Magnet or Sticky? An OSS Project-by-Project Typology

Kazuhiro Yamashita[1], Shane McIntosh[2], Yasutaka Kamei[1] and Naoyasu Ubayashi[1]
[1]Principles of Software Languages Group (POSL), Kyushu University, Japan
[2]Software Analysis and Intelligence Lab (SAIL), Queen's University, Canada
[1]yamashita@posl.ait.kyushu-u.ac.jp, {kamei, ubayashi}@ait.kyushu-u.ac.jp,
[2]mcintosh@cs.queensu.ca

ABSTRACT

For Open Source Software (OSS) projects, retaining existing contributors and attracting new ones is a major concern. In this paper, we expand and adapt a pair of population migration metrics to analyze migration trends in a collection of open source projects. Namely, we study: (1) project stickiness, i.e., its tendency to retain existing contributors and (2) project magnetism, i.e., its tendency to attract new contributors. Using quadrant plots, we classify projects as attractive (highly magnetic and sticky), stagnant (highly sticky, weakly magnetic), fluctuating (highly magnetic, weakly sticky), or terminal (weakly magnetic and sticky). Through analysis of the MSR challenge dataset, we find that: (1) quadrant plots can effectively identify at-risk projects, (2) stickiness is often motivated by professional activity and (3) transitions among quadrants as a project ages often coincides with interesting events in the evolution history of a project.

Categories and Subject Descriptors

D.2.8 [**Software Engineering**]: Metrics—*Process metrics*

General Terms

Measurement

Keywords

Magnet, Sticky, Developer migration, Open source

1. INTRODUCTION

Using census data, the Pew Research Center, a research body that studies the issues, attitudes and trends shaping America and the world, launched a Social & Demographic Trends study. The study reports that just 28% of adults born in Alaska still live there. Furthermore, 86% of adult residents of Nevada migrated there from a different state, suggesting that Nevada is quite "magnetic". Such population studies illustrate the migratory trends of citizens in America.

MSR'14, May 31 – June 1, 2014, Hyderabad, India
Copyright 2014 ACM 978-1-4503-2863-0/14/05...$15.00
http://dx.doi.org/10.1145/2597073.2597116

For software engineers, migratory trends of open source contributors is of interest. To become a popular and long-lived project, maintainers need to retain existing contributors and attract new ones. Mockus *et al.* [4] find that although the Apache and Mozilla open source projects have a small core team of developers, there is a larger community of contributors. While prior work has explored contributor immigration [1] and Long Term Contributors (LTCs) [5], to the best of our knowledge, the "sticky" and "magnetic" nature of open source projects has not yet been explored. Hence in this paper, we set out to adapt the "magnet" and "sticky" metrics described in the Pew Research study for the context of open source projects. We then use these metrics to explore the sticky or magnetic nature of open source projects in the MSR challenge dataset [2]. Using the dataset, we address the following two research questions:

(RQ1) What are typical values of magnet and sticky in OSS?
Although most of the studied projects tend to be more sticky than they are magnetic, highly magnetic projects like `Homebrew` consistently attract many developers by simplifying the contribution process.

(RQ2) How do magnet and sticky values change over time?
Our quadrant analysis can identify projects at risk of becoming obsolete. Furthermore, quadrant transitions are often accompanied by interesting events.

Paper organization. The remainder of the paper is organized as follows. Section 2 provides our definitions of magnet and sticky metrics for the open source context. Section 3 describes the areas of the MSR challenge dataset that we leverage for our study, for which the results are presented in Section 4. Finally, Section 5 draws conclusions.

2. MAGNET OR STICKY?

The Pew Research Center report[1] defines *magnet states* as those states where a large proportion of adults who live there have moved from another state. Thus, the magnet metric for a state is the proportion of adult residents of a state who were not born in the state. Furthermore, the report also defines *sticky states* as those states where a large proportion of adults who were born there continue to live there. Thus, the sticky metric for a state is the proportion of adult residents who were born in the state.

These definitions are sound for a study of populations, where a single adult can only occupy one state at a time.

[1]`http://www.pewsocialtrends.org/2009/03/11/magnet-or-sticky/`

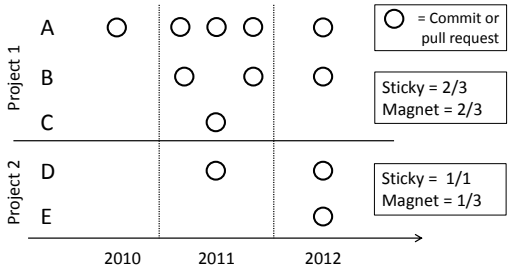

Figure 1: Example of Magnet or Sticky of values by our definition in 2011

However, the definition cannot be applied directly to open source projects, where a contributor can contribute to several projects at the same time. Therefore, we expand original definition to apply to open source projects as follows:

Magnet projects are those that attract a large proportion of new contributors. Thus, we calculate the magnetism of a project as the proportion of contributors who made their first contribution in the time period under study who contribute to a given project.

Sticky projects are those where a large proportion of the contributors will keep making contributions in the time period the following and under study. Thus, we calculate the stickiness of a project as the proportion of the contributors in the time period under study who have also made contributions in the following time period.

While our definitions are based on contributors, in this paper, we focus on developers. We plan to explore other types of contributions in future work. Furthermore, our definition is based on time periods in the abstract sense. We select development years as the granularity for this study because we believe that years provide a coarse enough granularity to observe high-level trends. Coarser or finer granularities can also be explored.

Figure 1 shows an example of our definition. In this example, we examine the 2011 time period. There are five developers (A, B, C, D and E) and two projects (1 and 2). To calculate the magnet metric, we observe that there are three new developers (B, C and D), and two of them contribute to project 1 (B and C), while one developer (D) contributes to project 2. In this case, Magnet value of project 1 is 2/3 and project 2 is 1/3.

To calculate the sticky metric, in project 1, three developers contribute in 2011 (A, B and C) two of them also contribute in 2012 (A and B). Hence, the sticky value of project 1 is 2/3. In project 2, one developer contributes to the project in 2011 (D) and two developers (D and E) contribute to in 2012. However, the sticky metric only considers the number of developers who contribute to the project in the studied time period and the next one. Hence, Sticky value of project 2 is not 2/1, but rather 1/1.

3. DATASET

In this paper, we analyze the GitHub dataset provided by Gousios [2]. The dataset includes a variety of software evolution data from 90 OSS projects, such as issues, pull requests, organizations, followers, stars and labels. However, in this paper, we only operate on code authorship data in the commits and pull requests tables. Although these 90 OSS projects are forked thousands of times, we focus on the commit and pull request activity in the original repository.

Table 1: An overview of the dataset collected using queries like: "select count(id) from TABLE".

# Users	# Projects	# Commits
499,485	108,718	555,325

An overview of the data we study is presented in Table 1. Note that each of the IDs are unique for each studied table. Therefore, Table 1 shows the number of unique users, projects and commits.

In this paper, we consider a developer to be one who authors code changes to a project. In the GitHub dataset, a developer can either perform the commit themselves or send a pull request to an upstream repository maintainer. We consider both actions as development activity for our magnet and sticky analysis.

4. STUDY RESULTS

We now present the results of our study with respect to our two research questions. For each question, we discuss our approach, quantitative and qualitative results.

(RQ1) What are typical values of magnet and sticky in OSS?

Approach. We calculate magnet and sticky values for the studied OSS projects as described in Section 2. To visualize the data, we plot magnet and sticky values for each project against each other, and (similar to Khomh *et al.* [3]) divide the plot into four quadrants:

Attractive: Projects with high magnet and sticky values. Attractive projects are successful in both attracting new developers and retaining existing ones.

Fluctuating: Projects with high magnet values, but low sticky ones. Fluctuating projects are successful at attracting new developers, but unsuccessful at retaining existing ones.

Stagnant: Projects with low magnet values, but high sticky ones. Stagnant projects retain the existing development team well, but struggle to attract new members.

Terminal: Projects with low magnet and sticky values. Terminal projects struggle to retain existing developers and do not attract new ones.

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

As described in Section 2, the sticky value of a year under study depends on the number of developers in that year and the following one. Hence, to address RQ1, we focus on the most recent year (2011) that has a complete year of historical data recorded after it (2012). We are not able to use the data of 2012 or other more recent years because the dataset only contains partial results from the 2013 (i.e., until October) and no results from 2014 yet.

Note that the sticky value depends on the number of developers who contribute in the target year (Figure 1). If the number of developers in the target year is small, the sticky value tends to be high. Therefore, to reduce the influence of noise on our results, we filter away projects that have ten or fewer developers.

Quantitative results. Figure 2 shows the magnet vs. sticky quadrant plot of OSS projects in 2011. Attractive

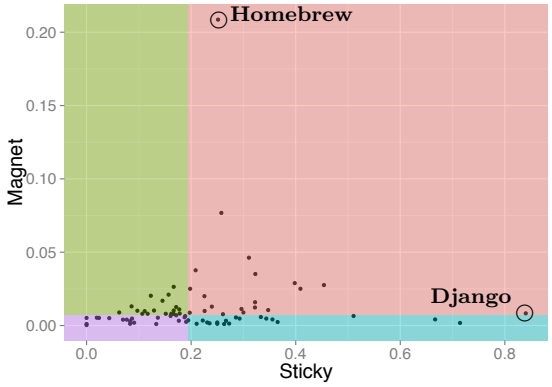

Figure 2: Distribution of magnet and sticky values for the studied projects.

projects land in the red quadrant, fluctuating projects land in the green quadrant, stagnant projects land in the blue quadrant and terminal projects land in the purple quadrant.

Figure 2 shows that magnet values tend to be much smaller than the sticky values. Indeed, most projects have sticky values that are larger than their magnet values. In other words, stickiness is a more common attribute of a software project than magnetism. This finding is in agreement with the original magnet vs. sticky study of American states, where the median of sticky value was 0.580 and the median of magnet value was 0.39[2]. Like citizens who become accustomed to their environment, developers who contribute to a project are likely to continue contributing to the same project.

Manual analysis. Figure 2 shows that there are projects with extreme magnet and sticky values. We manually inspect two such projects, i.e., the projects that have the highest magnet and sticky values.

The `Homebrew`[3] project has the largest magnet value in Figure 2. `Homebrew` is a popular package management tool for Mac OS X that began development in 2009. We suspect that in addition to `Homebrew`'s popularity, it is especially magnetic because it is relatively easy to contribute to it. For example, the smallest contribution that one can make to `Homebrew` is a "Formula cookbook", i.e., a package description that specifies the URL of a package and how it can be installed. Since most package descriptions are written in high-level scripting languages,[4] the barrier to entry for newcomers is relatively low.

The `django` project has the largest sticky value. `Django` is a high-level python web framework that began development in 2005. `Django` is used by several popular web applications, such as: (1) Instagram – a popular social image sharing application[5] and (2) ReviewBoard – a peer reviewing web application.[6] To investigate why the `django` project is so sticky, we studied the professional activity of the ten most active and loyal developers.

We find that although prior work by Zhou *et al.* reports that the most important factor to becoming a long term contributor to an open source project is a pro-community attitude [5], many of the top contributors are motivated by their professional activity. For example, many of the most loyal `django` developers began contributing during the initial development period or shortly after it. Of the ten top contributors, eight are paid to develop web applications professionally, and use `django` to do so. Their work on `django` is thus likely motivated by their professional activity.

Stickiness is a more common project attribute than magnetism. Especially sticky projects like `django` *are used professionally by many of the most loyal contributors. Especially magnetic projects like* `Homebrew` *have simple contribution processes.*

(RQ2) How do magnet and sticky values change over time?

Approach. We first analyze how all of the studied projects transition among the quadrants of Figure 2 as they age. Since the boundaries of the quadrants will likely change, we recalculate the boundaries for each studied year.

Quantitative results. Figure 3 illustrates quadrant transition likelihood using a state transition diagram. Each value describes 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, Figure 3 shows that the likelihood of transitioning from the attractive quadrant to the fluctuating one is 22%. We use the "*" state to represent the years when projects did not satisfy our filtering criteria (ten or more developers). To improve the readability of the figure, we plot two "*" states, however they are semantically identical.

This figure shows that terminal quadrant projects are the only ones that drop into the filtered away state ("*"). This agrees with our intuition, in that terminal quadrant projects are losing team members and struggling to attract new ones, which if continued for a long enough period of time would lead to the death of a project. Projects in the other quadrants are successful at either attracting new developers (high magnet) or retaining the existing ones (high sticky), and are unlikely to decay in size like those in the terminal quadrant.

We also find that fluctuating projects have same likelihood of transition to other three categories (i.e., attractive, stagnant, terminal). This also agrees with intuition, in that fluctuating projects have plenty of turnover in the development team, and hence could transition to any of the other quadrants at any time.

Figure 3 also shows that there is often a higher likelihood of transition from higher quadrants to lower ones than vice versa. This is likely due to the fact that increasing the stickiness or magnetic nature of a project requires effort to obtain (and retain) more developers. Nonetheless, in two out of nine cases (i.e., attractive-stagnant and fluctuating-terminal), projects are more likely to transition from the lower quadrant to the higher one than vice versa. While the difference in the fluctuating-terminal transition flow is minimal (i.e., only four percentage points differentiate the two directions), the attractive-stagnant case has a much broader separation. The reversal of transition flow in the attractive-stagnant case is likely because 70% of the studied projects only began development after 2009 (see Table 2). The short nature of much of the studied project history causes "trendy", short-lived, but highly popular applications to influence our results.

[2] http://www.pewsocialtrends.org/2009/03/11/sticky-states/
[3] http://brew.sh/
[4] https://github.com/Homebrew/homebrew/wiki/Formula-Cookbook
[5] http://instagram-engineering.tumblr.com/post/13649370142/what-powers-instagram-hundreds-of-instances-dozens-of
[6] http://www.reviewboard.org/

Table 2: Quadrant transitions of long-lived open source projects.

Quadrant in 2011	Project Name	2004	2005	2006	2007	2008	2009	2010
Attractive	rails	*	Terminal	Stagnant	Stagnant	Fluctuating	Fluctuating	Attractive
	xbmc	-	Stagnant	Attractive	Attractive	Stagnant	Attractive	Terminal
	django	-	*	*	Stagnant	Stagnant	Stagnant	Stagnant
Fluctuating	jquery	-	-	Fluctuating	Terminal	*	Attractive	Fluctuating
	paperclip	-	-	-	*	Fluctuating	Terminal	Fluctuating
	compass	-	-	-	-	*	Fluctuating	Fluctuating
Stagnant	scala	Terminal	Terminal	*	*	Attractive	Terminal	Fluctuating
	memcached	*	*	Terminal	Fluctuating	Stagnant	Terminal	*
	clojure	-	-	*	*	*	Attractive	Attractive
Terminal	django-debug-toolbar	-	-	-	-	Terminal	Terminal	Terminal
	jekyll	-	-	-	-	Fluctuating	Fluctuating	Terminal
	blueprint-css	-	-	-	*	*	Terminal	*

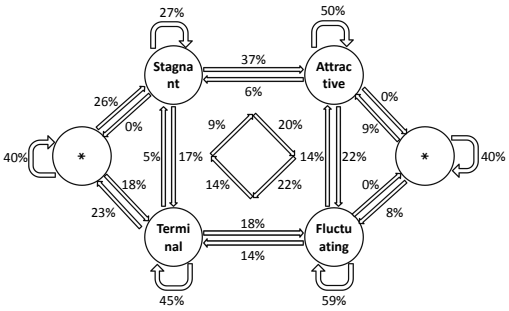

Figure 3: The likelihood of quadrant transitions

Furthermore, to confirm the risk of becoming obsolete, we check how many of the projects that enter the filtered state ("*") were in the terminal state just before. We find that two-thirds of filtered state projects originate from the terminal state.

Manual analysis. We select three projects from each quadrant that have the longest history, and study each quadrant transition that they make in depth. Table 2 shows the quadrant transition history of the 12 selected projects. In this table, "-" means that the project was not yet under development, and "*" means that the project started, yet, lacked a large enough contributing team to satisfy our filtering criteria (i.e., more than ten active developers).

We first discuss the `Rails` project, which is one of the most popular web application frameworks, as it transitions through the quadrant states. The first version of `Rails` was released in 2004, version 1.0 was released in Dec. 2005, version 2.0 was released in Dec. 2007 and version 3.0 released in Aug. 2010. From the Table 2, `Rails` became a highly magnetic project in 2008, which coincides with the release of version 1.2 (1.2.6 is a stable version in ver. 1.x) and 2.0. We suspect that the appearance of the stable version increased the visibility of `Rails`, which in turn increased developer interest. Furthermore, the `Rails` 2.0 introduced `SQLite3` as the backend datastore and encouraged the use of REpresentational State Transfer (REST), which also likely intrigued technology-motivated developers to participate.

Next, we discuss the `jQuery` project, which began development in 2006 and is one of the most popular JavaScript libraries today. Interestingly, Table 2 shows that `jQuery` has transitioned between three quadrants and "*" state during its lifespan. The bursty nature agrees with the hypothesis of Bird *et al.* [1] that the rate of immigration in open source projects is non-monotonic. Early in development, `jQuery` transitioned from the terminal quadrant to "*" in 2008 and then to the attractive quadrant in 2009. Although the transition from the terminal quadrant to "*" state is not uncommon, the transition from "*" state directly to the attractive quadrant is interesting and worth exploring. In fact, on Sep. 2008, Microsoft and Nokia announce their support for jQuery[7]. This news generated much interest in `jQuery`.

23% of terminal quadrant projects eventually decay into a state where they have ten or fewer contributors, suggesting that our quadrant analysis can successfully identify projects at risk of becoming obsolete. Furthermore, we find that many quadrant transitions are accompanied by interesting events in a project's history.

5. CONCLUSIONS

Migratory trends of open source developers are of interest for software engineers. Project maintainers would like to retain the active developers that they have and would also like to attract new ones to grow the community.

This paper applied the magnet and sticky population concepts to a set of open source projects. We find that:

- Stickiness is a more common project attribute than magnetism, which can be motivated by more than a pro-community attitude [5], but also by professional activity (e.g., `django`).
- Quadrant plots can effectively identify at-risk projects.
- Quadrant transitions often coincide with interesting events in a project's history.

6. ACKNOWLEDGEMENTS

This research was partially supported by JSPS KAKENHI Grant Numbers 24680003 and 25540026 and the Natural Sciences and Engineering Research Council of Canada (NSERC).

References

[1] C. Bird, A. Gourley, P. Devanbu, A. Swaminathan, and G. Hsu. Open borders? immigration in open source projects. In *Proc. MSR*, pages 6–13, 2007.

[2] G. Gousios. The ghtorrent dataset and tool suite. In *Proc. MSR*, pages 233–236, 2013.

[3] F. Khomh, B. Chan, Y. Zou, and A. E. Hassan. An entropy evaluation approach for triaging field crashes: A case study of mozilla firefox. In *Proc. WCRE*, pages 261–270, 2011.

[4] A. Mockus, R. T. Fielding, and J. D. Herbsleb. Two case studies of open source software development: Apache and mozilla. *TOSEM*, 11(3):309–346, 2002.

[5] M. Zhou and A. Mockus. What make long term contributors: Willingness and opportunity in OSS community. In *Proc. ICSE*, pages 518–528, 2012.

[7] `http://blog.jquery.com/2008/09/28/jquery-microsoft-nokia/`