deleted files	Int	Review size
	RenamedFiles	# of renamed files	Int	Review size
	DeleteLOC	LoC deleted	Int	LoC size
	AddLoC	LoC added	Int	LoC size
	ModLoC	LoC modified	Int	LoC size

TABLE III
REFINEDMEMBERS

Features	Attributes	Description	Data Type	Rationale
Review Info	ID	Member ID	Int	ID
	Name	Name of the member	String	Name
	email	Email address of the member	String	Details
	domain	Extracted from 'email'	String	Organization
Submission Activity	#open	# of opened reviews	Int	Review counter
	#merged	# of merged reviews	Int	Review counter
	#abandoned	# of abandoned reviews	Int	Review counter
	#author	# of patches he/she wrote	Int	Review counter
	#owners	# of reviews that member is the owner	Int	Review counter
	#submitted	# of reviews submitted	Int	Review counter
Review Activity	#PostComments	# of comments posted by the member	Int	Review counter
	#assigned	# of reviews assigned to review	Int	Review counter
	#approved	# of reviews approve	Int	Review counter
	#verified	# of reviews verified	Int	Review counter
	#codeReviewer	# of reviews performed (excluding verify/approve)	Int	Review counter
Code Merge	#committed	# of reviews merged into the source code	Int	Source code counter
	#realReviewer	# of reviews performed (including verify/approve)	Int	Review counter

over 150 attributes in a single review. There are many useful libraries to handle and analyze JSON data[9]

Gerrit can set a role and privileges to the members of any project. The members can also set scores to the patch under revision. To differentiate the different roles of members for a review, we analyzed the reviewers scores of the AOSP for their two review activities, verification (VRIF) and code review (CRVW). As seen in Table IV, for example, *Code Reviewers* have the ability to score -2 to +1 for CRVW. In order to approve the review, +1 from the *Verifier* and +2 from the *Approver* are required. Any -1 from the *Verifier* or -2 from the *Approver* rejects the patch submission. *Code Reviewers* can additionally write comments and express their opinions. *Approvers* and *Verifiers* are experienced members such as Google employees.

[9]For example, MongoDB is a JSON-friendly document-oriented database system http://www.mongodb.org/, and JSON-Path is a simple library to extract parts of a given JSON document like XPath for XML http://goessner.net/articles/JsonPath/.

B. Refined Datasets

The refined datasets are *RefinedReviews* and *RefinedMembers*, which detailed descriptions are provided in Tables II and III.

The *refinedReviews* contains data related to four features of the review: *Review info*, *review related info*, *patch set info* and

TABLE IV
ROLES DERIVED FROM REVIEWER SCORES IN AOSP

Member Role	Score	Raw Data Notation
Verifier	-1 , 0, +1	VRIF
Code Reviewer	-2 , -1 , 0 , +1	CRVW
Approver	+2	CRVW

source code info.

The *refinedMembers* contains information about each member in the project. The data contains features related to member's general information: *Submission activity*, *review activity* and *code merge*. Note that submission refers to submission of reviews and not to the submitter roles of a member.

V. APPLICATIONS/PUBLICATIONS

Our methodology was replicated using the OpenStack[10] , Qt[11] and Chromium Projects[12]. However, we found that the project roles differ from project to project, thus care has to be taken in the pre-processing. The JSON raw datasets are also available online[13].

The original motivation for our datasets was the analysis of self reviews [4]. However, we later realized that it could be applied based on the reviewers roles. Using our refined datasets, other datasets were derived for three different applications, two of them have been published:

A. Member Profiling based on Peer Review Activities

For this study, we used our refined datasets, we investigated their benefits such as identifying hidden experts, inactive or disinterested members to gauge the health of the OSS project and assisting aspiring members to monitor performance, as well as opportunities for career improvement. Using this information, we developed a profiling chart that was later published in [5].

B. Social Network Analysis of Peer Reviews

This study investigated the importance of OSS peer review contributor roles and their review activities by using Social Network Analysis (SNA), proposed as Peer Review Social Network (PeRSoN). To the best of our knowledge, this is the first research constructing social networks from mining a peer review repository. Our results that were published [6], provided hints on relationships among the OSS peer review contributor roles, their activities, and the network structure.

To build PeRSoN, at first, Pajeck[14] was used to generate a set of SNA metrics from the JSON raw dataset. Later, using the refined datasets, the roles of every PeRSoN member were determined.

Additionally these datasets can be used for mining information related to self reviews, merged and abandoned patches, review complexity, and patch complexity.

[10]http://www.openstack.org/
[11]http://qt-project.org/
[12]http://www.chromium.org/
[13]http://sdlab.naist.jp/reviewmining/
[14]http://vlado.fmf.uni-lj.si/pub/networks/pajek/

VI. LIMITATION

Members identification is based on the registration, thus the main threat is email aliasing where members may use multiple accounts. By using semi-manual processes of cross-checking the username, name and email address for duplicates ensures confidence of members details.

The post-processed refined datasets that we present in this paper are only a summary of the data of the review, such as "X reviewers were involved in Z review" and "Y comments were written in Z review". To provide more detailed information, such as "number of expert reviewers involved in the review" and "the reviewers that found the greatest number of defects", parsing additionally raw datasets would be required.

VII. CONCLUSION AND FUTURE WORK

We believe that our datasets fully integrates both source code and peer review repositories in a single dataset. Additionally, providing our raw and refined datasets, users should be able to utilise them for more reliable quantitative analysis. Our datasets open possibilities for mining and collecting review process metrics, thus providing more detailed insights into the review process.

We provide our extraction tools[15] as an OSS and Gerrit API specifications, so that our datasets can be maintained by the OSS community and they can generate new datasets according to their own aims and needs.

REFERENCES

[1] B. Boehm and V. Basili, "Top 10 list [software development]," *Computer*, vol. 34, no. 1, pp. 135 –137, Jan 2001.

[2] M. Mantyla and C. Lassenius, "What types of defects are really discovered in code reviews?" *IEEE Transactions on Software Engineering*, vol. 35, no. 3, pp. 430–448, May-June 2009.

[3] J. Howison and K. Crowston, "The perils and pitfalls of mining sourceforge," in *Proceedings of the 1st International Workshop on Mining Software Repositories (MSR 2004)*, 2004, pp. 7–11.

[4] K. Hamasaki, K. Fujiwara, N. Yoshida, R. G. Kula, K. Fushida, and H. Iida, "Analysis of patch reviews in the android open source project," in *IPSJ SIG Technical Reports*, vol. 2012-SE-176, no. 12, May 2012 (In Japanese).

[5] R. G. Kula, A. E. Camargo Cruz, N. Yoshida, K. Hamasaki, K. Fujiwara, X. Yang, and H. Iida, "Using profiling metrics to categorise peer review types in the android project," in *Supplemental Proceedings of the IEEE 23rd International Symposium on Software Reliability Engineering (ISSRE 2012)*, Nov 2012, pp. 146–151.

[6] X. Yang, R. G. Kula, A. E. Camargo Cruz, N. Yoshida, K. Hamasaki, K. Fujiwara, and H. Iida, "Understanding oss peer review roles in peer review social network (PeRSoN)," in *Proceedings of the 19th Asia-Pacific Software Engeering Conference (APSEC 2012)*, Dec 2012, pp. 709–712.

[15]https://sd-dev.naist.jp/rhodecode/SDLAB/ReviewMining/GerritCrawler/