PAPER

Investigating and Projecting Population Structures in Open Source Software Projects: A Case Study of Projects in GitHub

Saya ONOUE[†a)], **Hideaki HATA**[†b)], ***Nonmembers***, **Akito MONDEN**[††c)], ***Member***, ***and*** **Kenichi MATSUMOTO**[†d)], ***Fellow***

SUMMARY GitHub is a developers' social networking service that hosts a great number of open source software (OSS) projects. Although some of the hosted projects are growing and have many developers, most projects are organized by a few developers and face difficulties in terms of sustainability. OSS projects depend mainly on volunteer developers, and attracting and retaining these volunteers are major concerns of the project stakeholders. To investigate the population structures of OSS development communities in detail and conduct software analytics to obtain actionable information, we apply a demographic approach. Demography is the scientific study of population and seeks to identify the levels and trends in the size and components of a population. This paper presents a case study, investigating the characteristics of the population structures of OSS projects on GitHub, and shows population projections generated with the well-known cohort component method. We found that there are four types of population structures in OSS development communities in terms of experiences and contributions. In addition, we projected the future population accurately using a cohort component population projection method. This method predicts a population of the next period using a survival rate calculated from past population. To the best of our knowledge, this is the first study that applied demography to the field of OSS research. Our approach addressing OSS-related problems based on demography will bring new insights, since studying population is novel in OSS research. Understanding current and future structures of OSS projects can help practitioners to monitor a project, gain awareness of what is happening, manage risks, and evaluate past decisions.

key words: *OSS, Software Development Communities, Software Population Pyramids, Demography*

1. Introduction

As of 2014, GitHub reported having over 3.4 million users and 16.7 million repositories[*]. Why does GitHub attract so many developers? Several studies have identified the essence of this success [1]–[4]. GitHub is a distributed version control system (DVCS) and a web-based hosting service for Git repositories. Brindescu et al. assessed the differences between the centralized version control system (CVCS) and DVCS [1]. They reported that developers prefer DVCS because of its useful features, such as the ability to commit locally, work offline while retaining full project history, and create merging branches cheaply. Muşlu et al. reported that developers moved from CVCS to DVCS because DVCS has the ability to work offline, to work incrementally, and to context switch and do exploratory coding efficiently [2]. GitHub has tapped into the opportunity to facilitate pull-based development by offering workflow support tools, such as code reviewing systems and integrated issue trackers. Gousios et al. reported the impacts of pull-based development based on mining repository data: fast development, transparency in project management, attracting contributions, crowd sourcing of code review, and democratizing development [3]. GitHub is also considered as a developers' social networking service, and it promotes software development through formal and informal collaboration, called social coding. Dabbish et al. examined the value of transparency and collaboration in OSS, reporting that developers form a rich set of social inferences, including inferring technical goals and vision and trying to identify projects with similar, and developers combine these inferences into effective strategies for coordinating work, advancing technical skills, and managing their reputations [4].

Although GitHub has many attractive features and many users and repositories, most projects are inactive and have very few commits. Based on a qualitative manual analysis of GitHub repositories, Kalliamvakou et al. reported that the majority of the projects are personal and inactive [5]. To survive and succeed, software development communities need to attract and retain contributors. Yamashita et al. proposed a pair of population metrics, namely, magnetism and stickiness [6]. Magnet projects are defined as those that attract a large proportion of new contributors, and sticky projects as those where a large proportion of the contributors will continue to make contributions. With the two values of magnetism and stickiness, OSS projects are classified into the following four categories: (1) Attractive projects have high magnet and high sticky values. These projects are successful both at attracting new contributors and at retaining existing ones. (2) Fluctuating projects have high magnet but low sticky values. These projects are successful at attracting new contributors but unsuccessful at retaining them. Therefore, the members of these

Manuscript received September 4, 2015.
Manuscript revised January 4, 2016.
Manuscript publicized February 5, 2016.

[†]The authors are with Nara Institute of Science and Technology, Ikoma-shi 630-0192 Japan
[††]The authors is with Okayama University, Okayama-shi, 700-0082 Japan
a) E-mail: onoue.saya.og0@is.naist.jp
b) E-mail: hata@is.naist.jp
c) E-mail: monden@okayama-u.ac.jp
d) E-mail: matumoto@is.naist.jp

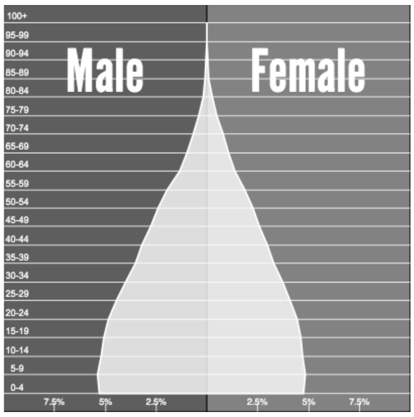
DOI: 10.1587/transinf.2015EDP7363

[*]Marisa Whitaker, "Former UC student establishes a celebrated website in GitHub that simplifies coding collaboration for millions of users," University of Cincinnati, April 2014, http://magazine.uc.edu/favorites/web-only/wanstrath.html.

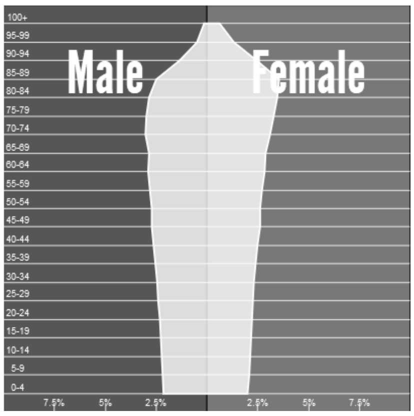
OSS development communities fluctuate from year to year. (3) Stagnant projects have low magnet but high sticky values. In contrast to fluctuating projects, stagnant projects retain existing contributors but cannot attract new ones. (4) Terminal projects have low magnet and low sticky values. Based on this classification, Yamashita et al. empirically studied OSS project histories and identified at-risk projects.

The work of Yamashita et al. suggests that we should go further to analyze moving human resources of OSS projects in more details not only for the evaluation of project sustainability but also for providing actionable information to help practitioners monitor a project, know what is really working, improve efficiency, manage risk, anticipate changes, and evaluate past decisions [7]. For example, if one could know that a project is attracting new contributors but loosing experienced contributors, then it can be considered that the project is changing its direction originally intended by the experienced contributors. For a straightforward way to analyze such moving human resources, this paper focuses on the populations of development communities. And, to conduct software analytics in populations of OSS development communities, we apply an approach taken from demography. Demography is the scientific study of population. Demographers seek to uncover the levels and trends in a population's size and components [8]. Every population has a different composition: the number and proportion of males and females in each age group. This structure can have considerable impact on the population's current and future social and economic situation. Government policymakers and planners worldwide use population projections to gauge future demand for services and to forecast future demographic characteristics. We believe this perspective, that is, demography for actionable information, is also important for OSS projects to manage sustainable development communities.

A population pyramid is a graphical illustration of the distribution of the various age groups in a population. Depending on the countries' conditions, the shape of population pyramids varies. Population pyramids are used to show the current status of a country's population and provide insights about political and social stability, as well as economic growth. Population projection is a powerful approach to discuss future populations [9]. In a previous study, we applied population pyramids to OSS development communities. We dubbed this approach software population pyramids [10]. In software population pyramids, contributors are grouped by their experiences in the communities. Extending the previous study [10], this paper investigates the characteristics of the population structures observed in OSS projects in GitHub by introducing demographic analysis. In addition, we project the future of population structures using the well-known cohort component method. We address the following research questions in this paper: What characteristics of population structures exist in OSS projects in GitHub, and can we project future population structures? The differences between this study and the Yamashita et al. study [6] can be summarized as follows:



(a) India in 2010



(b) Japan in 2050

Fig. 1 General population pyramid. (http://populationpyramid.net/)

- The study of Yamashita et al. is based on population migration metrics. Therefore, their research specialized in the migration and remaining of developers. In contrast, we introduced the demographic approach. Therefore, we can investigate population structures deeply and predict the future of development communities with a well-known method.
- Yamashita et al. considered developers to be authors of code changes, so they focused only on the commit and pull request activities. However, we are also interested in other contributors who send issues and comments. So we analyze other activities as well as commit and pull request activities, which makes it possible to understand development communities in detail.
- Our software population pyramids consist of various experience groups in a software development communities. Our method thus allows us to see long-term contributors, though the previous study did not distinguish between the experiences of individual developers.

2. Demographic Analysis

2.1 Population Structures

Age and sex are the most basic characteristics of a population. Every population has a different age and sex composition, and this population structure can have considerable impact on the population's current and future social and economic situation [8]. A population pyramid graphically displays a population's age and sex composition.

In a general population pyramid, the population is distributed along the horizontal axis, with males shown on the left and females on the right. The male and female populations are broken down into five-year age groups represented by horizontal bars along the vertical axis, with the youngest age groups at the bottom and the oldest at the top. The shape of the population pyramid gradually evolves over time, following trends in fertility, mortality, and international migration. We can understand the status of a country just by looking at population pyramids. Figure 1(a) shows the population pyramid of India in 2010. This pyramid is large toward bottom, a form that is common in developing countries. Population pyramids are also useful for predicting the future composition of a population. Figure 1(b) shows the projected population pyramid of Japan in 2050. It seems like a tower rather than a pyramid. This form is common in low birth rate and high longevity countries. Depending on the countries' status, the shape of population pyramids varies.

2.2 Software Population Pyramids

There are various contributors to the OSS project. There are, for example, bug reporters, commenters, reviewers, and coding contributors. All contributions and various contributors are important for OSS projects. For example, bug reporters assume an important role in improving the quality of OSS [11]. Also, developers can keep up motivation by getting some comments of thanks, admiration, or opinion. However, coding contributors, bug reporters, and commenters differ essentially. Contributors can comment or report bugs without a deep understanding of source code files, but coding contributors need to understand them. So, coding contributors are considered to be required to have specific skills, unlike bug reporters or commenters. Therefore, in this study, we distinguish coding contributors with other, non-coding contributors.

It is often the case that core developers contribute to both coding and non-coding activities. We identify individuals as coding contributors if the contributors have experienced code-related activities at least once in his/her existing period. If a contributor only has non-coding activities in a given period, he/she is regarded as a non-coding contributor. Then if the contributor begins code-related activities later, he/she will be classified as a coding contributor. To clarify such transitions, we call such contributors "moved contributors". End users play an important role in maintaining contributors' motivation [12]. However, because contributions of end users are not recorded in software repositories in general, our study do not consider them.

We have proposed software population pyramids: population pyramids of software development communities [10]. Contributors are considered to be the constituent member of the communities, and the contribution periods are regarded as existing periods or lifetimes. A software population pyramid consists of two back-to-back bar graphs, with the population plotted on the X-axis and experience on the Y-axis. The bar graph on the right shows coding contributors, and the bar graph on the left shows non-coding contributors in a particular population in three-month experience groups. In a general population pyramid, the populations are broken down into five-year age groups. However, we should make software population pyramids with shorter periods because the five-year length is too long for OSS projects. In our previous study, we analyzed software population pyramids in one year length[10]. However, we found that many contributors leave projects within a year. In addition, less than three months is too short for many projects to obtain enough data to draw a population pyramid. So, in this study, we make software population pyramids with three months groups, and analyze population of contributors in OSS projects.

There are some differences between our software population pyramids and the general population pyramids.

- Whereas a population pyramid consists of bars for males and females, a software population pyramid consists of coding bars for contributors and non-coding contributors.
- In a general population pyramid, people appear at birth and disappear when they die, but in a software population pyramid, contributors start their experiences when they enter and finish when they leave the development communities. In this study, we consider that a contributor left a project when he/she did not give any contribution on that project for more than three months. However, very few contributors might come back to the project after three-month (or more) interval. They disappear from the pyramids while they are inactive temporarily. In that case, we consider them as experienced contributors when they come back to the project.
- The height of general population pyramids are similar to each other, because maximal life-span of human is not so different in each country. However, software population pyramids have different heights, because OSS projects have different existing periods and people can leave freely.
- Because the parent-child relationships exist in population pyramids, there are correlations between the volume of the parent population and the population of children. However, software population pyramid do not exhibit such relationships. This can cause the pyramid to change dramatically.

Table 1 GitHub development activities

Development Activities	Overview	Separation
commits	Commit to the repository	coding
pull_requests	Request a commit to committers	
commit_comments	Comment against commit	non-coding
issues	An issue associated with a repository	
issue_comments	Comment against commit	
pull_request_comments	Comment against commit pull_request	
events	This is a read-only API to the GitHub events.	excluded in this study
followers	A follower to a user.	
forks	A copy of a repository	
org_members	Users that are members of an organization.	
repo_collaborators	Users with access to the repository.	
repo_labels	Label list is labeled to the repositories	
repos	A dump of every public repository	
issues_events	An event on an issue	
users	Github users.	
watchers	Users that have starred (was watched) a project	

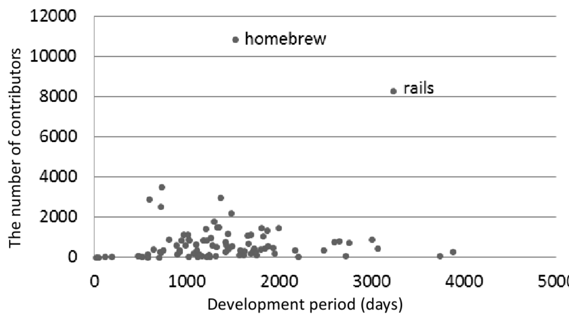


Fig. 2 Distribution of development periods and the number of contributors.

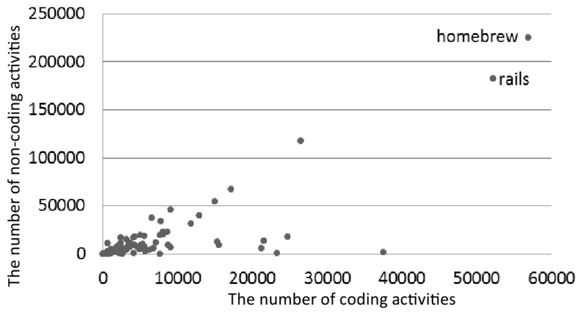


Fig. 3 Distribution of the number of coding contributors and the number of non-coding contributors.

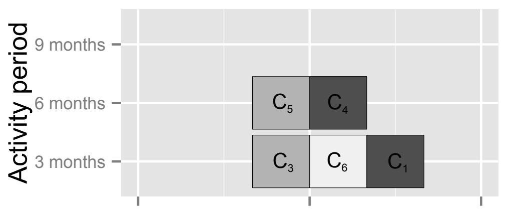
2.3 Datasets

We analyze the GitHub dataset provided by Gousios [13]. This dataset includes developers' activity histories for 90 OSS projects. Figures 2 and 3 show point diagrams that plot metrics of projects. Figure 2 shows the distribution of development periods and the number of contributors. From Figure 2, we can see that homebrew has many contributors and that the development period of rails is long. Figure 3 shows the distribution of the number of coding contributors and the number of non-coding contributors by project. From Figure 3, we see that homebrew and rails have many coding and non-coding contributors. From small to large-scale projects, this dataset includes various types of projects. In total, this dataset includes 16 development activities, but to focus on contributors' activities, we use only six development activities. Table 1 shows the name of 16 development activities and an explanation of the content of these activities. Pull_request and commits are considered to be coding-related activities, whereas commit_comments, issue_comments, pull_request_comments and issue_events are considered non-coding activities. Events, followers, org_members, repo_collaborators, repo_labels, repos, users and watchers are not related to contributors' activities. "Forks" is generally a contributor's activity; however, fork itself does not contribute to the development, and also fork is often done by a person before he/she participate in a development as a contributor. So we excluded it from the contributors' activity list.

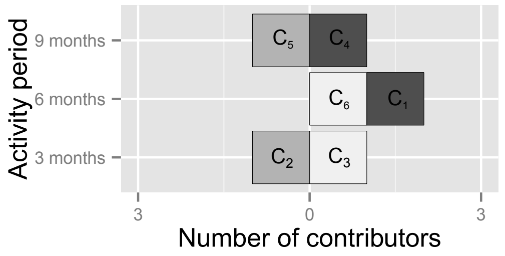
Table 2 Example of data of activity and activity periods of contributors in t_1 and t_2.

	Pyramids in t_1		Pyramids in t_2	
	Working months		Working months	
Contributor	coding	non-coding	coding	non-coding
C_1	1	-	4	-
C_2	-	-	-	2
C_3	-	3	2	6
C_4	5	2	8	5
C_5	-	4	-	7
C_6	3	5	6	8

We classified contributors as coding and non-coding contributors. **Coding** contributors are contributors who have at least one code-related activity in their existing periods. **Non-coding** contributors are contributors who have not experienced code-related activities but have experienced non-coding activities. We obtained the dates of those events for each contributor, and identified the contribution period from the first event until the last event. Details of how to obtain the data are explained in Appendix A. Contribution periods are divided into **coding periods** and **non-coding periods** based on the classification of the activity events. If a contributor has only non-coding activities in his/her early period, the period is regarded as a non-coding period and he/she is regarded as a non-coding contributor. If a contributor has coding-related activities, the period is regarded as a coding period and he/she is regarded as a coding contributor.



(a) Pyramids in t_1.



(b) Pyramids in t_2.

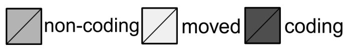


Fig. 4 Examples of software population pyramids in t_1 and t_2

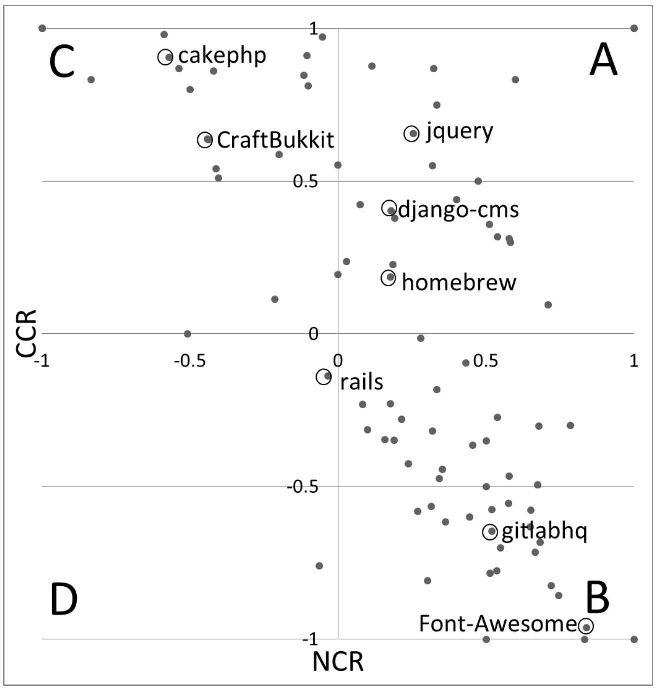


Fig. 5 Distribution of CCR and NCR of OSS projects in GitHub.

Table 2 shows an example of data of activity and activity periods of contributors, and Figure 4 shows software population pyramids that plot the data of Table 2. The time t_2 is three months later to the time t_1. The X-axis is the number of contributors. The center is zero, the right side shows the number of coding contributors, and left side shows non-coding contributors. The Y-axis is the activity period of contributors. For example, contributor C_1 has coding activity periods of one month in t_1 and four months in t_2. Therefore, he/she is plotted as C_1 in location in Figure 4 (a) and Figure 4 (b) as a coding contributor. Contributor C_2 has a non-coding activity period of two months in t_2. He/she is plotted as C_2 in location in Figure 4 (b) as a non-coding contributor. In contrast, contributor C_3 has a non-coding activity period of three months in t_1. He/she is plotted as C_3 in location in Figure 4 (a) as a non-coding contributor. However, he/she has a coding activity period of two months in t_2. Therefore, he/she is plotted as C_3 in location in Figure 4 (b), having moved to the contributor side.

The method of calculating activity periods used here does not take into account actual activity between start and end. In a previous study, we analyzed the frequency of activities of contributors, finding that, although some contributors continued to make small contributions for long periods, there is no contributor that stops activities in a project and then rejoins the project later [14]. However, it is important to take into account the frequencies of contributions. This could be the future work of this study.

3. Characteristics of Population Structures

We classify the shapes of software population pyramid, and investigate their characteristics. For this purpose, we propose two new measures. One is the proportion of the number of non-coding contributors (non) to the number of coding contributors (coding), called the Coding Contributors Ratio (CCR). CCR is defined as follows:

$$CCR = \begin{cases} \dfrac{coding - non}{coding} & (coding \geq non) \\ \dfrac{coding - non}{non} & (coding < non) \end{cases}$$

CCR ranges from −1 to 1. Higher values mean that more contributors are coding contributors, and lower values mean that more contributors are non-coding contributors. If the value is close to 0, the number of coding contributors and the number of non-coding contributors are similar.

The other proposed measure is the proportion of the number of experienced contributors to the number of newcomers (new contributors), called the New Contributors Ratio (NCR). In this study, we define newcomers as contributors who have less than three months of activity periods, and we define experienced contributors as those with longer activity periods. NCR is defined as follows:

$$NCR = \begin{cases} \dfrac{new - experience}{new} & (new \geq experience) \\ \dfrac{new - experience}{experience} & (new < experience) \end{cases}$$

NCR ranges from −1 to 1. Higher values mean that more contributors are new contributors, and lower values mean that more contributors are experienced contributors. If the value is close to 0, the number of new contributors and the number of experienced contributors are similar.

Figure 5 shows the distribution of the projects using the CCR and the NCR in September 2013. Because four

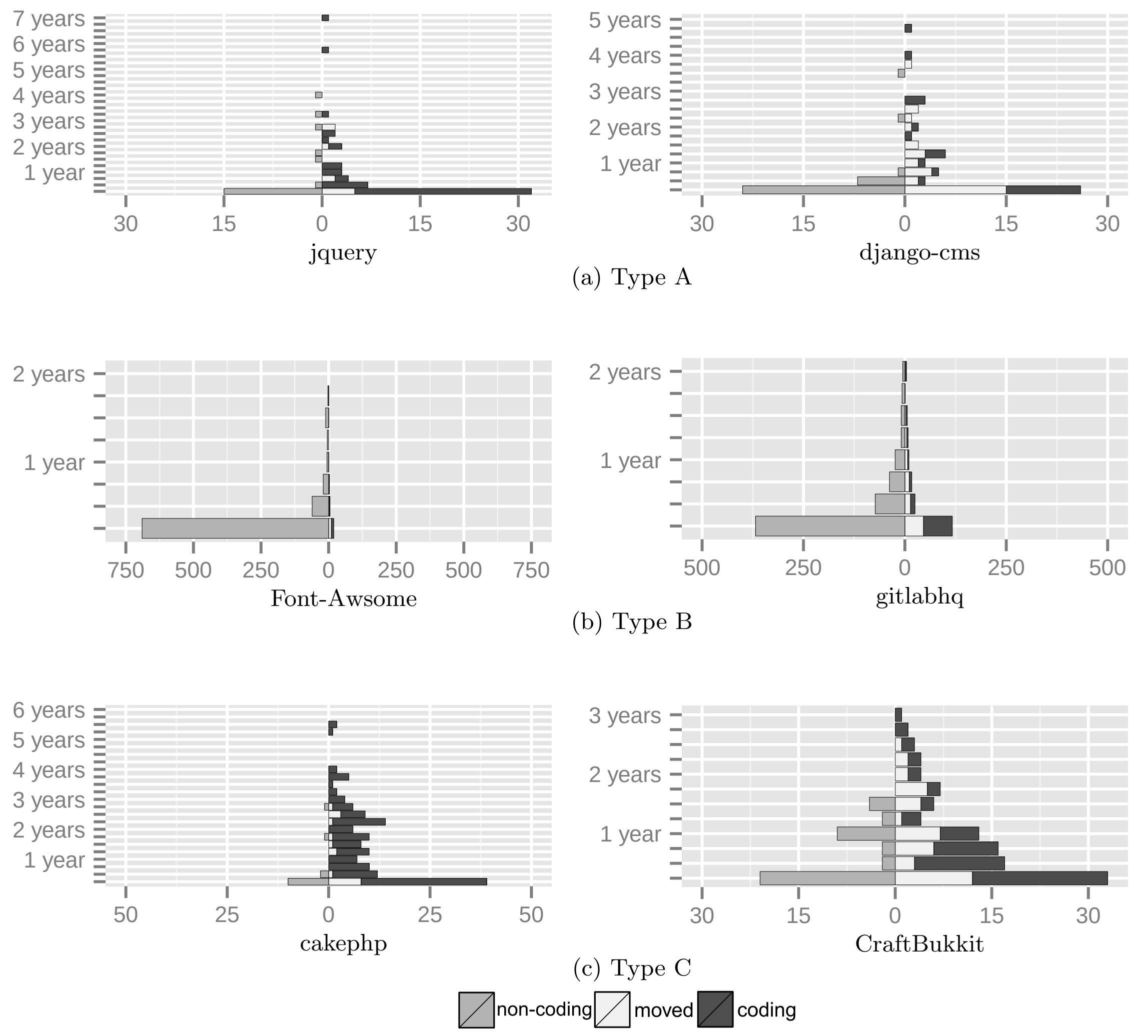


Fig. 6 Examples of software population pyramids of each type. (Note that scales are different)

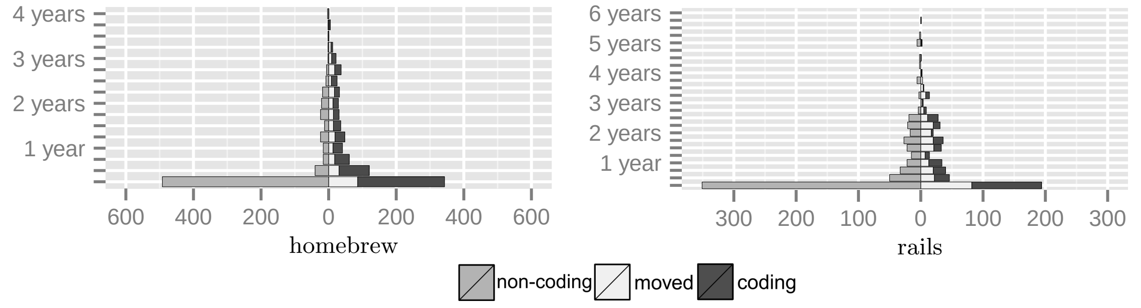


Fig. 7 Examples of software population pyramids (CCR and NCR are close to 0). Scales are different.

projects did not have any contributors in this period, we could not plot them. For that reason, there are 86 projects displayed in Figure 5. With this distribution, we can classify the projects into the following four types:

- Type A: There are more newcomers than experienced contributors, and more coding contributors than non-coding ones in a project. So, the shape of software population pyramid on the right side is larger than the left side, and the bottom is larger than others, also experienced contributors are plotted intermittently.
- Type B: There are more newcomers than experienced contributors, and more non-coding contributors than coding ones in a project. So, the shape of software population pyramid on the right side is larger than the left side,and the bottom part is larger than other parts. Also, experienced contributors are plotted intermittently.
- Type C: There are more experienced contributors than newcomers, and more coding contributors than non-coding ones in a project. So, the shape of software population pyramid on the left side is larger than the right side, and the bottom part is larger than other parts. Also, experienced contributors are plotted intermittently.
- Type D: There are more experienced contributors than newcomers, and more non-coding contributors than coding ones in a project. So, the shape of software population pyramid on the left side is larger than the right side, and experienced contributors are plotted continuously.

There are 23 projects categorized Type A, 42 projects as Type B, 18 projects as Type C, and three projects as Type D.Figure 6 presents examples of software population pyramids belonging to Type A, Type B, and Type C.

(a) Type A In these projects, there are few experienced non-coding contributors. In django-cms, there are many moved contributors. Because many developers moved from non-coding to coding, these projects have many coding contributors.

(b) Type B Font-Awesome has many non-coding contributors. This project makes Web icon fonts, and many people sent requests for new icons to this project. Therefore, many non-coding contributors leave this project immediately following a short period of contribution.

(c) Type C There are many moved contributors in CraftBukkit. Also, there are many coding newcomers. However, many coding contributors continue their activities, because there are more experienced contributors than there are newcomers.

In Figure 6, we see that the shapes of the pyramids are different from each other. OSS projects are managed by voluntary contributors, so contributors may not correspond to the many bug reports in projects of Type B. Additionally, it is difficult to obtain coding newcomers, because there are no moved newcomers. However, there are a few contributors that report bugs, such as in Type A or C, so these projects have little chance of improving the quality of the OSS through bug reports.

Figure 7 shows the software population pyramids of homebrew and rails. Homebrew belongs to Type A, and rails belongs to Type D. In these projects, both CCR and NCR values are close to 0. These projects are continually gaining contributors because their software population pyramids do not have intermittent bars. In addition, there are many moved contributors. We can see that these projects succeeded in attracting and retaining new/experienced and coding/non-coding contributors.

Projects that are plotted close to the center of the graph are well balanced in CCR and NCR. In these projects, there are an almost equal the number of contributors and newcomers, and almost equal the number of non-coding contributors and coding ones too. It is our important future work to consider adding another (5th) project type to distinguish such projects from others. For example, if we distinguish the projects that plotted around the origin belonging to the top 10% and others, six project such as homebrew, rails, bitcoin, diaspora, openFrameworks and redis meet that definition.

4. Population Projection

For project managers, it is important to maintain experienced contributors. Therefore, we propose a population projection method of the number of contributors in OSS projects using demographic methods.

4.1 Cohort component population projection

We predict the number of contributors using a simplified cohort component population projection. In demography, a cohort is a group of subjects share a particular event during a particular time span. Cohort component population projection is the simplest population projection method. Isserman offers a way to project the size of populations [15]. Isserman's method uses the survival rate [16], as well as fertility, mortality, and migration data. We can project the size of populations at a certain age cohort using following formulas:

$$\begin{aligned} &Population\ of\ age\ (X+n)\ in\ year\ (T+n) = \\ &Survival\ Rate \times Population\ of\ age\ X\ in\ year\ T \end{aligned}$$

where

$$\begin{aligned} &Survival\ Rate = \\ &\frac{Population\ of\ age\ (X+n)\ in\ year\ T}{Population\ of\ age\ X\ in\ year\ (T-n)} \end{aligned}$$

X is the age of the cohort being examined, n is an interval of time usually set at ten years representing the period of time between the two most recent censuses, and T is the year of the most recent census. We replace each variable in our software population pyramids such that X is the activity period of the cohort being examined, n is an interval of time set at three months representing the

Table 3 Median of ABRE.

	non-coding		moved		coding		All	
	cohort	baseline method	cohort	baseline method	cohort	baseline method	cohort	baseline method
Type A	0.4993	0.5000	0.3027	0.5000	0.1865	0.3731	0.2534	0.5000
Type B	0.5000	0.6332	0.3923	0.5000	0.5000	0.5750	0.5000	0.5917
Type C	0.6711	1.0000	0.2500	0.7179	0.3684	0.6250	0.3333	0.7500
Type D	0.3137	1.0000	0.2378	0.2500	0.4808	0.6154	0.2875	0.6667
All types	0.5000	0.6667	0.3299	0.5000	0.4074	0.5417	0.4000	0.6000

period of time between the two most recent contributors counting, and *3 m in year T* is the month of most recent contributors counting.

For example, we consider a case of a projection 10 to 19 year-old population in 2020. In this projection, we use 0 to 9 year-old population in 2000 and 10 to 19 year-old population in 2010 to calculate a survival rate of 0 to 9 year-old population. Here, the survival rate is calculated as follows.

$$Survival\ Rate = \frac{Population\ of\ age\ (10\ to\ 19)\ in\ 2010}{Population\ of\ age\ (0\ to\ 9)\ in\ 2000}$$

where

$$Population\ of\ age\ (10\ to\ 19)\ in\ 2020 = Survival\ Rate\ of\ 0\ to\ 9 \times Population\ of\ age\ (0\ to\ 9)\ in\ 2010$$

In this way, to calculate the population of each cohort and to sum them. The cohort component method includes birth and net migrants in general. Births are the same as newcomers to an OSS project in our study. Births are derived from the number of mothers and the birth rate. However, these input data do not exist for the number of contributors, so we use following very simple formula to calculate newcomers:

$$newcomer = (P\_\{T\} + P\_\{T - n\}) / 2$$

where

$$P\_\{T\} = Population\ activity\ period\ 3\ m\ in\ year\ T$$

Additionally, we do not consider that contributors move to other projects in our study, so we do not calculate net migration.

4.2 Evaluation

With the cohort component population projection method, we project a future population size for the 36 projects that have more than 100 contributors. There are four projects categorized as Type A, 21 projects as Type B, nine projects as Type C, and two projects as Type D. In this study, we project the number of contributors of September 2013 by calculating the survival rate from the number of contributors of March and June 2013. In order to verify the projection accuracy of our proposed method, we compared it with the baseline method, which assumes that the number of contributors of September and June 2013 are the same. Populations are projected for non-coding, moved, and coding contributors, separately.

To evaluate the projection accuracy, we compare the projection error of our propose method to the baseline method one. MRE (Magnitude of Relative Error) [17] or MER (Magnitude of Error Relative to estimate) [18] are used to evaluate the prediction accuracy. We use ABRE (Absolute Balanced Relative Error) [19] as an evaluation metric of the prediction accuracy for the number of contributors remaining.

Measured values of the number of contributors is denoted as x, and the predicted value of the number of contributors is denoted as $\widehat{x}$. Each indicator is determined by the following equation:

$$MRE = \frac{|x - \widehat{x}|}{x}$$

$$MER = \frac{|x - \widehat{x}|}{\widehat{x}}$$

$$ABRE = \begin{cases} \dfrac{|\widehat{x} - x|}{x} & (\widehat{x} - x \geq 0) \\ \dfrac{|\widehat{x} - x|}{\widehat{x}} & (\widehat{x} - x < 0) \end{cases}$$

For these metrics, lower values indicate higher accuracy. MRE is the relative error of the predicted value to the actual value and MER is the relative error between predicted and actual values to the predicted value. However, MRE and MER share the problem that these measures cannot distinguish excessive prediction and too little prediction. In this study, we evaluated projection accuracy by using the ABRE to evaluate the balance between excessive prediction and too little prediction.

To investigate the projection accuracy, we used the Wilcoxon non-parametric statistical hypothesis test. Wilcoxon test is generally used when comparing two related samples to assess whether their population mean ranks differ. It can be used as an alternative to the paired Student's t-test, t-test for matching pairs, or the t-test for dependent samples when the population cannot be assumed to be normally distributed. With the Wilcoxon test, we test the difference between the ABREs of our propose method and the ABREs of the baseline method. In particular, we test each project type (Type A-D) and each contribution type (non-coding, moved, coding). Then, we test their measures of central tendency, and investigate whether there are significant differences in projection accuracy.

Table 3 shows the median of the ABREs. If the

Table 4 Result of Wilcoxon test.

	p-value			
	non-coding	moved	coding	All
Type A	**0.02748**	**0.00158**	0.26867	**0.00014**
Type B	**0.00037**	**0.01845**	0.06200	**0.00001**
Type C	0.05935	**0.00035**	0.16000	**0.00013**
Type D	**0.00001**	**0.02700**	**0.02901**	**0.00000**
All types	**0.00000**	**0.00000**	**0.00116**	**0.00000**

ABRE value is close to 0, the projection accuracy is high. In Table 3, projection accuracies of our proposed method are higher than the baseline method in all predictions. Table 4 shows the result of the Wilcoxon test (95% confidence), where bold numbers indicate the statistical significant improvements by the proposed (cohort) method. In Table 4, projection of non-coding contributors has no significant difference in Type C, and projections of coding contributors have no significant difference in Type A, B, and C. However, projection of all contributors and all types have significant difference. In this result, the projection accuracy of our proposed method was higher than the baseline method. On the other hand, there was no difference in the predictive accuracy between different project types in this projection. This result shows the possibility that the reduction of contributors depends little on type of activities, the number of newcomers or experienced contributors.

Figure 8 shows actual software population pyramids and lines of predicted values. We can see that most of the lines are surprisingly well fitted to the observed data, especially near the top of each pyramid. In this study, we define short-term contributors as contributors that have activity period of less than one year, and define long-term contributors as contributors that have activity period of equal to or more than one year. The median of ABRE of short-term contributors is 0.4055, and median of ABRE of long-term contributors is 0.3333. The result of the Wilcoxon rank test (95% confidence) showed that the difference is significant (p-value = 0.0460), which indicates that the projection of the number of long-term contributors is higher accuracy than the projection of the number of short-term contributors.

5. Related Work

Web services for software developers, such as GitHub[†] and Open Hub[††], are popular. Therefore, we can easily understand the activity of contributors in OSS. Dabbish et al. asserted that exposing the activity of contributors through social network services is likely to rapidly promote cooperation and learning in OSS development. They interviewed GitHub users and examined cooperation and learning in the OSS community [4]. They found that contributors build a set from the activity information in GitHub, such as consulting someone else's technique when they edit code.

There are many studies that focus on the social aspects of software development. Bogdan noted the success of an OSS project depends to a large extent on the social aspects. He sought to increase understanding of how human aspects, gamification, and social media influence OSS [20]. Phillips et al. considered the building of team effectiveness and found that many challenges are social, not technical. They applied insights from group dynamics and organizations to inform the design of engineering tools and practices to improve the building of team effectiveness [21].

We considered that homebrew and rails are successful projects. However, the success of software development cannot be decided easily. Ralph et al. interviewed an interdisciplinary sample of 191 design professionals concerning their perceptions of software engineering success. They concluded that stakeholder impacts are driven by project efficiency, artifact quality, and market performance [22]. However, we consider the success of OSS to be related to contributors as well.

There are studies about prediction of contributors and the continuity of contributors' activity. Steinmacher et al. argued that the sustainability of many OSS projects relies on retaining newcomers. They discussed some barriers faced by newcomers to OSS [6]. Additionally, they found that 20% of new contributors become long-term contributors [23].

Rastogi et al. presented a framework that characterizes the stability of the community in software maintenance projects using community participation patterns. They modeled community participation patterns of contributors and forecast future behavior to help plan and support informed decision making [24]. Also, Loyala et al. proposed a methodology that adapts Lotka-Volterra-based biological models used for describing host-parasite interactions to understand how the population of OSS contributors evolves over time. Their experiments showed that the proposed approach performs effectively in terms of providing an estimation of the population of developers for each project over time [25].

Zhou and Mockus studied long-term contributors (LTC), analyzing the behavior of individual participants in Gnome and Mozilla [26]. They reported that future LTCs tend to be more active and show more community-oriented attitudes than do other joiners during their first month.

Social network analysis is a research area related to this study. For example, Bird et al. reported that developers play a significant social role in email lists [27]. Similarly, Bird et al. analyzed email addresses in OSS projects to examine the community structure among developers [28].

Onoue et al. studied the characteristics of developers' activities, finding that various developers Are characterized by different types of development activities [14].

Hata et al. studied the characteristics of sustainable OSS projects through game theoretical analysis and empirical analysis[29]. They reported that to have coding developers, that is, to incentivize developers to code, the projects should prepare documents, or the projects or third parties should hire developers. These strategies were in fact implemented by sustainable projects in GitHub.

[†]GitHub https://github.com/

[††]Open Hub https://www.openhub.net/

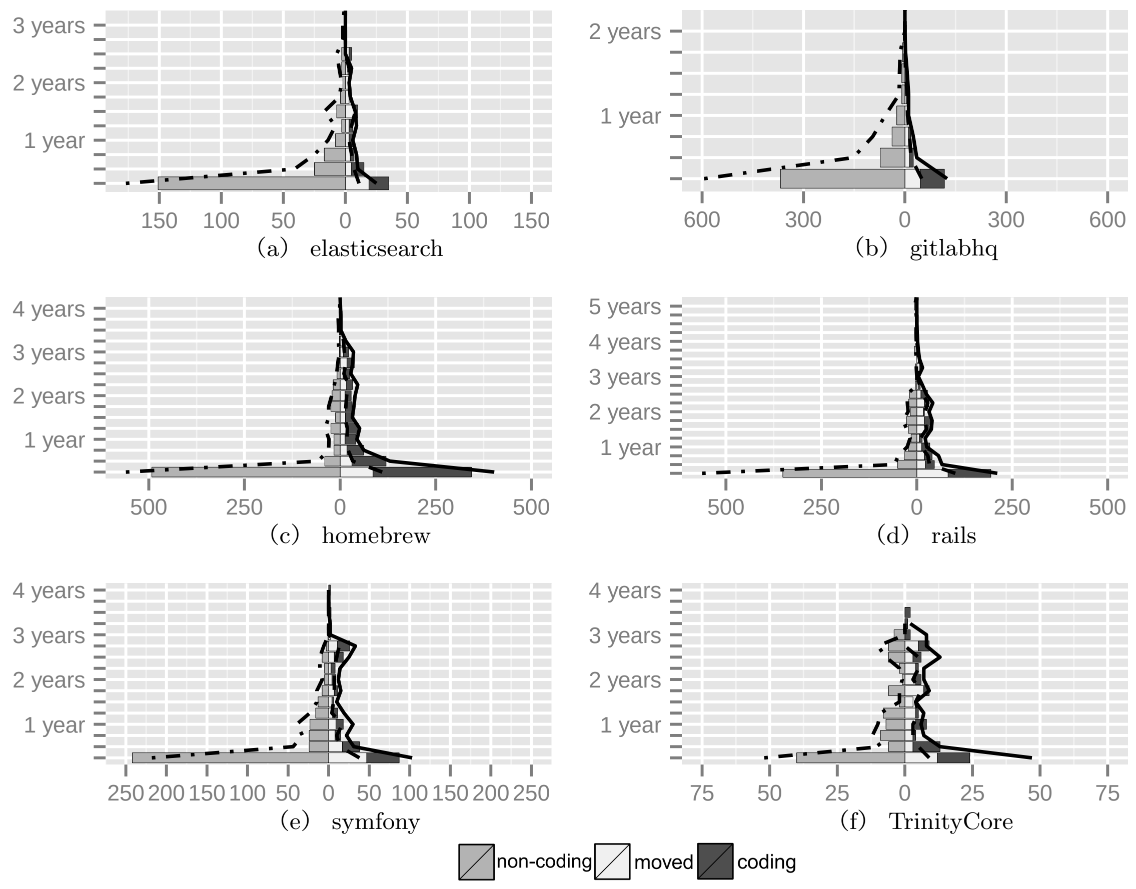


Fig. 8 Comparing measured and predicted values of the number of contributors.

6. Discussion and Conclusion

In this study, we focus on contributors to OSS projects using a demographic approach. In OSS projects, many people are involved in development, meaning that human resources are very important for OSS. We conclude that we can predict the future of participation in OSS projects by analyzing them from a demographic perspective.

In the field of demography, researchers create population pyramids to analyze the current situation of selected countries. We proposed a population pyramid for OSS projects called the software population pyramid. Contributors are considered the constituent member of the communities, and the contribution periods are regarded as experience periods or lifetimes. A software population pyramid consists of two back-to-back bar graphs, with the population plotted on the X-axis and experience on the Y-axis. One of the bar graphs shows coding contributors and the other shows non-coding contributors in a particular population in three-month experience groups.

We classified shapes of the software population pyramid and compared them. To classify the shape of these pyramids, we proposed two new measures, CCR and NCR. CCR is the proportion of the number of non-coding to the number of coding contributors, and NCR is the proportion of the number of experienced contributors to the number of newcomers. Using these measures, we classified 86 software population pyramids into four types as follows.

- Type A: There are more newcomers than experienced contributors, and more coding contributors than non-coding ones in a project.
- Type B: There are more newcomers than experienced contributors, and more non-coding contributors than coding ones in a project.
- Type C: There are more experienced contributors than newcomers, and more coding contributors than non-coding ones in a project.
- Type D: There are more experienced contributors than newcomers, and more non-coding contributors than coding ones in a project.

There were 23 projects categorized as Type A, 42 projects as Type B, 18 projects as Type C, and three projects as Type D. The result indicates that, for projects in Type A and Type B, contributors do not stay long time in their projects after their contributions. For projects of Type C and Type D, they can not get enough newcomers; thus, they should consider how to recruit newcomers. So, those projects should consider how to get newcomers. Especially, Type C projects should attract non-coding newcomers, e.g. who post many issues, so that some of them might become coding newcomers as well.

Through empirical research, we found that the shapes and the transitions of software population pyramids vary depending on the status of the development communities. However, it is difficult to clarify the components of the contributor population of OSS projects using only these values. Other, new metrics are needed to clarify contributors' components in OSS projects. For example, the number of activities or frequency of activities should be considered.

The demographic approach of population projection is a powerful way to predict future population dynamics. In this study, we projected the number of contributors of September 2013 using the simplified cohort component population projection that calculates the survival rate from the number of contributors of March and June 2013. In order to verify the projection accuracy of our proposed method, we compare it with the baseline method, which assumes that the number of contributors of September and June 2013 are the same. To statistically compare the projection accuracy, we used the Wilcoxon non-parametric statistical hypothesis test. As a result, the projection accuracy of our proposed method was higher than the baseline method. However, this projection method cannot predict long-term contribution patterns because it does not predict newcomers in a narrow sense. Therefore, our future work includes improving the accuracy of these predictions and expanding the prediction to account for newcomers and to extend predictions into the long-term future. We believe this perspective is also important for OSS projects to manage sustainable development communities.

Acknowledgments

This work has been supported by JSPS KAKENHI Grant Number 26540029 and Program for Advancing Strategic International Networks to Accelerate the Circulation of Talented Researchers: Interdisciplinary Global Networks for Accelerating Theory and Practice in Software Ecosystem.

References

[1] C. Brindescu, M. Codoban, S. Shmarkatiuk, and D. Dig, "How do centralized and distributed version control systems impact software changes?," Proc. of 36th Int. Conf. on Softw. Eng., ICSE 2014, New York, NY, USA, pp.322–333, ACM, 2014.

[2] K. Muşlu, C. Bird, N. Nagappan, and J. Czerwonka, "Transition from centralized to decentralized version control systems: A case study on reasons, barriers, and outcomes," Proc. of 36th Int. Conf. on Softw. Eng., ICSE 2014, New York, NY, USA, pp.334–344, ACM, 2014.

[3] G. Gousios, M. Pinzger, and A.v. Deursen, "An exploratory study of the pull-based software development model," Proc. of 36th Int. Conf. on Softw. Eng., ICSE 2014, New York, NY, USA, pp.345–355, ACM, 2014.

[4] L. Dabbish, C. Stuart, J. Tsay, and J. Herbsleb, "Social coding in github: Transparency and collaboration in an open software repository," Proc. of 2012 ACM Conf. on Comput. Supported Cooperative Work, CSCW '12, New York, NY, USA, pp.1277–1286, ACM, 2012.

[5] E. Kalliamvakou, G. Gousios, K. Blincoe, L. Singer, D.M. German, and D. Damian, "The promises and perils of mining github," Proc. of 11th Work. Conf. on Mining Softw. Repositories, MSR '14, New York, NY, USA, pp.92–101, ACM, 2014.

[6] K. Yamashita, S. McIntosh, Y. Kamei, and N. Ubayashi, "Magnet or sticky? an oss project-by-project typology," Proc. of 11th Work. Conf. on Mining Softw. Repositories, MSR '14, New York, NY, USA, pp.344–347, ACM, 2014.

[7] R.P.L. Buse and T. Zimmermann, "Information needs for software development analytics," Proc. of 34th Int. Conf. on Softw. Eng., ICSE '12, Piscataway, NJ, USA, pp.987–996, IEEE Press, 2012.

[8] A. Haupt, T. Kane, and C. Haub, PRB's Population Handbook 6th ed., Population Reference Bureau, 2011.

[9] S. Pennec, APPSIM-Cohort component population projections to validate and align the dynamic microsimulation model APPSIM, National Centre for Social and Economic Modelling, 2009.

[10] S. Onoue, H. Hata, and K. Matsumoto, "Software population pyramids: The current and the future of oss development communities," Proc. of 8th ACM/IEEE Int. Symp. on Empirical Softw. Eng. and Measurement, ESEM '14, New York, NY, USA, pp.34:1–34:4, ACM, 2014.

[11] E. Raymond, The cathedral and the bazaar : musings on linux and open source by an accidental revolutionary, " O'Reilly Media, Inc.", Sebastapol, CA, O' Reilly Media, Oct. 1990.

[12] M. Zhou and A. Mockus, "Does the initial environment impact the future of developers?," Proc. of 33rd Int. Conf. on Softw. Eng., ICSE '11, New York, NY, USA, pp.271–280, ACM, 2011.

[13] G. Gousios, "The ghtorent dataset and tool suite," Proc. of 10th Work. Conf. on Mining Softw. Repositories, MSR '13, Piscataway, NJ, USA, pp.233–236, IEEE Press, 2013.

[14] S. Onoue, H. Hata, and K.I. Matsumoto, "A study of the characteristics of developers' activities in github," Proc. of 5th Int. Workshop on Empirical Softw. Eng. in Practice, IWESEP '13, pp.7–12, Dec 2013.

[15] A.M. Isserman, The Right People, the Right Rates, Journal of the American Planning Association 59.1, 1993.

[16] J.C. Raymondo, Survival Rates: Census and Life Table Methods, Population Estimation and Projection, Quorum Books, New York, 1992.

[17] S.D. Conte, H.E. Dunsmore, and V.Y. Shen, Software Engineering Metrics and Models, Benjamin-Cummings Publishing Co., Inc., Redwood City, CA, USA, 1986.

[18] B. Kitchenham, L. Pickard, S. MacDonell, and M. Shepperd, "What accuracy statistics really measure [software estimation]," Software, IEE Proceedings -, vol.148, no.3, pp.81–85, Jun 2001.

[19] Y. Miyazaki, M. Terakado, K. Ozaki, and H. Nozaki, "Robust regression for developing software estimation models," J. Syst. Softw., vol.27, no.1, pp.3–16, Oct. 1994.

[20] B. Vasilescu, "Human aspects, gamification, and social media in collaborative software engineering," Companion Proceedings of the 36th International Conference on Software Engineering, ICSE Companion 2014, New York, NY, USA, pp.646–649, ACM, 2014.

[21] S. Phillips, T. Zimmermann, and C. Bird, "Understanding and improving software build teams," Proceedings of the 36th International Conference on Software Engineering, ICSE 2014, New York, NY, USA, pp.735–744, ACM, 2014.

[22] P. Ralph and P. Kelly, "The dimensions of software engineering success," Proceedings of the 36th International Conference on Software Engineering, ICSE 2014, New York, NY, USA, pp.24–35, ACM, 2014.

[23] I. Steinmacher, I.S. Wiese, T. Conte, Gerosa, and M. Aurélio, "Why do newcomers abandon open source software projects?," Proc. of 6th Int. Workshop on Cooperative and Human Aspects of Softw. Eng., CHASE '13, pp.25–32, May 2013.

[24] A. Rastogi and A. Sureka, "What community contribution pattern says about stability of software project?," Software Engineering Conference (APSEC), 2014 21st Asia-Pacific, pp.31–34, Dec 2014.

[25] P. Loyola and I. Ko, "Population Dynamics in Open Source Communities: An Ecological Approach Applied to Github," Proceedings of the 23rd International Conference on World Wide Web, WWW '14 Companion, pp.993–998, April 2014.

[26] M. Zhou and A. Mockus, "What make long term contributors: Willingness and opportunity in oss community," Proc. of 34th Int. Conf. on Softw. Eng., ICSE '12, Piscataway, NJ, USA, pp.518–528, IEEE Press, 2012.

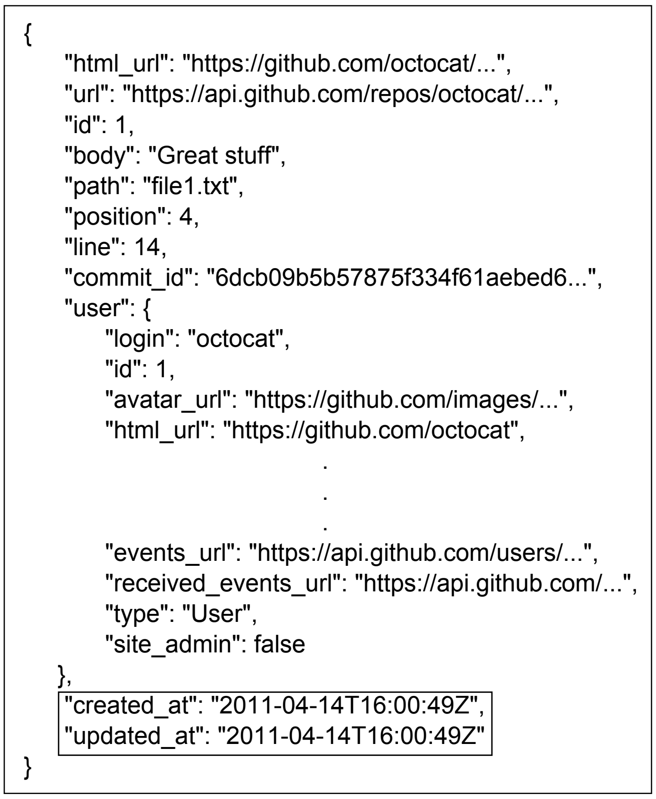
[27] C. Bird, A. Gourley, P. Devanbu, M. Gertz, and A. Swaminathan, "Mining email social networks," Proc. of 3rd Int. Workshop on Mining Softw. Repositories, MSR '06, New York, NY, USA, pp.137–143, ACM, 2006.

[28] C. Bird, D. Pattison, R. D'Souza, V. Filkov, and P. Devanbu, "Latent social structure in open source projects," Proc. of 16th ACM SIGSOFT Int. Symp. on Found. of Softw. Eng., SIGSOFT '08/FSE-16, New York, NY, USA, pp.24–35, ACM, 2008.

[29] H. Hata, T. Todo, S. Onoue, and K. Matsumoto, "Characteristics of sustainable oss projects: A theoretical and empirical study," Proc. of 8th Int. Workshop on Cooperative and Human Aspects of Softw. Eng., pp.15–21, 5 2015.

Appendix A: The method for obtaining data

In this appendix, we show the method for obtaining data. We obtained the dates of development activities for each contributor, and identified the contribution period from the first activity to the last activity. Figure A· 1 shows an example of the data of commit_comments. This is a record for one of the commit comment. It has dates such as "created_at" and "updated_at". We identify the contribution period of contributors from those dates. Contribution periods are divided into **coding periods** and **non-coding periods** based on the classification of the activities as shown in Table 1.



```
{
    "html_url": "https://github.com/octocat/...",
    "url": "https://api.github.com/repos/octocat/...",
    "id": 1,
    "body": "Great stuff",
    "path": "file1.txt",
    "position": 4,
    "line": 14,
    "commit_id": "6dcb09b5b57875f334f61aebed6...",
    "user": {
        "login": "octocat",
        "id": 1,
        "avatar_url": "https://github.com/images/...",
        "html_url": "https://github.com/octocat",
                        .
                        .
                        .
        "events_url": "https://api.github.com/users/...",
        "received_events_url": "https://api.github.com/...",
        "type": "User",
        "site_admin": false
    },
    "created_at": "2011-04-14T16:00:49Z",
    "updated_at": "2011-04-14T16:00:49Z"
}
```

Fig. A· 1 An examples of commit_comments data.

Saya Onoue received a BE degree in information science from the Nara Women's University in 2013, and she received an ME degree in information engineering from the Nara Institute of Science and Technology (NAIST) in 2015. She is currently a Ph.D student at NAIST. Her research interest is activity of contributors in OSS projects.

Hideaki Hata received a PhD degree in information science and technology from Osaka University in 2012. He is now an assistant professor at Nara Institute of Science and Technology. His research interests include software analytics, software economics, and human-software interaction. He is a member of the IPSJ, JSSST, IEEE, and ACM.

Akito Monden received a BE degree in 1994 in electrical engineering from Nagoya University and ME and DE degrees in 1996 and 1998 in information science from the Nara Institute of Science and Technology. He is currently a professor at Okayama University. He was an honorary research fellow at the University of Auckland (2003-2004). He is a member of the IEEE, ACM, and IEICE,

Kenichi Matsumoto received a PhD degree in information and computer sciences from Osaka University. He is a professor in the Graduate School of Information Science at the Nara Institute of Science and Technology. His research interests include software measurement and software process. He is a senior member of the IEEE, and a member of the ACM, the IEICE, and the IPSJ.