

Analysis of Coordination between Developers and Users in The Apache Community

Yasutaka Kamei¹, Shinsuke Matsumoto¹, Hirotaka Maeshima¹,
Yoji Onishi¹, Masao Ohira¹, and Ken-ichi Matsumoto¹

Graduate School of Information Science,
Nara Institute of Science and Technology
8916-5 Takayama, Ikoma, Nara 630-0192, Japan
+81-743-72-5312

{yasuta-k,shinsuke-m,hirotaka-m,yoji-o,masao,matumoto}@is.naist.jp

Abstract. Coordination is one of the keys for success of open source software (OSS) communities, because geographically distributed members need to collaborate on their work using communication tools (e.g. mailing list, bulletin board systems, bug tracking systems, and so on). In this paper, we investigated the informal social structure among developers and users. We analyzed two mailing lists of developers and users in the Apache community based on the betweenness centrality, one of the centrality measures proposed by Freeman. From the results of analysis, we have found that (1) participants with high betweenness coordinated activities between developers and users and (2) some of the participants would have been playing a role of a coordinator in the community for a long term.

1 Introduction

Today, open source software (OSS) is widely used by both individual users and users in administrative agencies and educational institutions, since many OSS products with high functionality and quality can be used for free. Generally, OSS is developed in OSS communities on the WWW, where developers and users discuss, share, and realize diverse, innovative ideas on OSS products under development through using electronic media such as mailing lists and bulletin board systems[3].

However, the community-based development process is occasionally stopped or remains stagnant due to some social factor (e.g. dissension among developers, downturn in users' demand, emergence of another superior OSS, and so forth). The cessation of OSS development means end-users cannot take continual, sufficient support from the community, such as bug fixes and functional enhancement. Therefore, system administrators in an organization need to carefully decide whether they should adopt OSS as an alternative for existing commercial software.

As OSS systems become critical information infrastructure in our society, many researchers and practitioners are very interested in understanding *why*

OSS communities sometimes can achieve a big success and vice versa, how participants in OSS communities are able to make the community sustainable, what we can do to foresee whether OSS communities will keep their activities in the future and so forth. Recent studies have been trying to reveal the process of OSS development and success factors in OSS communities[7, 9].

Toward understanding success factors of OSS communities, for instance, Howison et al. analyzed the informal social structure which is constructed from communication patterns among developers in OSS communities[6]. One of their findings was that a long-lived community does not dynamically change the structure of communications among developers[2, 6]. Raymond pointed out users' feedback to OSS products is also an important success factor[10]. Because feedbacks from users such as requests of new features and bug reports indicate extent of the demand for their products, responding to such the feedbacks affords developers to feel they are helping to fill social needs. That is, users play an important role to motivate developers to continuous OSS development.

As Raymond suggested, we also believe that feedbacks from users are much important as a success factor of OSS development. From this point of view, the prior study [6] which attempted to reveal the informal social structure in OSS communities, might be insufficient to understand relationships among developers and users. In this paper, we investigate the informal social structure among developers and users, using history data of communications in OSS communities (e.g. mailing lists).

In what follows, Section 2 introduces the related work. Section 3 explains Informal social structure in OSS communities, and Section 4 describes the details of analysis method. Section 5 provides our case study, and Section 6 discusses the result of analysis. Section 7 summarizes the paper and presents some future topics of research.

2 Related Work

Many studies on software development in OSS communities have been reported. Mockus et al.[9] investigated some assumptions on OSS development using CVS logs and bug reports, and revealed that only 4% of Apache developers contributed to 88% of added lines of code and 66% of fixed defects. [5] also reported similar results about community-driven OSS development. Although [5, 9] introduced the reality of OSS development from the perspective on software production by OSS developers, it did not suggest interactions among OSS developers (*how they collaborate each other and coordinate their activities*). In contrast with these studies, Jensen et al.[7] illustrated role migration and advancement processes in OSS communities including Apache community, based on qualitative and ethnographic methods. In this paper, we also analyze an aspect of OSS development processes in Apache community, especially focusing on coordination processes in OSS development.

There already exist some studies reporting on the coordination process in OSS communities. Yamauchi et al. described how geographically distributed developers coordinated their activities and how electronic media was used in the coordination, by analyzing mailing lists of FreeBSD Newconfig and GNU GCC community[11]. Howison et al. analyzed the outdegree centrality of developers in 5 OSS communities based on bug report data in a bug tracking system[6]. [6] showed the majority of communities had a single core developer for long periods of time and that larger communities do not change core developers, comparing with smaller communities. Bird et al. examined the correlation between centralities and Apache developers' contributions from the developers mailing list and the change history of source code[1]. The results of analysis indicated that developers communicating with many other developers more contributed to changes of source code. Above the three studies engaged in clarifying coordination and collaboration processes in OSS communities, particularly focusing on developers' activities. In this paper, we are interested in analyzing coordination between developers and users from the reason described previously.

3 Informal social structure in OSS communities

General OSS communities forms a kind of online community. Participants are geographically distributed and communicate each other through electronic media such as mailing lists and bulletin board systems (BBS). They can freely discuss ideas and exchange information on an equal footing, without the influence of seniority or one's social position. OSS communities often provide participants with multiple mailing lists and/or BBS, because diverse topics are discussed in the community, depending on roles of participants. So, most of OSS communities have several subgroups (e.g. developers using development mailing list) to encourage specific discussions.

The informal social structure with above the characteristics can be represented as the communication network in Figure 1. Here the network is defined by regarding each node as a sender of a message and an edge as a sender-receiver relationship. For instance, if participant A sends a message (M) to BBS and then participant B reply to M, A and B are a sender and receiver respectively and then they have a sender-receiver relationship. According to individual roles and purposes of participants in OSS communities, each participant joins one or more subgroups to communicates with other participants. In what follows, we call participants who have discussions only in a developers group as P_{dev} , participants who have discussions only in a users group as P_{user} , and participants who have discussions in both developers and users group as $P_{d \cap u}$.

In OSS communities, discussions in the developers group and the users group are independent each other by nature. Therefore, the community requires people who moderately coordinate the activities to keep activities in OSS communities energetic. We consider that the existence of a person who coordinate activi-

ties between developers and users (e.g. the center-black node in Figure 1) have important implications on OSS communities' activities as a whole.

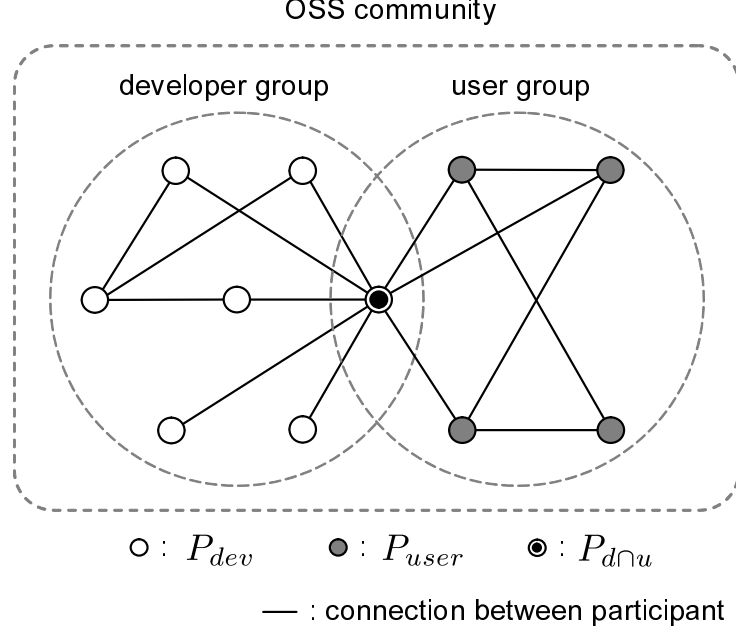


Fig. 1. Informal social structure of an OSS community

4 Analysis

4.1 Analysis of coordinators

In this paper, we suppose $P_{d \cap u}$ who discussed in both developers and users groups to be a coordinator who arranges interactions between developers and users. Especially we consider a coordinator would be a person who strongly intermediates between them. $P_{d \cap u}$ with the high degree of the intermediation could tightly connect developers and users, and would play a role of coordinating activities between developers and users. Howison et al have applied the outdegree centrality measure to the developers group in Apache, in order to identify the developers who provide a variety of information with other developers[6]. In this paper, we would like to analyze and identify the developers who strongly connects developers and users in Apache. We use the betweenness centrality, which is one of the centrality measures proposed by Freeman[4], as the degree

of intermediating between the both groups. Our analysis is composed of the followings.

Visualizing the informal social structure

Visualizing the informal social structure to understand the whole picture of the structure.

Calculating the betweenness centrality

Identifying $P_{d \cap u}$ with the high degree of the intermediation between developers and users by calculating the betweenness centrality for each actor in the network.

Reading contents of messages

Reading contents of messages which were posted by $P_{d \cap u}$ showing high betweenness, in order to confirm whether $P_{d \cap u}$ actually coordinates activities between the both groups, or not.

4.2 Betweenness centrality

The following describes the betweenness centrality measure proposed by Freeman [4]. Betweenness Centrality $C_{betweenness}(v_i)$ is formulated as below.

$$C_{betweenness}(v_i) = \frac{\sum_{j < k}^n \sum_{j < k}^n p_{jk}(v_i)}{\sum_{j < k}^n \sum_{j < k}^n p_{jk}} \quad (1)$$

where n shows the number of nodes in the network, p_{jk} shows the shortest path from v_j to v_k , $\sum_{j < k}^n \sum_{j < k}^n p_{jk}$ shows the number of paths from v_j to v_k , and $\sum_{j < k}^n \sum_{j < k}^n p_{jk}(v_i)$ shows the number of paths from v_j to v_k including v_i .

$C_{betweenness}(v_i)$ takes a value from 0 to 1, and a node with higher betweenness indicates the node which plays a role of an intermediary in the network. In OSS communities, we can assume a node with high betweenness centrality would be a participant who strongly connect between other participants. Thus, if a participant with high betweenness centrality suddenly drops off the project, other participants might become hard to communicate each other.

5 Case Study

This section describes our case study of the Apache HTTP Server community which has been developing a web server software product with the biggest market share.

5.1 Data Source

In the case study, we used the following archival source of data.

Developers mailing list

Anyone with an interest in working on Apache development can join this mailing list, which has been archived monthly since March, 1995. It is mainly used for discussions on development of Apache, including technical topics and bug fix.

Users mailing list

Anyone who needs support for the usage of Apache can join this mailing list, which has been archived monthly since November, 2001. It is mainly used for users support, including questions and answers about the installation of Apache and reports about error comments.

We analyzed the two mailing lists above for three months between before and after the latest major version of Apache (2.2.0) released.

5.2 Data Cleaning

Some of participants in the mailing lists may have several email addresses. So we need to clean the data to identify such the participants before our analysis. We have tried to identify the same person in the mailing lists according to the following criteria.

1. *A sender of messages with the same email address and different "Name"*: If the same email address are used by different "Name" senders such as "Alice" and "Bobby", we consider they are the same person.
2. *A sender of messages with the same name at "From" and partially the same address before at mark(@)*: If two email addresses such as chirs@domain1.com and chris@domain2.com are used by the same "Name" sender such as "Chris", we consider they are used by the same person.
3. *A sender of messages with the same name at "From" and different email addresses*: If two email addresses such as daniel@domain3.org and johnson@domain4.edu are used by the same "Name" sender such as "Daniel", we can judge they are used by the same person after confirming the body of messages (e.g. messages with the same signature).

In cleaning the data, we found 678 unique senders by applying the procedure 1 and 2 to the data. Only 5 of 678 senders match the procedure 3.

5.3 Result of analyses

Informal social structure We visualized the informal social structure in the Apache community based on the definition described in Section 3 in order to understand the whole picture of the community structure.

Figure 2 represents the network of the Apache community for the period of version 2.2.0 released. In Figure 2, developers P_{dev} , users P_{user} , and $P_{d \cap u}$ are positioned respectively at the left, right, and center. We can see there are $P_{d \cap u}$ with many edges between the both users and developers.

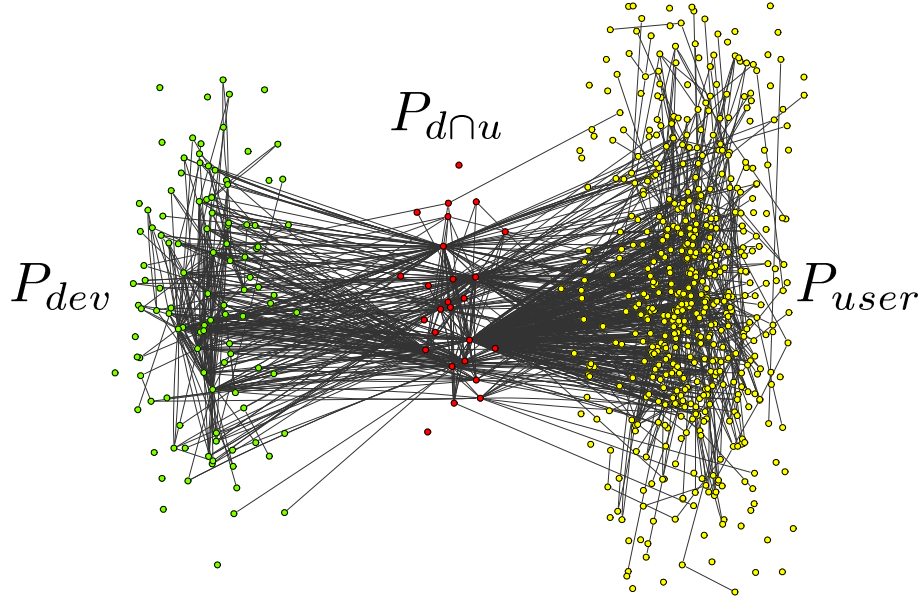


Fig. 2. Informal social structure in the Apache community

Betweenness centrality We calculated the betweenness centrality of each $P_{d \cap u}$ in the network to identify $P_{d \cap u}$ with the high degree of the intermediation between P_{dev} and P_{user} . Figure 3 shows the network of the top 5 $P_{d \cap u}$ of the betweenness centrality. Comparing to the two networks in Figure 2 and 3, we can see the top 5 $P_{d \cap u}$ intermediate between more than half of users and developers. Counting the number of edges of the top 5 $P_{d \cap u}$, we found that they communicated with 55 of 112 P_{dev} in the developers mailing list and with 249 of 540 P_{user} in the users mailing list.

Table 1. Statistics of top 5 $P_{d \cap u}$

	$P1_{d \cap u}$	$P2_{d \cap u}$	$P3_{d \cap u}$	$P4_{d \cap u}$	$P5_{d \cap u}$	median
Betweenness	0.179	0.044	0.043	0.022	0.019	0.001
Num. of email	592	193	261	127	62	17
Num. of degrees with developers	15	29	33	18	11	2
Num. of degrees with users	189	31	28	19	21	1

Table 1 indicates the statistics value of the top 5 $P_{d \cap u}$. Of the top 5 $P_{d \cap u}$, $P1_{d \cap u}$ with the highest betweenness has extremely a larger number of degrees

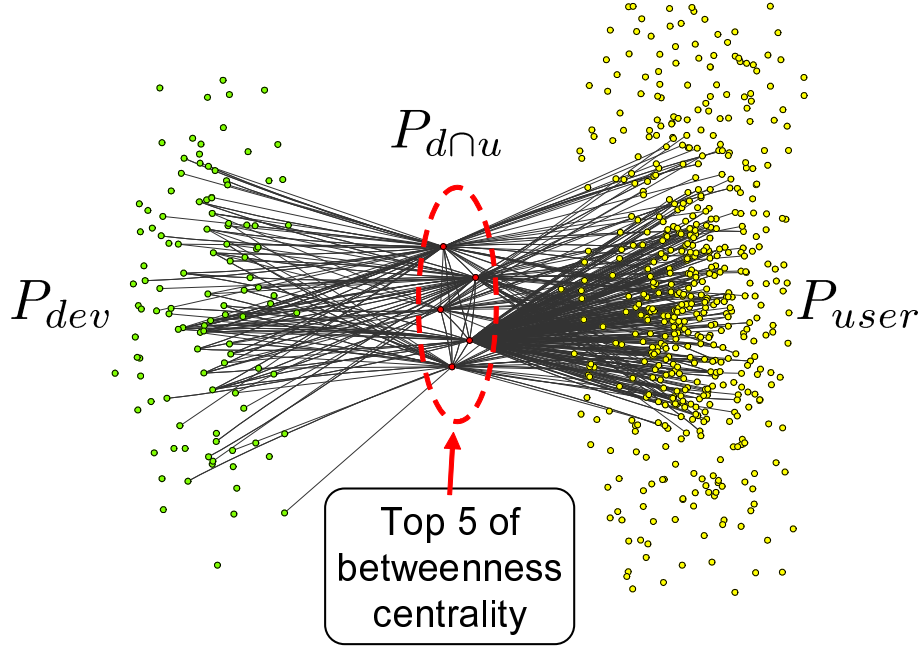


Fig. 3. Informal social structure of the top 5 $P_{d \cap u}$

with developers (P_d), comparing with that of the left $P_{d \cap u}$. The betweenness centrality of the top 5 $P_{d \cap u}$ is 10 times higher than the median, that means they intermediate between many developers and users.

Contents of messages We check the contents of messages which were posted by the top 5 $P_{d \cap u}$ to confirm whether they actually coordinated activities between the both P_d and P_{user} . Five experimenters have judged whether the messages relate to coordination actions of $P_{d \cap u}$. If more than three experimenters judge that a message implied the coordination actions, we decided it as a message relates to the coordination actions.

From the experimenters' review of the contents of messages, we have found that the top 5 $P_{d \cap u}$ with the high betweenness centrality coordinated activities between developers and users. Due to the page limitation of the paper, we only show a part of the reviews in Table 2, 3, 4.

Table 2 shows a kind of coordinative action, guide of discussions (users $\rightarrow$ developers). If developers-related topics were discussed in a users group and $P_{d \cap u}$ guided them to be discussed in a developer group, we considered such the action of $P_{d \cap u}$ as coordination. For instance, as the case in Table 2, it would be desirable to be discussed in a developers group, because a topic is about a

matter of implementation. In this situation, coordination is required for guiding the discussions to an appropriate place (developers group).

Table 2. Guide of discussions: users→developers

sender	dialog	notes
user	<i>I implemented a module of Apache on the Windows environment. I want to run it on Linux.</i>	The user is asking a question that is on the module development of Apache, to users groups.
$P1_{d \cap u}$	<i>It should be discussed on developers' ML because it is a matter of implementation.</i>	$P2_{d \cap u}$ is guiding the user to the developers group.

Table 3 shows a kind of coordinative action, information transfer (users→developers). If $P_{d \cap u}$ transferred information which only users had, to a developers group, we considered such the action of $P_{d \cap u}$ as coordination. For instance, as the case in 3, requests for new features and evaluations of products should be informed to developers for further development. In this situation, a required coordination is to inform users' feedbacks to a developers group.

Table 3. Information transfer: users→developers

sender	dialog	notes
developer	<i>Some versions seems not to be very popular.</i>	The developer is arguing in the developers group that some versions of Apache should be stopped to be in public because they are not popular among users.
$P3_{d \cap u}$	<i>I personally received emails from users regarding to their versions and then I knew the versions are still popular.</i>	$P3_{d \cap u}$ is telling the developers the fact that the versions mentioned by the developers are still popular to some extent. He is motivating developers to keep the development of Apache by informing the popularity of the Apache among users.

Table 4 shows a kind of coordinative action, a request for participation (developers→users). The shortage of members in one group means that members in not only the group but also the developers group have to have a bigger burden. For instance, the shortage of testers like the case in Table 4 lead to impose on not only current testers but also developers, because developers have to engage in both of the development and test of software. In this situation, a required coordination is to ask users to participate in a group with few members.

6 Discussion

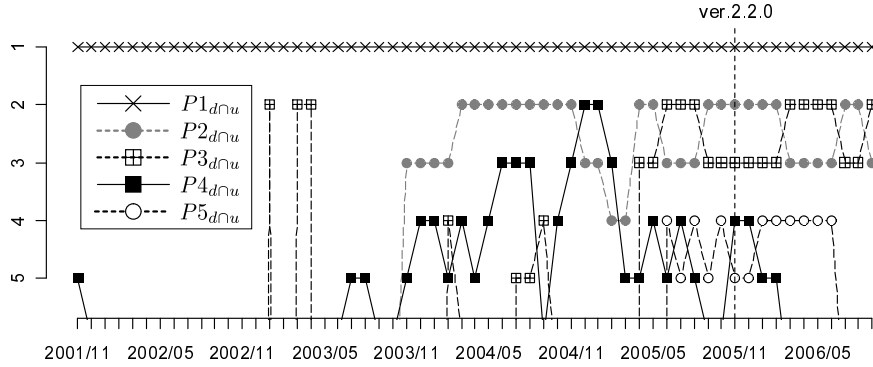
From the results of our analysis focusing on the betweenness centrality, we have confirmed that the existence of coordinators (the top 5 $P_{d \cap u}$ of the betweenness) who supports activities of developers and users. However, since the target

Table 4. Request for participation: developers→users

sender	dialog	notes
$P3_{d \cap u}$	<i>It is short of testers for some minor OS. We need your contributions as testers.</i>	Because of the shortage of testers who test the some versions of Apache compiled for some minor OS, $P3_{d \cap u}$ ask the users group to participate in the testers group. This coordination would contribute to the continuous development by the reduction of inordinate burden of developers.

of analysis in the section 5.3 was only the data for three months between before and after the latest major version of Apache (2.2.0) released, it is not sure if the top 5 $P_{d \cap u}$ support activities of developers and users only for the analysis period or also for the other periods. In this section, we look over the transition of the betweenness centrality of the top 5 $P_{d \cap u}$ for the period from November 2001 to September 2006.

Figure 4 shows the change of the rank order of the top 5 $P_{d \cap u}$ over time. The x-axis and y-axis indicate the time and the betweenness centrality respectively. We calculated the betweenness centrality by using the sliding time method[8]. Here, the centrality is calculated by sliding the three months analysis window month-by-month.

**Fig. 4.** Change of the rank order of the top 5 $P_{d \cap u}$ over time

From the Figure 4, we can see the existence of the particular $P_{d \cap u}$ who have been intermediating between developers and users for a long term. The betweenness centrality of $P1_{d \cap u}$ was the highest for the whole period. Though we did not checked the all messages posted by $P1_{d \cap u}$, we think that the coordinator with consistently higher betweenness such as $P1_{d \cap u}$ keeps on coordinating activities between developers and users. In the Linux community which is one of the most successful communities, Linus Torvalds, coordinator of the Linux Kernel community also has been contributing his efforts to the development of Linux Kernel since the start of the community (1991). We could think one of

the success factors of OSS communities is the existence of coordinators who facilitates and coordinates activities among people in OSS communities.

7 Conclusion and Future Work

In this paper, we investigated the informal social structure among developers and users, using two mailing lists of developers and users in the Apache community. The findings of the case study are following.

- participants with high betweenness coordinated activities between developers, and
- some of the participants would have been playing a role of a coordinator in the community for a long term.

Here, notice that our results of the analysis are applicable to the data from the mailing lists used in the Apache community. We need to investigate other dataset (e.g. history data in bug tracking system) and other communities to increase the appropriateness of the results. Furthermore, the data cleaning described in section might not be imperfect because we did not find a message sender who used different names at “From” and different email addresses. We need to sophisticate the data cleaning method in the near future.

Acknowledgments

This research is being conducted as a part of the EASE project in Comprehensive Development of e-Society Foundation Software program, and Stage Project, the Development of Next Generation IT Infrastructure, and Grant-in-aid for Scientific Research (B) 17300007, 2007 and for Young Scientists (B), 17700111, 2007, by the Ministry of Education, Culture, Sports, Science and Technology of Japan.

References

1. Christian Bird, Alex Gourley, Prem Devanbu, Michael Gertz, and Anand Swaminathan. Mining email social networks. In *Proceedings of the 2006 International Workshop on Mining Software Repositories (MSR’06)*, pp. 137–143, 2006.
2. Kevin Crowston and James Howison. The social structure of free and open source software development. *First Monday*, Vol. 10, No. 2, 2005.
3. Joseph Feller and Brian Fitzgerald. *Understanding Open Source Software Development*. Addison-Wesley, 2002.
4. Linton C. Freeman. Centrality in social networks: Conceptual clarification. *Social Networks*, Vol. 1, No. 3, pp. 215–239, 1979.

5. Daniel German and Audris Mockus. Automating the measurement of open source projects. In *Proceedings of the 3rd Workshop on Open Source Software Engineering*, pp. 63–67, Portland, Oregon, 2003.
6. James Howison, Keisuke Inoue, and Kevin Crowston. Social dynamics of free and open source team communications. In *Proceedings of the 2nd International Conference on Open Source Systems (OSS'06)*, pp. 319–330, 2006.
7. Chris Jensen and Walt Scacchi. Role migration and advancement processes in ossd projects: A comparative case study. In *Proceedings of the 29th International Conference on Software Engineering (ICSE'07)*, pp. 364–374, 2007.
8. Takeshi Kakimoto, Yasutaka Kamei, Masao Ohira, and Kenichi Matsumoto. Social network analysis on communications for knowledge collaboration in oss communities. In *Proceedings of the International Workshop on Supporting Knowledge Collaboration in Software Development (KCSD'06)*, pp. 35–41, September 2006.
9. Audris Mockus, Roy T Fielding, and James D Herbsleb. Two case studies of open source software development: Apache and mozilla. *ACM Transactions on Software Engineering and Methodology*, Vol. 11, No. 3, pp. 309–346, 2002.
10. Eric S. Raymond. *The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary*. O'Reilly and Associates, 1999.
11. Yutaka Yamauchi, Makoto Yokozawa, Takeshi Shinohara, and Toru Ishida. Collaboration with lean media: How open-source software succeeds. In *Proceedings of the 2000 Conference on Computer Supported Cooperative Work (CSCW'00)*, pp. 329–338, 2000.