Understanding the Process of Participating in Open Source Communities

Bianca Shibuya and Tetsuo Tamai
The University of Tokyo
3-8-1 Komaba, Meguro-ku
Building 15, Tokyo, Japan
{bshibuya,tamai}@graco.c.u-tokyo.ac.jp

Abstract

The number of participants in Open Source Software (OSS) communities has increased. Not only volunteers participate, but also companies and their employees. The motivation of the participants vary from extrinsic to intrinsic values. Community-managed and sponsored OSS projects try to explore these motivations to attract and keep these participants. This paper analyses three different OSS projects: MySQL, OpenOffice.org, and GNOME. Each has a different organizational structure that influences participants behavior. This study analyzes qualitative data from publicly available documents, such as project's wiki pages and project's webpages, and quantitative data from bug tracking systems and source code repositories. One of our findings is that the number of active developers does not change significantly when the total number of committers increases for the selected OSS projects.

1. Introduction

The longevity and success of an OSS project is strongly dependent on the community that provides an infrastructure for developers, users, and potential developers to collaborate with each other. It is important to analyze how the system and community evolve together to develop an OSS product. However, they evolve differently depending on the goal of the project and the structure of the community. In view of that, we consider two different types of OSS communities according to their current governance structure [13]: *autonomous (community-managed)* and *sponsored* communities.

During the system and community evolutions, the roles executed by the members are not static; they change according to the tasks executed by each member. For example, a newcomer starts as a passive member, reading messages posted in the mailing lists, then becomes a bug reporter, which does not require much technical knowledge. After gaining understanding of the system and community, this member may start writing or modifying source code, becoming a developer. Finally, after some important contributions are accepted, the member may obtain the right of committing code directly to the source code repository, becoming a committer. This process is called *joining script* [12], also referred as "immigration process" [2] and "socialization" [3]. The reverse process also happens; after gaining write-access to the repository, if a member reduces his participation, s/he will take less important roles.

An assertion is that the longevity of an OSS project is related to the size growth of the developer community. Because if there are no project members interested in keeping working on the source code and improving it, the project is going to stop and fail. An example is the GIMP project [14], which started as an academic project. When the creators left the university and decided to work on something different, this project stopped for more than a year until someone else decided to take over its control. Even if there are enough members at the moment, it is important to keep attracting people. The project should be ready in case core members decide to leave for some reason.

However, developers face obstacles to join in OSS projects. Software development is a knowledge-intensive activity that requires a very high level of domain knowledge, experience, and intensive learning. Those that contribute should write code that anyone can understand, and follow the project's standard. Therefore, this research tries to understand the participation of newcomers and experienced members and identifies the barriers for newcomers by analyzing the project's characteristics. We study three different OSS projects where the cost of contributing varies depending on their characteristics.

Research that covers the participation process in OSS projects is relatively scarce. To get useful information about members participation is difficult due to the lack of transparency in some projects' activities. This is especially true for sponsored OSS projects. In these projects core members usually work for a same company and for this reason they have other means of communications, internally in the company. There is not so much research covering these sponsored projects compared to community-managed projects. For this reason we decided to select two sponsored and one community-managed projects for our research.

This paper is organized in six sections. Section 2 presents the background in OSS joining process. Section 3 introduces our research questions. Section 4 describes our methodology. Section 5 answers our research questions and presents some limitations of our research. Section 6 presents final comments on our research and further work.

2. Background

The process of joining in an OSS project has been studied in the context of a few OSS projects. G. von Krogh *et al.* studied the joining script of the Freenet project [12]. They used data gathered from interviews, publicly available documents such as FAQs, e-mail archives from development mailing list, and source code repositories. They discovered that certain types of e-mail actions, such as offering bug fixes, are much more common among newcomers who eventually become developers. They also found that a significant period of observation (lurking), ranging from a couple of weeks to several months, was needed before someone felt s/he could contribute to a technical discussion. The process of joining is not an easy activity and takes time.

N. Ducheneaut examined the Python project and the interactions of a particular individual in his transitions from a newcomer to a developer using data from mailing lists and source code repository [3]. He points that prior technical activity and social standing in the community were strong factors in achieving the developer status. He also describes a series of well-defined steps that those who reached the status of developer in Python had to go through. Not every successful Python participant followed this exact trajectory, but most were reasonably close to it. The particular steps involved were:

1. Peripheral monitoring of the development activity;
2. Reporting bugs and simultaneous suggestions for patches;
3. Obtaining write-access to repository and directly fixing bugs;
4. Taking charge of a "module size" project;
5. Developing this project, gathering support for it, defending it publicly; and
6. Obtaining the approval of the core members and getting modules integrated into the project's architecture.

C. Bird *et al.* applied the hazard rate analysis, which is used to study time-dependent phenomena such as mortality and employment durations [2]. They modeled the duration between activities, for example, the first appearance on the mailing lists to the first commit to the source code repository. They extracted data from mailing lists and source code repository of three projects: Apache server, Postgres, and Python. One of their findings was that in Apache and Python projects, prior history of patch submission has a very strong effect on becoming a developer. This effect is not strongly observed in the Postgres project, where a much greater proportion (50%) of patches are submitted by non-developers than in Apache (31%) and Python (23%).

I. Herraiz *et al.* have found two groups with clearly different joining patterns. The difference in behavior between volunteers and firm sponsored developers influenced the joining pattern. Volunteers tend to follow a step-by-step joining process, while firm sponsored developers usually experience a "sudden" integration [4].

We study three different OSS projects that are different in their level of openness: MySQL, OpenOffice.org, and GNOME. There are two distinct types of openness according to J. West [13]: *transparency* and *accessibility*. Transparency allows outsiders to understand what is happening and why and the use of the community's final product, the source code. Accessibility allows external participants to directly influence the direction of the community to meet their specific needs, regardless of whether the external part is a volunteer or a firm sponsored individual.

There is a lot of research covering large OSS projects, such as Apache and Linux, but sponsored projects still remain not much explored. We analyze two sponsored projects and one autonomous project. MySQL, the most popular open source relational database management system, is an attractive alternative to higher-cost relational systems from commercial vendors. MySQL is currently owned and developed by Sun Microsystems. OpenOffice.org is a free cross-platform office application suite available for a number of different operating systems, such as Linux, Solaris, and Microsoft Windows. It is considered the main competitor to Microsoft Office. GNOME, the GNU network object model environment, is an open source software project for building a desktop environment for users and an application framework for software developers.

Comparing their characteristics, MySQL is the less open project, which does not permit external participants to become part of its group of developers, controlled by its owner. OpenOffice.org is accessible for outsiders to become developers but it is still under strong influence of Sun Microsystems, its major sponsor. And finally, GNOME is a well-known example of projects created by volunteers and maintained by the diversified community.

3. Research questions

We have defined the following research questions to be addressed in this research using data from different sources typically available in OSS projects:

RQ1: *Is there a path to follow to become a member with write-access to source code repository (be a committer)?* I. Herraiz *et al.* [4] found two types of joining process in the GNOME project: step-by-step process taken by volunteers with evidences of contributions before gaining the access to repository and "sudden" integration of employees paid by the sponsor company or any company interested in the project without evidences of previous contributions. Are these two types present in our selected OSS projects?

RQ2: *What are the difficulties for newcomers to join an OSS project?* This question tries to identify what are the barriers for people interested in starting their participation in OSS projects based on information from publicly available documents and e-mail exchanged with core developers.

RQ3: *What are the patterns of contributor activities?* These activities include bug report submissions and source code commits from the beginning of the projects to 2008/August. We define active contributors (bug reporters or committers) as those contributors who submit at least one contribution in a specified month.

RQ4: *Is most part of the development executed by a few members of an OSS project?* Previous studies of other OSS projects [9, 1, 6] show that most part of the development is executed by a few developers. Is this true for our selected projects?

RQ5: *Is the number of active committers proportional to the total number of committers during the evolution of the project?* We define active committers as those contributors who execute at least one commit to the source code repository in a specified month.

For MySQL project, only the RQ2 and RQ3 could be answered. Because only Sun Microsystems employees can commit changes to source code repositories. External contributions are difficult to track due to different ways of submitting contributions. Therefore, data from MySQL source code repository were not collected.

4. Methodology

As the selected OSS projects are considered large projects, we focus on the most important sub-project of each project: MySQL server of MySQL project, Writer of OpenOffice.org, and GTK+ of GNOME.

We tried to answer the research questions by gathering *qualitative* and *quantitative* data from the selected projects. Qualitative information (subjective) was collected from publicly available documents, such as project's main pages, wiki pages written by projects' developers, and research articles. The quantitative information was extracted from bug tracking systems and revision control systems like Concurrent Versions System (CVS) and Subversion (SVN).

For gathering initial information about the selected OSS projects and their development process, a questionnaire was created partially based on an evaluation model of OSS [11]. Two types of byproducts were generated based on information initially collected from projects' publicly available documents using the questionnaire: *workflow of participation* and *rich hypermedia* [5, 7]. We used workows to graphically represent the process of submitting contributions to OSS projects. Rich hypermedia were used to organize and reason about all the information that is provided from informal documents like webpages and mailing list archives.

These artifacts were created with the objective of identifying characteristics of the projects (existent roles, tools used for development, for instance) and understanding the process of contributing to OSS projects. As the questionnaire may contain out of date projects' information, these two types of artifacts had to be validated by each project's developers.

As quantitative data, we gathered them from bug tracking systems and CVS or SVN source code repositories. We collected data from the beginning of the projects until 2008/August for all repositories. Scripts were written in the Perl language to collect data automatically from bug tracking systems and stored into SQLite database. We also used a tool called CVSAnalY [8] for getting data from source code repositories, not only for CVS repositories, but also for SVN repositories and store them into MySQL database.

For analysis of qualitative data, community's attributes, such as governance, were considered as described by J. West [13]. These attributes affect the opportunities for others to participate in OSS projects. For analysis of quantitative data from bug tracking systems and source code repositories, we created charts to visualize the evolution of quantity of contributions from the beginning of the project until 2008/August.

We also use a methodology similar to the one applied by I. Herraiz *et al.* [4] to analyze our data in order to find some pattern of participation in OSS projects. For each user in the source code repository, who has its write-access, we computed the dates of the first and last commits, then we tried to identify those same users in the bug tracking systems to compute the number of bug reports and number of patches posted by them before gaining the access to repository, in other words, before the first commit. As there are other ways to submit patches to OSS projects, we also tried to look for evidences of patch submissions in the changelog file available for each source code repository.

5. Results and limitations

This section shows the five research questions and their answers. At the end of this section, we also present some limitations of our research.

RQ1: *Is there a path to follow to become a member with write-access to source code repository (be a committer)?*

The two types of paths described by I. Herraiz *et al.* [4] are present in our selected projects. The sudden integration of employees could be viewed clearly only in the Writer project. The majority of committers with higher number of commits for Writer project are employees of Sun Microsystems. A few members have taken a step-by-step joining process, at least for the most active committers. As we analyzed sub-projects of large projects, the "sudden" integration can be taken by developers for a different reason other than employment: previous participation in other sub-projects. For GTK+ project, the majority of committers came from another GNOME sub-project, the GIMP, which is a project GTK+ originated from.

RQ2: *What are the difficulties for newcomers to join an OSS project?*

We found the following difficulties faced by newcomers: selection of a suitable task from development process according to his/her skills, lack of up-to-date development documents, constraints imposed by contributor license agreement, no response from core developers for their doubts and support request, and need to learn a new tool.

RQ3: *What are the patterns of contributor activities?*

Different from the community-managed GTK+ project, the sponsored MySQL Server and Writer projects had their number of bugs and number of active reporters decreased in the last months as displayed in Figure 1, 2, and 3. The cause is uncertain, but possible causes are that the member's interest in the project has decreased for different reasons or the product has been improved, since the number of commits does not decreased. For the number of source code commits and number of active committers, Writer and GTK+ projects have the same pattern with alternations between higher and lower values (figures were omitted due to space restriction); and in Figures 4 and 5 for the number of active committers.

RQ4: *Is most part of the development executed by a few members of an OSS project?*

For both Writer and GTK+ projects, a sponsored and a community managed project, the development is also executed by a few members, who seem to be the core members of the projects. Listing the fifteen committers with the highest number of commits, we can note that 81.38% and 84.80% of the total number of commits are executed by these fifteen committers for the Writer and GTK+ projects respectively. These fifteen committers correspond to 13.89% and 7.14% of total number of committers, who commits only code files (for example, files with .cpp, .c, and .h extensions) for the Writer and GTK+ projects respectively. The percentage of these commits may also include pieces of code contributed by people who do not have write-access to the repository, but these are not significant compared to the total amount of commits executed by each core member.

RQ5: *Is the number of active committers proportional to the total number of committers during the evolution of the project?*

For both Writer and GTK+ projects, the total number of committers increases but the number of committers who actually execute commits per month does not change significantly as it can be viewed in Figures 4 and 5. It is possible to see some months of inactivity for most of project's members, mainly for the GTK+ project.

One of the limitations of our research is the fact that it was not possible to track participation of developers who started their participation in another sub-project of the same project. For this reason, a more complete analysis of members' participation should be done for all projects' modules. Other charts and tables, which give more details of the collected data, can be found in [10].

6. Conclusions and further work

Besides the source code being open, other sources of information about the development process should also be open to facilitate participation of new members. Well-structured OSS projects make available guidelines and wiki pages. All the selected projects provide different ways to facilitate this collaboration. MySQL project is working hard on its process of becoming more transparent to outsiders as the project has the necessity to retain all the control over the development, not allowing outsiders to become developers. OpenOffice.org, which faces intensive competition against a well-funded proprietary alternative, tries to become more accessible to newcomers. GNOME project, which was created by volunteers, tries to be transparent and accessible to attract and keep their members.

Taking the project's active committers per month and analyzing only the committers with higher number of commits, most of them are Sun employees for the Writer project. It is possible to see more commitment of these committers to the project. In other words, there are some committers who remain active from the first commit to the last. But there are also committers who have a few months of inactivity, ranging from one to three months. For the GTK+ project, committers with higher number of commits are current core members of the project or they were in the past. It is possible to verify some months of inactivity, ranging from one to eight months.

There is a lot of information that is kept inside the OSS

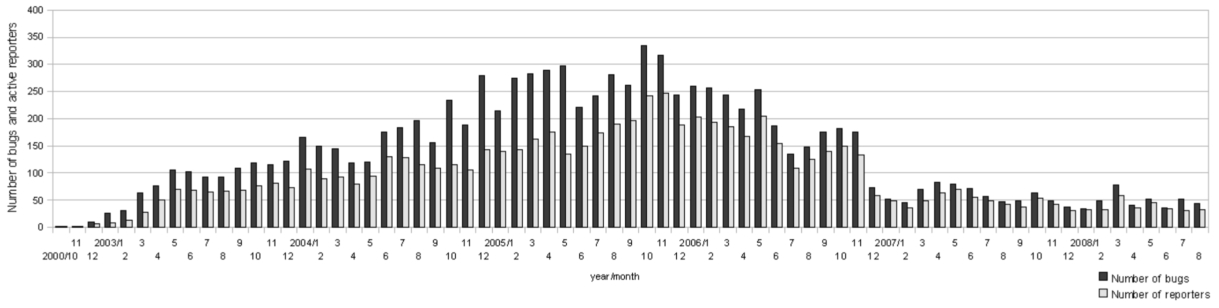

Figure 1. Chart showing the number of bugs and active reporters for MySQL server

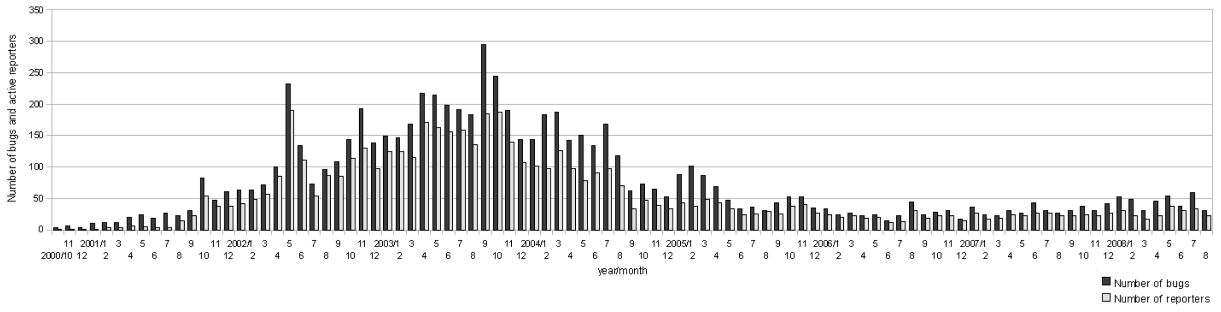

Figure 2. Chart showing the number of bugs and active reporters for Writer

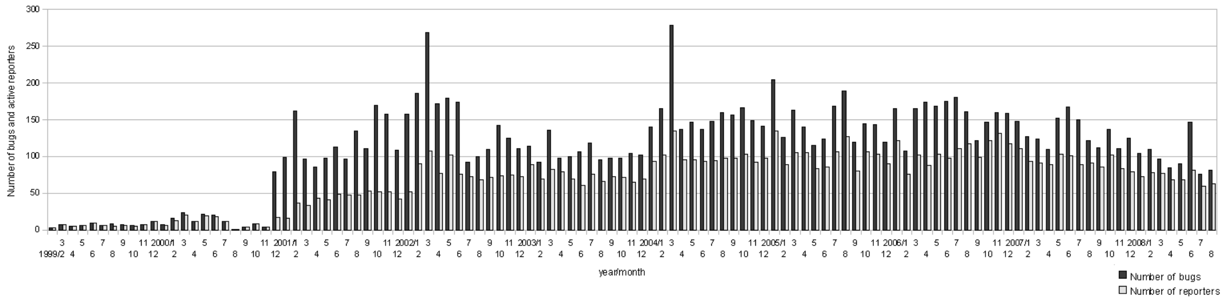

Figure 3. Chart showing the number of bugs and active reporters for GTK+

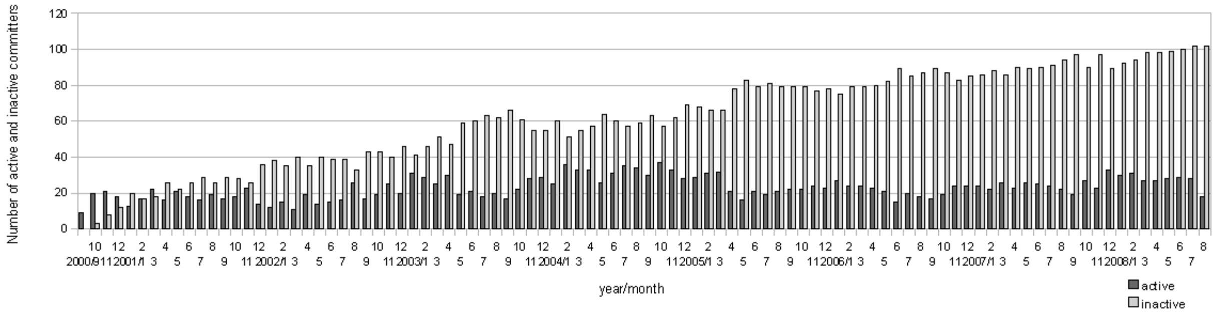

Figure 4. Chart showing the number of active and inactive committers for Writer

communities, for example how to improve the code and how to attract participants. This information is exchanged mainly among community's members during conferences organized by each OSS project. All the selected projects

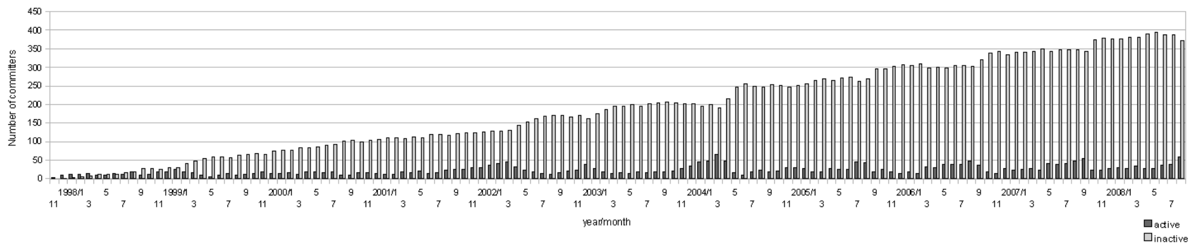

Figure 5. Chart showing the number of active and inactive committers for GTK+

have their conferences for discussing their problems and suggesting improvements. The material from these conferences are good source of information to be explored.

Most of OSS projects use CVS or SVN tools, which are centralized revision control systems. A few projects have decided to change their code repositories to a distributed one, for example, Linux Kernel project uses Git, OpenJDK project uses Mercurial, and recently MySQL changed to Bazaar. Proponents of Distributed Revision Control Systems (DRCS) states that one of its advantages is that they allow participation in projects without requiring permissions from projects core members, and thus better promote a culture of meritocracy instead of requiring status of committer. In other words, anyone can create a space for their code modications (repository branches) and make this space public for other members to review. DRCS permit the merge of modications from other branches to the main repository easily. This way of working seems to increase the participation of new members and to improve how patches are submitted to the project. Research covering participation using DRCS shall verify if this advantage stated by its proponents works.

7. Acknowledgments

We would like to thank all those who helped us, providing information from OSS projects: Behdad Esfahbod and Tim Janik from GNOME, Maho Nakata from OpenOffice.org, and especially Lenz Grimmer from MySQL, Michael Meeks and Mathias Bauer from OpenOffice.org for their explanations, discussions, and patience.

References

[1] J. H. Audris Mockus, Roy T. Fielding. Two case studies of Open Source Software development: Apache and Mozilla. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 11(3):309 – 346, 2002.

[2] C. Bird, A. Gourley, P. Devanbu, A. Swaminathan, and G. Hsu. Open borders? Immigration in open source projects. In *MSR '07: Proceedings of the Fourth International Workshop on Mining Software Repositories*, page 6, 2007.

[3] N. Ducheneaut. Socialization in an Open Source Software community: A socio-technical analysis. *Computer Supported Cooperative Work*, 14(4):323–368, 2005.

[4] I. Herraiz, G. Robles, J. J. Amor, T. Romera, and J. M. González-Barahona. The processes of joining in global distributed software projects. In *GSD '06: Proceedings of the 2006 international workshop on Global software development for the practitioner*, pages 27–33, 2006.

[5] C. Jensen and W. Scacchi. Process modeling across the web information infrastructure. *Software Process: Improvement and Practice*, 10:255–272, 2005.

[6] S. Koch and G. Schneider. Effort, co-operation and co-ordination in an Open Source Software project: GNOME. *Information Systems Journal*, 12(1):27–42, 2002.

[7] M. Oza, E. Nistor, S. Hu, C. Jensen, and W. Scacchi. A first look at the Netbeans requirements and release process. Institute for Software Research. University of California., 2002.

[8] G. Robles, S. Koch, and J. M. González-Barahona. Remote analysis and measurement of Libre Software Systems by means of the CVSAnalY tool. In *In Proceedings of the 2nd ICSE Workshop on Remote Analysis and Measurement of Software Systems (RAMSS)*, 2004.

[9] W. Scacchi, J. Feller, B. Fitzgerald, S. Hissam, and K. Lakhani. Understanding Free/Open Source Software development processes. *Software Process: Improvement and Practice*, 11(2):95–105, 2006.

[10] B. Shibuya. Understanding the process of participating in open source communities. Master's thesis, The University of Tokyo, 2009.

[11] K. van den Berg. Finding open options - an Open Source Software evaluation model with a case study on Course Management Systems. Master's thesis, Tilburg University, 2005.

[12] G. von Krogh, S. Spaeth, and K. R. Lakhani. Community, joining and specialization in Open Source Software innovation: A case study. *Research Policy*, 32(7):1217–1241, 2003.

[13] J. West and S. O'Mahony. The role of participation architecture in growing sponsored open source communities. *Econ-Papers - Industry & Innovation*, 15(issue 2):145–168, 2008.

[14] Y. Ye, K. Nakakoji, Y. Yamamoto, and K. Kishida. *Free/Open Source Software Development*, chapter 3 - The Co-Evolution of Systems and Communities in Free and Open Source Software Development, pages 59–82. 2005.