2012 19th Asia-Pacific Software Engineering Conference

Understanding OSS Peer Review Roles in Peer Review Social Network (PeRSoN)

Xin Yang, Raula Gaikovina Kula, Camargo Cruz Ana Erika, Norihiro Yoshida,
Kazuki Hamasaki, Kenji Fujiwara, and Hajimu Iida
Graduate School of Information Science, NAIST
8916-5 Takayama, Ikoma, NARA 630-0192, JAPAN
{kin-y, raula-k, camargo, yoshida, kazuki-h, kenji-f}@is.naist.jp, iida@itc.naist.jp

***Abstract*—Due to the distributed collaborations and the volunteering nature of Open Source Software (OSS), OSS peer review processes differs from traditional approaches. Despite the latest research efforts to understand OSS peer review processes, very little is known. Unlike related work, this study investigates OSS peer review processes from a different perspective. We investigate the importance of OSS peer review contributor roles and their review activities by using social network analysis (SNA), proposed as PeRSoN (Peer Review Social Network). As a case study, we extracted and analyzed the review process of Android Open Source Project (AOSP). To the best of our knowledge, this is the first research constructing social networks from mining a peer review repository. Our preliminary results provided hints on relationships among the OSS peer review contributor roles, their activities, and the network structure. The results raised issues that will be used to refine our approach in the future.**

***Keywords*-Peer Review; Open Source Software; Social Network Analysis;**

I. Introduction

Peer review is often regarded as one of the most important mechanisms for software quality assurance [1], [2]. With the recent rise of open source software (OSS), traditional peer review process techniques of industry or closed source software need to be changed. This could be due to the factors such as the lack of face-to-face communications, and the volunteering nature of OSS.

The motivation of this research is based on previous works by Rigby on OSS peer review [2] and Bird's study on social aspects of OSS [3]. Building on this research, we apply a novel approach to describe and understand better the OSS peer review process, using social network analysis (SNA). To the best of our knowledge, this is the first research constructing social networks from mining a peer review repository. A different related work performed by Surian [4] researched OSS developer network using graph pattern mining.

In this study, we investigate the importance of OSS peer review contributor roles and their review activities. We use Hong's approach [5] that generated developer social networks (DSN) from bug tracking system. We applied this approach for peer review system to generate a peer review social networks, which we named: PeRSoN (Peer Review Social Network). Then, we analyzed the Android Open Source Project (AOSP).

Our preliminary results gave us hints about the relationships among OSS peer review contributor roles, their activities, and the network structure, providing us with a better understanding of the peer review process in AOSP. Using SNA standard centrality measures, findings can be summarized as follows:

- There is a very strong correlation between verifier activity and degree centrality (*r = 0.922)* (A verifier is a contributor responsible for testing and approving patches).
- There is a strong correlation between verifier activity and betweenness centrality (*r = 0.756)*.
- However, there is no significant correlation between verifier activity and closeness centrality. From these results, we concludes that there are other factors to be considered.

In general, our findings are encouraging for us to refine our approach and investigate its usefulness applied to other projects.

II. PeRSoN

Social network is a kind of network structure which vertices represent people or groups of people, and edges represent social interaction between them [6]. In our study, we create PeRSoN by using peer review data of AOSP. This peer review data comes from a web-based code review system. We seek to investigate the relationship between peer review activities (such as reviewing of the patch or verification of the patch) and contributor's involved in the activities. We introduce and apply some standard measures to analyze our social networks. Centrality measures are simple and illuminating, for they show the most important and central vertices in a network. Freeman [7] defined three centrality measures: Degree, Closeness and Betweenness. He also suggested that these measures have social implications, which is described in Section IV.

Below are some of the important terms that will be used throughout this paper:

- **Contributor.** This represents a participant in code review activities, who can have one or more roles, which we will

TABLE I
Data evolution in PeRSoN.

Subject	2009fh	2009sh	2010fh	2010sh	2011fh
# of New Patches	127	139	148	170	172
# of Verifiers	43	56	92	120	133
# of Non-Verifiers	42	83	160	278	445
# of vertices	85	139	252	398	578
# of edges	127	266	287	318	342

1530-1362/12 $26.00 © 2012 IEEE
DOI 10.1109/APSEC.2012.63

explain later.

- **Contribution Activity.** This represents the main activities carried out by a contributor such as submission, review and verification in review process.
- **Author and Approver.** An `author` submits patches to the OSS peer review system and is the owner of the patch review report.
- **Verifier and Non-verifier.** A `verifier` is responsible for testing and verifying the patches is suitable for merging into the source code. After their verification, code review system submits patches to code repository automatically. To emphasize verifiers importance, we call other contributors `non-verifiers`.
- **Review Process.** A `review process` of AOSP[1] consist as the following steps in summary: First, authors submit new changes and generate new review reports. The system will notify approvers and verifiers. Second, approvers and verifiers review and test changes. Finally, the system merges verified changes to the code repository or abandons them.

III. Methodology

For our study, we use data from AOSP that uses Git[2] to manage its source code and Gerrit as its code review system. First, we extracted data from January 2009 to June 2011, which covers the main releases from version 1.5 to 3.1. Our raw data set is available to download[3]. We used every `6 months` as time-frame from January 2009 to June 2011. We also defined `fh` as first half of a year and `sh` as second half of a year (e.g., 2009fh represents the first half of 2009). Then, we used python scripts to extract the data for inputting into Pajek[4]. Finally, we analyzed every half year. We used Pajek for social network analysis and R Tools[5] for statistical analysis.

Our methodology consisted of 3 basic steps:

1) **Data Preparation.**
   We separated the whole data set by every 6 months and two types of roles (i.e., verifier and non-verifier). Then, we extracted the comment history from all patch review reports, including *contributor id*, *last updated date* and *report id*. We assumed that all comments were finished before the last updated date, and used it instead of the comments dates in the same report. In addition, we extracted a contributor's information list and grouped the contributors into verifiers and non-verifiers. We combined basic data of social network and activity data as shown in Table I.
2) **PeRSoN Generation.**
   To extract data from a peer review system, we applied the approach suggested by Hong et al. in their study [5]. Unlike their approach, our networks were unweighted and we reduced networks by vertices degree but not edges weight. We assumed that contributors have performed activities (including comments) in each patch review report as shown in Figure 1(a). Thus, all contributors participating in the same report are connected as shown in Figure 1(b). Because we are interested in those contributors who performed more activities in peer review process, we decided to reduce our network by removing the vertices whose degrees were less than *three*, which means those contributors performing more activities were kept as shown in Figure 1(c).
3) **Analysis.**
   We performed statistical analysis for each centrality measure. First, we calculated centrality measures and generated cumulative graphs of contributors frequency over time. Then, we discussed all the observations, using the statistical analysis to validate our hypothesis. Finally, we analyzed all centrality measures using correlation.

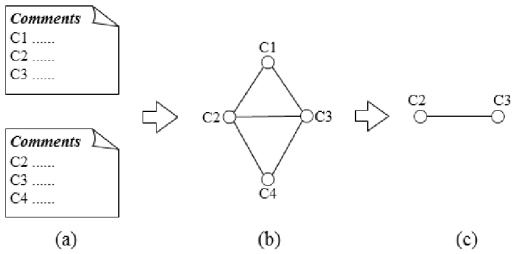

Fig. 1. An illustrative example of extracting PeRSoN.

IV. Preliminary Results

A. Degree Centrality

Degree centrality indicates the number of edges that a vertex has. We created a cumulative graph of contributors frequency to observe degree evolution as the number of contributors increases over time, as shown in Figure 2. The horizontal axis represents the degree centrality of contributors and the vertical axis represents the percentage of contributors. This Figure shows that *80%* of the Non-Verifiers have a degree measure of *less than 10* in 2009fh and *less than 50* in 2011fh, while Verifiers have a degree measure of *less than 50* in 2009fh and *less than 500* in 2011fh. From this figure, the 2 observations can be made: First, verifiers' degree increased overtime, while non-verifiers did not change. Second, verifiers have relatively greater degree than non-verifiers. In OSS projects, verifiers are often active and made more contribution. We assumed that active contributors may have a greater degree than inactive contributors. Because degree implies activity influence [7], we measured the correlations coefficient (spearman[6]) between activities and degree (see *Degree* column in Table II). These results show that verifier activities have stronger linear relationship to degree than non-verifiers.

[1] http://source.android.com/source/life-of-a-patch.html
[2] http://git-scm.com/
[3] http://sdlab.naist.jp/reviewmining/
[4] http://vlado.fmf.uni-lj.si/pub/networks/pajek/
[5] http://www.r-project.org/

[6] we use spearman because we take the measurements from ordinal scales.

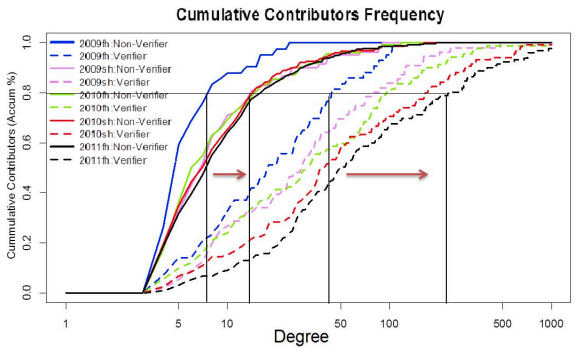

Fig. 2. Cumulative contributors frequency with degree evolution.

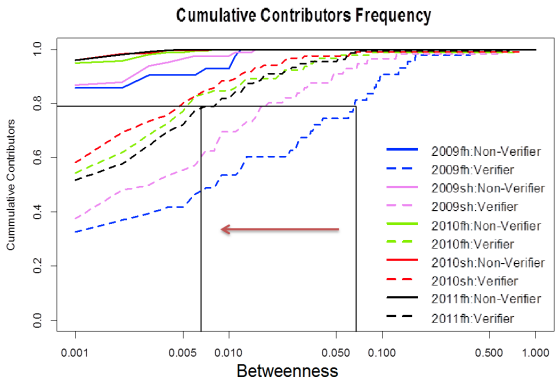

Fig. 3. Cumulative contributors frequency with betweenness evolution.

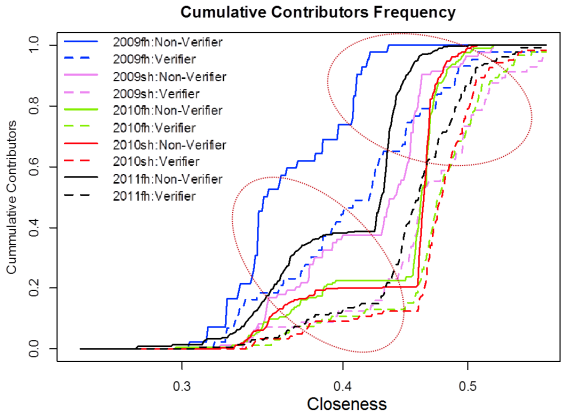

Fig. 4. Cumulative contributors frequency with closeness evolution.

B. Betweenness Centrality

Betweenness centrality of a given vertex indicates the number of shortest paths from all vertices to all other vertices that pass through that vertex. We created a cumulative graph of contributors frequency to observe betweenness evolution as the number of contributors increases over time, as shown in Figure 3. The horizontal axis represents the betweenness centrality of contributors and the vertical axis represents the percentage of contributors. This Figure shows that *80%* of the verifiers have a betweenness *less than 0.100* in 2009fh and *less than 0.010* in 2011fh, while most non-verifiers have a low betweenness. Moreover, the following observations can be made: First, verifiers' betweenness decreased overtime in general, while non-verifiers did not change. Second, verifiers have relatively greater betweenness than non-verifiers. In OSS development, verifiers often have more control than others. We assumed that verifiers may have greater betweenness. Because betweenness implies control influence [7], we measured the correlations coefficient (spearman) between activities and betweenness (see *Betweenness* column in Table II). These results show that activities of verifiers have stronger linear relationship to betweenness. In addition, we have found some exception cases. For example, one verifier's betweenness reached a high level of *0.051*, but he only verified once. He also contributed to *497 submissions* and *334 reviews*. Thus, we supposed there may be other factors influencing closeness. We plan to investigate these exception cases in our future work.

C. Closeness Centrality

The farness of a vertex is defined as the sum of its distances to all other vertices, and its closeness is defined as the inverse of the farness. We created a cumulative graph of contributors frequency to observe closeness evolution as the number of contributors increases over time, as shown in Figure 4. The horizontal axis represents the closeness centrality of contributors and the vertical axis represents the percentage of contributors. We found this graphs shows a bimodal distribution because there are *two peaks* in both verifiers and non-verifiers. We also created other graphs such as density distribution, and obtained the same results. We measured the correlations coefficient (spearman) between activities and closeness (see *Closeness* column in Table II). These results suggested that there is not a linear relationship between activities of all roles and closeness. Because closeness implies independence of vertices [7], and

TABLE II
CORRELATION OF ACTIVITY AND CENTRALITY MEASURE.

Activity	Degree	Betweenness	Closeness
# of Verifier Activity	0.922	0.756	0.621
# of Verifications	0.811	0.644	0.548
# of Non-Verifier Activity	0.617	0.210	0.318

TABLE III
EXCEPTION CASES FOR VERIFIERS.

	Low Degree	Low Closeness	Low Betweenness
High Degree	—	Active / Far away	—
High Closeness	Inactive / Central	—	—
High Betweenness	—	—	—

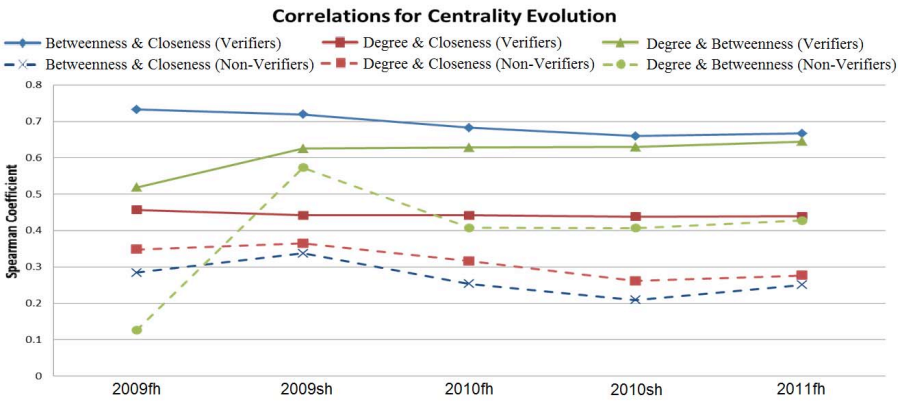

Fig. 5. correlations for three centrality measures evolution.

given the resulting bimodal distribution, we regard closeness as more complicated factor than degree and betweenness, which may be explained by studying other factors, different to roles and activities.

D. Correlations of Centrality Measures

Because we supposed that verifiers should be the most central and important role in our study, and we found that there are different correlations between the different centrality measure and verifier activities, we perform the following.

We compared across these three centrality measures, to measure correlation coefficients among them as shown in Figure 5. This figure shows that there is a strong correlation between verifiers' betweenness and closeness, and also between their degree and betweenness.

We analyzed exception cases in which:

- A verifier had high-degree and low-closeness. Having high-degree means he or she performed more activity, and having low-closeness means he or she is far away from the network center.
- A verifier had low-degree and high-closeness. Having low-degree means he or she performed few activity, and having high-closeness means he or she is close to the network center.

In addition, we combined correlation results (see Figure 5) and exception cases to create Table III. From which the following observations can be made: First, verifiers with high-closeness (close to network center) may have high-betweenness (more control). Second, verifiers with high-degree (more contribution) may have high-betweenness (more control). However, there is no significant correlation between verifiers' degree (contribution activity) and closeness (network location). Finally, from these findings, some suggestion for the exception cases can be provided:

- Verifiers with high-degree and low-closeness may be experts in specialized fields, because they perform more activity and have few connections with other members or work teams. Therefore, if they result to be specialists, we could suggest that the most important review requests are sent to them.
- Verifiers with low-degree and high-closeness may contribute less; however, being close to the network center, they may represent key figures tied to many other people. So, we see these verifiers as high authorities, for which we would suggest not to remove them or change their roles thoughtlessly.

V. Conclusion

Our preliminary results shed light on how contributors work and communicate together and how their activities can affect whole projects in peer review processes. Our findings gave us hints about the relationships among member roles, their activities and the network structure, these findings also raise additional questions.

- What factors can be related to closeness?
- What can the exception cases tell us in detail?
- How can we use PeRSoN to improve the OSS Peer review?

We hope our future work may address these pertinent questions to refine our approach and obtain better results.

Acknowledgment

This work was supported by JSPS KAKENHI Grant Number 22500027.

References

[1] P. C. Rigby, D. M. German, and M.-A. Storey, "Open source software peer review practices: a case study of the apache server," in *Proc. of ICSE*, 2008, pp. 541–550.

[2] P. C. Rigby and M.-A. Storey, "Understanding broadcast based peer review on open source software projects," in *Proc. of ICSE*, 2011, pp. 541–550.

[3] C. Bird, D. Pattison, R. D'Souza, V. Filkov, and P. Devanbu, "Latent Social Structure in Open Source Projects," in *Proc. of SIGSOFT /FSE*, 2008, pp. 24–35.

[4] D. Surian, D. Lo, and E.-P. Lim, "Mining collaboration patterns from a large developer network," in *WCRE*, 2010, pp. 269–273.

[5] Q. Hong, S. Kim, S. Cheung, and C. Bird, "Understanding a developer social network and its evolution," in *Proc. of ICSM*, 2011, pp. 323 –332.

[6] M. Newman, *Networks: An Introduction*. New York, NY, USA: Oxford University Press, Inc., 2010.

[7] L. C. Freeman, "Centrality in social networks conceptual clarification," *Social Networks*, vol. 1, no. 3, pp. 215–239, 1978-1979.