Analysis of Coordination between Developers and Users in the Apache Community

Yasutaka Kamei[1], Shinsuke Matsumoto[1], Hirotaka Maeshima[1], Yoji Onishi[1], Masao Ohira[1], and Ken-ichi Matsumoto[1]

1 Graduate School of Information Science, Nara Institute of Science and Technology 8916-5 Takayama, Ikoma, Nara 630-0192, Japan
{yasuta-k,shinsuke-m,hirotaka-m,yoji-o,masao,matumoto}@is.naist.jp

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

1 Introduction

Today, open source software (OSS) is widely used by both individuals and users in administrative agencies and educational institutions, since many OSS products with high functionality and quality are 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 using such electronic media as mailing lists and bulletin board systems [3].

However, the community-based development process occasionally bogs down or remains stagnant due to such social factors as dissension among developers, downturns in user demand, emergence of a superior OSS, and so forth. The cessation of OSS development means end-users cannot obtain continual, sufficient support from the community for 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 are sometimes very successful and vice versa, how participants in OSS communities can create sustainable communities, what can be done to anticipate whether OSS communities will maintain their activities in the future* and so forth. Recent studies

have tried to reveal the process of OSS development and success factors in OSS communities [7, 9].

For understanding the success factors of OSS communities, for instance, Howison et al. analyzed the informal social structure constructed from communication patterns among developers in OSS communities [6]. They found that a long-lived community does not dynamically change the structure of communications among developers [2, 6]. Raymond pointed out that user feedback to OSS products is also an important success factor [10]. Because such feedback from users as requests for new features and bug reports indicates the demand for their products, responding to such feedback allows developers to address social needs. That is, users play an important role in motivating developers to continuous OSS development.

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

The rest of our paper is organized as follows. Section 2 introduces related work. Section 3 explains the informal social structure in OSS communities, and Section 4 describes the details of our analysis method. Section 5 provides our case study, and Section 6 discusses the analysis results. Section 7 summarizes our 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 assumptions of OSS development using CVS logs and bug reports and revealed that a mere 4% of Apache developers contributed to 88% of the added lines of code and 66% of the fixed defects. [5] also reported similar results about community-driven OSS development. Although [5, 9] introduced the reality of OSS development from the perspective of software production by OSS developers, they did not suggest interactions among OSS developers (how they collaborate with 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 the Apache community, based on qualitative and ethnographic methods. In this paper, we also analyze an aspect of OSS development processes in Apache communities by focusing on coordination processes in OSS development.

Studies have already reported the coordination process in OSS communities. Yamauchi et al. described how geographically distributed developers coordinated their activities and how electronic media were used in the coordination by analyzing the mailing lists of FreeBSD Newconfig and GNU GCC community [11]. Howison et al. analyzed the outdegree centrality of developers in five OSS communities based on bug report data in a bug tracking system [6]. [6] showed that most communities had

a single core developer for long periods of time and that larger communities do not change core developers compared with smaller communities. Bird et al. examined the correlation between centralities and Apache developer contributions from developer mailing lists and the changed history of source code [1]. The analysis results indicated that developers communicating with many other developers contributed more to source code changes. The above three studies clarified the coordination and collaboration processes in OSS communities by focusing on developer activities. In this paper, we analyze the coordination between developers and users based on the above described reasons.

3 Informal social structure in OSS communities

General OSS communities form a kind of online community in which participants are geographically distributed and communicate with 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 social position. OSS communities often provide participants with multiple mailing lists and/or BBS because diverse topics are discussed in the community, depending on the roles of participants. Most OSS communities have several subgroups (e.g., developers using the development mailing list) to encourage specific discussions.

The informal social structure with the above characteristics is represented as a 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 message (M) to BBS and then participant B replies to M, A and B are the sender and receiver, respectively, creating a sender-receiver relationship. Based on the individual roles and purposes of the participants in OSS communities, each participant joins one or more subgroups to communicate with other participants. Below, we call participants who only join discussions in a developer group P_{dev}, participants who only have discussions in user group P_{user}, and participants who join discussions in both the developer and user groups $P_{d \cap u}$.

In OSS communities, discussions in the developer and user groups are independent of each other by nature. Therefore, the community requires people who moderately coordinate the activities to maintain their momentum in the OSS communities. We consider that the existence of a person who coordinates the activities between developers and users (e.g., center-black node in Figure 1) has important implications for OSS community activities as a whole.

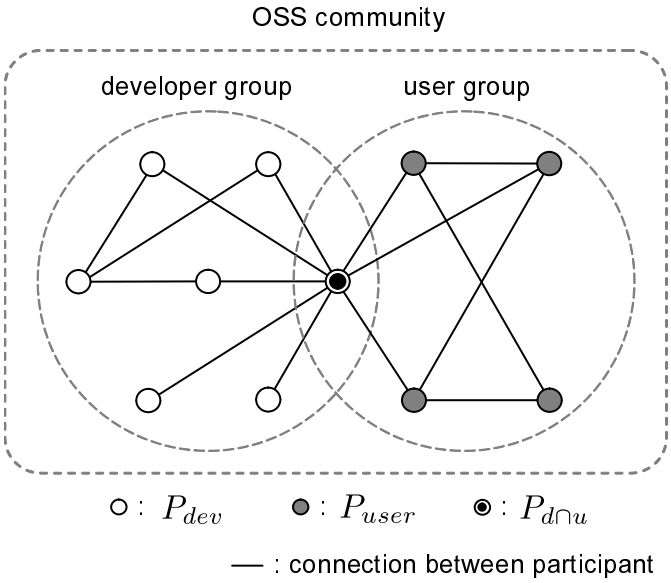

Fig. 1. Informal social structure of OSS community

4 Analysis

4.1 Analysis of coordinators

In this paper, we suppose $P_{d \cap u}$, which was discussed in both the developer and user groups, as a coordinator who arranges interactions between developers and users. We focus on a coordinator as a person who actively mediates between them. $P_{d \cap u}$ with a high degree of intermediation tightly connects developers and users and coordinates activities between developers and users. Howison et al. applied the outdegree centrality measure to the developer group in Apache to identify developers who provide a variety of information with other developers [6]. In this paper, we analyze and identify the developers who strongly connect developers and users in Apache. We use betweenness centrality, one of the centrality measures proposed by Freeman [4], as the degree of intermediation between both groups. Our analysis is composed of the following.

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 a high degree of intermediation between developers and users by calculating the betweenness centrality for each actor in the network.

Reading contents of messages

Reading the contents of messages posted by $P_{d \cap u}$ that show high betweenness to confirm whether $P_{d \cap u}$ actually coordinates activities between both groups.

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.

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

, where n shows the number of nodes in the network, p_{jk} shows the shortest path from v_j to v_k, $\sum^{n}\sum_{j<k}^{n} p_{jk}$ shows the number of paths from v_j to v_k, and $\sum^{n}\sum_{j<k}^{n} p_{jk}(v_i)$ 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 a higher betweenness indicates a node that acts as an intermediator in the network. In OSS communities, we assume a node with high betweenness centrality is a participant who strongly connects other participants. Thus, if a participant with high betweenness centrality suddenly leaves the project, other participants might have trouble communicating with each other.

5 Case Study

This section describes our case study of the Apache HTTP Server community that 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 data sources.

Developer mailing lists

Anyone interested in working on Apache development can join this mailing list that has been archived monthly since March 1995. It is mainly used for discussions on the development of Apache, including technical topics and bug fixes.

User mailing lists

Anyone who needs support to use Apache can join this mailing list that has been archived monthly since November 2001. It is mainly used for user support, including questions and answers about Apache installation and reports about error comments.

We analyzed the above two mailing lists for three months before and after the release of the latest major version of Apache (2.2.0). We consider the person who only sends e-mails to the developer mailing lists as P_{dev}, the person who only sends emails to the user mailing lists only as P_{user} and the person who sends emails to both of them as $P_{d \cap u}$.

5.2 Data cleaning

Since some participants in the mailing lists may have several email addresses, we need to clean the data to identify such participants. We identified the same person in the mailing lists based on the following steps.

Step 1. *A sender of messages with the same email address and ent "Name"*: If the same email addresses are used by different "Name" senders such as "Alice" and "Bobby," we consider them the same person.

Step 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 chris@domain1.com and chris@domain2.com are used by the same "Name" sender such as "Chris," we consider them the same person.

Step 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 judge they are used by the same person after confirming the body of the messages (e.g., messages with the same signature).

In cleaning the data, we found 740 unique senders by applying step 1 to the data, 683 unique senders by applying step 2 to the data after step 1, and 678 unique senders by applying step 3 to the data after step 2.

5.3 Analysis results

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

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

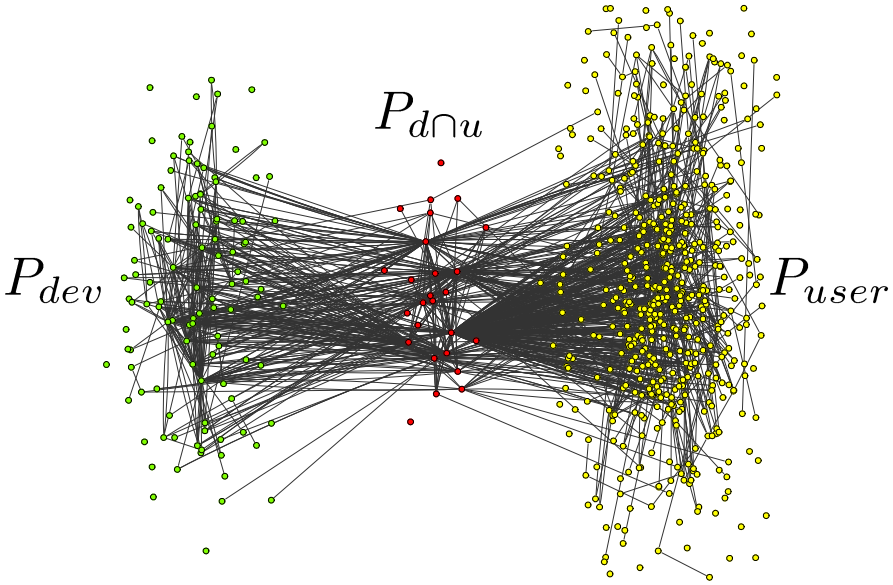

Fig. 2. Informal social structure in Apache community

Betweenness centrality We calculated the betweenness centrality of each $P_{d\cap u}$ in the network to identify the $P_{d\cap u}$ with high degree of intermediation between P_{dev} and P_{user}. Figure 3 shows the network of the top 5 $P_{d\cap u}$ of betweenness centrality. Comparing the two networks in Figures 2 and 3, we can see the top 5 $P_{d\cap u}$ intermediates between more than half of the users and developers. Counting the number of edges of the top 5 $P_{d\cap u}$, they communicated with 55 of 112 P_{dev} in the developer mailing list and with 249 of 540 P_{user} in the user 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 emails	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 statistical values of the top 5 $P_{d\cap u}$. $P1_{d\cap u}$ with the highest betweenness has an extremely large number of degrees with developers (P_{dev}), compared with the left $P_{d\cap u}$. The betweenness centrality of the top 5 $P_{d\cap u}$ is 10 times

higher than the median, meaning they act as intermediates between many developers and users.

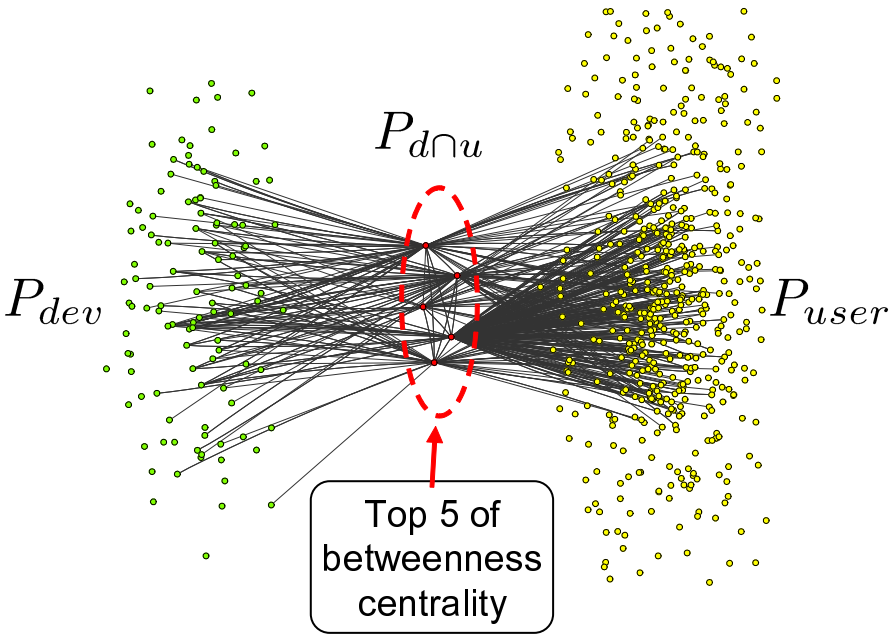

Fig. 3. Informal social structure of top 5 $P_{d \cap u}$

Content of messages We checked the contents of messages posted by the top 5 $P_{d \cap u}$ to confirm whether they actually coordinated activities between both P_{dev} and P_{user}. Five experimenters judged whether the messages related to the coordination actions of $P_{d \cap u}$. If more than three experimenters judged that a message implied coordination actions, we decided the message is related to coordination actions.

From experimenter reviews of the content of messages, we found that the top 5 $P_{d \cap u}$ with high betweenness centrality coordinated activities between developers and users. Due to space limitations in this paper, we can only show part of the reviews in Tables 2, 3, and 4.

Table 2 shows a kind of coordinative action, which is a guide of discussions (users→developers). If developer-related topics were discussed in a user group and $P_{d \cap u}$ guided them to be discussed in a developer group, we considered such action of $P_{d \cap u}$ coordination. For instance, as in Table 2, this should be discussed in a developer group, because the topic concerns implementation. In this situation, coordination is required to guide the discussions to an appropriate place (developer group).

Table 2. Guide of discussions: users→developers

sender	dialog	notes
user	*I implemented a module of Apache in a Windows environment. I want to run it on Linux.*	The user is asking a user group about the module development of Apache.
$P1_{d \cap u}$	*This should be discussed at developer ML because it is an implementation matter.*	$P1_{d \cap u}$ is guiding the user to the developer group.

Table 3 shows a kind of coordinative action, which is information transfer (users →developers). If $P_{d \cap u}$ transferred information to a developer group that only users had, we considered such action of $P_{d \cap u}$ coordination. For instance, as in Figure 3, requests for new features and product evaluations should be conveyed to developers for further development. In this situation, coordination is required to provide user feedback to a developer group.

Table 3. Information transfer: users→developers

sender	dialog	notes
developer	*Some versions do not seem very popular.*	The developer is suggesting in the developer group that some versions of Apache should be stopped from being made public because they are not popular.
$P3_{d \cap u}$	*Since I have personally received emails from users regarding their versions, I think the versions are still popular.*	$P3_{d \cap u}$ is telling the developers that the versions mentioned by the developers are still popular. He is motivating the developers to continue Apache development by describing its popularity among users.

Table 4 shows a kind of coordinative action, which is a request for participation (developers→users). The shortage of members in one group means that members not only in the group but also the developer group face bigger burdens. For instance, the shortage of testers, as in Table 4, imposes not only on current testers but also on developers because they have to develop and test the software. In this situation, coordination is required to ask users to participate in a group with few members.

Table 4. Request for participation: developers→users

sender	dialog	Notes
$P3_{d \cap u}$	*It is short of testers for some minor OS. We need your contributions as testers.*	Due to the shortage of testers for some versions of Apache compiled for some minor OS, $P3_{d \cap u}$ asks the user group to participate in the tester group. Such coordination would contribute to the development by reducing the burden of developers.

6 Discussion

From the analysis results focusing on the betweenness centrality, we confirmed the existence of coordinators (the top 5 $P_{d\cap u}$ of the betweenness) who support the activities of developers and users. However, since the analysis target in Section 5.3 was only the data for the three months before and after the latest major version of Apache (2.2.0) was released, it remains unclear whether the top 5 $P_{d\cap u}$ only supported the activities of developers and users for the analysis period or also for other periods. In this section, we analyze the transition of the betweenness centrality of the top 5 $P_{d\cap u}$ from November 2001 to September 2006.

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

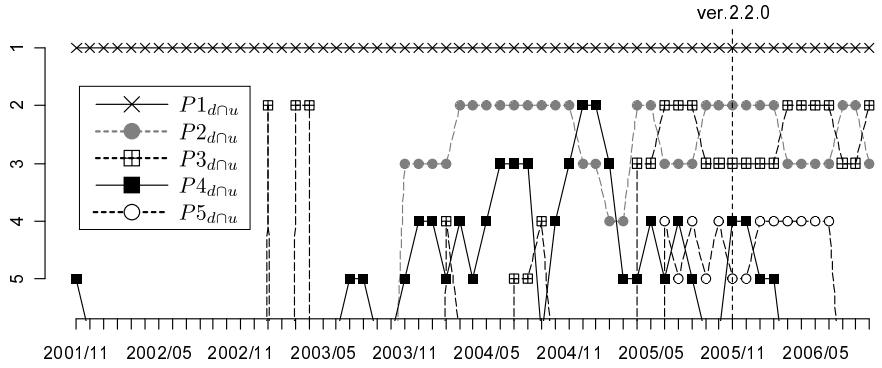

Fig. 3. Informal social structure of top 5 $P_{d\cap u}$

Figure 4 shows the existence of particular $P_{d\cap u}$ who have been intermediating between developers and users for a long time. The betweenness centrality of $P1_{d\cap u}$ was the highest for the whole period. Though we did not check all messages posted by $P1_{d\cap u}$, we believe that coordinators with consistently higher betweenness such as $P1_{d\cap u}$ continues coordinating activities between developers and users. In the Linux community, one of the most successful communities, Linus Torvalds, coordinator of the Linux Kernel community, has been contributing to its development since its start in 1991. One of the success factors of the OSS communities is the existence of coordinators who facilitate and coordinate activities among the members of 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 following are the findings of our case study.

- participants with high betweenness coordinated activities between developers, and
- some participants have been functioning as coordinators in the community for a long time.

Here, note that our analysis results are applicable to the data from the mailing lists used in the Apache community. We need to investigate other datasets (e.g., history data in bug tracking systems) and other communities to increase the appropriateness of our results. Furthermore, the data cleaning described in Section might be imperfect because we did not find any message senders who used different names at "From" and different email addresses. We need to increase the sophistication of the data cleaning method in the near future.

Acknowledgments

This research was conducted as part of the EASE project in the Comprehensive Development of e-Society Foundation Software program, and the 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.