Hindawi Publishing Corporation
ISRN Software Engineering
Volume 2014, Article ID 535724, 10 pages
http://dx.doi.org/10.1155/2014/535724

Research Article

Understanding Contributor to Developer Turnover Patterns in OSS Projects: A Case Study of Apache Projects

Aftab Iqbal

INSIGHT, NUI, Galway, Ireland

Correspondence should be addressed to Aftab Iqbal; aftab.iqbal@deri.org

Received 31 August 2013; Accepted 17 November 2013; Published 19 January 2014

Academic Editors: Y. Dittrich and Y. K. Malaiya

OSS projects are dynamic in nature. Developers contribute to a project for a certain period of time and later leave the project or join other projects of high interest. Hence, the OSS community always welcomes members who can attain the role of a developer in a project. In this paper, we investigate contributions made by members who have attained the role of a developer. In particular, we study the contributions made by the members in terms of bugs reported, comments on bugs, source-code patch submissions, and their social relation with other members of an OSS community. Further, we study the significance of nondevelopers contribution and investigate if and to what extent they play a role in the long-term survival of an OSS project. Moreover, we investigate the ratio of contributions made by a member before and after attaining the role of a developer. We have outlined 4 research questions in this regard and further discuss our findings based on the research questions by taking into account data from software repositories of 4 different *Apache* projects.

1. Introduction and Motivation

Open source software (OSS) is a good example of global software development. It has gained a lot of attraction from the public and the software engineering community over the past decade. The success of an OSS project is highly dependent on the infrastructure provided by the community to the developers and users in order to collaborate with each other [1]. It is important to understand how the OSS project and the community surrounding it evolve over time. During the project and community evolution, the roles of the members change significantly, depending on how much the member wants to get involved into the community. Unlike a project member in a software company whose role is determined by a project manager and remains unchanged for a long period of time until the member is promoted or leaves, the role in an OSS project is not preassigned and is assumed by a member as he/she interacts with other members. An active and determined member usually becomes a "core member" through the following path: a newcomer starts as a "reader", reading messages on the mailing lists, going through the wiki pages and other documentation, and so forth, in order to understand how the system works. Later, he starts to discover and report bugs, which does not require any technical knowledge, and becomes a "bug reporter". After gaining good understanding of the system and community, he may start fixing small and easy bugs which he identifies himself or are reported by other members of the system, hence playing the role of either a "bug fixer," "peripheral developer," or an "active developer." To this stage, his bug fixes are usually accepted through patches submitted on the mailing lists or bug tracking system. Finally, after some important contributions are accepted by the core developers, the member may obtain the right of committing source code directly to the source control repository, hence becoming the "core member" of the project. This process is also called "joining script" [2], also referred to as "immigration process" [3]. The general layered structure of OSS communities as discussed above is further depicted in Figure 1, in which the role closer to the center has a larger radius of influence on the system.

The figure depicts an ideal model of role change in the OSS community. However, not all members want to be or become the "core member." Some remain "passive user" and

some stop somewhere in the middle. The key point is that OSS makes it possible for an aspiring and determined developer to be part of the "core members" group of developers through continuous contributions. On the other hand, the sustainability of an OSS project is related to the growth of the developer community. The community surrounding an OSS must regenerate itself through the contributions of their members and continuous emergence of new "core members" otherwise the project is going to stop or fail. An example is the GIMP project (http://www.gimp.org/) [5], which started as an academic project. When the creators left the university and decided to work on something else, the project stopped for more than a year until someone else decided to take over the control and resume working on the project. Therefore, attracting or integrating new members is an important aspect to keep the system and the community evolve over time.

Given these precedents, the research goal of the study presented in this paper is to understand the pattern of contributions made by members who eventually attained the role of a developer in an OSS project, that is, joining the "core members" group of developers. We are interested to investigate the key factors which led members towards attaining the role of a developer. We studied the immigration process in OSS projects as done in the past by others but using a quantitative approach based on extensive data mining. The contribution of this paper is manifolds: we study the contributions made by the members in terms of bugs reported, comments on different bugs, attachments or source-code patch submissions to fix certain bugs, and social relation with other members on the mailing list in a particular OSS community/project. Further, we analyze the contributions made by members before and after attaining the role of a developer. Moreover, we compute the ratio of average contributions made by a developer (before attaining the role of a developer) and compare it with the average contributions made by other members of the project.

The rest of the paper is structured as follows: the related work comparable to our approach will be discussed in Section 2. Research questions are outlined in Section 3. In Section 4, the methodology we used to extract information from different software repositories is described. Section 5 presents the results based on the research questions, and finally, in Section 6, we conclude our work.

2. Related Work

The process of joining an OSS project has been studied by many researchers in the past. In this line, the best known model which describes the organizational structure of an OSS project is the "onion model" [10] (cf. Figure 1), a visual analogy which depicts how the members of an OSS project are positioned within a community. The onion-like structure represents only a static picture of the project, lacking the time dimension which is required to study the role transformation (i.e., promotion) from being a passive user to the core member of the project. Ye et al. complemented this shortcoming with a more theoretical identification and description of roles [5]. According to this model, a core member is supposed to go through all the roles, starting as a passive user, until he/she attains the role of a core member. In this regard, Jensen and Scacchi also studied and modeled the process of role migration in OSS projects [11], focusing on end users who eventually become core members. They identified different paths for the joining process and concluded that the organizational structure of studied OSS projects is highly dynamic in nature.

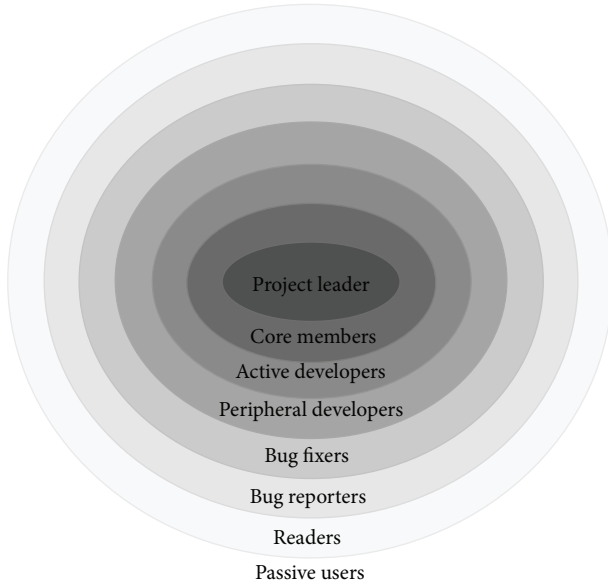

Figure 1: General structure of an OSS community based on the onion model described in [4].

Von Krogh et al. studied the joining and specialization process of FreeNet project [2]. Based on the data gathered from publicly available documents, mailing list archives, and the source control repository, they discovered that offering bug fixes is much common among newcomers who eventually become core members of the project. They also found that a certain period of time, ranging from couple of weeks to several months, was required before a newcomer could contribute to a technical discussion. There also exist few research studies which have reported and quantified the onion-like structure of a community for many OSS projects. For example, Mockus et al. [12] studied the *Apache httpd server* and *Mozilla web browser* projects and Dinh-Trong and Bieman [13] studied the *FreeBSD* project. According to their findings, the "core members" group is composed of small number of members. Surrounding the "core members" group is a large group of contributors (i.e., active developers, peripheral developers, etc.) who submit bug reports, offer bug fixes and participate heavily in discussions on the mailing lists.

In an ethnographic study, Ducheneaut studied the *Python* project in order to investigate the contribution of the members during their role transition from being a newcomer towards attaining the role of a core member by taking into account data from mailing lists and source control repository [14]. He found that prior technical commitment and good

social standing in the community were strong factors in joining the core members group of developers having write-access to the source control repository. Bird et al. [3] used the mailing lists and source control repository to investigate the time required for members to be invited into the "core members" group of an OSS project. They applied hazard rate analysis or survival analysis [15] to model the time-dependent phenomena such as employment duration. They used survival analysis to understand which factors influence the duration and occurrence of such events and to what degree. They modeled the duration between activities by considering the first appearance of a member on the mailing list to the first commit on the source control repository. One of their findings was that prior patch submission had a strong effect on becoming part of the "core members" group of a project. Herraiz et al. [16] studied the GNOME project and found two different patterns of joining the project: (1) volunteers/contributors who follow the "onion model", and (2) firm/organization sponsored developers who do not. They found that hired developers gain knowledge quickly enough to start writing code than the volunteers.

Although these research studies were carried out in detail on different OSS projects, they considered data only from mailing lists and source control repositories. However, we also take into account bug repositories and quantify the contributions made by members in terms of the following bugs reported, comments on bugs, social relation with other members based on comments, social relation with other members based on email exchanges on the mailing list, and patch submissions on bug repositories. In addition to that, there is no published work known to us which studies the contributions made by a member *before* and *after* attaining the role of a developer in an OSS project. Therefore, we have quantified and analyzed the average rate of contributions made by a member *before* and *after* attaining the role of a developer which makes this work unique in contrast to other related pieces of work which have been done so far in this area.

3. Research Questions

As mentioned earlier, the success of an OSS project is in its long-term survival which is potentially due to the existence of a community surrounding the project. We are particularly interested to identify the role of a community in the long-term survival of an OSS project as well as the key factors which promote a nondeveloper (In this paper, we will use the term "nondeveloper" to refer to all those members who do not have write-access to the source control repository.) to the role of a developer (In this paper, we will use the term "developer" to refer to all those members who have write-access to the source control repository.). Further, we are interested to know if the potential developers (In this paper, we will use the term "potential developer"' to refer to all those members who started as a passive user and later attained the role of a developer.) follow the onion model or if there is a sudden integration of developers into the "core members" group of an OSS project. In order to address these key points, we have outlined few research questions in the following which will be addressed using data from publicly available software repositories of few selected *Apache* projects:

(1) RQ-1: *What is the ratio of contributions made by the developers and nondevelopers to the project over the period of time?*

Previous studies [12, 17] on various OSS projects have shown that most part of the source-code development is carried out by the developers of those projects. We will investigate what are the contributions of nondevelopers if the source-code development is mostly done by the developers of those projects? In particular, we will investigate the contributions of nondevelopers in terms of reporting bugs, commenting on bugs, and exchanging emails. Further, we are interested to investigate the role of nondevelopers in the long-term success and maturity of an OSS project.

(2) RQ-2: *What is the ratio of contributions made by a potential developer before and after attaining the role of a developer?*

Attaining a higher role comes with more responsibilities and commitments to the project. We will investigate if a potential developer after attaining the role of a developer contributes (except source-code modification or bug fixing) more in contrast to contributing as a nondeveloper. Does the contribution pattern change with the change in role of a potential developer? To be more precise, does his/her contribution to the project in terms of bugs reporting and interaction with the community increase or decrease? We hypothesize that after attaining the role of a developer, he/she will participate actively in technical discussions on the bug tracking systems or on the mailing lists and report bugs effectively.

(3) RQ-3: *What is the average rate of contributions made by a potential developer comparing to other members of the project before attaining the role of a developer?*

We will investigate if the average contributions made by a potential developer are more than the average contributions made by nondevelopers who were also active during his/her time period. It has been addressed in previous studies [3] that demonstration of technical commitment and social status with other members will positively influence in attaining the role of a developer. We will investigate if a potential developer was more active (i.e., technically skilled and higher social status) than nondevelopers before attaining the role of a developer.

(4) RQ-4: *Does a potential developer follow onion model in order to attain the role of a developer?*

We will investigate if a potential developer follows the onion model in order to attain the role of a developer, that is, joining the "core members" group of the project. Not every member who is contributing to an OSS project eventually becomes a developer. It depends on the level of involvement of a member in an OSS project and also on the needs to promote

Table 1: Apache projects data range.

Apache projects	Date range
Apache Ant [6]	2000–2010
Apache Lucene [7]	2001–2010
Apache Maven [8]	2003–2010
Apache Solr [9]	2006–2010

Table 2: Dataset overview.

	Attachments	Bugs	Commits	Emails
Apache Ant	1,345	5,480	6,025	84,737
Apache Lucene	2,865	3,116	5,790	59,616
Apache Maven	1,169	3,902	8,815	87,611
Apache Solr	2,146	2,528	4,288	25,173

a nondeveloper to the role of a developer. There is no static or standard timeline for a member to join the "core members" group of a project. The time period required to attain the role of a developer varies from project to project and also from member to member. Members often start contributing to a project by participating in the mailing list conversations to get themselves familiar with the project before contributing source-code patches to the project. We will study the appearance of a potential developer on different software repositories by comparing the timestamp value of their first activity on these software repositories in order to validate if he/she actually followed the onion model.

4. Data Extraction Process

In this section, we describe our data extraction methodology and the *Apache* projects selected for evaluation. We extracted data from 4 different *Apache* projects as shown in Table 1. The range of data selected for each project is different because of the difference in the starting date of each project. The reason of choosing these *Apache* projects is that the repositories of these projects are on the Web and available to be downloaded (i.e., mailing list archives, bugs, subversion logs, etc.).

Most *Apache* projects have at least 3 different mailing lists: `user`, `dev`, and `commits`, but some have more than 3 mailing lists (e.g., `announcements`, `notifications`, etc.). For our study, we downloaded only the dev mailing list archives of each *Apache* project under consideration. The reason is that software developers communicate often with each other on the `dev` mailing list rather than on any other mailing lists. We developed our own scripts which were used to extract information from mailing list archives in a similar manner to previous research [12, 18]. For example, each email was processed to extract information like *sender name, email address, subject, date, message-id*, and *reference*. The *reference* field contains *message-id(s)*, if the email is a reply to previous thread(s). We used the *reference* field information to build a social network correspondence and computed social network measures [19] of all the members of a project.

We retrieved all the bugs (related to the *Apache* projects we considered for our study) which are publicly available through the Bugzilla and JIRA Web interface (https://issues.apache.org/) and extracted the required information using our custom written scripts. For further details on the information extracted from each email and bug, we refer the readers to [20]. We computed the social relation correspondence among members on the bug tracking system based on the bug comments exchanged among themselves. Bird et al. [21] findings indicated the detection and acceptance of source-code patches through the mailing list, but we discovered that source-code patches were always attached to the respective bugs on the bug tracking system rather than sending it on the mailing list. Prior research has indicated the importance of offering bug fixing and its acceptance as an influential factor in gaining the developer status [14]; therefore, we have also analyzed how many source-code patches were submitted by the members on bug repositories.

In order to get information from source control repository, we wrote our script (see [22] for details) and extracted necessary information (i.e., *log number, date of commit, author id, and files committed*). We only considered those subversion logs where a particular source-code file (i.e., "*.*java*" because *Apache* projects under consideration are Java-based) was committed. These subversion log files were further analyzed by our script in order to identify if it fixes any bug by looking for patterns, such as, "PR:xxx," "MNG-xxx," "SOLR-xxx," "LUCENE-xxx," and patch acceptance acknowledgements such as "patch provided by xxx", "patch submitted by xxx." On the identification of such patterns, the bugs were queried to retrieve source-code patches associated with those bugs. This would help to identify source-code patches that are accepted by the "core members" group of the project. Further, it allows to identify members who possess strong technical skills required for attaining the role of a developer in the project. Table 2 gives an overview on the raw data sources we extracted from different software repositories of the selected *Apache* projects based on the methodology described.

The values for *Apache Ant* show that there were `1,345` source-code patches found for a total of `5,480` bugs reported on the bug tracking system. Further, `6,025` subversion logs were extracted from the source control repository where source-code files (i.e., *.java) were committed and `84,737` emails were extracted from the *Apache Ant* mailing list archives between 2000 and 2010.

5. Empirical Analysis

Before we address each of the research questions in detail, we present a high level overview on the development activity of each *Apache* project under consideration over the period of time in Figure 2. This would give an insight into how much contributions were made each year to a project and the peak development years of a project. For each *Apache* project under consideration, we computed the number of contributions with respect to the number of people who made those contributions. For example, we computed the number

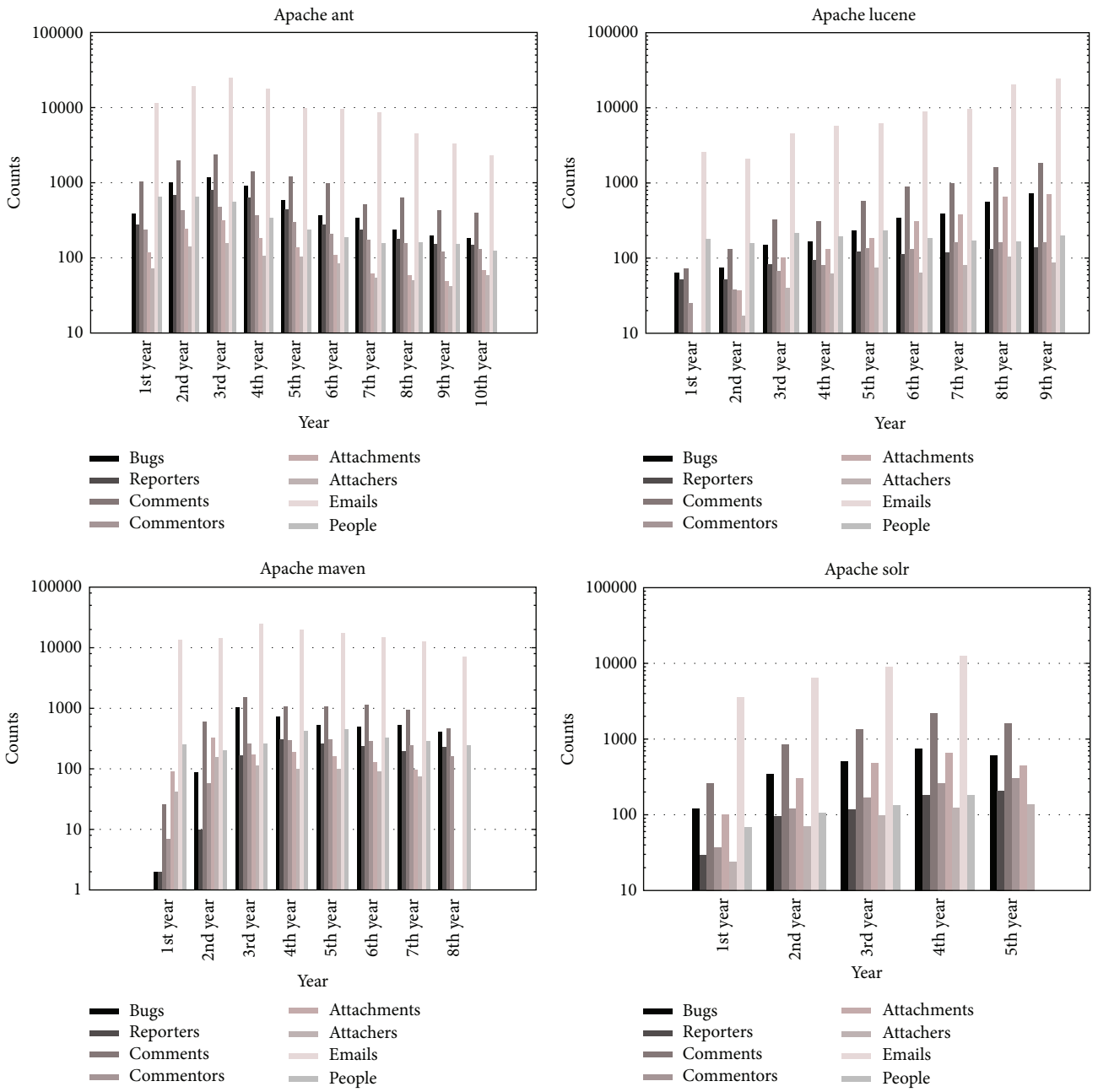

Figure 2: Development activity of Apache projects over the period of time.

of distinct bugs reported each year along with the number of distinct reporters who submitted those bug reports (cf. Figure 2). This would make it easier to answer simple questions like: how many bugs were reported and how many members were involved in the bug reporting process during the 2nd year of a project?

Preliminaries: let $\mathbf{C}$ be the total number of members (i.e., developers, nondevelopers, etc.) who worked on the project:

$$\mathbf{C} = \{c_1, c_2, \ldots, c_n\}. \quad (1)$$

Let $\mathbf{Y}$ be the total number of years of a project under consideration:

$$\mathbf{Y} = \{y_1, y_2, \ldots, y_n\}, \quad (2)$$

where y_1 is considered to be the first year of the project, y_2 is considered to be the second year of the project, and so on. Let $\grave{\mathbf{C}}$ be the set of members (i.e., developers, nondevelopers, etc.) who were active in a time period y:

$$\grave{\mathbf{C}} = \{c_1, c_2, \ldots, c_n\}, \quad \grave{\mathbf{C}} \subseteq \mathbf{C}, \quad (3)$$

and let `Immig` be the immigrants (i.e., potential developers) who started as contributors and later become the developers

of the project. We classified only those members as immigrants/potential developers who had an activity in the project (i.e., number of bugs reported, number of bugs commented, number of patches submitted, or number of emails sent) at least 4 months prior to their first commit on the source control repository

$$\mathbf{Immig} = \{\mathrm{Immig}_1, \mathrm{Immig}_2, \ldots, \mathrm{Immig}_n\},$$
$$\mathbf{Immig} \subseteq D_y \subseteq \mathbf{C}. \tag{4}$$

Let D_y be the set of developers who have made commits before and during time period y such that $D_y \subseteq \mathbf{C}$. Let the total number of bugs reported and commented and emails sent by a set of members in a time period y be represented as follows:

$$\mathrm{Contribution}_{\mathrm{bugs}}(C, y),$$
$$\mathrm{Contribution}_{\mathrm{comments}}(C, y), \tag{5}$$
$$\mathrm{Contribution}_{\mathrm{emails}}(C, y),$$

whereas the number of bugs reported and commented and emails sent by the developers in a given period of time y is represented as

$$\mathrm{Contribution}_{\mathrm{bugs}}\left(D_y, y\right),$$
$$\mathrm{Contribution}_{\mathrm{comments}}\left(D_y, y\right), \tag{6}$$
$$\mathrm{Contribution}_{\mathrm{emails}}\left(D_y, y\right),$$

respectively. Let d be a single developer and let `commitDate`(d) return the first commit date of a developer. The yearly average contribution of a member before and after attaining the role of a developer is represented as

$$\mathrm{Contribution}_{\mathrm{before}}(d, \mathrm{commitDate}(d)),$$
$$\mathrm{Contribution}_{\mathrm{after}}(d, \mathrm{commitDate}(d)), \tag{7}$$

and the total number of bugs reported and commented and emails sent by an immigrant before becoming a developer is represented as

$$\mathrm{Contribution}_{\mathrm{bugs}}(\mathrm{Immig}, \mathrm{commitDate}(\mathrm{Immig})),$$
$$\mathrm{Contribution}_{\mathrm{comments}}(\mathrm{Immig}, \mathrm{commitDate}(\mathrm{Immig})), \tag{8}$$
$$\mathrm{Contribution}_{\mathrm{emails}}(\mathrm{Immig}, \mathrm{commitDate}(\mathrm{Immig})).$$

RQ-1: *What is the ratio of contributions made by the developers and nondevelopers to the project over the period of time?*

In order to compute the contributions, we need to distinguish between the developers and nondevelopers of the project. As each subversion log has a time stamp associated to it, we queried all subversion logs from the *start date* of the project till the last *commit date* of the year under consideration. Based on this, we get a list of all developers IDs who have contributed to the source control repository till that particular year. For each developer ID, we computed the contributions (i.e., bugs reported, comments on bugs, emails, etc.) made to the project on yearly basis and add up the contributions made by all the developers for each year. Similarly, we computed the contributions made by nondevelopers on yearly basis and add up all their contributions for each year. Later, we plotted the contributions made by the developers and nondevelopers for each year in the form of a chart which is shown in Figure 3. Figure 3 shows the comparison of contributions made by the developers and nondevelopers of each *Apache* project under consideration. Further, we computed the average rate of contributions made by the developers and nondevelopers as well as the average number of developers and nondevelopers who made those contributions per year, which is shown in Table 3. For example, the number of bugs reported by the nondevelopers in a given period of time y is computed as follows:

$$\mathrm{Contribution}_{\mathrm{bugs}}\left(\frac{C}{D_y}, y\right), \tag{9}$$

and the average number of bugs reported by the developers and nondevelopers is computed as follows:

$$\sum_{y \in \mathbf{Y}} \frac{\mathrm{Contribution}_{\mathrm{bugs}}\left(D_y, y\right)}{|y|},$$
$$\sum_{y \in \mathbf{Y}} \frac{\mathrm{Contribution}_{\mathrm{bugs}}\left(C/D_y, y\right)}{|y|}. \tag{10}$$

Let us assume that the nondevelopers who were active in a certain period of time is calculated by `nonDev`($\dot{\mathbf{C}}/D_y, y$),

Table 3: Average rate of contributions made by developers and non developers.

Variable	Contributions		Participants	
	Dev.	Non dev.	Dev.	Non dev.
	Apache Ant			
bugs reported	29.70	509.30	6.30	377.40
bug comments	681.50	416.01	11.30	248.90
emails	4,773.91	5,628.00	14.54	284.91
	Apache Lucene			
bugs reported	141.11	156.77	8.66	91.33
bug comments	507.11	241.33	10.77	95.33
emails	3,547.00	5,745.66	13.66	173.22
	Apache Maven			
bugs reported	204.87	268.25	11.00	165.00
bug comments	569.28	334.28	13.85	195.14
emails	4,328.37	9,505.75	13.37	260.87
	Apache Solr			
bugs reported	201.10	265.60	10.20	116.00
bug comments	819.03	432.80	11.00	166.60
emails	1,808.25	6,047.75	9.00	112.50

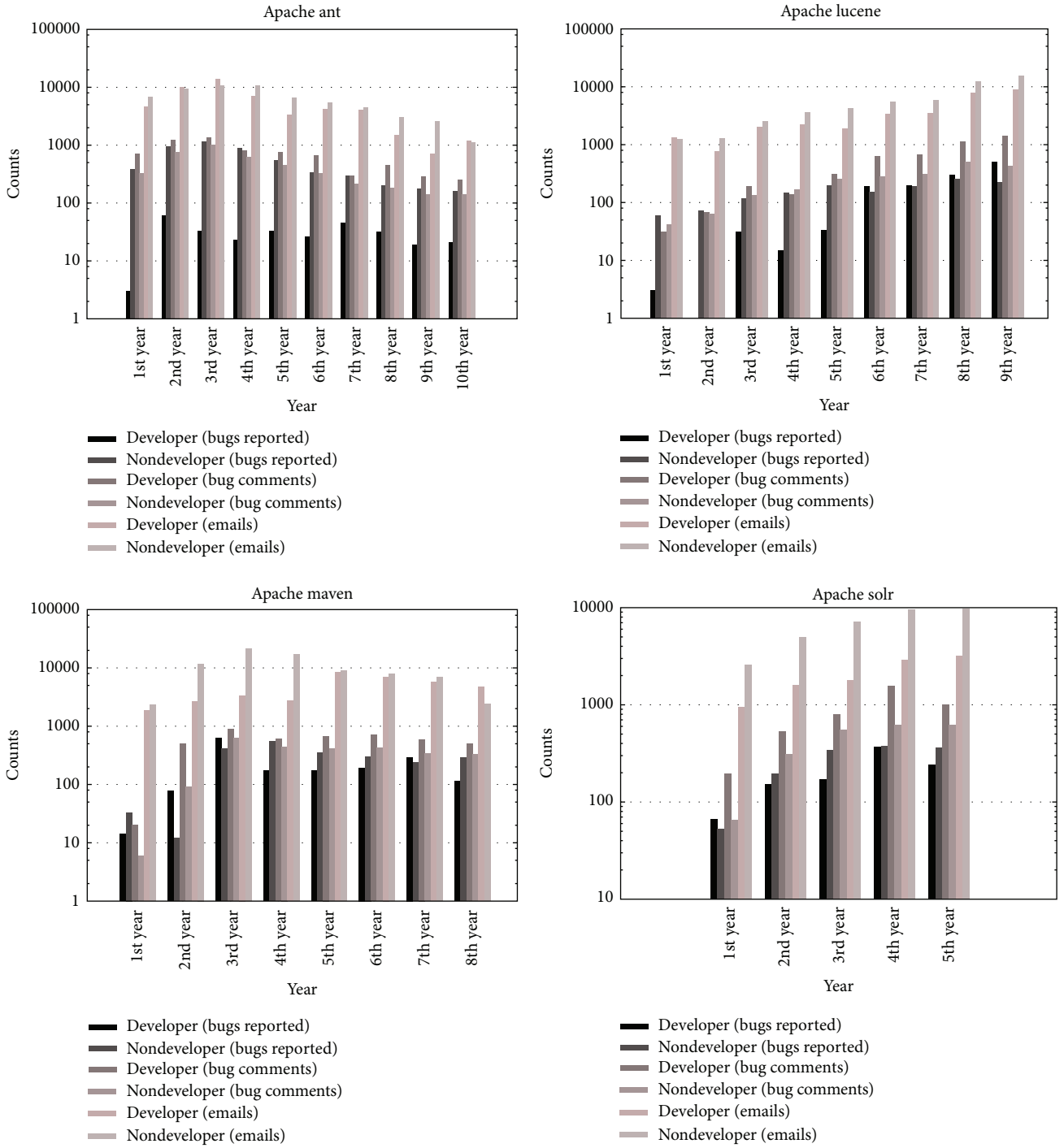

Figure 3: Contributions made by developers and nondevelopers over the period of time.

the average participation ratio of developers and nondevelopers is computed as follows:

$$\sum_{y \in \mathbf{Y}} \frac{D_y}{|y|}, \qquad \sum_{y \in \mathbf{Y}} \frac{\text{nonDev}\left(\grave{\mathbf{C}}/D_y, y\right)}{|y|}. \tag{11}$$

The results in Table 3 show that nondevelopers are highly involved (i.e., contributing more than the developers) in reporting bugs and participating in discussions on the mailing list. One potential reason for this is the existence of a huge community surrounding these *Apache* projects. Given that discussing/commenting on a bug report requires technical knowledge about the project which is why developers appear to be more active in commenting on the bug reports than nondevelopers, it is quite obvious from Table 3 (also see Figure 3) that nondevelopers play a significant role in

the projects under consideration, and hence it is one of the major factors in the long-term survival, success and maturity of these projects over the period of time.

The high ratio of nondevelopers involvement in the project (cf. Figure 3 and Table 3) allows the core members to select or vote for the potential developers to be invited to the "core members" group of the project.

RQ-2: What is the ratio of contributions made by a potential developer before and after attaining the role of a developer?

We are only interested in those developers who did not start contributing directly to the project but instead follows the onion model (cf. Figure 1). In order to select those developers, we retrieved all developers from subversion logs. Later for each developer, we compared his first commit date on the project to his first appearance on any of the project repositories (i.e., first bug reporting date, bug comment date, attachment, or email date) in order to compute the number of days or months before he started to contribute as a developer. Although there is no fixed or standard timeline for attaining the role of a developer in the project, we considered only those developers who had an activity (bug report, bug comment, attachment or email) on the project at least 4 months prior to their first commit on the source control repository of the project.

For each of those selected developers, we queried the contributions made to the project *before* and *after* the first *commit date* of each developer. As the time period of attaining the role of a developer is different for each developer, we computed the average yearly rate of contributions made by a developer *before* and *after* attaining the role of a developer. We do not show each individual's contribution to the project due to the privacy issues, and hence we have summarized the aggregated results of each project as shown in Table 4. All the variables (except n) used in our study represent the contribution of potential developers on yearly basis. For each *Apache* project, n represents the number of potential developers who have attained the role of a developer. The average yearly rate of contributions by a potential developer *before* and *after* attaining the role of a developer is calculated as follows:

$$\sum_{d \in \mathbf{Immig}} \frac{\text{Contribution}_{\text{before}}\,(d, \text{commitDate}\,(d))}{|d|}, \quad \sum_{d \in \mathbf{Immig}} \frac{\text{Contribution}_{\text{after}}\,(d, \text{commitDate}\,(d))}{|d|}. \tag{12}$$

Based on Table 4, we find that the bugs reporting pattern does not change much *before* and *after* attaining the role of a developer in *Apache Maven* and *Apache Solr* projects. However, in *Apache Ant*, it decreased tremendously after attaining the role of a developer. As shown in Figure 3, there are only few bugs reported by the developers in contrast to nondevelopers in the *Apache Ant* project which is also reflected by the value of `bugs reported` variable for the *Apache Ant* project. Members after joining the "core members" group participate more often in technical discussions on the bug tracking system which is reflected by the value of `bug comment` variable. However, an increase in the participation in technical discussions did not increase the social relation of the developers on the bug tracking system (i.e., `bug social relation`) in the case of *Apache Ant* and *Apache Maven* project. One reason could be that, after attaining the role of a developer, they focused only on certain modules of a project and hence involved in discussions on bugs relevant to those modules with other developers of the project. There is also a tremendous increase in the number of emails sent by the members after attaining the role of a developer which eventually increases the value of `email social relation` variable.

Table 4: Yearly average contribution ratio of a potential developer before and after attaining the role of a developer.

Variable	Mean		St. Dev	
	Before	After	Before	After
Apache Ant ($n = 13$)				
bugs reported	17.23	2.35	17.98	3.65
bug comments	41.44	39.02	37.66	34.83
bug social relation	44.65	31.57	51.41	23.34
emails	230.49	280.34	259	224.52
email social relation	30.5	20.31	30.35	13.55
Apache Lucene ($n = 22$)				
bugs reported	14.12	23.73	9.12	32.74
bug comments	24.84	73.21	18.18	91.85
bug social relation	33.57	34.67	23.75	30.02
emails	130	398.71	124.87	508.02
email social relation	19.59	21.69	16.53	15.32
Apache Maven ($n = 21$)				
bugs reported	28.40	27.82	53.27	79.49
bug comments	16.42	27.86	12.07	58.68
bug social relation	27.16	24.08	21.43	30.85
emails	158.41	185.55	218.27	174.89
email social relation	18.33	23.34	18.31	15.73
Apache Solr ($n = 13$)				
bugs reported	24.77	23.75	30.38	17.50
bug comments	44.72	84.77	41.35	86.25
bug social relation	68.04	80.46	52.18	52.39
emails	156.01	313.82	227.57	333.19
email social relation	14.72	38.47	7.85	33.04

Based on the *Apache* projects under consideration, we found that members after attaining the role of a developer tend to participate actively in technical discussions either on the mailing list or bug tracking system which also increases their social relation networks except the case of *Apache Ant* project. The bugs reporting behavior of these members varies in our studied *Apache* projects, and hence it is difficult to say if they report more bugs after attaining the role of a developer.

RQ-3: What is the average rate of contributions made by a potential developer comparing to other members of the project before attaining the role of a developer?

For each potential developer, we took the first time-stamp value where he first appears on the project and the second time-stamp value when he actually made the first commit

Table 5: Average contribution rate of a potential developer comparing to other members of the project before attaining the role of a developer.

Variable	Mean	St. dev
Apache Ant ($n = 13$)		
bugs reported	11.15	13.25
bug comments	10.43	8.35
bug social relation	8.42	6.65
emails	19.89	16.25
email social relation	10.97	8.42
Apache Lucene ($n = 22$)		
bugs reported	4.53	3.09
bug comments	4.08	2.89
bug social relation	3.18	1.73
emails	3.91	3.27
email social relation	6.23	4.42
Apache Maven ($n = 21$)		
bugs reported	4.05	3.68
bug comments	2.36	2.01
bug social relation	2.44	1.98
emails	4.05	5.76
email social relation	4.31	3.34
Apache Solr ($n = 13$)		
bugs reported	4.56	5.77
bug comments	4.4	4.72
bug social relation	2.52	2.15
emails	1.05	1.07
email social relation	2.71	1.95

to the source control repository of the project. We extracted the contributions (i.e., bugs reported, comments, emails, etc.) made by a potential developer between those time-stamp values. Using the same time-stamp values, we computed the contributions made by other members who were also active during that specific time period. Later, we divided the contributions of a potential developer by the average contributions of all other members in order to determine the average rate of contributions made by a potential developer comparing to other members of the project. We do not show each individual's contribution rate due to the privacy issues, and hence we have summarized the aggregated results of each project which is shown in Table 5. For example, the average rate of bugs reported by an immigrant comparing to other members who were active during the same time-stamp is calculated as follows:

$$\sum_{\text{Immig}\in\mathbf{Immig}} \frac{\text{Contribution}_{\text{bugs}}\left(\text{Immig}, \text{commitDate}\left(\text{Immig}\right)\right)}{\sum_{c\in\mathbf{C}} \text{Contribution}_{\text{bugs}}\left(c, \text{commitDate}\left(\text{Immig}\right)\right) / |c|}. \tag{13}$$

The results in Table 5 can be understand as follows: the average rate of reporting bugs by a potential developer of *Apache Lucene* project is `4.53` times the average rate of reporting bugs by all other members who were active during that time period. Although the average rate of contributions made by potential developers varies in all the projects under consideration, it is quite obvious from each variable value that the contributions made by potential developers are more than the average contributions of all other members. Hence, we can say that they were the most active contributors (i.e., technically skilled and higher social status) before attaining the developer status in the project.

Table 6: Appearance of a potential developer on different software repositories prior to attaining the role of a developer.

Apache projects	Patch submission (no. of days)	Bugs reported (no. of days)	Emails (no. of days)
Apache Ant	544.15	553.84	706.53
Apache Lucene	457.41	526.32	706.22
Apache Maven	385.00	396.31	709.71
Apache Solr	269.30	237.92	452.46

RQ-4: Does a potential developer follow onion model in order to attain the role of a developer?

For each potential developer, we computed the time-stamp value between his/her first commit date to his/her first activity on the different software repositories in terms of days. Table 6 presents the appearance of a potential developer in terms of average number of days on different software repositories prior to attaining the role of a developer. The result shows that all the potential developers started from the mailing list (cf. Table 6) because the email activity is the oldest for all *Apache* projects under consideration followed by the bugs reporting/commenting, and the latest activity before attaining the role of a developer was the source-code patch submissions (i.e., bugs fixing). The results shown in Table 6 closely match to the onion model (see Figure 1) where a member starts as a reader followed by reporting bugs and later fixing bugs before attaining the role of a developer.

Let `ActivityDate(Immig,mL)` return the number of days between the first commit date of an immigrant (i.e., potential developer) on the source control repository to his first activity date on the mailing list of a project. The average number of days for an immigrant to appear on a mailing list prior to his/her first commit date is calculated as follows:

$$\sum_{\text{Immig}\in\mathbf{Immig}} \frac{\text{ActivityDate}\left(\text{Immig}, \text{ml}\right)}{|\text{Immig}|}. \tag{14}$$

The results (Table 6) also show that it took almost 2 years for a potential developer of *Apache Ant, Apache Lucene*, and *Apache Maven* projects to attain the role of a developer. However, we cannot say that it is the standard time as the time varies dramatically from project to project as it can be seen in the results of *Apache Solr* project comparing to other *Apache* projects under consideration.

6. Conclusion

In this paper, we have investigated in detail the patterns of contributions made by those members who have attained

the role of a developer in the project. First, we investigated the significant role played by nondevelopers in the long-term survival of an OSS project and observed that nondevelopers who do not have write-access to the source control repository participate actively in reporting bugs and email discussions, thus contributing to the maturity of an OSS project. Our investigation based on the contribution of potential developers before and after attaining the role of a developer showed that after attaining a higher position in the community, developers tend to contribute more efficiently than nondevelopers of the project by actively participating in technical discussions along with fixing bugs. Moreover, we observed that the members who attained the role of a developer had more contributions in contrast to the average number of contributions made by other members of the project who were active during his/her time period. This makes it obvious that one of the important factors in order to attain the role of a developer is the demonstration of technical skills and commitment to the project in an efficient manner.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

References

[1] B. Shibuya and T. Tamai, "Understanding the process of participating in open source communities," in *Proceedings of the ICSE Workshop on Emerging Trends in Free/Libre/Open Source Software Research and Development (FLOSS '09)*, pp. 1–6, IEEE Computer Society, Washington, DC, USA, May 2009.

[2] G. Von Krogh, S. Spaeth, and K. R. Lakhani, "Community, joining, and specialization in open source software innovation: a case study," *Research Policy*, vol. 32, no. 7, pp. 1217–1241, 2003.

[3] C. Bird, A. Gourley, P. Devanbu, A. Swaminathan, and G. Hsu, "Open borders? Immigration in open source projects," in *Proceedings of the 4th International Workshop on Mining Software Repositories (MSR '07)*, Washington, DC, USA, May 2007.

[4] M. Antikainen, T. Aaltonen, and J. Väisänen, "The role of trust in OSS communities—case Linux Kernel community," *IFIP International Federation for Information Processing*, vol. 234, pp. 223–228, 2007.

[5] Y. Ye, K. Nakakoji, Y. Yamamoto, and K. Kishida, "The co-evolution of systems and communities in Free and Open Source Software Development," in *Free/Open Source Software Development*, pp. 59–82, Idea Group, Hershey, Pa, USA, 2004.

[6] http://ant.apache.org/.

[7] http://lucene.apache.org/.

[8] http://maven.apache.org/.

[9] http://lucene.apache.org/solr/.

[10] K. Crowston and J. Howison, "The social structure of free and open source software development," in *Proceedings of the International Conference on Information Systems*, Seattle, Wash, USA, 2003.

[11] C. Jensen and W. Scacchi, "Modelling recruitment and role migration process in oosd projects," in *Proceedings of the 6th International Workshop on Software Process Simulation and Modeling*, St. Louis, Mo, USA, 2005.

[12] A. Mockus, R. T. Fielding, and J. 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.

[13] T. Dinh-Trong and J. M. Bieman, "Open source software development: a case study of freeBSD," in *Proceedings of the 10th International Symposium on Software Metrics (METRICS '04)*, pp. 96–105, Washington, DC, USA, September 2004.

[14] N. Ducheneaut, "Socialization in an open source software community: a socio-technical analysis," *Computer Supported Cooperative Work*, vol. 14, no. 4, pp. 323–368, 2005.

[15] D. Cox and D. Oakes, *Analysis of Survival Data: Monographs on Statistics and Applied Probability*, Chapman and Hall, 1984.

[16] I. Herraiz, G. Robles, J. J. Amor, T. Romera, and J. M. G. Barahona, "The processes of joining in global distributed software projects," in *Proceedings of the International Workshop on Global Software Development for the Practitioner (GSD '06)*, pp. 27–33, 2006.

[17] W. Scacchi, J. Feller, B. Fitzgerald, S. Hissam, and K. Lakhani, "Understanding free/open source software development processes," *Software Process Improvement and Practice*, vol. 11, no. 2, pp. 95–105, 2006.

[18] M. Fischer, M. Pinzger, and H. Gall, "Populating a release history database from version control and bug tracking systems," in *Proceedings of the International Conference on Software Maintenance (ICSM '03)*, pp. 23–32, IEEE Computer Society, Washington, DC, USA, September 2003.

[19] S. Wasserman and K. Faust, *Social Network Analysis: Methods and Applications*, Cambridge University Press, 1994.

[20] A. Iqbal and M. Hausenblas, "Interlinking developer identities within and across open source projects: the linked data approach," *ISRN Software Engineering*, vol. 2013, Article ID 584731, 12 pages, 2013.

[21] C. Bird, A. Gourley, and P. Devanbu, "Detecting patch submission and acceptance in OSS projects," in *Proceedings of the 4th International Workshop on Mining Software Repositories (MSR '07)*, Washington, DC, USA, May 2007.

[22] A. Iqbal, O. Ureche, M. Hausenblas, and G. Tummarello, "LD2SD: linked data driven software development," in *Proceedings of the 21st International Conference on Software Engineering and Knowledge Engineering (SEKE '09)*, pp. 240–245, Boston, Mass, USA, July 2009.