

Temporal Changes of the Openness of an OSS community: A Case study of the Apache HTTP Server Community

Mizuki Yamamoto, Masao Ohira, Yasutaka Kamei,
Shinsuke Matsumoto and Ken-ichi Matsumoto

Graduate School of Information Science,
Nara Institute of Science and Technology
8916-5 Takayama, Ikoma, Nara 630-0192, Japan
+81-743-72-5312
{mizuki-y, masao, yasuta-k, shinsuke-m, matumoto}@is.naist.jp

Abstract. Today most OSS products are developed according to the “bazaar” style of software development. Certain products (e.g., Linux, Apache, PostgreSQL) favorably compare with proprietary software products and are widely prevalent in our daily life. On the contrary, a large number of OSS products and communities result in failure or stagnation due to various factors. This implies that “bazaar-like” software development might not be a critical success factor of OSS products and communities. What is the difference between successful and non-successful “bazaar-like” software development? To answer this question, in this paper we develop a hypothesis, *the condition of “bazaar” in a successful OSS community with a long history changes over time in evolving to adapt itself and its products to the rapidly changing external environment* and try to test the hypothesis by conducting a case study of the Apache HTTP Server community which is a representative of “bazaar-like” software development communities. From the results of our analysis, we have found that in the first half of the analysis period (1997-2002), the community tended to accept non-core developers’ opinions (patches) in the same way as core developers and emphasized to evolve the products, while in the last half of the analysis period (2003-2007) the community tended to place much value on providing quality services and the community has shifted the condition from fresh and/or vital to stable or mature.

1. Introduction

As globally distributed software development becomes more common in software development companies, the development style of open source software (OSS) has been drawing more attention from researchers and practitioners [6, 7, 8, 12]. Typical OSS such as Linux and Apache are traditionally developed in a geographically distributed environment. A better understanding of OSS development would be helpful for distributed software development in a software development company.

The development style of OSS is widely recognized as “bazaar-like” software development which has been conceptualized by Eric Raymond [11]. In “bazaar-like”

software development, anyone can participate in a community where an OSS product is collaboratively developed through the Internet. That is, “bazaar-like” software development is used as a metaphor to represent the openness of an OSS development and community to the public, assimilating the concept of a bazaar where people can choose and buy what they want. Using communication media (e.g., mailing lists and bulletin boards) and development support tools (e.g., configuration management systems and bug tracking systems), participants including not only developers but also users share, discuss, and implement a variety of ideas with each other, and then release OSS products to the public. In the form of participation in software development, OSS development differs substantially from traditional proprietary software development in a company, which is called “cathedral-like” software development.

Most current OSS products are developed according to the “bazaar” style of software development. Certain products (e.g., Linux, Apache, PostgreSQL) favorably compare with proprietary software products and are widely prevalent in our daily life. So, we consider that they play an important role as a part of social infrastructure. While socially successful OSS products exist, a large number of OSS products and communities result in failure or stagnation [10] due to various factors, (e.g., discord among participants, low market demand, low number of downloads, appearance of superior kinds of the same products, and so forth). This implies that “bazaar-like” software development might not be a critical success factor of OSS products and communities. *If so, how could certain products and communities such as Linux and Apache achieve great success and survive in a free competitive market for more than a decade?* Our study is motivated to answer this question.

In order to answer the research question above, we first suspect that OSS products and communities keep coevolving to adapt to a continuously and rapidly changing external environment such as market needs, competitors, and appearance of new technologies. In this paper we develop the hypothesis that *the condition of “bazaar” in a successful OSS community with a long history changes over time in evolving to adapt itself and its products to the rapidly changing external environment* and try to test the hypothesis by conducting a case study of the Apache HTTP Server community, which is a representative of “bazaar-like” software development communities. From the case study, we would like to provide a new perspective on a key factor of OSS development. In what follows, we introduce related work on the “bazaar” style OSS development and organization models of OSS communities in Section 2 and present our hypotheses in Section 3. Section 4 describes our case study of the Apache HTTP Server community and then Section 5 shows the result of the case study. We discuss lessons learned from the case study in Section 6. Section 7 summarizes the paper and states our future work.

2. Related work

The literature on OSS has made considerable efforts to better understand practical open software development [2, 3, 13]. Mockus et al. [8] investigated various claims on the nature of OSS development through the case study of Apache and Mozilla. One of the findings in [8] is that both the Apache and Mozilla communities have a small number of core developers (10 to 15 people) who control the code base and create approximately 80% or more of the new functionality. German et al. [4] also reported similar results for the Ximian project. These facts provide us a clue that OSS communities might not be supported by a fully open, liberalized “bazaar” but rather by some sort of centralized mechanism (e.g., authority).

An organizational structure of an OSS community has been illustrated as an “Onion” model [1, 14], which has become a widely received view of participants’ roles in an OSS community. If OSS products are created according to fully open “bazaar-like” software development, a process of role changes between the roles (promotion or demotion) in an OSS community must be more flexible than that in a business organization. To understand the actual process of role migration and advancement in OSS communities, Jensen et al. carefully observed three OSS communities (Mozilla.org, Apache.org, and NetBeans.org) and confirmed the existence of a complex process of role migration and advancement, which cannot be captured by simple, onion-like multiple tracks of roles modeled for an organizational hierarchy in an OSS community. They also confirmed that the important roles in a community (e.g., project manager) are not already available to community members from the general public.

Nakakoji et al. [9] classified the evolution patterns of OSS products and communities into three types (exploration-oriented, utility-oriented, and service-oriented) and discussed relationships of co-evolution among OSS products and communities.

Exploration-Oriented and Utility-Oriented OSS projects experience rapid evolution, mostly in linear fashion [5]. As the projects mature into Service-Oriented ones, the speed of evolution will slow down to a stable growth. A new round of rapid evolution will start again if the stable OSS system inspires new ideas or new requirements, giving birth to New Exploration-Oriented or Utility-Oriented OSS projects, which will again mature into new Service-Oriented ones. This process is similar to the biological evolutionary process...

Inspired by the studies above, we were motivated to investigate if OSS products and communities comply with “bazaar-like” software development and if the management policy of community participation changes over time as products mature. Especially, since the previous studies did not strongly focus on time-series analysis of quantitative data, we would like to analyze it and provide empirical evidence for new understanding of open source-style software development.

3. Temporal Changes in “Bazaar”

This section describes our hypotheses on temporal changes in “bazaar-like” software development.

3.1. Openness of an OSS Community

The basic hypothesis in this paper is that *a condition of “bazaar” in a successful OSS community with a long history changes over time in evolving to adapt itself and its products to the rapidly changing external environment.* The previous studies described before seem to support the hypothesis to some extent because they imply that OSS products are not developed in a fully open, liberalized “bazaar” but rather under control of a centralized system (decision making by core members). So, what is the openness of “bazaar-like” software development in an OSS community? We consider here that the openness of “bazaar-like” software development is determined by ease of community participation and acceptability of contributions from non-core members. The ease of community participation would be ensured in most OSS communities because anybody can subscribe to mailing lists, report bugs, and send a patch to developers in most OSS communities. By contrast, the acceptability of contributions from non-core members might not be guaranteed, because all of the users’ feedback, bug reports, and patches are not necessarily accepted and used to change the quality of OSS products. Therefore, we use and analyze the acceptability as a measure for ascertaining the openness of OSS communities (i.e., the condition of “bazaar”).

3.2. Hypotheses

We would like to test the basic hypothesis by using time-series analysis of the acceptability of contributions from both core and non-core members in OSS communities. To do so, we developed more concrete hypotheses as follows.

- [H1] *A community with a (small/big) difference in the acceptability between core and non-core members will (acquire/lose) non-core members over time.*
- [H2] *A community with a (small/big) difference in the acceptability between core and non-core members will (increase/decrease) in number of contributions from non-core members over time.*

As [4, 8] indicated, a small number of core members greatly contribute their efforts to create quality products. The acceptability of contributions from core members would be consequently higher than from non-core members. On the contrary, the acceptability of contributions from non-core members would depend on results of decision making among core members. If core members ignore a contribution (e.g., bug report) from non-core members, non-core members are not likely to continue to devote their efforts to the community. We consider that a

community where core members treat contributions from non-core members in the same way as core members will attract non-core members and motivate them to contribute to the community. [H1] and [H2] are based on this consideration of community participation and contributions from non-core members.

In this paper the acceptability is defined by the number of contributions that are reviewed by core members. In particular, we focus on patches posted to a developer mailing list as contributions and derive the acceptability by counting the number of patches reviewed by core developers. For instance, if all the patches posted by core and non-core developers are reviewed as equals, the difference in the acceptability between core and non-core developers will be small, which implies that the community would regard contributions from core and non-core members as equally important and therefore the condition of community would be open. If only the patches posted by core developers are reviewed and the patches posted by non-core developers are ignored, the difference in the acceptability between core and non-core developers will be large, which implies that the condition of the community would not be open to the public (i.e., non-core developers).

4. 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. We first briefly introduce the Apache HTTP Server community and then describe the target data source for analysis, and explain metrics measured from the data source. Finally, we provide the criteria to define a core developer and non-core developer.

4.1. Apache HTTP Server community

The Apache HTTP Server community was built by some of developers who implemented patches for the earliest web server (called “NCSA HTTPd”). The developers extended functions and fixed bugs through communications with e-mails and then released version 1.0 of the Apache HTTP server in 1996. The community has been continued up to today and released versions 1.3, 2.0 and 2.2.

The same as many other OSS products, the Apache HTTP server is also developed according to the “bazaar” style of software development. Anyone can participate in the community and help the improvement of the product through the Internet. For instance, the community allows any developer to post a patch to a mailing list at anytime. The patch is discussed by other developers and then applied to the Apache HTTP server by the developers who have commit permissions to the repository for software documents and source codes.

In the Apache project, the commit permissions are shared by the developers who contribute to improving the product for many years, such as software document maintenance and bug fix. These developers are called “core-developers”. On the

other hand, “non-core developers” do not have the commit permissions due to a short duration of their participation, small contributions, or other reasons. However, the non-core developers still can post a patch to a mailing list and discuss other patches as can the core-developers since the development style in the Apache project is “bazaar-like” software development.

4.2. Data Sources

In this paper, we chose the developer’s mailing list in the Apache community as our target data source. It has been archived monthly since March 1995. We analyzed the mailing list for 11 years from Jan. 1, 1997 to Dec. 31, 2007. The number of message logs during the period was 93,669. The developer’s mailing list is mainly used for discussing changes to the source codes of Apache and related issues including technical topics and bug fixes. Anyone interested in working on Apache development can join this mailing list.

4.3. Classification of a core and non-core developer

In this paper, we classify each developer into a core developer or a non-core developer year by year. A non-core developer in one year can be classified as a core developer in another (later) year, since a non-core developer may be promoted to a core developer. For the classification of developers in the Apache community, in this paper we establish criteria as follows.

If there is a developer’s commit log in the CVS repository, then we classified the developer as a core developer, since core developers have commit permission to the repository and a commit log with a committer name is generated when committing files to the repository. Therefore, if there is at least one developer’s commit log in one year, the developer (i.e., committer) is classified as a core developer in that year. Developers that are not core developers are classified as non-core developers. If there is a developer’s commit log in 1995 and not in 1996, then the developer is classified as a core developer in 1995 and a non-core developer in 1996.

4.4. Metrics measurement

In order to test the two hypotheses described in Subsection 3.2, we measured the following metrics from the development mailing list every year: (1) the number of core and non-core developers (i.e., the number of email senders), (2) the number of contributions from core and non-core members (i.e., the number of patches), and (3) the acceptability of contributions from core and non-core members (i.e., the rate of patches reviewed by other developers).

The number of developers is counted from the number of posted emails in the development mailing list. The developers are classified into core and non-core developers, based on the criteria described in Subsection 4.3. The number of contributions is derived by extraction of emails with a string [PATCH] in the subject of an email. This rule relies on a previous study of Rigby et al. [12], which reported that developers in the Apache community posted attached patches with a string [PATCH] in the subject of an email. Emails with patches of the total number of 93,669 emails in total were 3,061. Furthermore, if other developers respond to a posted patch in the mailing list, the patch is regarded as reviewed. [12] randomly sampled 100 replied emails with patches and confirmed that non-reviewed emails were less than 5%.

5. Result

5.1. Number of developers

Fig. 1 shows changes in the number of core developers and non-core developers. The x-axis shows time and the y-axis shows the number of developers. The number of non-core developers rapidly increased until 2002 and decreased until 2007. On the other hand, the number of core developers increased steadily (15 to 50 developers). This result can be seen as the total growth of the Apache community.

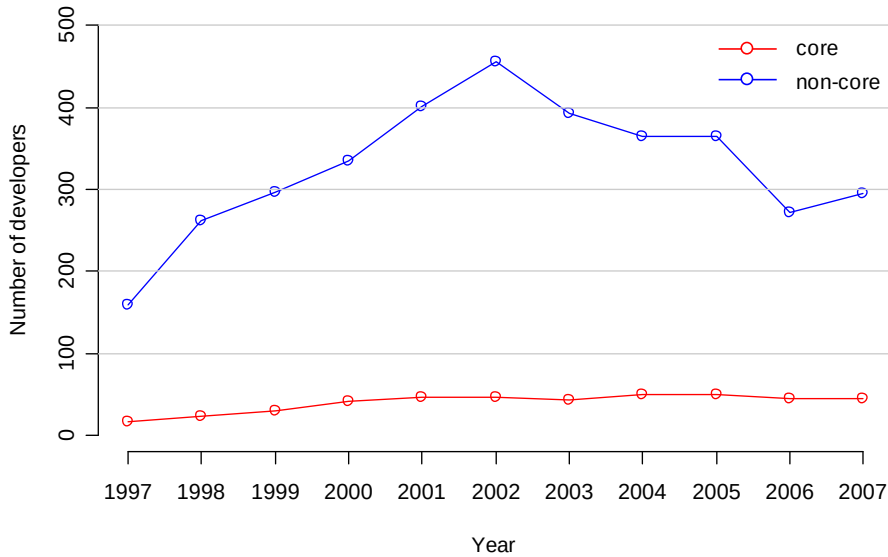


Fig. 1. Changes in number of developers

5.2. Number of patch mails

Fig. 2 shows the number of patch mails sent by core developers and non-core developers. The x-axis shows time and the y-axis shows the number of patch mails. The total number of patch mails during the analysis period was 3061. The core developers sent 1820 (about 60%) of the total number of patch mails.

In the early periods (1997-1998), most patch mails (about 86%) were sent by about 20 core developers. This result indicates most defects were detected and corrected by the core developers in the early periods. In the middle periods (1999-2002), non-core developers began to send many patch mails, while the number of patch mails sent by core developers significantly decreased. The total number of patch mails has decreased since 2002, and eventually only 52 were sent in 2007.

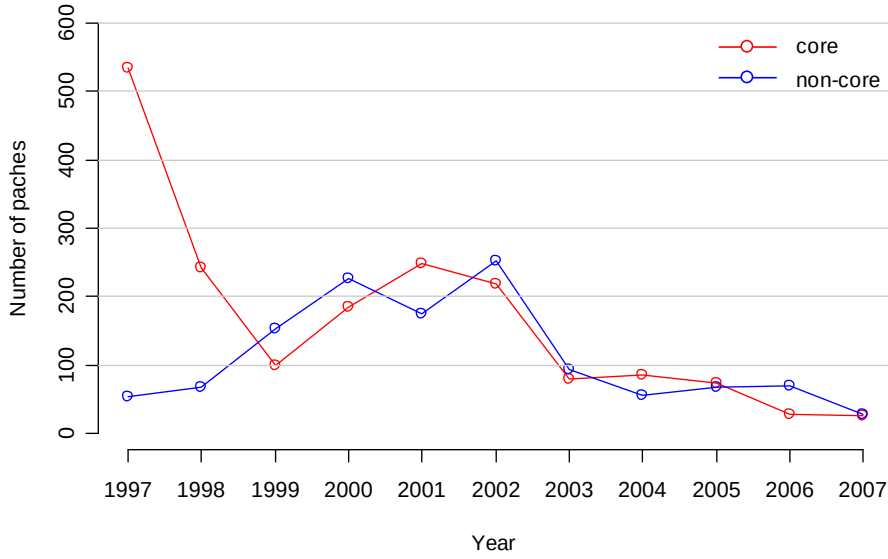


Fig. 2. Changes in number of patches

5.3. Review rate

We now focus on the review rate (percentage of patch mails that reviewed by other developers). Fig. 3 shows the review rates of core developers and non-core developers. The values written at the top of Fig. 3 show the p-value calculated by chi-square test. Bold letters indicate statistically significant ($P < 0.05$) differences between the review rate of core developers and non-core developers.

In 1997, the review ratio of core developers was notably higher (80.0%) than other periods, although a large number of patch mails (over 500) were sent by core developers. In the later periods (2002-2007), there were significant differences between the review rate of core and non-core developers. The review ratio of non-core was significantly low from 2004 to 2006. In particular, over the half of the patch mails sent by non-core developers had received no response in 2006. As a result, the review rate of non-core developer was the lowest in 2006.

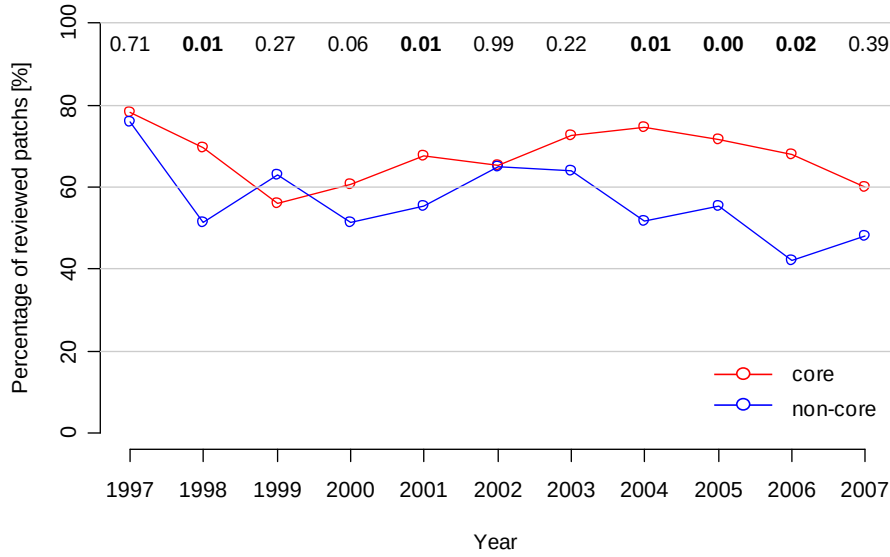


Fig. 3. Changes in percentage of reviewed patches

5.4. Hypotheses testing

In this subsection, we test our two hypotheses described in section 3.

[H1] *A community with a (small/big) difference in the acceptability between core and non-core members will (acquire/lose) non-core members over time.*

In Fig. 3, the differences in the review ratio between core and non-core developers are relatively small until around 2002 and become large from 2003. While Fig. 1 shows the rapid increase of non-core developers until 2002, it also showed the decrease of non-core developers. We can consider that the hypothesis [H1] is supported.

- [H2] *A community with a (small/big) difference in the acceptability between core and non-core members will (increase/decrease) in number of contributions from non core members*

In Fig. 2, the number of patch mails from non-core developers increased until 2002 and decreased from 2002. As with [H1], we can conclude that the hypothesis [H2] is supported.

6. Discussions

The results of our analysis supported our hypotheses [H1] *A community with a (small/big) difference of the acceptability between core and noncore members will (acquire/lose) noncore members over time* and [H2] *A community with a (small/big) difference of the acceptability between core and non-core members will (increase/decrease) in number of contributions from non core members*.

In the first half of the analysis period (1997-2002), the difference of the acceptability between core and non-core developers in the Apache community can be said as relatively small until 2002, since the difference of the review rate was only observed two times during 6 years (1997-2002). The result suggests that the Apache community equally treated each contribution sent by core and non-core developers. In other words, there was a positive stance toward widely taking contributions not only from core developers but also from non-core developers in this period, that is, the condition of the community was open. Therefore, we can conclude that the community attracted non-core developers and encouraged them to submit many contributions (patches) as shown in Fig. 1 and Fig. 2.

In the last half of the analysis period (2003-2007), however, the difference of the acceptability was relatively big. The difference of the review rate could be confirmed three times during 5 years. We consider that the community in this period lost the positive attitude as described above and the openness of the community also decreased. These factors would strongly affect the decreasing number of non-core developers and patches from non-core developers. Summarizing these results, we can conclude that the Apache community changes own openness over time.

Our empirical findings also can support the model of evolution patterns of OSS products and communities presented by Nakakoji et al. [9]. According to the classification of [9], the Apache HTTP Server community is a typical service-oriented OSS community in which a service-oriented system and community are not controlled by one project leader but several core members. The objective of service-oriented system and community is to provide stable and robust services. The feature of OSS products classified as service-oriented is widely used in our daily life. Therefore, a service-oriented OSS community tends to be conservative against rapid changes and evolution of current products. These characteristics of service oriented products and communities meet the policies of the Apache community [7]. Our findings, however, cannot be explained by only viewing the Apache community as

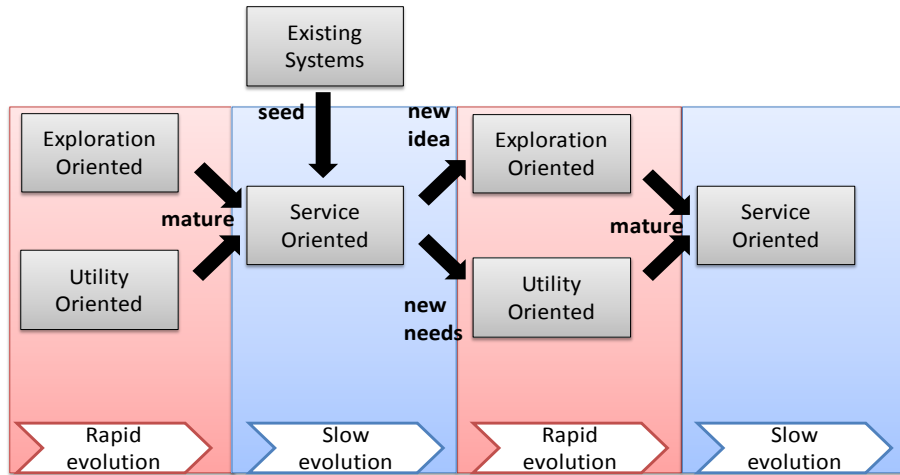


Fig. 4. Evolution patterns of OSS communities and products [8]

service-oriented, since the findings showed the openness of the Apache community changed over time.

Fig. 4 originally illustrated in [9] shows the evolution process of OSS communities, in which exploration-oriented (e.g., Perl and GNU systems) and utility-oriented (e.g., Linux systems excluding the kernel) OSS community experience rapid evolution, and the evolution speeds down to a stable growth as the community matures into service-oriented ones. Based on this evolution model and our findings, the condition of the Apache community can be applied to both a service-oriented and utility-oriented state. The first half of the analysis period (1997-2002) suggests the Apache community was open to public (non-core developers) and motivated non-core developers to join and contribute to the community, that is, the community was in a state of utility-oriented and rapid evolution. In contrast, the last half of our analysis period (2003-2007) suggests the community was regarded as service-oriented. As noted above, a service-oriented community is conservative because it should provide a stable and robust service. In the apache community, we have confirmed that the openness decreased in the last half period of the analysis. Therefore, contributions from core developers seemed to be considered as more important than ones from non-core developers.

In summary, our analysis based on quantitative data could capture a migration process from utility-oriented to service-oriented, as a part of empirical evidence of the evolutionary process of OSS communities and products. From the considerations of the results of our case study above, we can conclude that an OSS community and its leaders require to carefully adjust own conditions as the situation demands in order to keep the community and products successful.

7. Summary and Future work

In this paper, we developed a hypothesis that *the condition of “bazaar” in a successful OSS community with a long history changes over time in evolving to adapt itself and its products to the rapidly changing external environment* and analyzed the Apache HTTP Server community to test the hypothesis. From the results of our analysis, we have found that in the first half of the analysis period, the community tended to accept non-core developers' opinions (patches) in the same way as core developers and emphasized evolution of the products, while in the last half of the analysis period the community tended to place much value on providing quality services and moved to a stable condition. This implies that the Apache community and products have moved from utility-oriented to service-oriented.

In the near future, we need to analyze the Bugzilla database which is currently used for tracking and managing bug issues. Since in this paper we only focused on the development mailing list, we have not yet analyzed the review rate of patches posted to Bugzilla yet. Adding the analysis results of Bugzilla to the results in this paper would increase the validity of the results. Furthermore, we need to analyze not only the Apache community but also other OSS communities to assure the generalizability of the conclusions. We also would like to investigate the relationship between evolution processes and social structures [5] in OSS communities.

Acknowledgments

This research is being conducted as a part of the Next Generation IT Program, Grant-in-aid for Young Scientists (B), 20700028, 2008 and Grant-in-Aid for Japan Society for the Promotion of Science (JSPS) Fellows (Research No:20009220) by the Ministry of Education, Culture, Sports, Science and Technology, Japan.

References

1. Kevin Crowston and James Howison. The social structure of free and open source software development. First Monday, Vol. 10, No. 2, 2005.
2. Chris DiBona, Mark Stone, Danese Cooper. Open Sources 2.0: The Continuing Evolution, O'Reilly Media, Inc., 2005.
3. Joseph Feller and Brian Fitzgerald. Understanding Open Source Software Development. Addison-Wesley, 2002.
4. 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, OR., 2003.

5. Michael W. Godfrey and Qiang Tu. Evolution in Open Source Software: A Case Study. In Proceedings of 16th IEEE International Conference on Software Maintenance (ICSM'00), pp.131-140, San Jose, CA., 2000.
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. 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.
9. Kumiyo Nakakoji, Yasuhiro Yamamoto, Yoshiki Nishinaka, Koichi Kishida, Yunwen Ye. Evolution Patterns of Open-Source Software Systems and Communities, In Proceedings of 5th International Workshop on Principles of Software Evolution (IWPSE2002), Orlando, FL., pp.76-85, May, 2002.
10. Masao Ohira, Naoki Ohsugi, Tetsuya Ohoka, Ken-ichi Matsumoto. Accelerating cross-project knowledge collaboration using collaborative filtering and social networks. In Proceedings of International Workshop on Mining Software Repositories (MSR2005), pp.111-115, St.Louis, MO., 2005.
11. Eric S. Raymond. The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary. O'Reilly and Associates, 1999.
12. Peter C. Rigby, Daniel M. German, 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 (ICSE2008), pp.541-550, Leipzig, Germany, 2008.
13. Steven Weber. The Success of Open Source. Harvard University Press, 2004.
14. Yunwen Ye, Koichi Kishida. Towards an Understanding of the Motivation of Open Source Software Developers. In Proceedings of 25th International Conference of Software Engineering (ICSE2003), pp.419-429, Portland, OR., 2003.