2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR)

The Impact of the Adoption of Continuous Integration on Developer Attraction and Retention

Yash Gupta, Yusaira Khan, Keheliya Gallaba, and Shane McIntosh
Department of Electrical and Computer Engineering
McGill University
Montréal, Canada
{yash.gupta, yusaira.khan, keheliya.gallaba}@mail.mcgill.ca, shane.mcintosh@mcgill.ca

***Abstract*—Open-source projects rely on attracting new and retaining old contributors for achieving sustainable success. One may suspect that adopting new development practices like Continuous Integration (CI) should improve the attractiveness of a project. However, little is known about the impact that adoption of CI has on developer attraction and retention. To bridge this gap, we study how the introduction of TRAVIS CI—a popular CI service provider—impacts developer attraction and retention in 217 GITHUB repositories. Surprisingly, we find that heuristics that estimate the developer attraction and retention of a project are higher in the year before adopting TRAVIS CI than they are in the year following TRAVIS CI adoption. Moreover, the results are statistically significant (Wilcoxon signed rank test, $\alpha = 0.05$), with small but non-negligible effect sizes (Cliff's delta). Although we do not suspect a causal link, our results are worrisome. More work is needed to ascertain the relationship between CI and developer attraction and retention.**

I. INTRODUCTION

Thriving open-source software projects are often built by a vibrant community of contributors. Such communities are built using a platform of tools and resources that allow developers (and users) to collaborate [6]. Indeed, retaining existing contributors and attracting new contributors is essential to the sustained success of OSS projects [3]. Over the lifetime of a project, maintainers introduce new workflows and practices accompanied with new tools to increase developer productivity and also to make the product attractive to new contributors.

Continuous Integration (CI) is one such practice in which incremental changes to a software project are automatically built, tested for regression, and checked for common quality problems as they arrive (or just prior to arriving) in the project repository. TRAVIS CI stands out as one of the most popular cloud-based providers of CI services. Little is known about how the adoption of modern tools, like CI, impact developer attraction and retention.

In this paper, we perform an exploratory study of the relationship between TRAVIS CI adoption and developer attraction and retention. Through analysis of 217 projects from the MSR challenge dataset [1], we address the following research question:

How does the adoption of CI impact developer attraction and retention?

Surprisingly, we find that developer attraction and retention heuristics tend to be higher in the year prior to adopting TRAVIS CI than they are in the year afterwards. These results are statistically significant (Wilcoxon signed rank test, $\alpha = 0.05$), with small but non-negligible effect sizes (Cliff's delta).

Although we do not suspect a causal link, our results suggest that the link between CI adoption and developer attraction and retention is complex. More work is needed to ascertain the relationship between CI and developer attraction and retention.

Paper organization. In the remainder of the paper, we describe the design of our case study (Section II), its results (Section III), and some conclusions and promising avenues for future work (Section IV).

II. CASE STUDY DESIGN

The mining challenge dataset provides TRAVIS CI build information for 3,702,595 builds spanning 1,283 GITHUB projects. Figure 1 provides an overview of the three-step approach that we followed to answer the research question using the mining challenge dataset. To aid in future replication of our work, we make our data and scripts available online.[1]

A. Data Extraction

The mining challenge dataset contains version history after TRAVIS CI was introduced to each project. However, to gather relevant data about how the adoption of TRAVIS CI affects project attractiveness and retentiveness, we extract additional version history data dating back to the start of the project from GITHUB. Our data extraction approach is described below.

Stage DE: Extract Commit Metadata. We extract relevant metadata from the commit logs of the studied projects. More specifically, from each commit, we extract its commit ID (hash), author name, author email, and timestamp.

We use the author email to associate each commit with its contributor. However, email data is known to be noisy [2] and to combat this noise, we apply the email address disambiguation procedure proposed by Bird *et al.* [2].

B. Data Filtering

After extracting relevant metadata, we filtered the mining challenge dataset before using it to compute our metrics.

Stage DF 1: Branch Analysis. To conduct our analysis, we must first identify the main development branch of the studied

[1] https://dx.doi.org/10.6084/m9.figshare.4805848

978-1-5386-1544-7/17 $31.00 © 2017 IEEE
DOI 10.1109/MSR.2017.37

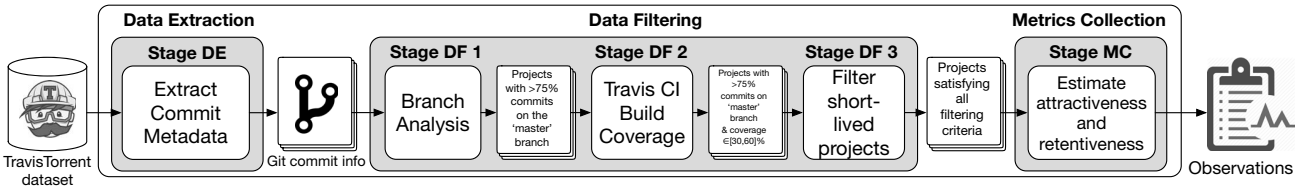


Fig. 1: An overview of our approach to study the relationship between CI adoption and developer attraction and retention.

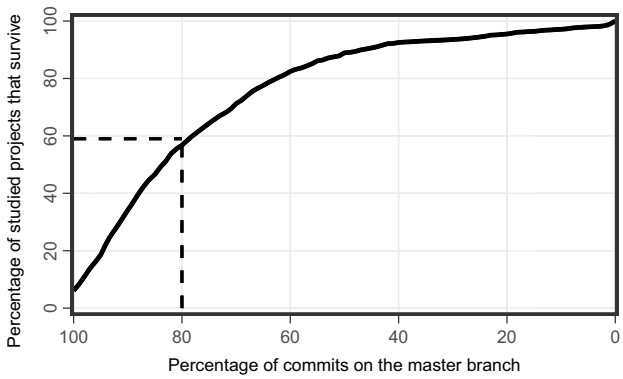


Fig. 2: Threshold plot showing the percentage of projects that survive (Y-axis) when various the thresholds for the percentage of commits on the `master` branch (X-axis) are considered.

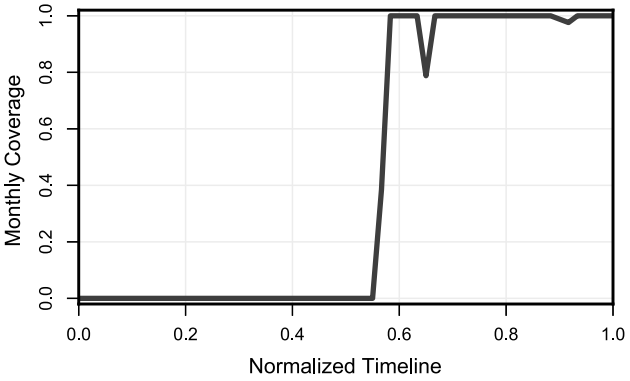


Fig. 3: TRAVIS CI Coverage Graph for the `data_axle/cassandra_object` project from the TRAVIS TORRENT database. The area under the curve is 0.427.

projects. However, projects may follow different branching strategies in their `git` repositories. We focus on repositories that follow the `master` branch strategy, where main development is performed on (or merged into) the `master` branch.

To detect the projects that are using the `master` branch strategy, we compute the proportion of commits that are found in the `master` branch for each of the candidate projects. To determine the suitable cutoff value for the proportion of commits on the `master` branch, Figure 2 plots potential thresholds against the number of surviving repositories.

By analyzing the figure, we selected a threshold value of 80% commits on `master`, which reduced our dataset to 758 projects (59.1% of the candidate projects in the dataset).

Stage DF 2: TRAVIS CI Build Coverage. Candidate projects must have historical data prior to and after adopting TRAVIS CI to ensure that developer attraction and retention values can be compared among the periods.

In order to detect such projects automatically, we first compute the *build coverage*, i.e., the proportion of commits that land on the `master` branch that can be associated with a TRAVIS CI build. To identify the commits that could be associated with a TRAVIS CI build, we use the TRAVIS TORRENT dataset. For each candidate project, we compute build coverage in every month that there was commit activity.

We then normalize the timeline of each project such that every month in the lifetime of the project could be represented by a number between 0–1. Note that the last month of the projects had to be truncated to August 31, 2015, since this was the final date of builds in the TRAVIS TORRENT dataset. We then plot the monthly build coverage against the normalized timeline, yielding a curve in the unit square space.

In order to detect projects that have sufficient pre- and post-TRAVIS CI data for our analysis, we compute the *Area Under the Build Coverage Curve* (AUBCC). Since the curve is in the unit square space, the values of AUBCC range between 0–1, where an AUBCC value of 0 indicates that the project has not used TRAVIS CI, and an AUBCC value of 1 indicates that the project has used TRAVIS CI since its inception.

Figure 3 shows the build coverage curve for the `data_axle/cassandra_object` project, which is an example of a candidate project that we would like to analyze. The AUBCC value of this project is 0.427.

Figure 4 shows a histogram of the AUBCC values for all of the candidate projects. The distribution of AUBCC values has a mean of 0.603 and a standard deviation of 0.273. Based on this histogram, we select the 261 candidate projects that have an AUBCC value between 0.30–0.60, which have a similar number of commits prior to and after TRAVIS CI adoption.

Stage DF 3: Filter short-lived projects. From the 261 candidate projects, we select those that had a lifetime of at least 12 months prior to and after the adoption of TRAVIS CI. 217 projects satisfy this criterion and are selected for analysis.

C. Compute Developer Attraction and Retention Heuristics

In this study, we focus on the developer attraction and retention properties of projects. Similar to prior work [4], [5], we operationalize these characteristics using the notions

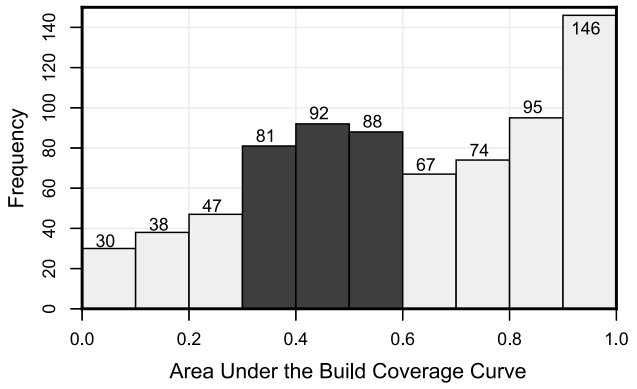


Fig. 4: A histogram of the Area Under the Build Coverage Curve (AUBCC) values for the candidate TRAVIS TORRENT repositories. The repositories which are in the darkly shaded area were selected for our analysis.

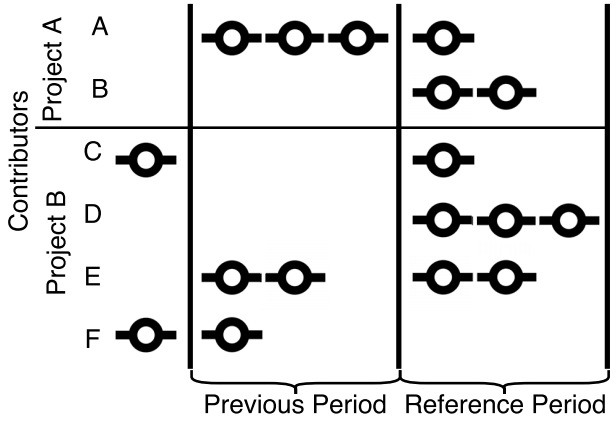


Fig. 5: Figure showing the calculation of *newContributors*, *contributors* and *returnContributors*

of *magnetism* and *stickiness*, which were proposed by a Pew Research Center report.[2] We define these concepts below:

- **Magnetism** is the proportion of contributors in the reference period who made their first commit during that period *(newContributors)* to the number of distinct contributors to the project in that period *(contributors)*, i.e., for a reference period t, $magnetism(t) = \frac{newContributors(t)}{contributors(t)}$.
- **Stickiness** is the proportion of contributors in the reference period who also made a commit during the period prior to the reference period *(returnContributors)* to the number of distinct contributors to the project in that previous period *(contributors)*, i.e., $stickiness(t) = \frac{returnContributors(t)}{contributors(t-1)}$.

Figure 5 provides an illustrative example of magnetism and stickiness. Project A had two contributors in the reference period, of which only one contributor made their first contribution during that period, i.e., $magnetism(t) = \frac{1}{2}$. Similarly, Project B had three contributors in the reference period, of which only one contributor made their first contribution during that period, i.e., $stickiness(t) = \frac{1}{3}$.

For each project, we compute magnetism and stickiness scores prior to and post-TRAVIS CI adoption. We analyze reference periods of length two months and one year to see if the period length has any impact on our findings. We use Wilcoxon signed rank tests ($\alpha = 0.05$) to check to see if there is a significant different in the pre- and post-TRAVIS CI values. Moreover, we use Cliff's delta to measure the effect size, which is *negligible* when $delta < 0.147$, *small* when $0.147 \leq delta < 0.33$, *medium* when $0.33 \leq delta < 0.474$, and *large* otherwise.

III. CASE STUDY RESULTS

In this section, we discuss the results of our case study. First, we describe our approach and then we describe the results.

Approach. First we extracted the contributor statistics from 7 to 12 months and from 1 to 6 months before TRAVIS CI was introduced. For getting the date when TRAVIS CI was introduced in the workflow of the project, we look in the mining challenge dataset for the first commit that triggered a build in TRAVIS CI. We take this commit as the demarcation point between the before and after time periods. Then we used these statistics to compute the magnetism and stickiness metrics for the period of one year before the introduction of CI.

Similarly, for the one year period after the introduction of CI, we extracted the contributor statistics first from 1 to 6 months and then from 7 to 12 months after TRAVIS CI was introduced. Then we used them to compute the magnetism and stickiness metrics for the period of one year after the introduction of CI.

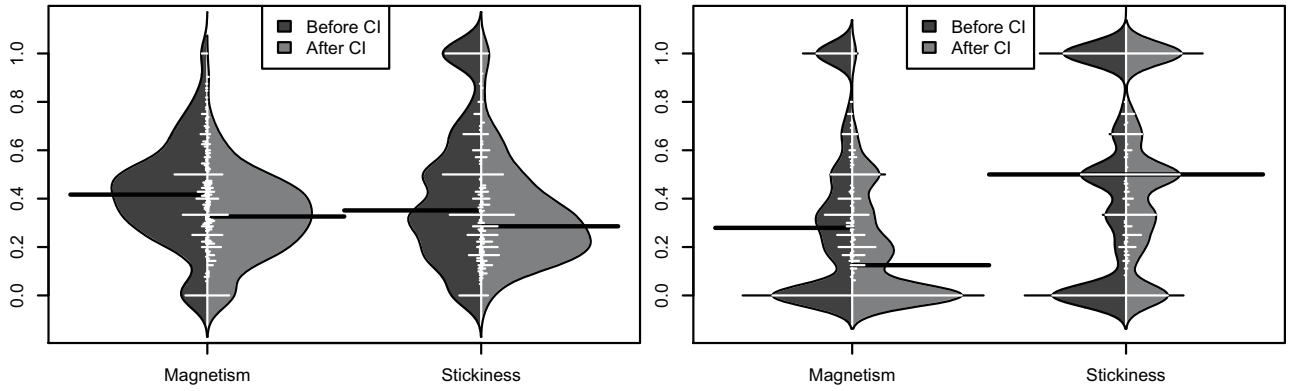
Results. Magnetism and stickiness values tend to drop after TRAVIS CI has been adopted. Figure 6a shows the magnetism and stickiness values for one year time periods. The median Magnetism one year before CI is 0.415, mean is 0.415 and the standard deviation is 0.208. One year after CI, the median is 0.326, mean is 0.320 and the standard deviation is 0.165. A Wilcoxon signed rank test reveals that the drop in magnetism is statistically significant ($p = 1.35 \times 10^{-7}$). Moreover, we obtain a Cliff's delta of 0.293, which is considered small, but non-negligible.

When we turn to stickiness, we find that it drops from: (i) a median of 0.349 before to 0.286 after; (ii) a mean of 0.425 before to 0.329 after; and (iii) a standard deviation of 0.289 to 0.197 after. A Wilcoxon signed rank test reveals that the drop in stickiness is statistically significant ($p = 4.42 \times 10^{-6}$), with a Cliff's delta of 0.186, which is again considered small, but non-negligible.

***Observation 1**: Magnetism and stickiness values tend to drop in the year that follows* TRAVIS CI *adoption.*

Figure 6b shows the magnetism and stickiness values for two month time periods. The median Magnetism two months before CI is 0.273, mean is 0.342, and the standard deviation is 0.346. Two months after CI, the median is 0.125, mean is 0.183 and the standard deviation is 0.222. A Wilcoxon signed rank test reveals that the drop in magnetism is statistically significant

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



(a) One year time periods

(b) Two month time periods

Fig. 6: A beanplot comparing the Magnetism and Stickiness scores prior to and post-TRAVIS CI adoption.

($p = 2.23 \times 10^{-8}$). Moreover, we obtain a Cliff's delta of 0.246, which is again considered small, but non-negligible.

Similarly, for Stickiness, the median Stickiness two months before CI is 0.500, mean is 0.460, and the standard deviation is 0.396. Two months after CI, the stickiness values have a median of 0.500, mean of 0.492, and a standard deviation of 0.347. A Wilcoxon signed rank tests reveals that we do not have enough evidence to reject the null hypothesis, i.e., the stickiness values pre- and post-TRAVIS CI are statistically indistinguishable. Moreover, we obtain a Cliff's Delta of -0.06, which is considered negligible.

Observation 2*: The magnetism two months prior to* TRAVIS CI *adoption is significantly higher than magnetism in the two months after* TRAVIS CI *adoption, with a small effect size. However, stickiness values are statistically indistinguishable before and after* TRAVIS CI *adoption.*

Implications. Surprisingly, magnetism and stickiness values tend to decrease after TRAVIS CI has been adopted in the studied systems. However, we caution against drawing naïve conclusions based on these observations (e.g., one should avoid adopting CI). There are several reasons to adopt a CI workflow, such as improvements in defect reporting and a reduction in developer overhead, since unlike less frequent build cycles, problems can be caught early, while tradeoffs and design decisions are still fresh in the developers' minds. If, however, the primary motivation for a move to a CI workflow is to attract or retain developers, our observations suggest that this is unlikely to come to fruition.

IV. CONCLUSIONS

Developer attraction and retention are key characteristics for the growth of a community around an open-source project. In theory, adoption of modern development techniques and tools should have a positive effect on developer attraction and retention. In this paper, we analyzed the relationship between adoption of TRAVIS CI—a popular CI service provider—and project magnetism (a heuristic for developer attraction) and stickiness (a heuristic for developer retention). Surprisingly, our results suggest that TRAVIS CI adoption is accompanied by a statistically significant drop in magnetism and stickiness.

While these observations are worrisome, we cannot claim that there is a causal link between the phenomena. A limitation of our current analysis approach is that we have not controlled for confounding factors, such as team or system size. An approach to address this limitation would be to train regression models to explain the studied relationship while controlling for several confounding factors. We plan to conduct such a study in future work. In future work, we also plan to analyze the trend of the magnetism and stickiness over the years rather than using snapshots of a year or 2 months.

REFERENCES

[1] M. Beller, G. Gousios, and A. Zaidman. Travistorrent: Synthesizing travis ci and github for full-stack research on continuous integration. In *Proceedings of the 14th working conference on mining software repositories*, 2017.

[2] C. Bird, A. Gourley, P. Devanbu, M. Gertz, and A. Swaminathan. Mining Email Social Networks. In *Proceedings of the 3rd Int'l Conf. on Mining Software Repositories (MSR)*, pages 137–143, 2006.

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

[4] M. Ortu. *Mining Software Repositories: Measuring Effectiveness and Affectiveness in Software Systems*. PhD thesis, University of Cagliari, 2014.

[5] K. Yamashita, S. McIntosh, Y. Kamei, and N. Ubayashi. Magnet or sticky? an oss project-by-project typology. In *Proceedings of the 11th working conference on mining software repositories*, pages 344–347. ACM, 2014.

[6] Y. Ye and K. Kishida. Toward an understanding of the motivation open source software developers. In *Proceedings of the 25th international conference on software engineering*, pages 419–429. IEEE Computer Society, 2003.