Journal of Information Processing Vol.24 No.2 339–348 (Mar. 2016)

[DOI: 10.2197/ipsjjip.24.339]

Regular Paper

Magnet or Sticky? Measuring Project Characteristics from the Perspective of Developer Attraction and Retention

Kazuhiro Yamashita[1,a)] Yasutaka Kamei[1,b)] Shane McIntosh[2,c)]
Ahmed E. Hassan[3,d)] Naoyasu Ubayashi[1,e)]

Received: June 30, 2015, Accepted: December 7, 2015

Abstract: Open Source Software (OSS) is vital to both end users and enterprises. As OSS systems are becoming a type of infrastructure, long-term OSS projects are desired. For the survival of OSS projects, the projects need to not only retain existing developers, but also attract new developers to grow. To better understand how projects retain and attract contributors, our preliminary study aimed to measure the personnel attraction and retention of OSS projects using a pair of population migration metrics, called Magnet (personnel attraction) and Sticky (retention) metrics. Because the preliminary study analyzed only 90 projects and the 90 projects are not representative of GitHub, this paper extend the preliminary study to better understand the generalizability of the results by analyzing 16,552 projects of GitHub. Furthermore, we also add a pilot study to investigate the typical duration between releases to find more appropriate release duration. The study results show that (1) approximately 23% of developers remain in the same projects that the developers contribute to, (2) the larger projects are likely to attract and retain more developers, (3) 53% of terminal projects eventually decay to a state of fewer than ten developers and (4) 55% of attractive projects remain in an attractive category.

Keywords: Magnet, Sticky, Developer Transition, Open Source Software, Mining Software Repositories

1. Introduction

Open source software (OSS) is vital to both end users and enterprises. According to a survey conducted by Black Duck, 78% of enterprises run part or all of their business operations on OSS systems [*1]. For example, Red Hat, a leading company of Linux distribution, is enhancing the Linux kernel to develop Enterprise Linux (an enterprise operation system). As OSS systems are becoming a type of infrastructure, long-term OSS projects are desired.

The survival of OSS projects depends on their ability to retain contributors. As contributors continue working on a project, they develop more efficient approaches and perform more complicated tasks. Long Term Contributors (LTCs) are particularly important for projects, because these contributors gradually become experts who can write good codes and perform core tasks (e.g., mentoring newcomers) [44].

Besides retaining contributors, OSS projects need to attract new contributors to grow the projects. The importance of new contributors has been emphasized in some studies. According to Kraut et al. [21], new contributors provide innovation, new ideas, and novel work procedures. Qureshi et al. [28] claim that new contributors promote a sustainable community to motivate, engage, and retain new contributors. New contributors may also become candidates of new LTCs.

For these reasons, retaining contributors and attracting new contributors are crucial for OSS projects. This fact has been widely recognized in literature [21], [31], [33], [43], [45]. These studies focus on the factors that attract and retain staff. For instance, Zhou et al. [45] reported that a pro-community attitude is most important for cultivating LTCs in open source projects. Steinmacher et al. [33] modeled 58 social barriers of OSS projects (e.g., not receiving an answer). However, to our knowledge, no study has measured these characteristics using developer transitions.

Therefore, in this study, we focus on the number of new and existing contributors in OSS projects. We measure the attractiveness and retention of contributors in OSS projects by two metrics, called Magnet and Sticky metrics [26]. The Magnet metric indicates the number of new developers attracted to a project and Sticky metric indicates the number of existing developers that stay with the project. The Magnet and Sticky metrics are defined in Section 2. These metrics are expected to capture the status of each project in terms of recruiting and retaining contributors.

Using the two metrics, we address the following two research questions:

[1] Kyushu University, Fukuoka 819–0395, Japan
[2] McGill University, Montréal, Québec, Canada
[3] Queen's University, Kingston, Ontario, Canada
[a)] yamashita@posl.ait.kyushu-u.ac.jp
[b)] kamei@ait.kyushu-u.ac.jp
[c)] shane.mcintosh@mcgill.ca
[d)] ahmed@cs.queensu.ca
[e)] ubayashi@ait.kyushu-u.ac.jp

[*1] https://www.blackducksoftware.com/future-of-open-source (accessed 2015-06-15)

(**RQ1**) **What Are the Typical Distributions of Projects from the Magnet and Sticky Perspectives?**
Motivation: Applying the concepts of Magnet and Sticky in an OSS context, we seek the project distributions of these concepts.
Result: 23% of contributors remain with a project. Lager projects attract larger number of new contributors than smaller projects.

(**RQ2**) **How do the Magnet and Sticky values change over time?**
Motivation: By investigating the transitions of Magnet and Sticky values, we can capture the temporal evolution and decay of the projects.
Result: 53% of terminal projects eventually decay into a state of have fewer than ten contributors. On the other hand, 55% of attractive projects keep their popularity. Furthermore, stagnant projects are more likely to decay than fluctuating projects.

To answer these research questions, we conducted experiments of 16,552 GitHub projects. GitHub is one of the largest social coding platforms in the world, hosting many types of OSS projects.

This study is an extended version of our preliminary study (as a short paper of an international conference) [42]. The largest extension is the dataset. Since the dataset that we used for preliminary study includes 90 projects that are not randomly selected and not representative of GitHub [*2], in this study, we use 16,552 projects that have more than ten forks and developers (cf., Section 3). Furthermore, we modify the definition of sticky value to improve the usability and investigate typical duration between releases to avoid ad hoc decision about duration.

The summary of extensions is as below:

- Modifying the definitions of Sticky values (Section 2).
- Adding a pilot study to investigate the typical duration between releases (Section 4).
- Extending the number of target projects from 90 to 16,552 (RQ1, 2).
- Adopting the typical release duration as time period in our experiments (RQ1, 2).

Paper organization. The remainder of the paper is organized as follows. Section 2 describes out measure of contributor attractiveness and retention in OSS projects. Section 3 discusses the motivations and approaches of the research questions, overview the dataset, and defines the terminology used throughout the paper. Sections 4 and 5 present the results of pilot studies investigating the release duration in OSS projects and studies that we conduct using the large-scale of OSS projects, respectively. Then, we discuss our results in Section 6. Section 7 surveys related work, and Section 8 concludes the paper.

2. Measuring Contributor Retention and Attraction in OSS

This section describes our measurements of personnel retention and attraction in OSS. In this study, we use the Magnet and Sticky metrics defined by the Pew Research Center [26] for illustrating the migratory trends of citizens in the United States. The Sticky metric revealed that just 28% of people are born in Alaska, but more than 75% of those born in Texas, remain in their birth states as adults. Furthermore, the Magnet metric revealed that 86% of adult residents of Nevada had migrated from a different state.

2.1 Magnet and Sticky in State Populations

The Pew Research Center report [26] defines *Magnet states* as states that attract a large proportion of adults from other states. Thus, the Magnet metric of a state is the proportion of adult residents who were not born in that state, relative to the total state population. The report also defines *Sticky states* as states that retain a large proportion of the people born in that state. Thus, the Sticky metric of a state is the proportion of adult residents who were born in that state, relative to the residents born and living in the entire United States.

2.2 Magnet and Sticky in OSS Projects

The definitions of Magnet and Sticky are unambiguous in population studies, in which a single adult occupies only one state at a time, but are not directly applicable to open source projects, because contributors can contribute to several projects at the same time. Furthermore, the birth and current residence states of a single adult are identified from certificates of residence; however, no such document records the projects contributed by a developer. Therefore, if a developer commits to a project during a certain period, we identify that the developer has joined the project during that period. Therefore, the identification depends on the duration of the contribution period. In our preliminary study [42], we tentatively assigned the time window of the analysis as one year. In the present study, we more rigorously assess the time window as six months in a pilot study, see Section 4.

Using this duration, we divide time into periods. The period of interest is denoted the target period (p_i). The periods immediately preceding and succeeding the target period are called the previous period (p_{i-1}) and the following period (p_{i+1}), respectively.

In our preliminary study [42], the Sticky metric was defined as the proportion of contributors in both p_i and p_{i+1}. In this study, we modify the definition to the proportion of contributors in p_{i-1} and p_i. In this manner, we can predict the status of the projects in p_{i+1} (i.e., the future status).

Therefore, we redefine the Magnet and Sticky metrics as follows:

Magnet projects are projects that attract a large proportion of new contributors. Thus, the magnetism of a project is the proportion of contributors who contributed during a particular period, but not during previous periods.

Sticky projects are projects in which many contributors continue making contributions. Thus, the stickiness of a project is the proportion of contributors who contributed during a particular period and also during previous periods.

2.3 Illustrative Example

The quantification of our definitions is demonstrated in **Fig. 1**.

[*2] http://2014.msrconf.org/challenge.php (Accessed 2015-06-15)

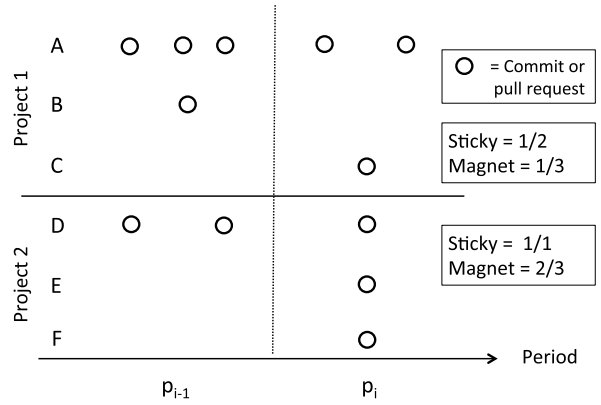


Fig. 1 Calculation examples of our newly defined Magnet and Sticky values.

In this example, we examine two projects during period p_i. There are six developers (A, B, C, D, E, and F) and two projects (1 and 2). Circles show the commits or pull requests contributed by the developers (listed down the left-hand side). For example, while developer A makes two contributions during period p_i, developer B makes no contributions during that period. To calculate the Magnet metric, we observe that three new developers (C, E, and F) join the team at p_i, one of whom contributes to project 1 (C), while the others (E and F) contribute to project 2. In this case, the Magnet values of projects 1 and 2 are $\frac{1}{3}$ and $\frac{2}{3}$, respectively.

To calculate the Sticky metrics, we note that two developers (A and B) contributed to project 1 at p_{i-1}, one of whom (A) also contributes at p_i. Hence, the Sticky value of project 1 is $\frac{1}{2}$. One developer (D) contributes to project 2 at p_{i-1}, and three developers (D, E, and F) contribute at p_i. However, the Sticky metric only considers the number of contributors during p_{i-1} and p_i. Hence, the Sticky value of project 2 is not $\frac{3}{1}$, but rather $\frac{1}{1}$.

3. Study Design

This section provides an overview of our study. First, we develop our research questions and motivations, then describe our dataset.

3.1 Research Questions — Motivation and Approach

(RQ1) What Are the Typical Distributions of Projects from the Magnet and Sticky Perspectives?

Motivation. First, we overview the trends of Magnet and Sticky values in the OSS context. This research question was addressed in our preliminary study [42], but here we expand the number of case study projects from 90 to 16,552. We also reconsider the time window. In this study, we established the time window as six months in a pilot study.

Approach. The Magnet and Sticky values of the studied OSS projects are calculated as described in Section 2. To visualize the data, we plot the Magnet and Sticky values of each project against each other project, and (similar to Khomh et al. [20]) divide the plot into four quadrants, as done in our preliminary study [42]:

Attractive projects (with high Magnet and Sticky values) successfully attract new developers while retaining their existing ones.

Fluctuating projects (with high Magnet values, and low Sticky values) successfully attract new developers but tend to lose existing ones.

Stagnant projects (with low Magnet values, and high Sticky values) retain their existing development team but struggle to attract new members.

Terminal projects (with low Magnet and Sticky values) struggle to retain existing developers while failing to attract new ones.

The quadrant thresholds can be dynamically configured. In this study, we use the median Magnet and Sticky values as the thresholds, as the median is a robust measure that is not heavily influenced by outliers.

As in our preliminary study [42], we focus on the latter six months of the most recently completed year of historical data (i.e., from July to December of 2013). The most recent dataset includes the largest number of projects.

Note that the Sticky value depends on the number of contributors in both the target and the previous time periods (Fig. 1). If few developers have contributed in the previous time period is small, the Sticky value tends to be high. Therefore, to reduce the noise in our results, we filter out projects with less than ten developers in the previous time period. We also consider the time period in which the project started. The Sticky value of a start-up project is 0, because all of the developers are new and no developer has contributed during the previous time period. Therefore, we filter out new projects in the target time period.

Besides an overview of the distribution, we also show typical values of differently sized projects. As mentioned above, the Magnet and Sticky metrics are influenced by the number of total developers in the target and previous time periods. Therefore, we divide projects according to their number of developers, and display the median Magnet and Sticky values of projects in each size category.

(RQ2) How do the Magnet and Sticky values change over time?

Motivation. By investigating the changes in Magnet and Sticky values, we can capture the temporal evolution and decay of the projects.

Approach. We analyze how the aging projects transit among the quadrants of Fig. 3. As the quadrant boundaries will likely change, the boundaries are recalculated in each time period. In this study, we track the Magnet and Sticky values from 2000 to 2013 (i.e., thorough 28 time periods).

3.2 Overview of Dataset

Our dataset is the GitHub dataset "GHTorrent" provided by Gousios [12] [*3]. Part of this dataset is provided in the MySQL database and includes diverse software evolution data from a large collection of OSS projects, such as issue reports, pull requests, organizations, followers, stars and labels. We focus on the code authorship data in the commits and pull requests tables.

GitHub has unique features such as *fork* and *pull request* for collaborative development. GitHub describes Fork as a copy of a repository. Forking a repository allows you to freely experi-

[*3] We use MySQL databases dump at 2014/04/02.

Table 1 Overview of the GHTorrent dataset used in this study.

Dataset	#Users	#Repos	#Commits	#PullReqs
This Study	3,426,046	8,510,504	96,999,485	3,200,428
Preliminary Study	499,485	108,718	555,325	78,955

ment with changes without affecting the original project [*4]. Pull request allows users to "tell others about changes you've pushed to a repository on GitHub. Once a pull request is sent, interested parties can review the set of changes, discuss potential modifications, and even push follow-up commits if necessary" [*5]. A typical development process on GitHub, driven by fork and pull requests, proceeds as follows:

(1) A developer forks a repository to which he or she hopes to contribute.
(2) The developer makes changes to the fork repository.
(3) The developer sends a pull request to the original repository to reflect his or her changes to the original repository.
(4) If the owner of the original repository allows the pull request, the changes are included in the original repository.

Table 1 overviews the dataset used in our study. This dataset is 800 times larger than that accessed in our preliminary study [42]. Besides the higher number of repositories, the current dataset includes more users, commits and pull requests than the dataset of the preliminary study.

As described above, the entire dataset is divided into sub-datasets covering different time periods. Therefore, we examine the column `created_at`. However, this column contains uninterpretable or nonsensical dates such as '0000-00-00', '0000-00-00 00:00:00', and 2025 (as the commit year). As the dates of commit and pull requests are critical to our analysis, we filtered out such cases.

3.3 Developers

In this study, a developer is a person who alters software code. In the GitHub dataset, developers can either perform the commit themselves or send a pull request to an upstream repository maintainer. Both actions are viewed as developmental activity in our Magnet and Sticky analyses. According to Kalliamvakou et al. [17], most of the accepted pull requests sent from fork repositories are absent in the histories of original repositories. Therefore, we mine the developer information from both the original and fork repositories. In particular, we obtain the author information from the retrieved commits. From the pull requests, we obtain the information of actors who send (i.e., open) the pull requests.

The GitHub system identifies authors as registered or non-registered from the email addresses of the commits [*6]. If the author of a commit is not registered, GitHub records the author information that can be obtained from Git, such as name and email address, along with a unique id. In this system, some developers are assigned multiple user ids. Therefore, we clean the data using the tool [*7] that matches users with their information recorded in GitHub (e.g., login name, actual name, email address and location).

3.4 Projects

Not all of the repositories included in our dataset are software projects [17]. Other repository categories include, but are not limited to, *Experimental* (e.g., examples, demonstrations and samples,) and *Storage* (e.g., configuration files and personal use). We assume that the number of fork repositories and developers is negligible in these categories, since these repositories do not require collaboration with others. To identify software projects, we note the number of fork repositories and number of developers, both of which indicate collaborative activity. Projects with less than 10 fork repositories and 10 developers are filtered out. The post-filtered dataset includes 16,552 original repositories.

Our study focuses on projects adopting the pull-based model, which excludes the 55% of the GitHub projects using shared repository models [13]. Moreover, we filtered projects with fewer than 10 forks. Therefore, our findings are not generalizable to shared repository models.

4. Pilot Study

In our preliminary study [42], we tentatively assigned the target period of the magnetism and retention calculations as one year. However, the validity of this assignment was not discussed. In the present study, the appropriate period is identified in a pilot study.

In defect prediction, code review and other studies relying on Mining Software Repositories (MSR), experiments are conducted at the release-level [18], [24]. However, when conducting experiments across multiple projects, the release-level is inappropriate for two reasons. First, we desire to compare metrics at the same time; second, multiple projects are not released simultaneously.

Instead of the release-level, we therefore adopt the representative release duration. Some of the large projects regularly update their products [19]; Google Chrome and Mozilla Firefox update their products every six weeks (i.e., adopt a rapid release model). If all the projects in our dataset are periodically updated at the same rate, that period becomes a useful parameter in the magnetism and retention calculations. Therefore, we manually inspect some projects to determine the constancy of their update periods. Unfortunately, unlike Google Chrome and Mozilla Firefox, most projects are not regularly upgraded. Hence, to identify the typical release period of the GitHub projects, we calculate the durations between the releases of each project.

Approach. GitHub releases the products [*8] and provides the API to access the released information. We extract the release information (version number and published date) of all target projects from the GitHub API. The published and git tag dates are independent, although both dates have the same version name and release date of the updated version onto GitHub. Although

[*4] https://help.github.com/articles/fork-a-repo/ (Accessed 2015-06-15)
[*5] https://help.github.com/articles/using-pull-requests/ (Accessed 2015-06-15)
[*6] https://help.github.com/articles/why-are-my-commits-linked-to-the-wrong-user/ (Accessed 2015-06-15)
[*7] https://github.com/bvasiles/ght_unmasking_aliases (Accessed 2015-06-15)
[*8] https://help.github.com/articles/creating-releases/ (Accessed 2015-06-15)

Table 2 Release durations of major, minor, patch upgrades of GitHub projects (days).

	Major Release	Minor Release	Patch Release
Min	9	1	1
1st Qu.	68.8	20	6
Median	167.5	52	18
Mean	202.8	84.2	38.5
3rd Qu.	316.3	117	45
Max	650	620	609
NumberOfUpdate	98	2,021	6,092

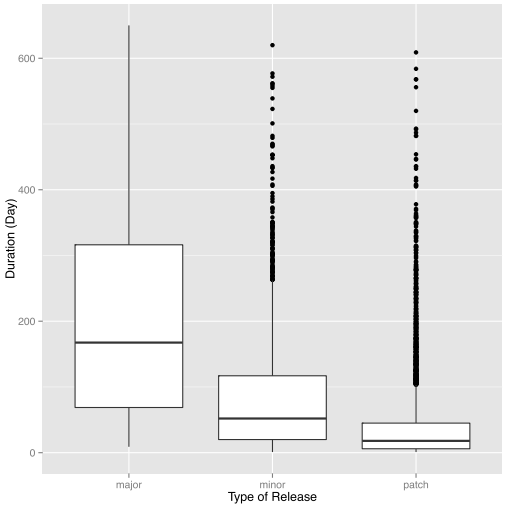


Fig. 2 Release Duration (days).

GitHub recommends the semantic labeling of new versions (in MAJOR.MINOR.PATCH number format) [27], some projects do not follow this recommendation. Projects not adopting the semantic versioning system are removed from our analysis. We also remove alpha versions and release candidates (e.g., 1.0.0-alpha, 1.0.0-pre), because such versions are candidates rather than official releases. After filtering, we extract the release information of 16,682 versions of 1,778 projects. From this information, we calculate the durations of *Major*, *Minor* and *Patch* releases.

In the semantic versioning system [27], a major release denotes an update of incompatible API changes, and the version number changes from (`x.0.0`) to (`x+1.0.0`). Minor releases add functionality to a project in a backwards-compatible manner, and alter the version number from (`x.y.0`) to (`x.y+1.0`). Patch releases correct backwards-compatible bugs, and are marked by version number changes from (`x.y.z`) to (`x.y.z+1`).

Multiple versions (even major upgrades) were occasionally released on the same day, and in different order from their version numbers. We presumed that such projects had been moved to GitHub from another hosting service (e.g., SourceForge), and had been previously released. A developer could then release all versions onto GitHub on the same day. Therefore, we filter out updates with durations below one day and released in different order from their version numbers.

Results. The duration distributions of the major, minor, and patch updates are presented in **Table 2** and **Fig. 2**. To improve the accuracy of the pilot study, we focus on durations between the 1st and 3rd quantiles.

Figure 2 reveals clear duration differences between the major, minor, and patch releases. At the patch level, the durations at the 1st and 3rd quantiles are 6 days and 45 days, respectively, with a median of 18 days (approximately half a month). For minor upgrades, the durations at the 1st and 3rd quantiles are 20 days and 117 days, respectively, with a median of 52 days (approximately two months). At the major level, the durations of the 1st and 3rd quantiles are 69 days and 316 days, respectively, and the median is 168 days (approximately half a year).

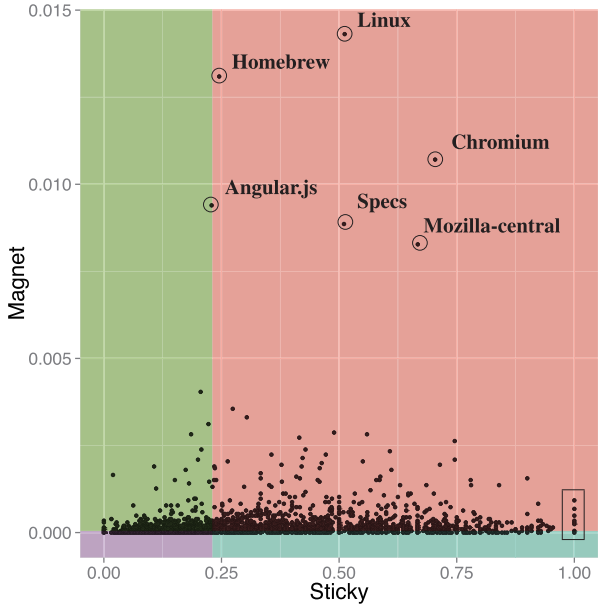


Fig. 3 Distribution of Magnet and Sticky values for the studied projects.

New versions of GitHub projects are released in 18 days at the patch level, 52 days at the minor level and 168 days at the major level.

The pilot study revealed the typical durations of each level of releases. In the following study, we adopt the median duration of the major release as the time window, because the major release is the most important update of a project.

5. Study Results

(RQ1) What Are the Typical Distributions of Projects from the Magnet and Sticky Perspectives?

Figure 3 presents a Magnet vs. Sticky quadrant plot of the OSS projects released on GitHub during the latest time period (July to December of 2013). Attractive, fluctuating, stagnant, and terminal projects land in the red (upper-right), green (upper-left), blue (lower-right), and purple (lower-left) quadrants, respectively. The names of the extremely attractive projects are annotated in the figure.

The median Magnet value is quite small, and the median Sticky value is only 0.23 (Fig. 3). Although the Magnet value is typically below 0.005 (marked by the horizontal division on the plot), some projects have large Magnet values. These findings suggest that the distribution of the number of new developers in each project is highly skewed, and that approximately 23% of developers remain in the same projects.

The results are similar to our preliminary study [42]. In preliminary study, Magnet values are much smaller than Sticky values

Table 3 Projects with Sticky values of 1.0.

Name
DIRACGRID/DIRAC
JetBrains/MPS
georchestra/georgestra
dxw/wordpress
virtual-world-framework/vwf
open-mpi/ompi-svn-mirror
rose-compiler/edg4x-rose
stackforge/savanna
PCGen/pcgen
crosswalk-project/crosswalk

and only a few projects have large Magnet value. Furthermore, the median Sticky value is approximately 20%.

Six of the projects have exceptionally high Magnet, namely, `Linux`, `Homebrew`, `Chromium`, `Angular.js`, `Specs`, and `Mozilla-central`. `Linux` is among the most famous projects, and `Homebrew` is a popular package management tool for Mac OS X. The web browser project `Chromium` is basis of Google Chrome. `Angular.js` is the web framework for JavaScript, `Specs` is a repository for the public CocoaPods [*9] specification, and `Mozilla-central` is a repository for source codes implemented by the Mozilla foundation such as Firefox web browser.

The `Linux`, `Chromium`, `Homebrew` and `Mozilla-central` projects are well-known and universally popular. Therefore, many developers are expected to join these projects.

The popularity of `Angular.js` and `CocoaPods` during the analysis period was checked by Google Trends [*10], which records the number of query searches on Google in chronological order. In Refs. [6], [40], the popularity of a project is assessed from the numbers of web pages indexed by Google and views of the project page. However, the popularity trends of the projects are difficult to identify by these indicators. Therefore, we identify the popularities of the projects thorough Google Trends. The search numbers of both `Angular.js` [*11] and `CocoaPods` [*12] were increasing from 2013. Therefore, we assume that as the projects gained popularity from 2013, they increasingly attracted newcomers to their development. This finding suggests that the Magnet and Sticky values well-indicate the fame and popularity of a project.

Ten projects in Fig. 3 have a Sticky value of 1.0 (we put a framed box around the projects). The names of these ten projects are listed in **Table 3**. To identify the reason for such high Sticky values, we check their web pages and the developers' affiliations to find out the primary developers and maintainers of the projects. If more than half of developers belong to companies, we consider that the projects are supported by those companies. All the projects in Table 3 are found to be developed or supported by companies or laboratories. In general, non-company developers are likely to join OSS projects as hobbyists [22], but company and laboratory developers probably join OSS projects as part of their work [22], [29]. Therefore, projects supported by company or laboratory developers are more likely to be constantly contributed by the same developers than projects supported by non-company developers.

[*9] CocoaPods is the dependency manager for Swift and Objective-C.
[*10] https://www.google.co.jp/trends/ (Accessed 2015-10-15)
[*11] https://www.google.co.jp/trends/explore#q=angularjs (Accessed 2015-10-15)
[*12] https://www.google.co.jp/trends/explore#q=CocoaPods (Accessed 2015-10-15)

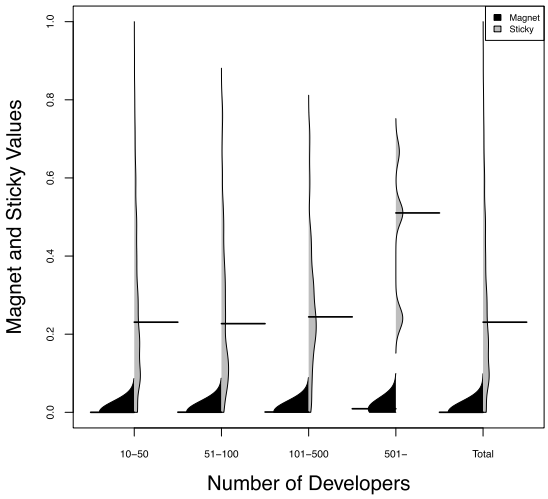


Fig. 4 Beanplots of Magnet and Sticky values, grouped by developer size.

Table 4 Median values of Magnet and Sticky OSS projects released on GitHub.

Metrics	# of Total Developers in Project				
	10-50	51-100	101-500	501-	Total
Median Magnet	0	2.9e-04	7.5e-04	9.1e-03	4.9e-05
Median Sticky	0.23	0.23	0.24	0.51	0.23
# of Projects	4,275	217	112	8	4,612

We then study the impact of the number of project developers on the Magnet and Sticky values. **Figure 4** shows beanplots of the Magnet and Sticky values of differently sized projects (the medians are listed in **Table 4**). In these plots, the left (black) regions and right (gray) regions indicate the Magnet and Sticky values, respectively. From left to right, the number of developers is binned into 10–50, 51–100, 101–500, 501– plus, and all sizes.

From Fig. 4 and Table 4, we find that the Magnet and Sticky values are generally higher for larger projects than for smaller projects. As the denominator of the Sticky value is the total number of developers in the previous time period, the Sticky value is inversely proportional to the number of developers. However, large projects tend to have large Sticky values, consistent with our intuition that developers prefer to join and contribute long-term to such projects.

Larger projects attract and retain more developers than smaller projects. 23% of developers remain with the same project irrespective of size (total number of developers), and new developers tend to join popular and famous projects.

(RQ2) How do the Magnet and Sticky values change over time?

Figure 5 illustrates the quadrant transition likelihood on a state transition diagram. Percentages describe the likelihood of a transition from one quadrant to another (or the same) quadrant. The direction of the arrow indicates the direction of the quadrant change. For example, the likelihood of moving from the attractive to the terminal quadrant is 13%. States marked with "*" indicate

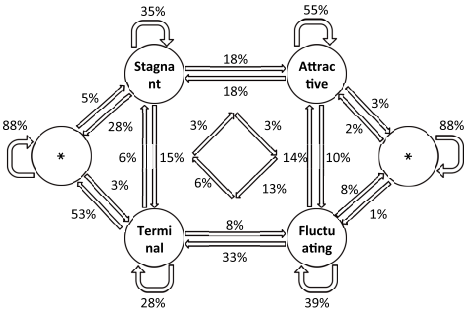


Fig. 5 The likelihood of quadrant transitions.

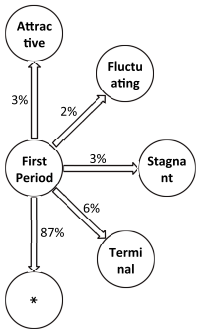


Fig. 6 Likelihood of quadrant transitions from the first period.

projects that failed our filtering criteria (ten or more developers) during some time periods. To improve the readability of the figure, we plot two "*" states, although these states are semantically identical.

According to this figure, 3%, 8%, 28%, and 53% of the attractive, fluctuating, stagnant, and terminal projects entered the filtered out state ("*"). Although any project can drop into the "*" state, the probability is much higher for terminal projects than for projects in other quadrants. Therefore, terminal projects are very likely to decay into the "*" state. Intuitively, we expect that as terminal quadrant projects are losing team members and struggling to attract new ones, they will eventually die.

This result is different from our preliminary study [42]. In our preliminary study, projects decay into "*" state only from terminal quadrant, however, in this study, we found that projects that are in other three quadrants decay into "*" state.

Interestingly, 28% of the stagnant projects, but only 8% of the fluctuating ones, decay into the "*" state. In both quadrants, one of the two metrics (Magnet or Sticky) is high; therefore, we expected that both quadrants would enter the "*" state with similar likelihood. The observed asymmetry might reflect the impact of number of developers. As fluctuating (stagnant) projects are characterized by high (low) Magnet and low (high) Sticky values, it appears that Magnet measure is more affected by number of developers than Sticky.

Moreover, projects in the fluctuating, stagnant, and terminal quadrants do not easily transit to the attractive quadrant. Only 18% of the projects entered the attractive quadrant from other quadrants, but 55% of the attractive projects maintained their high magnetism and stickiness. This phenomenon indicates that attractive projects are more stable than projects in other quadrants.

In Fig. 5, we filtered start-up projects during the time period because the Sticky value of such projects is 0, as earlier described in RQ1. However, the status transitions from the first time period to the next warrant investigation. **Figure 6** shows the likelihood of quadrant transitions from the first to the second time period. Only 13% of the projects maintained ten or more developers in next one, indicating the difficulty of retaining and acquiring developers after initiating a project.

53% of the terminal projects eventually decayed into a state of ten or fewer contributors, while 55% of the attractive projects maintained their popularity. Only 13% of the projects identified in the first time period had maintained ten or more developers in the second period.

6. Discussion

This section discusses our analysis and results.

6.1 Discussion of RQ1

From the result of calculating Magnet and Sticky values at latest time period, we obtained the distributions of the values of projects such as the median Magnet value is 0.05 and Sticky value is 0.23. The results are similar to our preliminary study [42]. Furthermore, we found that larger projects attract and retain larger number of developers. These findings fit our intuition. The large projects are already known by many people and there are more information of the projects compared to small projects. Hence, we assume that new developers can find the projects and the information easily. For existing developers, contributing at fame and popular project is proud thing and motives them. From these expectations, we assume that both types of developers (new developers and existing developers) have good motivation to contribute to the projects in larger projects.

Also, we showed the median of Magnet and Sticky values at the latest time period. We assume that the values act as a gauge of project health. If Magnet and Sticky values of a project are lower than the median values, the project faces a risk of decaying. In particular, Sticky value is stable across total number of developers. Hence, projects that have lower Sticky values are especially risky.

6.2 Discussion of RQ2

From the result of calculating likelihood of quadrant transitions, we found that 53% of terminal projects eventually decay into a state where they have fewer than ten contributors and 55% of attractive projects keep the popularity. We also revealed some different trends from our preliminary study [42]. In preliminary study, only terminal projects decay into the "*" state, but in this study, attractive, fluctuating, and stagnant projects also decayed into the "*" state. We attribute these differences to the much larger dataset in this study.

We plan to study project survivability (i.e., project keep maintaining) using the transition in our future work. Chengalur-Smith et al. [4] showed that the number of developers positively correlates with project survivability. Therefore, we expect that project survivability can be predicted from the analyzed trends and the definition of project failure, as proposed by English and

Schweik [11]. This estimation is planned for future work, but we must consider the definition of project death. In this study, the "*" state represents projects with fewer than ten developers and we consider projects moved to the "*" state as a type of obsolete projects. However, some projects with few developers are robustly sustained. Therefore, a small number of developers does not signify that the project will die (i.e., project stops its development). We must consider the definition of obsolete project in future work.

7. Related Work

7.1 Role Migration in Open Source Software

Some studies have investigated the role migration in OSS. Nakakoji et al. [25], Ye et al. [43] and Jensen and Scacchi [16] found that the extent to which each developer influences an OSS project establishes a hierarchy among the developers. Nakakoji et al. [25] claimed that a sustainable OSS project must evolve both the systems and the community. They identified three evolution patterns exploration-oriented, utility-oriented, and service-oriented. Ye et al. [43] sought to understand why people participate in OSS projects. They assume that learning in practice motivates OSS developers. Along with this learning process, a developer's role transformation in the OSS community provides extrinsic motivation. Jensen and Scacchi [16] investigated the role migration and project career advancement processes of OSS developers, focusing on three large OSS projects. They discussed the roles and layers in each projects and the migration between the roles and layers of developers who joined the projects.

Von Krogh et al. [38], Ducheneaut [10], Herraiz et al. [14], Bird et al. [2], and Shibuya et al. [32] also studied the role immigration process of OSS participants. Von Krogh et al. [38] found that new joiners to the Freenet project will more likely undertake certain actions than long-term developers. Drawing on personal experience, Deucheneaut described the six steps toward becoming a Python developer. In their experiments on three large projects, Bird et al. found that a submission history of patch upgrades can effectively elevate a joiner to developer status. Herraiz et al. discovered two groups of role migration; volunteer developers who proceed in a step-by-step fashion, and sponsored developers who suddenly migrate their roles. Shibuya and Tamai studied the openness (transparency and accessibility [41]) of three projects. They found that each project facilitates participation of new developers in different ways.

Similar to project roles, Robles et al. [30], Hindle et al. [15], and Vasilescu et al. [37] studied the various activities in projects. Hindle et al. distinguished four types of files and Robles et al. proposed eight different activities. Recently, Vasilescu et al. extended this number to 14 activities and empirically studied how the workloads of projects/contributors varied across the software ecosystem.

These studies focused on the role migration in development projects. In contrast, we investigate developer's migration between projects. By measuring the numbers of new and existing developers, we attempt to understand the underlying characteristics of projects.

7.2 Success of Open Source Software

After conducting a literature review, DeLone and McLean [9] proposed the Information Systems (IS) Success Model. They also reformed the model and its dimensions of *information quality*, *system quality*, *service quality*, *use*, *user satisfaction*, and *net benefits* [8]. Crowston et al. [5] similarly conducted a literature review and proposed dimensions that determine the success of an OSS project. Next, they trialed their success measures by interviewing SlashDot developers. The interviewed developers rated developer dimensions (such as developer involvement and satisfaction) most highly, followed by user dimensions (such as user satisfaction and involvement). Crowston et al. also performed an empirical study of their success measures [6].

Based on interviews with OSS developers, English and Schweik [11] proposed six ranks of OSS success and failure. Their classes are defined by several factors, such as the number of public releases, activity and age of the project. Evaluating their classification on SourceForge projects, and they classified only 15% of the projects as "Success, Growth."

Bonaccorsi et al. [3] claimed that two factors shape the lifecycle of a successful OSS project; a widely accepted leadership and effective co-ordination among the developers.

As described above, there are many methods of evaluating the success of OSS projects. In this study, we focus on developer attraction and retention (i.e., the success of developer growth).

7.3 GitHub

GitHub projects have been analyzed in various studies. Wagstrom et al. [39] proposed a dataset for investigating the relationship between Ruby on Rails and ecosystems, which includes approximately 1,000 projects. Thung et al. [34] investigated the relationships among projects and among developers, and identified the most successful projects/developers by their page rank. These studies focused on the relationships among and between projects and developers during a project's evolution.

McDonald et al. [23] interviewed 10 lead and core members of three large OSS projects hosted on GitHub. Most of the interviewees measured of a project's success by the numbers of existing and new contributors. In a quantitative analysis, Tsay et al. [36] studied two measures of a project's success on GitHub. Bettenburg and Hassan [1] studied the effect of social interactions on software quality. They found that social interactions consistently influence software quality and complexity. Dabbish et al. [7] examined how GitHub users interpreted and use the information obtained from GitHub.

Gousios et al. [13] and Tsay et al. [35] investigated Pull Requests on GitHub. Gousios et al. found that most of the pull requests are accepted or rejected within one day, and that accepting users submit their pull request within very similar timeframes. Tsay et al. studied the factors affecting pull request acceptance rates. They found that highly discussed pull requests are likely to be rejected, but, this effect is moderated by the submitter's prior interactions. Kalliamvakou et al. [17] discussed the perils and promises of researching on GitHub.

Our research applies Magnet and Sticky concept derived from social study and the Pew Research Center (a nonpartisan think

tank) in an OSS context, and thereby measures the attractiveness of GitHub projects.

8. Conclusion

Building on our preliminary study [42], we aimed to better understand how OSS projects attract and retain contributors. First, we extended the dataset from 90 to 16,552 projects to generalize out preliminary results. Second, we redefined the Sticky metric to better suit our purpose. Third, we experimentally identified the typical duration between product releases.

In this study, we obtained similar and different (RQ2) results to our preliminary study [42]. Roughly, the results of RQ1 are similar and of RQ2 are different to our preliminary study.

The main results of the experiments are summarized below as follows:

- 23% of developers remain in the same projects.
- Larger projects attract and retain more developers.
- 53% of terminal projects eventually decay into a state of fewer than ten contributors.
- 55% of attractive projects remain in the attractive quadrant.

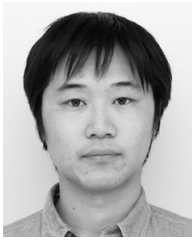
As mentioned in Section 6, our future work will investigate the relationship between the Magnet and Sticky metrics and project survivability. We found that developers tend to abandon terminal projects, whereas attractive projects are frequently sustained. These findings indicate a relationship between survivability and the proposed metrics, but cannot quantify this relationship. In particular, we did not define the failure of projects. Therefore, when investigating this relationship in future work, we should adopt a definition of project failure, as proposed English and Schweik [11].

Acknowledgments This research was partially supported by JSPS KAKENHI Grant Numbers 15H05306 and the Program for Advancing Strategic International Networks to Accelerate the Circulation of Talented Researchers.

References

[1] Bettenburg, N. and Hassan, A.: Studying the Impact of Social Structures on Software Quality, *Proc. Int'l Conf. Program Comprehension* (*ICPC*), pp.124–133 (2010).
[2] Bird, C., Gourley, A., Devanbu, P., Swaminathan, A. and Hsu, G.: Open Borders? Immigration in Open Source Projects, *Proc. Int'l Working Conf. Mining Software Repositories* (*MSR*), pp.6–13 (2007).
[3] Bonaccorsi, A. and Rossi, C.: Why Open Source software can succeed, *Research Policy*, Vol.32, No.7, pp.1243–1258 (2003).
[4] Chengalur-Smith, I.N., Sidorova, A. and Daniel, S.L.: Sustainability of Free/Libre Open Source Projects: A Longitudinal Study., *Journal of the Association for Information Systems*, Vol.11, No.11 (2010).
[5] Crowston, K., Annabi, H. and Howison, J.: Defining Open Source Software Project Success, *Proc. Int'l Conf. Information Systems* (*ICIS*), pp.327–340 (2003).
[6] Crowston, K., Howison, J. and Annabi, H.: Information systems success in Free and Open Source Software development: Theory and measures, *Software Process–Improvement and Practice*, Vol.11, No.2, pp.123–148 (2006).
[7] Dabbish, L., Stuart, C., Tsay, J. and Herbsleb, J.: Social Coding in GitHub: Transparency and Collaboration in an Open Software Repository, *Proc. Conf. Computer Supported Cooperative Work* (*CSCW*), pp.1277–1286 (2012).
[8] DeLone, W. and McLean, E.: Information Systems Success Revisited, *Proc. Int'l Conf. System Sciences* (*HICSS*), pp.238–248 (2002).
[9] DeLone, W.H. and McLean, E.R.: Information Systems Success: The Quest for the Dependent Variable, *Information Systems Research*, Vol.3, No.1, pp.60–95 (1992).
[10] Ducheneaut, N.: Socialization in an Open Source Software Community: A Socio-Technical Analysis, *Comput. Supported Coop. Work*, Vol.14, No.4, pp.323–368 (2005).
[11] English, R. and Schweik, C.M.: Identifying Success and Tragedy of FLOSS Commons: A Preliminary Classification of Sourceforge.net Projects, *Proc. Int'l Workshop Emerging Trends in FLOSS Research and Development* (*FLOSS*), pp.54–59 (2007).
[12] Gousios, G.: The GHTorrent dataset and tool suite, *Proc. Int'l Working Conf. Mining Software Repositories* (*MSR*), pp.233–236 (2013).
[13] Gousios, G., Pinzger, M. and van Deursen, A.: An Exploratory Study of the Pull-based Software Development Model, *Proc. Int'l Conf. Software Engineering* (*ICSE*), pp.345–355 (2014).
[14] Herraiz, I., Robles, G., Amor, J.J., Romera, T. and González Barahona, J.M.: The Processes of Joining in Global Distributed Software Projects, *Proc. Int'l Workshop Global Software Development for the Practitioner* (*GSD*), pp.27–33 (2006).
[15] Hindle, A., Godfrey, M. and Holt, R.: Release Pattern Discovery via Partitioning: Methodology and Case Study, *Proc. Int'l Workshop Mining Software Repositories* (*MSR*), p.19 (2007).
[16] Jensen, C. and Scacchi, W.: Role Migration and Advancement Processes in OSSD Projects: A Comparative Case Study, *Proc. Int'l Conf. Software Engineering* (*ICSE*), pp.364–374 (2007).
[17] Kalliamvakou, E., Gousios, G., Blincoe, K., Singer, L., German, D.M. and Damian, D.: The Promises and Perils of Mining GitHub, *Proc. Working Conf. Mining Software Repositories* (*MSR*), pp.92–101 (2014).
[18] Kamei, Y., Matsumoto, S., Monden, A., Matsumoto, K., Adams, B. and Hassan, A.E.: Revisiting Common Bug Prediction Findings Using Effort Aware Models, *Proc. Int'l Conf. Software Maintenance* (*ICSM*), pp.1–10 (2010).
[19] Khomh, F., Adams, B., Dhaliwal, T. and Zou, Y.: Understanding the impact of rapid releases on software quality, *Empirical Software Engineering*, Vol.20, No.2, pp.336–373 (2015) (online), available from ⟨http://dx.doi.org/10.1007/s10664-014-9308-x⟩.
[20] Khomh, F., Chan, B., Zou, Y. and Hassan, A.E.: An Entropy Evaluation Approach for Triaging Field Crashes: A Case Study of Mozilla Firefox, *Proc. Int'l Working Conf. Reverse Engineering* (*WCRE*), pp.261–270 (2011).
[21] Kraut, R., Burke, M., Riedl, J. and Resnick, P.: The Challenges of Dealing with Newcomers, *MIT Press*, pp.179–230 (2012).
[22] Lakhani, K. and Wolf, R.: *Why Hackers Do What They Do: Understanding Motivation and Effort in Free/Open Source Software Projects.*, MIT Press (2005).
[23] McDonald, N. and Goggins, S.: Performance and Participation in Open Source Software on GitHub, *CHI '13 Extended Abstracts on Human Factors in Computing Systems*, pp.139–144 (2013).
[24] McIntosh, S., Kamei, Y., Adams, B. and Hassan, A.E.: The Impact of Code Review Coverage and Code Review Participation on Software Quality: A Case Study of the Qt, VTK, and ITK Projects, *Proc. Working Conf. Mining Software Repositories* (*MSR*), pp.192–201 (2014).
[25] Nakakoji, K., Yamamoto, Y., Nishinaka, Y., Kishida, K. and Ye, Y.: Evolution Patterns of Open-source Software Systems and Communities, *Proc. Int'l Workshop on Principles of Software Evolution* (*IWPSE*), pp.76–85 (2002).
[26] Pew Research Social & Demographic Trends: Magnet or Sticky?: A State-by-State Typology, available from ⟨http://www.pewsocialtrends.org/2009/03/11/magnet-or-sticky/⟩ (accessed 2015-06-15).
[27] Preston-Werner, T.: Semantic Versioning 2.0.0, available from ⟨http://semver.org⟩ (accessed 2015-06-15).
[28] Qureshi, I. and Fang, Y.: Socialization in Open Source Software Projects: A Growth Mixture Modeling Approach, *Organizational Research Methods*, Vol.14, No.1, pp.208–238 (2011).
[29] Riehle, D., Riemer, P., Kolassa, C. and Schmidt, M.: Paid vs. Volunteer Work in Open Source, *Proc. Hawaii Int'l Conf. System Sciences* (*HICSS*), pp.3286–3295 (2014).
[30] Robles, G., Gonzalez-Barahona, J.M. and Merelo, J.J.: Beyond Source Code: The Importance of Other Artifacts in Software Development (a Case Study), *J. Syst. Softw.*, Vol.79, No.9, pp.1233–1248 (2006).
[31] Schilling, A., Laumer, S. and Weitzel, T.: Who Will Remain? An Evaluation of Actual Person-Job and Person-Team Fit to Predict Developer Retention in FLOSS Projects, *Proc. Hawaii Int'l Conf. System Science* (*HICSS*), pp.3446–3455 (2012).
[32] Shibuya, B. and Tamai, T.: Understanding the process of participating in open source communities, *Proc. Int'l Workshop on Emerging Trends in Free/Libre/Open Source Software Research and Development* (*FLOSS*), pp.1–6 (2009).
[33] Steinmacher, I., Conte, T., Gerosa, M.A. and Redmiles, D.: Social Barriers Faced by Newcomers Placing Their First Contribution in Open Source Software Projects, *Proc. Conf. Computer Supported Cooperative Work and Social Computing* (*CSCW*), pp.1379–1392 (2015).

[34] Thung, F., Bissyande, T.F., Lo, D. and Jiang, L.: Network Structure of Social Coding in GitHub, *Proc. European Conf. on Software Maintenance and Reengineering* (*CSMR*), pp.323–326 (2013).
[35] Tsay, J., Dabbish, L. and Herbsleb, J.: Influence of Social and Technical Factors for Evaluating Contribution in GitHub, *Proc. Int'l Conf. Software Engineering* (*ICSE*), pp.356–366 (2014).
[36] Tsay, J.T., Dabbish, L. and Herbsleb, J.: Social Media and Success in Open Source Projects, *Proc. Conf. on Computer Supported Cooperative Work Companion* (*CSCW*), pp.223–226 (2012).
[37] Vasilescu, B., Serebrenik, A., Goeminne, M. and Mens, T.: On the variation and specialisation of workload—A case study of the Gnome ecosystem community, *Empirical Software Engineering*, Vol.19, No.4, pp.955–1008 (2014).
[38] von Krogh, G., Spaeth, S. and Lakhani, K.R.: Community, joining, and specialization in open source software innovation: A case study, *Research Policy*, Vol.32, No.7, pp.1217–1241 (2003).
[39] Wagstrom, P., Jergensen, C. and Sarma, A.: A network of Rails a graph dataset of Ruby on Rails and associated projects, *Proc. Int'l Working Conf. Mining Software Repositories* (*MSR*), pp.229–232 (2013).
[40] Weiss, D.: Measuring Success of Open Source Projects Using Web Search Engines, *Proc. Int'l Conf. Open Source Systems*, pp.93–99 (2005).
[41] West, J. and O'mahony, S.: The Role of Participation Architecture in Growing Sponsored Open Source Communities, *Industry and Innovation*, Vol.15, No.2, pp.145–168 (2008).
[42] Yamashita, K., McIntosh, S., Kamei, Y. and Ubayashi, N.: Magnet or Sticky? An OSS Project-by-project Typology, *Proc. Int'l Working Conf. Mining Software Repositories* (*MSR*), pp.344–347 (2014).
[43] Ye, Y. and Kishida, K.: Toward an Understanding of the Motivation Open Source Software Developers, *Proc. Int'l Conf. Software Engineering* (*ICSE*), pp.419–429 (2003).
[44] Zhou, M. and Mockus, A.: Developer Fluency: Achieving True Mastery in Software Projects, *Proc. Int'l Symposium on Foundations of Software Engineering* (*FSE*), pp.137–146 (2010).
[45] Zhou, M. and Mockus, A.: What Make Long Term Contributors: Willingness and Opportunity in OSS Community, *Proc. Int'l Conf. Software Engineering* (*ICSE*), pp.518–528 (2012).



Kazuhiro Yamashita is a Ph.D. candidate at Kyushu University. He received his Bachelor's degree and Master's degree from Kyushu University. His research interests include software engineering, data mining, mining software repositories (MSR).

Yasutaka Kamei is an associate professor at Kyushu University in Japan. He has been a research fellow of the JSPS (PD) from July 2009 to March 2010. From April 2010 to March 2011, he was a postdoctoral fellow at Queen's University in Canada. He received his B.E. degree in Informatics from Kansai University, and M.E. degree and Ph.D. degree in Information Science from Nara Institute of Science and Technology. His research interests include empirical software engineering, open source software engineering and Mining Software Repositories (MSR). His work has been published at premier venues like ICSE, FSE, ESEM, MSR and ICSM, as well as in major journals like TSE, EMSE and IST. More information is available online at http://posl.ait.kyushu-u.ac.jp/˜kamei/.

Shane McIntosh is an assistant professor in the Department of Electrical and Computer Engineering at McGill University. He received his Bachelor's degree in Applied Computing from the University of Guelph and his M.Sc. and Ph.D. in Computer Science from Queen's University. In his research, Shane uses empirical software engineering techniques to study software build systems, release engineering, and software quality. His research has been published at several top-tier software engineering venues, such as the International Conference on Software Engineering (ICSE), the International Symposium on the Foundations of Software Engineering (FSE), and the Springer Journal of Empirical Software Engineering (EMSE). More about Shane and his work is available online at http://shanemcintosh.org/.

Ahmed E. Hassan is a Canada Research Chair in Software Analytics and the NSERC/Blackberry Industrial Research Chair at the School of Computing in Queen's University. Dr. Hassan serves on the editorial board of the IEEE Transactions on Software Engineering and the Journal of Empirical Software Engineering. He spearheaded the organization and creation of the Mining Software Repositories (MSR) conference and its research community. Early tools and techniques developed by Dr. Hassan's team are already integrated into products used by millions of users worldwide. Dr. Hassan industrial experience includes helping architect the Blackberry wireless platform, and working for IBM Research at the Almaden Research Lab and the Computer Research Lab at Nortel Networks. Dr. Hassan is the named inventor of patents at several jurisdictions around the world including the United States, Europe, India, Canada, and Japan. More information at: http://sail.cs.queensu.ca/.

Naoyasu Ubayashi is a professor at Kyushu University since 2010. He is leading the POSL (Principles of Software Languages) research group at Kyushu University. Before joining Kyushu University, he worked for Toshiba Corporation and Kyushu Institute of Technology. He received his Ph.D. from the University of Tokyo. He is a member of ACM SIGPLAN, IEEE Computer Society, and Information Processing Society of Japan (IPSJ). He received "IPSJ SIG Research Award 2003."