How Do Social Interaction Networks Influence Peer Impressions Formation? A Case Study

Amiangshu Bosu and Jeffrey C. Carver

University of Alabama, Tuscaloosa, AL, USA
asbosu@ua.edu, carver@cs.ua.edu

Abstract. Due to their lack of physical interaction, Free and Open Source Software (FOSS) participants form impressions of their teammates largely based on sociotechnical mechanisms including: code commits, code reviews, mailing-lists, and bug comments. These mechanisms may have different effects on peer impression formation. This paper describes a social network analysis of the WikiMedia project to determine which type of interaction has the most favorable characteristics for impressions formation. The results suggest that due to lower centralization, high interactivity, and high degree of interactions between participants, the code review interactions have the most favorable characteristics to support impression formation among FOSS participants.

Keywords: Open Source, OSS, FOSS, social network analysis, collaboration.

1 Introduction

Impression Formation (i.e., obtaining an accurate perception of teammates' abilities) is difficult for Free Open Source Software (FOSS) developers because of the lack of physical interaction with their distributed, virtual teammates. Inaccurate perceptions of teammates' abilities and expertise may reduce the team's productivity because some developers improperly disregard the opinions or contributions of certain teammates. Due to the lack of physical interaction, members of distributed teams form impressions based on tasks [16]. This task-focus increases the potential effects of *socio-technical interactions* (i.e. social interactions facilitated by technological solutions) on impression formation. Common FOSS socio-technical interaction mechanisms include: project mailing-lists, bug repository, code review repository, and code repository.

Each of these socio-technical mechanisms facilitates different types of interactions among project participants. The results of our recent large-scale survey of FOSS developers suggest that developers believe code review is beneficial in the impression formation process [6]. Our hypothesis in this work is that due to the nature of interaction supported by each of these mediums, some may be more beneficial than others for impression formation. We conducted a social network analysis (SNA) of the interactions facilitated by each socio-technical mechanism to determine which of those mechanisms tended to support impression formation.

L. Corral et al. (Eds.): OSS 2014, IFIP AICT 427, pp. 31–40, 2014.

A social network is a theoretical construct that represents the relationships between individuals, organizations, or societies. These networks are typically modeled as a graph in which vertices represent individuals or organizations and edges represent a relationship between individuals or organizations. SNA focuses on understanding the patterns of interactions and the relative positions of individuals in a social settings [10]. SNA is commonly used in many domains, e.g., sociology, biology, anthropology, communication studies, information science, organizational studies, and psychology.

Here we summarize the results of previous SNA analysis on interactions facilitated by socio-technical mechanisms. *Code-commit* social networks: 1) exhibit 'small world' characteristics[1], 2) show that developers who work on the same file are more likely to interact in the mailing-list [4], and 3) identify the influence of a sponsoring company on the development process, release management, and leadership turnover [15]. *Bug-fixing* social networks show: 1) decentralized interactions in larger projects [8] and 2) a stable core-periphery structure with decreasing group centralization over time [14]. *Mailing-list* social networks: 1) exhibit 'small world' characteristics with developers having higher status [3] and 2) the core members have disproportionately large share of communication with the periphery members [17]. *Code-review* social networks show those responsible for testing and approving patches are the central actors [19].

Previous studies applying SNA to FOSS projects focused on only one or two of these social networks. However, due to the differing characteristics of each network, it is important to compare the networks to understand the various types of interactions within FOSS communities. Therefore this study has two primary goals: 1) Identify the key characteristics of each FOSS social interaction networks, and 2) Compare those networks to identify which have characteristics that best support peer impression formation.

The remainder of the paper is organized as following. Section 2 describes the research method. Section 3 defines the metrics used in this study. Section 4 discusses the study results. Finally, Section 5 concludes the paper.

2 Research Method

We performed a case study to analyze four types of social networks. This section details the methods for project selection, data collection, and data analysis.

2.1 Project Selection

To allow for the desired analysis, we needed to identify a FOSS project with a large, active developer community who performed code reviews that had publicly accessible repositories. The FOSS projects maintained by the WikiMedia foundation (`www.wikimedia.org`) satisfy the criteria. WikiMedia foundation is a non-profit organization that maintains many active, open-source, web-based

[1] The average distance between pairs of nodes in a large empirical networks are often much shorter than in random graphs of the same size [18].

projects including: MediaWiki, and Wikdata. Contributors to these projects include a good mix of volunteers and paid Wikimedia employees. This characteristic is representative of many popular FOSS projects and therefore makes these projects ideal for this research.

2.2 Data Collection

The data from WikiMedia required to conduct SNA on the four sociotechnical mechanisms is available in online repositories. The WikiMedia project development policy[2] requires all code changes to be reviewed prior to inclusion in the project. To facilitate this review process, the WikiMedia projects use Gerrit[3], an online code review tool. Because all changes were submitted to Gerrit, this repository contains the code review and the code commit information. WikiMedia's BugZilla[4] repository contains the fault data. We developed Java applications to mine these repositories. WikiMedia's mailing-list archives (i.e. wikitech-l[5]) contains the development mailing-list data. We used the Mailing List Stats tool [13] to mine the development mailing-list data. All tools populated a local MySQL database.

2.3 Data Analysis

We cleaned the data by removing 'bot' accounts from the code review and bug repositories. A manual inspection of the comments by accounts that contained 'bot' in the name (e.g. L10n-bot or jenkins-bot) revealed that the messages were automated rather than human-generated. We also merged multiple mailing-list accounts from the same developer using an existing approach [3] to resolve multiple email aliases belonging to same developer.

We then wrote scripts to calculate the number of interactions between developers captured by data in each repository. For each data type, we created social networks as undirected, weighted graphs where nodes represent developers and edge weights represent the quantity of interactions between the developers. Table 1 provides an overview of the characteristics of the four social interaction networks. We then used Gephi [2] for SNA and UCINet [5] for calculating network centralization scores.

3 Social Network Metrics

This section describes the commonly used SNA metrics employed in this study.

Average Degree: Average of the degrees of all the nodes in the network, where *degree* is the number of edges connected to a node. This metric quantifies the average number of teammates with which each person interacts.

[2] `http://www.mediawiki.org/wiki/Developmentpolicy`
[3] `https://gerrit.wikimedia.org`
[4] `https://bugzilla.wikimedia.org/`
[5] `https://lists.wikimedia.org/mailman/listinfo/wikitech-l`

Table 1. Overview of the four social interaction networks

Network	Data source	Edge weight between nodes
Mailing-list	29,038 emails	# of mutual email threads
Bug interaction	9,952 bugs	# of bugs where both have commented
Code review	90,186 requests	# of mutual code review requests
Code commit	90,186 commits	# of files where both have made at least one change

Average Weighted Degree: Average of the weighted degrees of all the nodes in the network, where *weighted degree* is the sum of the edge weights of all edges connected to a node. This metric indicates the average volume of interaction between any two teammates.

Network Density: The ratio of the number of edges that exist to the maximum number of edges possible. This metric indicates how many pairs of teammates are interacting with each other.

Because many real-world networks exhibit small-world *properties, the next three metrics quantify the small-world characteristics of the networks.*

Average Path Length: The average length of the shortest paths (i.e. number of steps) between every pair of nodes. This metric indicates the average distance in the interaction network between any two teammates.

Average Clustering Coefficient: The *clustering coefficient* of a node is a measure of the connectedness of its neighbors. That is, it is the percentage of all possible edges between a node's neighbors that are actually present or the probability that any two neighbors of the node are connected. The *Average Clustering Coefficient* of a network is the average of the clustering coefficients of the nodes in the graph. Small-world networks have a higher average clustering coefficient relative to other networks of same size [18].

Network Centralization: Network centralization is a measure of the relative centrality of the most central nodes in a network [9]. A higher centralization means that there are a relatively small number of central nodes. According to Freeman [9], network centralization can be calculated as

$$C_X = \frac{\sum_{i=1}^{n}[C_X(p*) - C_X(p_i)]}{max\sum_{i=1}^{n}[C_X(p*) - C_X(p_i)]}$$

where n= number of nodes in the network, $C_X(p_i)$ = any centrality measure of point i, $C_X(p*)$ = largest value of $C_X(p_i)$ for any node in the network, and $max\sum_{i=1}^{n}[C_X(p*) - C_X(p_i)]$= the maximum possible sum of differences in point centrality for a network on n nodes. Among the different methods for calculating network centrality, Freeman recommends using *Betweenness Centrality* (i.e. the number of times a node acts as a bridge along the shortest path between two

other nodes [9]), to identify the central nodes in a network [11]. Therefore, we used *Betweenness Centrality* as an indicator of the extent to which a small number of central nodes dominate the interactions.

4 Results

Figure 1 shows the social networks built from the four data sources using the Fruchterman-Reingold layout algorithm [12]. To make the diagrams more comprehensible, we first filtered out the low-weight edges and then excluded the isolated nodes. Node size is based on weighted degrees. Edge widths are based on edge weights. Intensity of the node color is based on betweenness centrality.

A visual inspection of the diagrams suggests that the *bug interaction* and *mailing-list* networks are similar. They each contain only a few nodes with high weighted degree (i.e. node size) and high betweenness centrality (i.e. color intensity). There are also a small number of edges between non-central nodes, which indicates little interaction among them. Conversely, the *code review* and *code commit networks* have more nodes with high weighted degree and high betweenness centrality. Furthermore, in the *code commit network*, the distribution of nodes based on weighted degree and betweenness centrality is more uniform and contains a lot of edges between the non-central nodes. These graphs suggest that the *mailing-list* and *bug interactions* networks are the most centralized, followed by the *code review* network and finally the *code commit* network, which is the most decentralized.

The following subsections provide a more detailed comparison of the networks using the metrics defined in Table 2.

Table 2. Metrics describing each social interaction network

	Mailing-list	**Bug interaction**	**Code review**	**Code commit**
# of nodes	2,518	3,416	522	448
# of edges	28,208	34,200	5,052	21,353
Avg. degree	22.33	20.02	19.36	95.33
Avg. weighted degree	28.35	39.12	196.93	688.55
Avg. path length	2.67	2.6	2.53	1.88
Network density	0.009	0.006	0.037	0.213
Avg. clustering coefficient	0.762	0.853	0.693	0.839
Network centralization	28.49%	32.18%	10.94%	5.94%

4.1 Network Size

The *bug interaction* network has the most nodes, followed by the *mailing-list* network, the *code review* network and finally the *code commit* network. The *bug interaction network* is the largest because any user can create an account and post bugs in the BugZilla repository, while the development *mailing-list*

is only for development discussion and posting requires approval from the list administrator. The *code review* network is slightly larger than the *code commit* network because a contributor without commit access can submit a patch for code review. If that patch passes the code review, a core developer commits it to the main branch. Finally, because only long time contributors earn commit access, the *code commit* network is the smallest.

Participants have a similar number of peers in all networks except for the *code commit* network, where they have more. This larger number of peers may indicate that developers work on many modules. Although participants in the *mailing-list*, *bug*, and *code review* networks have a similar number of peers, participants

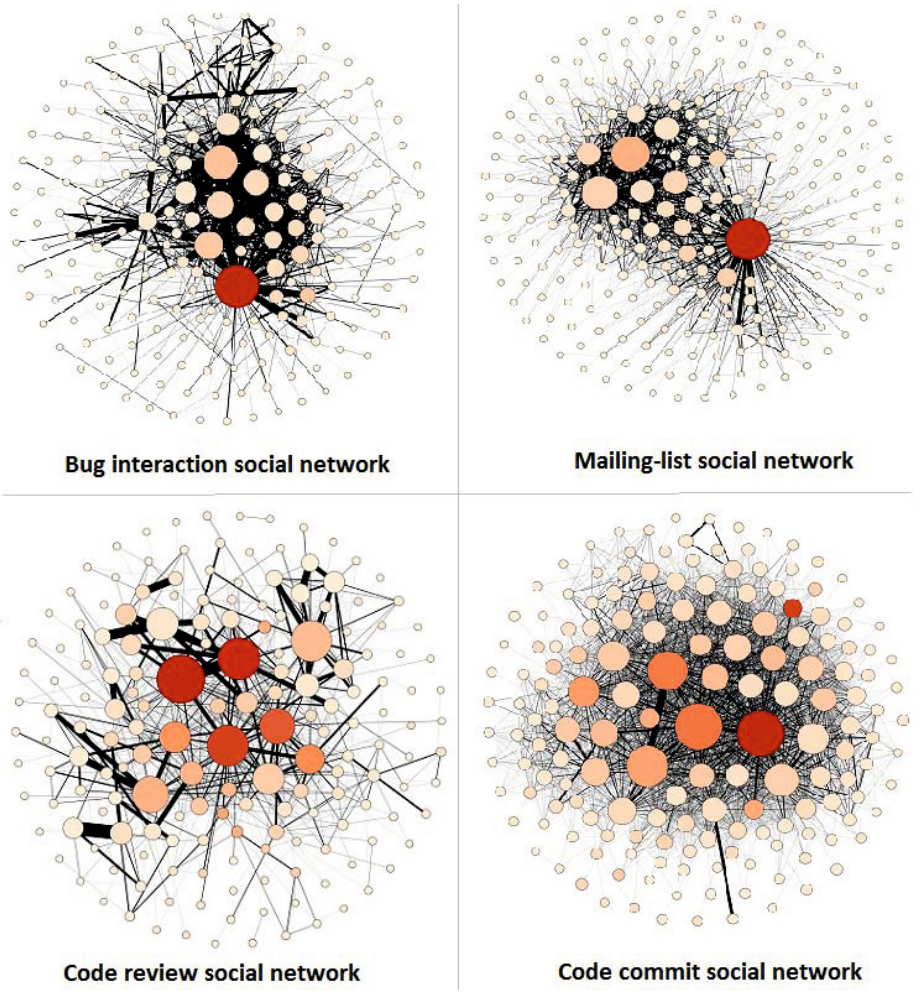

Fig. 1. Social network diagrams

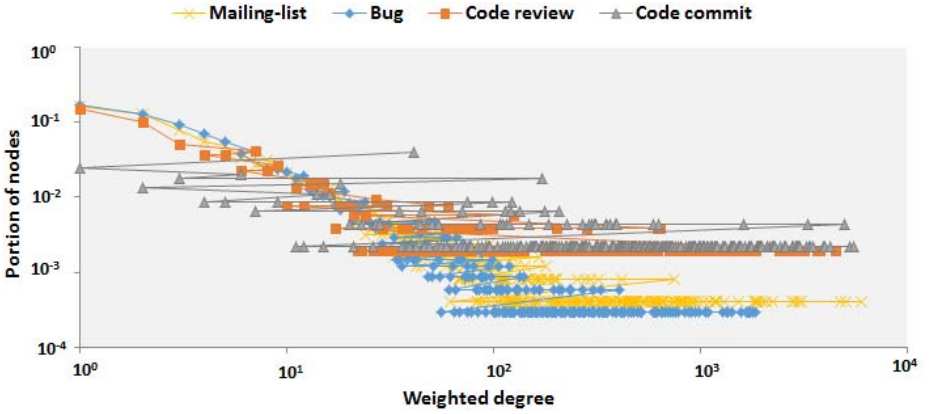

Fig. 2. Weighted degree distributions (log-log)

in the *code review* network interact more frequently with those peers (i.e. higher average weighted degree). Because of the very high average degree, the *code commit* network has the lowest average path length, and highest density. The *mailing-list* and *bug interaction* networks have similar densities and average path lengths. However, the *code review* network has a lower average path length, and higher density, indicating that two contributors are more likely to interact through code review than through a mailing-list or a bug. Considering the higher probability (i.e. density) and higher volume (i.e weighted degree) of *code commit* and *code review* interactions, developers are more likely to form impressions with their teammates via those interactions.

4.2 Network Model

Figure 2 shows that for three of the four networks (excluding the code commit network), the weighted degree distribution may follow the power law distribution ($p[X = x] \propto (x)^{-\alpha}$)[6]. We used the R - poweRlaw package [7] to estimate the α values (i.e. power law exponent) and the goodness-of-fit[7] (p) for the power law distributions. The α values for the *mailing-list*, *bug interaction*, and *code review* networks are estimated as 1.73 (p=.976), 1.7 (p=.994), and 1.9 (p=.851), respectively. This result suggests that like many other real world social networks, the *mailing-list*, *bug interaction*, and *code review* networks follow the power law distribution. The low value of $\alpha(< 3)$, short average path length (Table 2), 'small world' property (i.e. high clustering coefficients), and presence of hubs that indicate those three graphs follow the *scale-free network* model [1].

[6] In a power law distribution, small occurrences are extremely common and large instances are extremely rare. Because the probability of a large value decreases exponentially, this disribution is referred to as the 'power law'.

[7] Kolmogorov-Smirnov goodness of fit test measures if the dataset comes from a specific distribution. p $\approx$ 0 indicates the model does not provide a plausible fit.

The results of previous research [1] into understanding the properties and modeling the growth of scale-free networks can be helpful for understanding the interactions among FOSS participants. For example, scale-free networks are robust because the removal of any randomly-chosen node is not likely to break network connectivity. Conversely, hub nodes are points of failure whose removal can lead to network disintegration. As a result, the unavailability of a central developer may break FOSS network connectivity and affect the productivity of the other dependent developers. In terms of growth, scale-free networks follow a *preferential attachment* model in which a new node is more likely to begin interactions with high degree nodes than with low degree nodes. Therefore, in scale-free FOSS networks, new members are more likely to begin their interactions with contributors who are popular (i.e. the higher degree nodes).

4.3 Network Centralization

The betweenness centralization scores indicate that the *bug interaction* and the *mailing-list* networks are the most centralized. They depend more upon the central nodes than do the *code commit* and *code review* networks.

High centralization might be beneficial in some cases (i.e. making quick decisions). But for a large FOSS network, high centralization has many drawbacks. First, central nodes may become bottlenecks due to overload. Second, because most interaction goes through the central hubs, there is less interaction among the non-central nodes. In this situation, the central nodes become familiar with most of the network while the non-central nodes become familiar only with the hubs. Therefore, high centralization is not helpful for impression formation in a FOSS community because most nodes are non-central nodes. As a result, lower centralization may be better for impressions formation.

Because it has the lowest centralization score, the *code commit* network has the most favorable characteristic for impression formation. While working on code written by a teammate could lead to impression formation, the lack of the type of direct interaction that occurs during code review suggests that interactions around code commits may not be the best for impression formation. The *code review* network has the second lowest centralization score. As opposed to the code commit interactions, code review interactions are highly interactive between the author and reviewer. Evaluating the quality of code written by a peer and discussing it's design helps to build mutual impressions between the author and reviewers. Although mailing-lists and bug interaction are also interactive, those interactions are highly centralized.

Therefore, considering both interactivity and centralization, we hypothesize that code review interactions should provide the best support for impression formation between FOSS developers. The results of our prior study, which indicate a high level of trust, reliability, perception of expertise, and friendship between FOSS peers who have participated in code review for a period of time, also provide evidence for this hypothesis [6].

5 Conclusion

Considering network structure, centralization, and number of interactions, interactions related to code commits have the most favorable characteristics for impression formation, followed by interactions related to code review. However, because the interactions related to code commits lack the type of direct interactivity present in those related to code review, we hypothesize that the *code review activity may be best suited to support impression formation.* Despite the differences in SNA metric values, we hypothesize that the code review and code commit interactions are complementary for impressions formation, because developers who work on the same module are more likely to review each others' code. While in this paper we did not study this hypothesis, we consider testing this hypothesis as a future goal.

Additionally, the characteristics of the four WikiMedia social networks are quite different, except for some similarities in the *mailing-list* and *bug interaction* networks. These different characteristics suggest that each social network provides a different perspective on interactions within an FOSS community. To fully understand the community interactions and the peer impression formation process, it is likely necessary to examine multiple perspectives rather than drawing a conclusion based only on one type of interaction.

The primary threats to validity for this are related to external validity. Due to the vast differences among FOSS projects, the results of this one case study are not generalizable to the entire FOSS domain. To build reliable empirical evidence, we need to study more FOSS projects. We carefully selected WikiMedia projects for this case study as those might be good representatives of community-driven FOSS projects. The results of this study can help to build research questions for future FOSS studies.

The contributions of this paper include:

- Identified the need for studying different interaction networks to understand FOSS community structure;
- Identified the scale-free network model to describe FOSS participants' interaction networks; and
- Identified the code review interaction as a likely method for impression formation among FOSS participants.

The future work for this research includes replicating this case study using more FOSS projects to verify the generalizability of this result.

Acknowledgment. This research is supported by US National Science Foundation grant 1322276.

References

1. Barabási, A.L., Albert, R.: Emergence of scaling in random networks. Science 286(5439), 509–512 (1999)
2. Bastian, M., Heymann, S., Jacomy, M.: Gephi: An open source software for exploring and manipulating networks. In: Proc. 3rd Int'l. AAAI Conf. on Weblogs and Social Media, San Jose, California (2009)
3. Bird, C., Gourley, A., Devanbu, P., Gertz, M., Swaminathan, A.: Mining email social networks. In: Proc. 3rd Int'l Wksp. on Mining Soft. Repositories, Shanghai, China, pp. 137–143 (2006)
4. Bird, C., Pattison, D., D'Souza, R., Filkov, V., Devanbu, P.: Latent social structure in open source projects. In: Proc. 16th ACM SIGSOFT Int'l Symp. on Foundations of Soft. Eng., Atlanta, Georgia, pp. 24–35 (2008)
5. Borgatti, S.P., Everett, M.G., Freeman, L.C.: Ucinet for windows: Software for social network analysis (2002)
6. Bosu, A., Carver, J.C.: Impact of peer code review on peer impression formation: A survey. In: Proc. 7th ACM/IEEE Int'l. Symp. on Empirical Soft. Eng. and Measurement, Baltimore, MD, USA, pp. 133–142 (2013)
7. Clauset, A., Shalizi, C., Newman, M.: Power-law distributions in empirical data. SIAM Review 51(4), 661–703 (2009)
8. Crowston, K., Howison, J.: The social structure of free and open source software development. First Monday 10(2-7) (2005)
9. Freeman, L.C.: Centrality in social networks conceptual clarification. Social Networks 1(3), 215–239 (1979)
10. Freeman, L.C.: The development of social network analysis: A study in the sociology of science, vol. 1. Empirical Press, Vancouver (2004)
11. Freeman, L.C., Roeder, D., Mulholland, R.R.: Centrality in social networks: II. experimental results. Social Networks 2(2), 119–141 (1980)
12. Fruchterman, T.M., Reingold, E.M.: Graph drawing by force-directed placement. Software: Practice and Experience 21(11), 1129–1164 (1991)
13. Herraiz, I., Perez, J.G.: Mailing list stats
14. Long, Y., Siau, K.: Social network structures in open source software development teams. Journal of Database Mgmt. 18(2), 25–40 (2007)
15. Martinez-Romo, J., Robles, G., Gonzalez-Barahona, J.M., Ortuño-Perez, M.: Using social network analysis techniques to study collaboration between a FLOSS community and a company. In: Russo, B., Damiani, E., Hissam, S., Lundell, B., Succi, G. (eds.) Open Source Development, Communities and Quality. IFIP, vol. 275, pp. 171–186. Springer, Boston (2008)
16. McKenna, K.Y., Bargh, J.A.: Plan 9 from cyberspace: The implications of the internet for personality and social psychology. Personality and Social Psychology Review 4(1), 57–75 (2000)
17. Oezbek, C., Prechelt, L., Thiel, F.: The onion has cancer: Some social network analysis visualizations of open source project communication. In: Proc. 3rd Int'l. Wksp. on Emerging Trends in Free/Libre/Open Source Soft. Research and Development, Cape Town, South Africa, pp. 5–10 (2010)
18. Watts, D.J., Strogatz, S.H.: Collective dynamics of 'small-world' networks. Nature 393(6684), 440–442 (1998)
19. Yang, X., Kula, R.G., Erika, C.C.A., Yoshida, N., Hamasaki, K., Fujiwara, K., Iida, H.: Understanding oss peer review roles in peer review social network (PeRSoN). In: Proc. 19th Asia-Pacific Soft. Eng. Conf., Hong Kong, pp. 709–712 (2012)