How Important is Mentoring for New Contributors in an OSS Project?

A Quantitative Study of the Rust Compiler Team

Vala Zeinali
School of Computer Science
Kent State University
Kent, OH, USA
vzeinali@kent.edu

Chris Bogart
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA, USA
cbogart@andrew.cmu.edu

Daniel Klug
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA, USA
dklug@cs.cmu.edu

James Herbsleb
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA, USA
jdh@cs.cmu.edu

ABSTRACT

In order for an open source software (OSS) project to survive, the project must attract and retain new contributors. There are many things ecosystems do to sustain a project. Our paper looks at one of those initiatives, mentorship. In this paper we use quantitative practices to investigate how OSS projects matriculate and retain new contributors. Our quantitative analysis shows new contributors that submit pull requests that reference issues with a mentor present are more likely to stay for their second year after their first year. We then introduce a preliminary qualitative approach that unengaged mentors cause new contributor disengagement. Overall, we conclude that mentorship is paramount to retain new contributors in the Rust OSS ecosystem project.

CCS CONCEPTS

• Empirical Software Engineering• Socio-Technical Complex Systems • Mining Software Repositories • Contributor Retention • Mentorship

KEYWORDS

Open source software, mentorship, quantitative analysis, data mining software repositories

ACM Reference format:

Vala Zeinali, Chris Bogart, Daniel Klug and James Herbsleb. 2020. How Important is Mentoring for New Contributors in an OSS Project? A Quantitative Study of the Rust Compiler Team. In *Proceedings of ACM CHASE conference workshop (CHASE'20). ACM, New York, NY, USA, 4 pages.* https://doi.org/10.1145/1234567890

1 Introduction

Contributor retention in OSS projects affects the sustainability of said project within the given ecosystem [2, 5, 13, 16, 17]. There have been studies done to determine variables that result in contributor retention and disengagement. Miller and colleagues [16] find that most disengagement comes from occupational reasons, i.e., a developer got a new job that doesn't support OSS, they changed role/project, or they left a job where they contributed to OSS. There are many reasons a contributor may leave an OSS project.

We perform similar analysis to Constantino [5], however we consider an additional factor that may contribute to retention in OSS projects. We investigate the effect on prospective developers when they contribute a pull request (PR) to issues that have been tagged as having a mentor available to help. We hypothesize that a first year Rust contributor will remain in the OSS for a second year if they experience mentorship within their assigned issue and/or contribute more effort in their PRs measured by lines of code (LOC). We explore the role of mentorship in the Rust compiler project through both a quantitative and preliminary qualitative approach [6, 7, 22].

We raise the following research questions:

RQ1: *Are contributors that put in more effort in their first year more likely to put in effort in their second year?*

RQ2: *Are contributors who author more pull requests (PR) that reference issues that contain the label "E-Mentor" more likely to produce effort in their second year?*

CHASE'20, MAY, 2020, Seoul, South Korea

https://doi.org/10.1145/1234567890

2 Related Work

Employee retention and turnover has been widely studied [4, 11, 12, 15]. There have been a few papers in the socio-technical systems track that break down possible reasons for contributor disengagement [2, 4, 5, 13, 16]. Miller identifies possible reasons for disengagement, such as occupational reasons like leaving a company that contributes to OSS, social reasons such as losing interest in the OSS project, and technical reasons like leaving GitHub for another repository system [3]. Likewise, Constantinou and Mens conducted an empirical comparison of developer retention in the rubygems and npm software ecosystems that focuses on the amount of communication a contributor makes and the frequency of commits the contributor makes in the ecosystem [5]. One of the best indicators in predicting the probability of a contributor survival is how often they commit to the project [5].

It seems that the overall pattern described in prior research is the more you put in, the more you get out and more effort equals a higher probability of surviving in the ecosystem. It is known that organizations want their employees to be engaged. There are indicators that engaged employees are more productive and there is a link between employee engagement and discretionary effort, innovation, customer loyalty, quality, and productivity [3, 10]. Such studies have led to increased interest in what drives employee engagement and this is where 'feeling valued' is important. Robinson et al's research on the NHS found that "The strongest driver of all (drivers) for engagement is a sense of feeling valued and involved" [19]. Instinctively, it is clear that employees want to feel valued at work or have a 'sense of value' and they report that this is what makes them feel engaged.

Humans yearn for connection with other humans. Connection comes in many forms, however, in our paper we will view human connection and feeling valued as mentorship within the Rust compiler team. The process of mentorship in Rust starts when an issue is created. A mentor is then assigned to the issue and then a new contributor declares himself as the mentee. Issues that have mentors in Rust are signaled by the label "*E-Mentor*" (*Figure 1*). The mentor is supposed to guide the mentee until the issue becomes closed [18]. Mentor's and mentees will communicate within the issue thread until the mentee or someone else merges a PR referencing the issue.

Figure 1: shows what an issue that is tagged with "E-Mentor" label looks like to a contributor in the project repository on GitHub.

3 Experimental Procedure

3.1 Dataset

Our dataset contains every PR and issue from every contributor (number of contributors = 3008) in the Rust compiler team from July 2010 to July 2019 (number of PRs. referencing issues = 84004, *figure 2)*. In addition, we have every issue ever created during that duration. We specifically look at how many lines of code a contributor adds and deletes per PR. In addition, we look at how many pull requests reference issues that have been labeled with the "*E-Mentor*" label. Labels are a signal to contributors, these labels signal to a contributor what type of issue is present in the system. In the Rust ecosystem, there are currently 264 types of signaling labels. Labels can range from denoting what type of feature an issue works on, the area of expertise, the type of problem the issue references, and the type of engagement required. Rust uses the "*E-Mentor*" label in a way to encourage and assist new contributors by signaling to them that they will be mentored through the issue [1]. In this paper we focus on the type of engagement required for the issue. Specifically, in this paper we will focus on the "*E-Mentor*" label. We match every PR pushed that references an issue that is labeled with the "*E-Mentor*" label to the corresponding contributor. We keep a count of the number of PRs pushed that reference an issue labeled with "*E-Mentor*" for each contributor for their first year in the environment.

3.2 Quantitative Methodology

First, we mine nine years of GitHub repository data on the Rust compiler team [23]. Then we explore the overall effort put in by all levels of contributors over time and within their first two years in the environment (effort is noted by additions of LOC + deletions of LOC). Next, we classify all contributors in the repository as "new" or "old" *(figure 2)* contributors for every year and graph the shift in two-year moving average contributor effort (*figure 3).* We then explore if issues labeled "*E-Mentor*" help new contributors stick around and put effort in their second year.

if(PrProposedYear != AuthorsFirstActionYear)
THEN
Status ← *OLD*
ELSE
STATUS ← *NEW*

Figure 2: shows contributor classification in dataset.

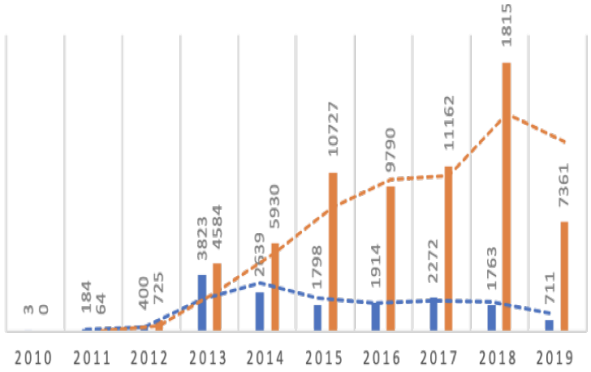

Figure 3: Rust Compiler Project Ecosystem Contributor Demographic Evolution represents the trend of new and old contributors authoring PRs. Orange bars represent old contributors (> 1 year in Rust). Blue bars represent new contributors. Orange and blue dotted line is two-year moving average for old contributors and new.

We label the contributors as "present" if they had effort in their second year, and "absent otherwise (*figure 4*). Lastly, for our quantitative approach, we run a multi-linear regression model to predict second year effort based first year effort and number of pushed PRs referencing issues that contain the label "*E-Mentor*" [14].

if(SecondYearEffort > 0) THEN
Status ← *PRESENT*
ELSE
STATUS ← *ABSENT*

Figure 4 shows how we classify our data.

4 Experimental Results

4.1 Quantitative Results

After running our multi-linear regression model, we find that developers' total lines of code added or deleted in their first year (effort_y1), additions of lines of code (addisions_y1), and number of lines of coded added for mentored issues (num_mentor_done_y1) are together predictive of the amount of effort in the second year (effort_y2, all factors significant at p<.001). Our model predicts that developers write 2.18 more lines of code in the second year for every line of code they write in the first year. In addition, we have statistically significant evidence (p < .001) to believe that effort in year one increases developers' chance of putting effort in year two.

```
Coefficients:
                   Estimate Std. Error t value Pr(>|t|)
(Intercept)        481.8266   209.5231   2.300   0.0216 *
effort_y1            2.1841     0.1188  18.378  < 2e-16 ***
additions_y1        -3.5891     0.2266 -15.838  < 2e-16 ***
num_mentor_done_y1 391.7194    96.1625   4.074  4.8e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Finally, second year effort will change by approximately 391.7194 units as number of PRs referencing issues labeled with "*E-Mentor*" increases by 1 (units) on average. Since our p-value < .001, we have reason to reject the null hypothesis and have reason to believe our alternative hypothesis that on average, pushing more PRs that reference issues labeled with "*E-Mentor*" increases the average amount of effort put in year two.

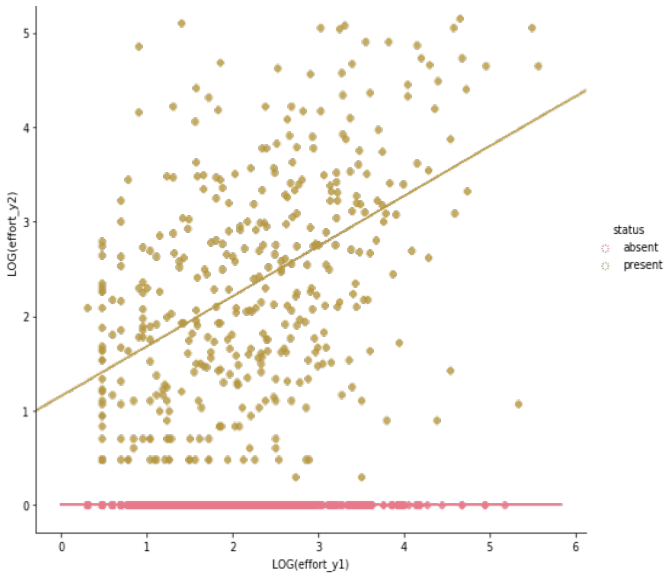

Figure 5: Effort in Year One (effort_y1) VS. Effort in Year Two (effort_y2), R^2=.19. Values in log form.

In our sample 19.0% (R-squared*100) of the variability in effort_y2 can be explained by the linear association between effort_y2 and effort_y1 (*figure 5*). Overall, since our p-value < .001, we have reason to reject the null hypothesis and have reason to believe our alternative hypothesis that on average, putting in more effort in year one increases the average amount of effort put in year two.

After running our quantitative empirical study on the Rust compiler team ecosystem, we answer our first two research questions (***RQ1*** and ***RQ2***). We then move on to make a preliminary inspection of the mentorship relationship with contributors.

5 Discussion

As a preliminary first look at how mentorship relationships actually play out, we randomly selected 5 new contributors that have pushed PRs referencing issues labeled with "*E-Mentor*" from Rust before 2019 and examine their interactions within the issue. We look at two things: within the issue thread, was there an assigned mentor to the issue (Y/N) and if so, was the mentor unengaged (did they respond to mentees questions). We then track our 5 individuals from the date of an issue and see how they evolved since the marked issue.

5.1 Preliminary Qualitative Results

Individual	Mentor Present?	Mentor Engaged?	Days in Eco. Before Leaving Rust
1	YES	NO	2130
2	YES	NO	9
3	YES	YES	5
4	YES	NO	Present*
5	YES	YES	Present*

*** Contributor has committed to the project within the last 50 days since data pulled 1/20/20**

In our random selection of 5 individuals, we create the table shown above. We anonymize the names of the individuals to protect their identity. In our random selection and examination, we found two cases of unengaged mentors. However, of those two mentors, only one mentee left immediately after the experience of an unengaged mentor. We show an example of an unengaged mentor in (*figure 6).* The new contributor left the Rust project five days later after getting no response from the assigned mentor (*figure 6*).

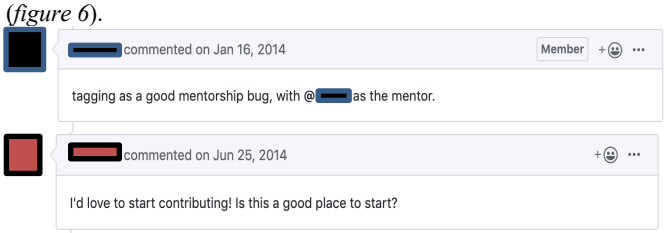

Figure 6: An exchange between individual 3 and an unengaged mentor. After the new contributor commented, the mentor never responded. The new contributor is denoted by the red square.

6 Experimental Limitations

One major threat to validity is the notion of confounding variables. It may very well be the case that there is a better

indicator to solving retention issues. Second, just because mentorship worked in one ecosystem, does not guarantee that mentorship will work in another ecosystem. Mentorship may work for a certain culture but be offensive or an insult to another culture. We simply analyze one ecosystem for signs of beneficiary factors to retain new contributors in an OSS ecosystem. We mine and analyze publicly available data on GitHub, our dataset does not include private conversations that may have taken place elsewhere from public Rust complier repository.

7 Conclusion / Future Work

We conclude that effort put in during your first year is indeed significant in predicting your survival in an OSS project for your second year. Likewise, we find that pushing more PRs that reference issues labeled with "*E-Mentor*" increases the average amount of effort put in year two and thus being retained in the Rust OSS ecosystem. Our qualitative examination showed that there is no direct magic in mentorship, and that some new contributors that had unengaged mentors continued to work on the project for more than a couple years and that some left the project in less than a week. Additionally, those with engaged mentors ranged from leaving Rust in less than two weeks to being present for close to three years.

We would like to continue our exploration of mentorship in a qualitative approach to cross validate our quantitative results. We plan on contacting past and present contributors in the Rust compiler project at random. We then would like to ask about their experience with working on issues with a mentor and if they are still working on the project and if not, why they aren't. In addition, we plan on scaling up our sample size in our preliminary qualitative method presented in the paper.

8 Acknowledgements

This research was possible by the REU-SE program at Carnegie Mellon University. I am very grateful for my experience in the program. Specifically, I want to thank Chris Bogart, Jim Herbsleb, and Daniel Klug for their time and knowledge. Thank you to my significant others for their support as always.

9 References

[1] Bauer, T. N. Erdogan, B. "Organizational socialization: The effective onboarding of new employees" in S. Zedeck (Ed.), APA Handbook of industrial and organizational psychology, Vol. 3, pp. 51-64. Washington, DC, USA, 2011, American Psychological Association.

[2] Benjamin Gidron. Predictors of retention and turnover among service volunteer workers. Journal of Social Service Research, 8(1):1–16, 1985.

[3] Blessing White, (2008), The State of Employee Engagement 2008: Highlights for U.K. and Ireland, 3. 12.

[4] Jailton Coelho and Marco Tulio Valente. Why modern open source projects fail. In Proc. Int'l Symposium Foundations of Software Engineering (FSE), pages 186–196. ACM, 2017.

[5] Eleni Constantinou and Tom Mens. An empirical comparison of developer retention in the rubygems and npm software ecosystems. Innovations in Systems and Software Engineering, 13(2-3):101–115, 2017.

[6] Creswell, J. W. (2003). Research design: Qualitative, quantitative, and mixed methods approaches (2nd ed.). Thousand Oaks, CA: Sage.

[7] John W Creswell and Vicki L Plano Clark. Designing and conducting mixed methods research. Wiley Online Library, 2007.

[8] Fagerholm, F., Sanchez Guinea A., Münch, J., Borenstein, J. "The Role of Mentoring and Project Characteristics for Onboarding in Open Source Software Projects". Empirical Software Engineering and Measurement (ESEM), 2014.

[9] Fuller, W.A. (1984). Least Squares and Related Analyses for Complex Survey Designs. Survey Methodology 10, 97-118.

[10] Gruman J.A. and Saks A.M., (2011), Manage Employee Engagement to Manage Performance, Industrial and Organizational Psychology, Vol. 4, Iss. 2, pp. 204–207, Wiley

[11] Peter W Hom, Thomas W Lee, Jason D Shaw, and John P Hausknecht.
One hundred years of employee turnover theory and research. Journal of Applied Psychology, 102(3):530, 2017.

[12] Hyosu Kim and Dennis Kao. A meta-analysis of turnover intention predictors among us child welfare workers. Children and Youth Services Review, 47:214–223, 2014.

[13] Irma Browne Jamison. Turnover and retention among volunteers in humanservice agencies. Review of Public Personnel Administration, 23(2):114–132,2003.

[14] Kutner MH, Nachtsheim CJ, Neter J, et al. Applied Linear Statistical Model. 5th ed. New York, NY: McGraw-Hill/Irwin; 2005:664 – 665, 1173–1183.

[15] Lynn E Miller, Gary N Powell, and Joseph Seltzer. Determinants of turnover among volunteers. Human Relations, 43(9):901–917, 1990

[16] Courtney Miller, David Widder, Christian Kastner, and Bogdan Vasilescu. Why do People Give Up FLOSSing? A Study of Contributor Disengagement in Open Source
10.1007/978-3-030-20883-7_11, 2019.

[17] Minghui Zhou, Audris Mockus, Xiujuan Ma, Lu Zhang, and Hong Mei. Inflow and retention in OSS communities with commercial involvement: A case study of three hybrid projects. ACM Trans. Softw. Eng. Methodol. (TOSEM), 25(2):13, 2016.

[18] Newby, T. J. and Heide, A. (1992), The Value of Mentoring. Performance Improvement Quarterly, 5: 2-15. doi:10.1111/j.1937-8327.1992.tb00562.x

[19] Robinson, D., Perryman, S. and Hayday, S. (2004), The Drivers of Employee Engagement, Report 408, Institute for Employment Studies, Brighton

[20] Sandelowski M. (1995) Focus on qualitative methods: sample size in qualitative research. Research in Nursing and Health 18, the information needs of the study. The ultimate aim is to 179–183.

[21] Steinmacher, I., Wiese, I., Chaves, A.P., Gerosa, M.A., "Why do newcomers abandon open source software projects?," 6th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE), pp. 25-32, 2013.

[22] Tashakkori, A., & Teddlie, C. (1998). Mixed methodology: Combining qualitative and quantitative approaches. Thousand Oaks, CA: Sage.

[23] T. Xie, S. Thummalapenta, D. Lo and C. Liu, Data mining for Software Engineering, IEEE Computer, 2009

[24] Trost, J.E. (1986). Statistically nonrepresentative stratified sampling: A sampling technique for qualitative studies. Qualitative Sociology, 9, 54-57.