Social Network Analysis in Open Source Software Peer Review

Xin Yang
Nara Institute of Science and Technology, Japan
kin-y@is.naist.jp

ABSTRACT

Software peer review (aka. code review) is regarded as one of the most important approaches to keep software quality and productivity. Due to the distributed collaborations and communication nature of Open Source Software (OSS), OSS review differs from traditional industry review. Unlike other related works, this study investigated OSS peer review processes from social perspective by using social network analysis (SNA). We analyzed the review history from three typical OSS projects. The results provide hints on relationships among the OSS reviewers which can help to understand how developers work and communicate with each other.

Categories and Subject Descriptors

D.2.5 [**Software Engineering**]: Testing and Debugging—*Code inspections and walk-throughs*; E.1 [**Data Structures**]: Graphs and networks

Keywords

open source, peer review, social network

1. INTRODUCTION

Software peer review refers to the code inspections by developers, rather than the author himself. It can be regarded as one of the most important activities for software development [2, 1]. Software project developers apply peer review for two main benefits: reducing defects and saving cost. The traditional peer review (or inspection) is established by Fagan, which performs peer review activities in the form of group meeting [6, 5].

In the past decade, open source software (OSS), which is a very unique and dynamic software development manner, has been adopted by many software projects. Most of the OSS projects have geographically distributed development environment. and there are very few chances for developers to have face-to-face communication. Unless changing the approach for peer review, the traditional meeting-based peer review seems hard to be applied on OSS projects. Moreover, developers' experiences, behaviors and attitude can affect the quality of source code. As a result, it is demanding experienced experienced developers could share their knowledge with new members by reviewing their code in OSS society. Furthermore, having a good development community to share those experiences and knowledge is demanded. We investigated some related studies that have been done in OSS peer review [12, 13], to the best of our knowledge, this is the first research constructing social networks from mining peer review repository and also performing social network analysis to study OSS peer review process.

In this study, we investigated the importance of OSS contributors by their roles (or positions) in peer review. We used social network analysis (SNA) to perform our study bug tracking system. We applied this approach on peer review system to generate peer review social networks named PeRSoN (Peer Review Social Network). Then, we analyzed three typical OSS projects: Android Open Source Project (AOSP), Qt and OpenStack as case study. The preliminary result gives us hints about the relationships among OSS peer review contributor roles, their activities, and the network structure. Our research questions can be summarized as follows:

RQ1 Is there any relationship between the contributors' activities and their network positions?

RQ2 Who is the most important contributor in OSS peer review?

For **RQ1**, we started from the most standard social network measures as centrality metrics to measure the network position of each contributor. Then we performed correlation for contributor activities and their social network measures. For **RQ2**, we summarized and separated all the peer review contributors into several different role groups and we successfully found that, most active reviewers who have Verification authority are the network center which implies the most important contributors.

2. RELATED WORK AND BACKGROUND

Some studies on OSS peer review have been done in recent years. Rigby et al. has examined Apache Server Project and created some metrics similar to traditional inspection in order to find an efficient and effective OSS review technique [12]. They also have studied the broadcast nature of OSS peer review, which is different from the traditional peer review method [13]. Some studies also suggest using review bot to reduce human effort and improve review quality [4].

FSE'14, November 16–21, 2014, Hong Kong, China
ACM 978-1-4503-3056-5/14/11
http://dx.doi.org/10.1145/2635868.2661682

Comparing with e-mail based peer review, researchers found web-based tools are preferred in modern peer review because contributors can easily check the review status of source code [3]. In our study, we analyzed the review community by its social networks.

Social network is a network structure in which the vertices represent people or groups of people, and the edges represent social interaction between them such as conversation or notification [11]. Andrew et al. applied social networks on failure prediction [10]. In our study, we create PeRSoN by extracting peer review history data from three OSS projects. The peer review history data comes from their code review systems. All three projects use Gerrit, a web-based code review system to perform code review from code repository (all these projects use Git as to manage source code). We sought to investigate the potential relationship between contributors' activities and their importance in the social network. By applying some standard centrality measures, we were able to analyze our PeRSoN from social aspect. Freeman defined three centrality measures: Degree, Closeness and Betweenness, which represent the importance of vertex from different perspective [7]. Below are some of the important terms that will be used throughout this paper:

- **Author and Reviewer**. An **author** represents the contributor who submits code patch and the owner of the code review report. A **reviewer** represents the contributor who review the patchset.
- **Approver and Verifier**. An **approver** is a special reviewer who is experienced and has approval authority by checking whether the code patch follows the best practices of the current project and fits the project's stated purpose. A **verifier** is responsible for building, testing and verifying the code patch and decides whether it is suitable for merging into the code repository. **Verifiers** can be either human contributors or automatic test tools.

3. APPROACH AND UNIQUENESS

The traditional research on software engineering always focuses on source code metrics and development history, but we applied a novel approach from social perspective to study peer review which is an important phase in software development. Our dataset comes from AOSP, Qt and OpenStack. All these projects use Git to manage the source codes and Gerrit to manage peer reviews. The main approach consisted of 3 main steps:

1) Preparation before experiment. Before the experiment, we extracted the raw review dataset from the Gerrit server of each project. The details of mining review repository approach can be found from the study of Hamasaki et al. [8]. Our raw data set is available to download[1]. Because we focused on social aspect of OSS peer review, the main dataset that we extracted is the contributors information and their activities history. In order to investigate the common of contributors, we grouped the contributors into several role groups by their different review activities.

2) PeRSoN Generation. After we have the contributors information and their review histories, we started to generate the peer review social network. We applied an approach which is used for generating bug report network [9]. However, our networks were generated from peer review reports.

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

Table 1: Correlation of Most Active Verifier Activity And Centrality Measure.

Projects	Degree	Betweenness	Closeness
AOSP	0.952	0.789	0.485
OpenStack	0.964	0.992	0.868
Qt	0.953	0.884	0.795

Based on the broadcasting nature of OSS peer review, we assumed that the contributors who appeared in the same review report must have communications with each other. Thus, we connected all the contributors who participated in the same report.

3) Analysis. We performed social network analysis (centrality measure) for each contributor and analyzed the relationship between contributors centrality measures and their activities using correlation[2]. We applied three standard centrality measures: degree, closeness and betweenness on the experiment to measure the importance of contributors from three different perspective. Then we compared the centrality of contributors with their review activities such as submissions, reviews, approvals, verifications. In order to find special features from the whole review community, we separated contributors into different role groups. At the end we applied Wilcoxon-Mann-Whitney test to evaluate the classification of roles.

4. RESULT AND CONTRIBUTION

Our results indicate there is a strong linear correlation between verifiers activities and their centrality measures in AOSP (degree and betweenness but not including closeness) [15]. Furthermore, we investigated the correlations between *the most active verifiers* activities and their centrality measures in three projects. The result in Table 1 indicates that in OpenStack and Qt, activities of these kind of verifiers have strong linear relationship to all centralities. In AOSP, their activities have strong linear relationships to degree and betweenness, but moderate correlation in closeness. Our preliminary result gives us hint to study the relationship of peer review contributors and their network positions.

In addition, we compared the different centrality distributions among contributor roles by applying Wilcoxon-Mann-Whitney test [14]. All three projects have been tested by comparing role groups existing in each project and all p-values of the comparison are below 0.05. As result, the most active verifiers have significant more centrality than the other contributors in these projects.

Based on these results, two main contribution can be summarized as follows:

- There is strong correlation relationship between *the most active verifiers*'s activities and their degree centrality and betweenness centrality.
- *The most active verifiers* are at the center of peer review networks, which implies they are the most important contributors in their community.

These contributions provide us with hints on how OSS peer review contributors work and communicate, which can help to understand peer review process and investigate the potential defect in peer review community.

[2]We use Pearson's correlation because of the normal distribution of data

5. REFERENCES

[1] A. Frank Ackerman, Priscilla J. Fowler, and Robert G. Ebenau. Software inspections and the industrial production of software. In *Proc. of a symposium on Software validation: inspection-testing-verification-alternatives*, pages 13–40, New York, NY, USA, 1984. Elsevier North-Holland, Inc.

[2] A.F. Ackerman, L.S. Buchwald, and F.H. Lewski. Software inspections: an effective verification process. *Software, IEEE*, 6(3):31–36, 1989.

[3] Alberto Bacchelli and Christian Bird. Expectations, outcomes, and challenges of modern code review. In *Proceedings of the 2013 International Conference on Software Engineering*, ICSE '13, pages 712–721, Piscataway, NJ, USA, 2013. IEEE Press.

[4] Vipin Balachandran. Reducing human effort and improving quality in peer code reviews using automatic static analysis and reviewer recommendation. In *Proceedings of the 2013 International Conference on Software Engineering*, ICSE '13, pages 931–940, Piscataway, NJ, USA, 2013. IEEE Press.

[5] M. E. Fagan. Design and code inspections to reduce errors in program development. *IBM Systems Journal*, 15(3):182–211, 1976.

[6] M.E. Fagan. Advances in software inspections. *Software Engineering, IEEE Transactions on*, SE-12(7):744–751, 1986.

[7] Linton C. Freeman. Centrality in social networks conceptual clarification. *Social Networks*, 1(3):215–239, 1978-1979.

[8] Kazuki Hamasaki, Raula Gaikovina Kula, Norihiro Yoshida, AE Cruz, Kenji Fujiwara, and Hajimu Iida. Who does what during a code review? datasets of oss peer review repositories. In *Proceedings of the Tenth International Workshop on Mining Software Repositories*, pages 49–52. IEEE Press, 2013.

[9] Qiaona Hong, Sunghun Kim, SC Cheung, and Christian Bird. Understanding a developer social network and its evolution. In *Proceedings of the 2011 27th IEEE International Conference on Software Maintenance*, pages 323–332. IEEE Computer Society, 2011.

[10] Andrew Meneely, Laurie Williams, Will Snipes, and Jason Osborne. Predicting failures with developer networks and social network analysis. In *Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, SIGSOFT '08/FSE-16, pages 13–23, New York, NY, USA, 2008. ACM.

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

[12] Peter C Rigby, Daniel M German, and Margaret-Anne Storey. Open source software peer review practices: a case study of the apache server. In *Proceedings of the 30th international conference on Software engineering*, pages 541–550. ACM, 2008.

[13] Peter C Rigby and Margaret-Anne Storey. Understanding broadcast based peer review on open source software projects. In *Proceedings of the 33rd International Conference on Software Engineering*, pages 541–550. ACM, 2011.

[14] Douglas A Wolfe and Myles Hollander. Nonparametric statistical methods. *Nonparametric statistical methods*, 1973.

[15] Xin Yang, Raula Gaikovina Kula, Camargo Cruz Ana Erika, Norihiro Yoshida, Kazuki Hamasaki, Kenji Fujiwara, and Hajimu Iida. Understanding oss peer review roles in peer review social network (person). In *Proceedings of the 2012 19th Asia-Pacific Software Engineering Conference-Volume 01*, pages 709–712. IEEE Computer Society, 2012.