How Long and How Much: What to Expect from Summer of Code Participants?

Jefferson O. Silva
Pontifical University of São Paulo
São Paulo, Brazil
silvajo@pucsp.br

Igor Wiese
Federal University of Technology, Paraná
Campo Mourão, Brazil
igor@utfpr.edu.br

Daniel German
University of Victoria
Victoria, Canada
dmg@uvic.ca

Igor Steinmacher
Federal University of Technology, Paraná
Campo Mourão, Brazil
igorfs@utfpr.edu.br

Marco A. Gerosa
Northern Arizona University
Flagstaff, USA
marco.gerosa@nau.edu

***Abstract*—Open Source Software (OSS) communities depend on continually recruiting new contributors. Some communities promote initiatives such as Summers of Code to foster contribution, but little is known about how successful these initiatives are. As a case study, we chose Google Summer of Code (GSoC), which is a three-month internship promoting software development by students in several OSS projects. We quantitatively investigated different aspects of students' contribution, including number of commits, code churn, and contribution date intervals. We found that 82% of the studied OSS projects merged at least one commit in codebase. When only newcomers are considered, ~54% of OSS projects merged at least one commit. We also found that ~23% of newcomers contributed to GSoC projects before knowing they would be accepted. Additionally, we found that the amount of commits and code of students with experience in the GSoC projects are strongly correlated with how much code they produced and how long they remained during and after GSoC. OSS communities can take advantage of our results to balance the trade-offs involved in entering CCEs, to set the communities' expectations about how much contribution they can expect to achieve, and for how long students will probably engage.**

Keywords: Google Summer of Code; Community Code Engagement; Open Source Software; Newcomers; Sustainability; Mining Software Repositories

I. INTRODUCTION

The sustainability and evolution of several open source software (OSS) communities depends on the influx of new volunteers [1]. Newcomers are needed not only to provide the communities with fresh ideas [2], but also to accomplish communities' valuable chores [3]. Many OSS communities have failed due to insufficient volunteer participation [4].

OSS communities are increasingly joining or promoting community code engagements (CCE), which are short-term software development initiatives. These engagements include Summer of Code internships that promote software development by students during the summer holidays [5]. Examples include Google Summer of Code (GSoC)[1], Rails Girls Summer of Code (RGSoC)[2], Julia Summer of Code (JSoC)[3], and Outreachy.[4] A variation on Summers of Code is the so-called Semester of Code engagements, which include Facebook Open Academy [5] and Undergraduate Capstone Open Source Projects.[6] While Summers of Code typically occur during holidays and may provide stipends and mentors, Semesters of Code occur along with regular course studies, possibly involving faculty members and providing students with academic credits [6].

Some CCEs are held by high profile organizations, such as Facebook, Yahoo!, and Google, which are potentially more attractive to newcomers than volunteer self-guided contribution to OSS [7], [8]. While contribution in CCEs may potentially provide students with attractive rewards, such as CV-building, stipends, and learning, little is known about how successfully CCEs retain students as committers, or whether the OSS projects merge the students' contribution into codebase. The current literature on CCEs primarily provides evidence on retention and code contribution for OSS projects in the scientific software domain [5], [9]–[11], with findings based on students' and mentors' subjective perceptions. Alternatively, Schilling *et al*. [12] mined software repositories to quantify students' retention, but only to the KDE project. Thus, not only little is known about how much CCEs promote code contribution in general, but there is also, to our best knowledge, no empirical study that quantitatively investigates retention and code contribution for more than one OSS project.

To understand the amount of contribution (*i.e*., code churn and commits) that are merged into codebase, and how long students contribute before and after GSoC, we answer the following research questions (RQ).

RQ1. How much code do CCE students contribute to codebase?

RQ1a. How many commits/code churn in codebase are contributed by the students?

Motivation: Answering RQ1 may help OSS communities set their expectations regarding the amount of code contributed by Summer of Code students.

[1] https://developers.google.com/open-source/gsoc/
[2] http://railsgirlssummerofcode.org/
[3] http://julialang.org/blog/2015/05/jsoc-cfp/
[4] https://wiki.gnome.org/Outreachy
[5] https://www.facebook.com/pg/OpenAcademyProgram/about
[6] http://ucosp.ca/

Approach: We split students into *newcomers* and *students-with-experience*. We considered that newcomers are students who did not have any commits before the announcement date of mentoring organizations and are not former GSoC students. We refer to the students who do not meet these criteria as students-with-experience. We also split the commits to GSoC projects into 3 contribution periods: *before*, *during*, and *after* GSoC. For each period, by using a unique commit identifier, we counted how many of the students' commits were merged in codebase. We assessed how much code students added by calculating the code churn (*i.e.*, lines added + lines removed) in each commit.

Findings: Merged commits occurred in all periods. Most OSS projects (~82%) merged at least one commit authored by students. When only newcomers are considered, ~54% of OSS projects merged at least one commit.

RQ2. How long do students contribute before and after CCEs?

RQ2a. What was the students' contribution before and after GSoC?

RQ2b. Is previous contribution associated with students' retention?

Motivation: The answer to this question may help OSS communities manage expectations for attracting new long-term contributors.

Approach: We estimated contribution *after* GSoC by studying the interval between GSoC end date and last commit, contributions, and the count of distinct contribution days (*i.e.*, distinct commit dates). Contribution *before* GSoC was estimated analogously, but considering the interval between first commit and GSoC kickoff. We tested correlation of metrics from both periods.

Findings: 23.1% of newcomers contributed to GSoC projects before knowing they would be accepted, while ~43% of them kept contributing longer than a month *after* GSoC, ~26% longer than six months, and ~16% longer than a year. Students-with-experience started contributing more than a year earlier than kickoff, while ~47% of them kept contributing longer than a month, ~33% longer than six months, and ~23% longer than a year. For newcomers and students-with-experience, the number of distinct contribution days was not proportional to longer contribution intervals. In addition, we found that the amount of commits and code are strongly correlated with increased levels of contribution *during* and *after* GSoC.

These findings provide empirical evidence for OSS communities on how much student contribution they can expect from GSoC participation, and how long students stay before and after the program.

II. BACKGROUND AND RELATED WORK

In this section, we present work related to newcomers' retention and community code engagements (CCE). We begin by explaining what Google Summer of Code is, how it works, and why we chose to study it.

A. Google Summer of Code

Google Summer of Code (GSoC) is a worldwide Google program that offers students a stipend to write code for OSS for a three-month period. We chose to study GSoC because it: is more well-known compared to other internships; has been in operation for more than 10 years; has a large number of students from all over the world, and provides students with a comprehensive set of participation rewards [5], including participating in a well-known large company's program, community bonding, skill development, personal enjoyment, career advancement, peer recognition, status, and stipends.

Since GSoC began in 2005, Google has paid[7] students who successfully complete all three program phases. GSoC has five goals.[8] Goals (*ii*) and (*iii*) inspire our RQs:

(*i*) "Create and release OSS code for the benefit of all"

(*ii*) "Inspire young developers to begin participating in OSS development"

(*iii*) "Help OSS projects identify and bring in new developers and committers"

(*iv*) "Provide students the opportunity to do work related to their academic pursuits (*flip bits, not burgers*)"

(*v*) "Give students more exposure to real-world software development scenarios"

Applicants must write and submit project proposals to the OSS organizations (previously approved by Google) they wish to work for, such as the Apache Software Foundation and Debian. The organizations' mentors—who are usually regular contributors—rank and decide which proposals to accept. When students effectively begin coding for their GSoC projects, Google issues them an initial payment. After the first half of the program, mentors assess their students' work and submit to Google a mid-term evaluation. For the passing students, Google issues mid-term payments. At the GSoC end, mentors submit their final evaluations to Google and students are required to submit their code. Passing students receive their remaining payment, and are invited to a summit in California.

B. Newcomers' Retention in OSS

Typically, studies on retention take the perspective of the individual developer. Thereby, intrinsic motivation (*e.g.*, [13]–[15]), social ties with team members (*e.g.*, [16]–[18]), project characteristics (*e.g.*, [19]–[21]), ideology (e.g., [22]), and incentives and rewards (*e.g.*, [7], [8], [23]) have been found most relevant for OSS developers to continue contributing.

Zhou and Mockus [24], for example, worked on identifying newcomers who are more likely to continue contributing to the project in order to offer active support for them to become long-term contributors. They found the individual's willingness and the project's climate to be associated with the odds that an individual would become a long-term contributor.

[7] From GSoC 2013 to 2015, Google paid an amount of US$ 5,500 to students

[8] At the time of this writing, GSoC had removed the reference webpage with these goals. However, the goals can be found in websites that support OSS communities, e.g: http://write.flossmanuals.net/gsocstudentguide/what-is-google-summer-of-code

TABLE I. SUMMER OF CODE SUMMARY

	Pays stipends?	Eligibility	# of participants 2016	Duration	Inception year	Internship sponsors (2016)
GSoC	Yes	Any 18+ year-old student enrolled at an accreditted university	1,206	3 months	2005	Google
JSoC	Yes	Since 2014, JSoC has been using GSoC's selection process	10	3 months	2013	MIT Lab in 2013. Google, after 2013
Outreachy	Yes	Women (cis and trans), trans men, and genderqueer people. Residents and nationals of the USA who are Black/African American, Hispanic/ Latin@, American Indian, Alaska Native, Native Hawaiian, or Pacific Islander.	46	3 months	2013	Mozilla, Bloomberg, Google, Intel, RedHat, Wikimedia Foundation, Z, Cadasta, CodeThink, Debian, Fedora, FFmpeg, Free Software Foundation, IBM, NodeJS, Open Source Robotics Foundation, Open Stack, Xen Project
RGSoC	Yes	All people with non-binary gender identities or who identify as women (transgender or cisgender)	50	3 months	2013	Softwire, innoQ, Mozilla, ThoughWorks, Exam Success, AgileBloom, Open Suse, Wooga, Apcera, Lauch School, Articulate, Ableton, Honeybadger, ActBlue, Basecamp, GitLab, CoreOS, Spotify

Fang and Neufeld [2] built upon Legitimate Peripheral Participation theory [25], which has been typically embraced to explain how an individual engages in a community of practice, to understand developers' motivation to continue sustainable contribution. The authors found that initial conditions to participate did not effectively predict long-term participation, but also that situated learning (*i.e.*, the process of acting knowledgeably and purposefully in the world) and identity construction (*i.e.*, the process of being identified within the community) behaviors were positively linked to sustained participation.

C. Community Code Engagements

As mentioned, community code engagements (CCE) are becoming a common initiative. Table I lists main differences among some programs. Despite its practical relevance, little research has examined how CCEs influence new volunteer contributions or how much of the code produced in these programs is indeed merged to the OSS projects.

Taking the communities' perspective, Schilling *et al*. [12] used the concepts of Person-Job (the congruence between an applicant's desire and job supplies) and Person-Team (the applicant's level of interpersonal compatibility with the existing team) from the recruitment literature to derive objective measures to predict the retention of 80 former GSoC students in the KDE project. Using a classification schema of prior code contributions to this project, they found that intermediate (4-94 commits) and high (>94 commits) levels of prior development were strongly associated with retention.

Trainer *et al*. [10] also took the OSS communities' perspective in a case study of a bioinformatics library called Biopython to investigate the outcomes of GSoC (for this project only). By analyzing interviews with the top 15 students ranked by the number of commits, the researchers identified three positive outcomes: (*i*) *the addition of new features to codebase,* finding that 50% of the GSoC projects were merged to codebase; (*ii*) *training*, finding that the students learned new software engineering skills, such as testing; and (*iii*) *personal development*, reporting that students use participation in GSoC for career advancement. The authors also found that mentors faced several challenges related to the GSoC process, such as issues with candidates' proposal submission and ranking; mentors are often volunteer contributors working in their spare time to help a large number of applicants write proposals.

Taking the perspective of Summer of Code organizers, Trainer *et al*. [5] conducted a multiple case study of 22 GSoC projects in the scientific domain to understand GSoC outcomes and the underlying practices that lead to them. They found that GSoC facilitated the creation of strong ties between mentors and students, reporting that 18% of the students (n=22) became mentors in subsequent editions.

While these previous works help enlighten understudied aspects of Summers of Code, their scope is restricted to a few GSoC projects and mainly to the scientific software domain; consequently, their conclusions may not be applicable to other projects. Only Schilling *et al*. [12] mined software repositories for quantifying student retention, but limited their analysis to KDE. Trainer *et al*. [5] and Trainer *et al*. [10] collected data through interviews. Although we understand the relevance of interviews for achieving their goals, their results on retention only represent the students' perception on whether students kept contributing. In addition, in the work of Schilling *et al*. [12], it is unclear whether any code written due to GSoC was merged in codebase.

We argue that CCEs have the potential to influence newcomers' experience and decision-making process. Legitimate Peripheral Participation theory [25], when applied to OSS, predicts that future contributors begin their involvement by observing before coding and then passively interacting with experienced members; this process culminates in the emergence of regular contributors. However, contributing to OSS by means of Summers of Code significantly alters this process: a contract binds students and mentors for a three-month period. Summer of Code students do not start at the margin; instead, they are individually guided—and sponsored—to become contributors. They have the time to dedicate themselves to the project, potentially developing strong social ties to both the mentor and other community members. While current studies on GSoC have targeted specific projects or domains, our study represents a more comprehensive investigation by analyzing data obtained from mining multiple software repositories.

III. RESEARCH METHOD

In this section, we present the method for data collection and analysis. For data collection, we searched for the students' assigned projects and mined repositories. For data analysis, we used descriptive statistics and statistical tests.

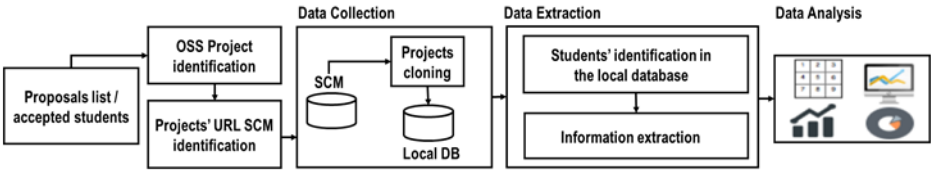


Figure 1. Method used to collect and analyze students' interaction with their GSoC projects.

A. Data Collection

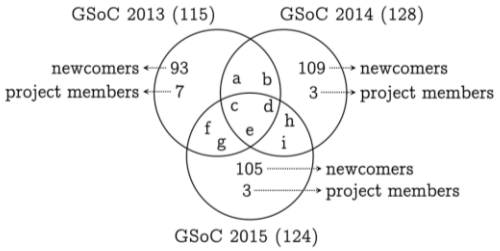
The research method followed in this study is depicted in Figure 1. The data collection phase involved many steps, since Google only publishes the names of the organizations and accepted candidates, making it hard to determine the specific project a given participant worked for. For example, Google informs that participant John Doe was accepted by Apache Software Foundation, but, generally, there is no information on which Apache project John has worked on. As the collection and verification of each student project is a laborious and time-consuming task, we limited our analysis to the GSoC 2013-2015 editions. We counted ~3,100 distinct accepted students for these editions.

We randomly sampled 866 students, which offers a confidence level of 95% and a margin of error of 5%. We manually searched for the students assigned GSoC projects in source code management systems (SCM) by using their names and the project description provided by GSoC. In most cases, the projects were hosted on GitHub. We determined that we found their assigned projects when we had clear evidence linking the projects in the SCM with the students' information and the organization (*e.g.*, when the projects' descriptions in the SCMs matched those of the GSoC's projects, or when we found web links in the students' blogs to the projects). We found the projects of 406 students (out of 866), all of which were hosted on GitHub.

The next step was to identify the students' IDs in the project logs. First, we used *MetricsGrimoire-CVSAnalY*[9] to extract information from Git repositories and store it in a local database. The database includes information not only about the project commits, but also about the contributors. Second, we searched for all the IDs that students' might have used. We used the students' names and emails (or combinations) to decide if the IDs belonged to the same student. Based on this, to identify the students we applied common disambiguation heuristics, such as the ones presented by Wiese *et al*. [26]. For instance, when the IDs where composed of the combination of the initials of the students' first name with their full last names, or when the IDs were composed of the students' names initials and these initials were used as the students' IDs on GitHub. This yielded a final working sample of 367 students (out of 406).

Additionally, for all students in our sample, we verified whether they participated in previous GSoC editions. Figure 2 illustrates students' contribution per GSoC edition. It is worth-mentioning that a student may have participated in more than one edition, but in our sample this student may appear in only one edition. Thus, for clarification, Figure 2 depicts how many students participated in two or three GSoC editions with letters (a-i) and we caption their meaning below. We summarize students' participation as follows: 32 (8.6%) participated in 2 editions, 15 (4.1%) participated in all 3 editions, 13 (3.6%) participated in one edition, but were already project members, and the remaining 307 (83.7%) are newcomers who participated in one of the three editions analyzed. For each edition, we include the total students in parenthesis. In addition, we found 16 students who participated in GSoC editions prior to 2013.

As the last step, for every student in our final working sample, we counted the number of participations as a student and as a mentor, using the list published by GSoC and considering the editions of GSoC 2005 (first edition) to 2015.



a = 5 students (from our 2013 sample) also participated in GSoC 2014
b = 5 students (from our 2014 sample) also participated in GSoC 2013
c = 5 students (from our 2013 sample) also participated in GSoC 2014 and 2015
d = 5 students (from our 2014 sample) also participated in GSoC 2013 and 2015
e = 5 students (from our 2015 sample) also participated in GSoC 2014 and 2015
f = 5 students (from our 2013 sample) also participated in GSoC 2015
g = 5 students (from our 2015 sample) also participated in GSoC 2013
h = 6 students (from our 2014 sample) also participated in GSoC 2015
i = 6 students (from our 2015 sample) also participated in GSoC 2014

Figure 2. Number of students by participation year

We used the student's name and the GSoC project name as a matching criterion. That is, when we had a match with a student name as both student and mentor, we analyzed both: whether the GSoC project of the mentor was related to the GSoC project of the student, and; whether the year of participation as a student was earlier than that as a mentor.

B. Data Analysis

To analyze the data, we split the students' contribution to GSoC into 3 periods: *before*, *during*, and *after* GSoC. We used the official timelines (*i.e.*, start and end dates) to classify the commits in each period.

[9] http://metricsgrimoire.github.io/CVSAnalY/

TABLE II. SAMPLE CHARACTERIZATION.

# of participations in GSoC	# of students who participated in GSoC as students	# of students who participated in GSoC as mentors	avg contrib interval (*days*) after GSoC (std dev)	avg contrib interval (*days*) before GSoC (std dev)	avg # of commits to the GSoC projects (std dev)	avg # of merged commits
1	307	9	52.0 (135)	36.3 (117)	97.0 (136)	64.3 (92)
2	48	3	77.5 (203)	108.6 (311)	216.9 (489)	115.5 (252)
3	8	0	74.4 (157)	37.2 (99)	115.5 (153)	89.1 (160)
4	3	0	3.5 (305)	0.0 (0)	155.0 (174)	20.0 (17)
5	0	0	0.0 (NA)	0.0 (NA)	0.0 (NA)	0.0 (NA)
6	1	0	476.0 (NA)	1,603.0 (NA)	477.0 (NA)	476.0 (NA)

Although students can contribute to an OSS community in different ways, such as opening issues, fixing bugs, or promoting events, we use the term students' **contribution** to refer to their commits (and consequently code churn) to the SCM. Thus, contribution before and after GSoC refer to the commits performed before GSoC kickoff and after GSoC ended, respectively. Students' **contribution interval** refers to the time in days that a student contributed (*i.e.*, committed). For instance, if a GSoC edition started on the 15th and a commit was performed on the 10th of the same month and year, then this contribution interval is 5 days before kickoff. Additionally, we use the concept of **distinct contribution days** (*i.e.*, distinct commit dates). For instance, if a student performed 3 commits on the 10th day, again 5 days before kickoff then the distinct contribution days' count before GSoC is 1 (*i.e.*, one distinct commit date before GSoC).

RQ1. How much code do CCE students contribute to codebase? To test whether a specific commit was merged to codebase, we compared each of the student's commits' Secure Hash Algorithm (SHA)—which uniquely identifies all commits—to the commits' SHAs belonging to codebase, and grouped them by participation period. The number of the students' commits in codebase was obtained by counting the number of commits in each group.

To determine how much code the students added to codebase, we used the *git-log* tool, which creates a log file for the projects repository, containing the commits' SHA, authors' name, and how many lines were added and removed for each file in a commit. Next, for every commit, we calculated the code churn and stored it in the database.

To test whether there are statistical differences in the number of commits among the participation periods, we used the Wilcoxon Signed-Rank Test, which can determine whether the corresponding data population distributions are statistically equivalent for non-normal distributions. The null hypothesis is that the commits' distribution in each period tuple—m(*before-during*), (*during-after*), (*before-after*)—are statistically equivalent. If the p-value is less than the 0.05 significance level, we reject the null hypothesis.

To quantify the strength of difference between two groups of observations beyond the p-values interpretation, we used the Cliff's Delta *d* statistic, a non-parametric effect size. For Cliff's Delta *d*, the effect size is considered negligible for $d < 0.147$, small for $0.147 \leq d < 0.33$, medium for $0.33 \leq d < 0.47$, and large for $d \geq 0.47$ [27].

RQ2. How long do students contribute before and after CCEs? To properly determine it, we distinguish newcomers from the students-with-experience in the assigned projects, such as former GSoC students and project members. To identify former GSoC students, we counted how many GSoC editions a student participated in. To do so, we used the GSoC's announcement date of the accepted mentoring organizations for each year as a *threshold*. Thus, if a developer started contributing after the announcement (*threshold*), we considered the developer a newcomer. Otherwise, we treated the developer as a project member. For GSoC 2013, the announcement of the mentoring organizations was made 70 days before the coding period started. For GSoC 2014 and 2015, the announcement was made 84 days before kickoff.

Therefore, **newcomers** are students who did not have commits older than the GSoC announcement date in relation to the start date of their first GSoC edition and are not former GSoC students. We refer to the students who did not meet these criteria as **students-with-experience**.

Lastly, we correlated the collected variables using Spearman's correlation to test their predictive strength across different participation periods. We used the following variables to generate the correlation matrix: contribution interval (before & after) GSoC; number of commits (all periods); number of merged commits (all periods); number of distinct contribution days (all periods); and code churn (all periods).

IV. RESULTS

This section reports our results. We start by characterizing our study sample.

A. Sample Characterization

Table II summarizes the characteristics of our sample in terms of: number of participations in the program as both students and mentors; participation before and after GSoC in the assigned project, and; the total and merged commits to the projects. Note that the rows regarding 3-6 participations may also include editions from GSoC 2010-2015.

It is worth mentioning that there are few students with 3+ participations, which can influence the analysis of participation (before/after) and commits (total/merged). Additionally, the students who participated in only one GSoC edition are not necessarily new to the project, and the ones with 2+ participations are not necessarily project members. We present our results by analyzing these cases.

In Figure 3, one can observe that, considering our sample, almost half of the students had code merged after the official GSoC timeline. In addition, many students (~19%) had code merged only *during* GSoC.

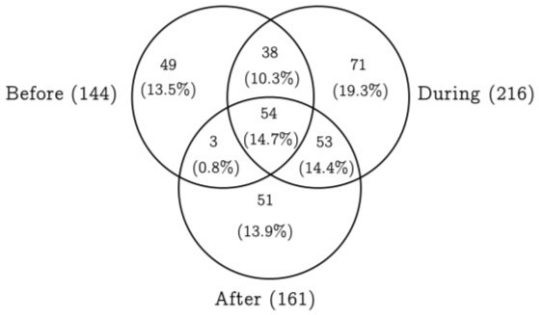


Figure 3. Number of students that had commits merged to codebase *before*, *during* and *after* GSoC. 48 (13.1%) students did not have anything merged.

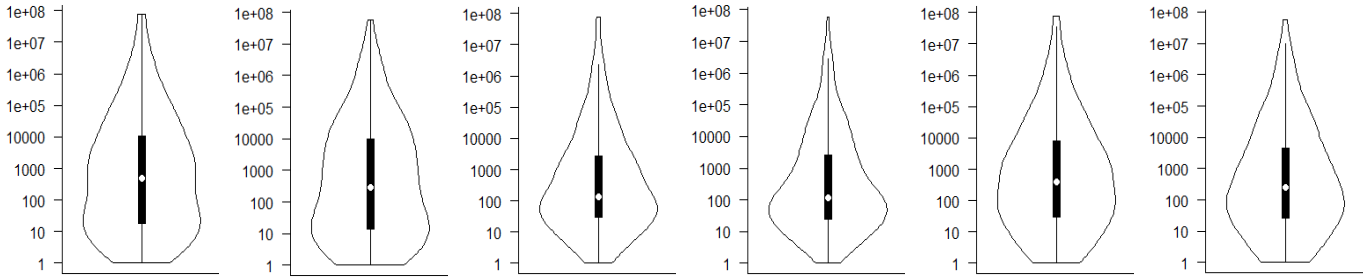
B. *RQ1a. How many commits/code churn in codebase are contributed by the students?*

To estimate how many of the students' commits were merged to the GSoC projects in each participation period, we present the violin plots in Figure 4. For better data visualization, we removed the students without any commits for that period from the plots. We report how many students were removed and the respective percentages in brackets after the figures' captions. Comparing Figure 4 (a) and (b), we can see that some of the students' commits were merged even before kickoff. These commits may have come from at least three distinct sources: students who were already project members; former GSoC students; and newcomers. A possible explanation for newcomers' commits is that **some candidates contribute to GSoC's projects to increase their odds of being accepted**. Indeed, we found mentors' blogs (*e.g.* [28]) with tips on how to be accepted.

We found support for this explanation in our data. Figure 5 depicts the number of distinct students who contributed to their GSoC project in the 180 days before kickoff. While the commits of students-with-experience (Figure 5b) to the project remained relatively constant until the start of the bonding period (~30 days before kickoff), some newcomers (Figure 5a) started committing ~80 days before kickoff. This means that newcomers started committing to the GSoC project before they knew they would be accepted, possibly attempting to show their skills to the community before selection.

Strictly speaking, most OSS projects of our sample benefited from participation in GSoC, since in ~87% of the cases they had at least one merged commit to codebase. When only newcomers are considered, ~54% of OSS projects merged at least one commit.

In Figure 4 (e), we can see that ~45% of the students did not commit anything after GSoC. The commits of the students who did, typically ranged between 31 (Q_1) and 8,064 (Q_3). In Figure 4 (f), we can observe that ~44% of the students had commits merged to codebase, typically ranging from 26 (Q_1) to 4,452 (Q_3). **Hence, code was merged to codebase in all periods**.



(a) Commits' distribution **before** GSoC [*189/367 (~51%) did not commit*]

(b) Merged commits' distribution **before** GSoC [*223/367 (~61%) did not have any commit merged*]

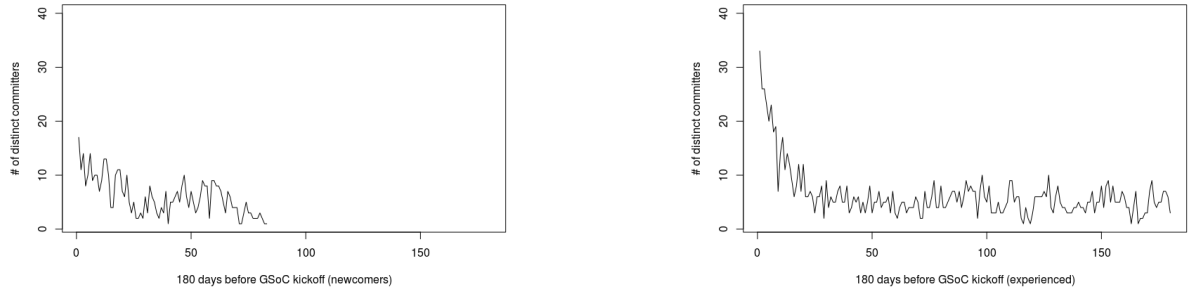
(c) Commits' distribution **during** GSoC [*99/367 (~27%) did not commit*]

(d) Merged commits' distribution **during** GSoC [*151/367 (~41%) did not have any commit merged*]

(e) Commits' distribution **after** GSoC [*166/367 (~45%) did not commit*]

(f) Merged commits' distribution **after** GSoC [*206/367 (~56%) did not have any commit merged*]

Figure 4. Commits and merged commits distribution by participation period (*before*, *during*, and *after*).



(a) Newcomers' contribution before GSoC (180 days)

(b) Students-with-experience contribution before GSoC (180 days)

Figure 5. Students' contribution 180 days before GSoC

Additionally, we used the Wilcoxon Signed-Rank Test to understand whether the amount of commits' distribution per participation period per year were statistically equivalent. In Table III, we can observe that for all years there is a large effect size when we compare the students' commits distribution performed *during* GSoC to the ones made *before* and *after* the program. However, when we compared the commits made *after* GSoC to the ones made *before* GSoC, we could only find statistical difference for GSoC 2014, still with a small effect size. For GSoC 2013 and GSoC 2015, we did not find any statistical difference.

TABLE III. EFFECT SIZE AND WILCOXON SIGNED RANK TEST COMPARING THE NUMBER OF COMMITS MADE BY GSOC STUDENTS' BY YEAR

	2013 *d*	**2014 *d***	**2015 *d***
During vs Before	0.73 (*large*) *	0.67 (*large*) *	0.66 (*large*) *
During vs After	0.62 (*large*) *	0.56 (*large*) *	0.58 (*large*) *
After vs Before	0.01 (*neglible*)	0.21 (*small*) *	0.08 (*neglible*)

* $p < 0.05$: significance level of the Wilcoxon Signed Rank Test. *d*: effect size computed with Cliff's Delta

However, when we measured the strength of this difference, we found it to be small, suggesting that, in the long run, the commits performed *after* tended to return to the levels *before* GSoC. This happens due to the commits of top contributors, which can be thousands of times higher than regular contributors. In addition, top contributors are mostly consisted of students-with-experience. We obtained similar results on the code churn statistical test.

Analyzing the students' commits provides one perspective on students' contributions; we analyzed code churn (how much code was merged) to offer an additional perspective. Figure 6 depicts the distribution of students' code churn per participation period. The churn *before* boxplot shows the distribution median as 1,482, with its top 25% ranging between 8,880 (Q_3) and 21,964 (upper whisker). For the *during* period, the distribution median was approximately six times higher (~8,900), with its top 25% ranging between 30,132 (Q_3) and 71,000 (upper whisker).

The churn *after* boxplot shows that the students' code churn significantly decreased after the program, with the distribution median decreasing to 2,435. The top 25% of the distribution remained high, ranging between 16,000 (Q_3) and 33,700 (upper whisker). We can understand the magnitude of the students' contribution when we add the code churns to the distributions. **In this way, we can see that the code churn *before* GSoC totaled 11.5M, *during*, 81.9M, and *after*, 19.1M.**

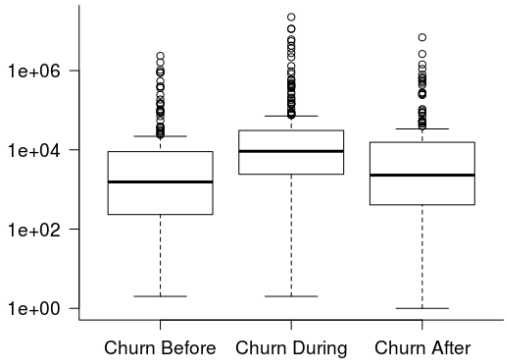


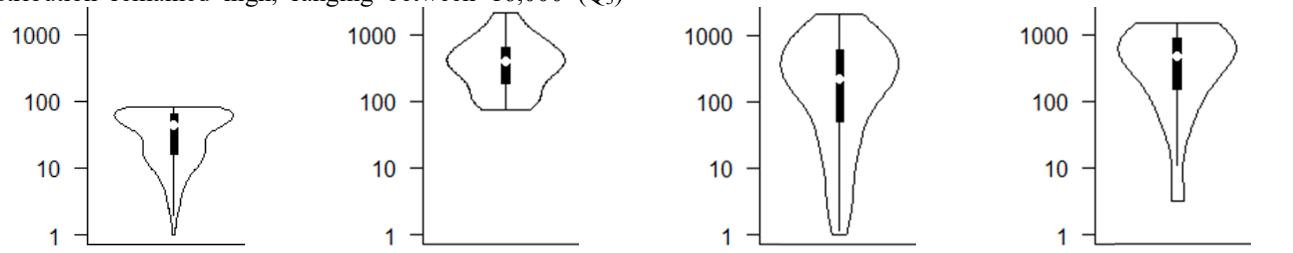
Figure 6. Students' code churn by participation period.

C. *RQ2a. What was the students' participation before and after GSoC?*

To understand how long the students contributed to GSoC projects, Figure 7 depicts the distribution of the students' contribution intervals before and after GSoC.

We split the students into newcomers and students-with-experience. Figure 7 (a) and (c) show newcomers' contribution intervals, in days, before and after GSoC, while Figure 7 (b) and (d) show the same information for the students-with-experience. As previously, for better data visualization we only show the students who kept contributing to the assigned projects. Thus, we report how many students were excluded and the respective percentage after the figures' captions, in brackets.

Figure 7 (a) complements a previous finding, by informing that many (~23%) of the newcomers contributed before knowing whether they would be accepted in. However, typically the students had not contributed to their GSoC projects before the program, which suggests that GSoC is indeed attracting potential contributors.



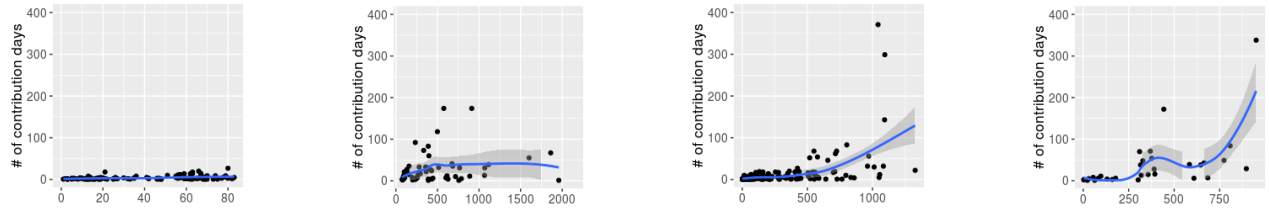
(a) Newcomers' contribution interval (*in days*) before GSoC [*189/307 (~62%) did not contribute before*]

(b) Students-with-experience contribution interval (*in days*) before GSoC [*0/60 (0%) did not contribute before*]

(c) Newcomers' contribution interval (*in days*) after GSoC [*137/307 (~45%) did not remain*]

(d) Students-with-experience contribution interval (*in days*) after GSoC [*29/60 (~48%) did not remain*]

Figure 7. Contribution interval before and after (*retention*) distribution for newcomers and students-with-experience.



(a) Distinct contribution days' count vs range between first commit to the project and GSoC kickoff for newcomers (*307 students*)

(b) Distinct contribution days' count vs range between first commit to the project and GSoC kickoff for students with-experience (*60 students*)

(c) Distinct contribution days' count vs range between GSoC end date and last commit to the project for newcomers (*307 students*)

(d) Distinct contribution days' count vs range between GSoC end date and last commit to project for students-with-experience (*60 students*)

Figure 8. Distinct contribution days' count (*# of days*) before and after GSoC vs the interval (*in days*) between: first commit and GSoC start date for newcomers (a) and students-with-experience (b); and GSoC end date and last commit for newcomers (c) and students-with-experience (d)

In Figure 7 (b), we can see that many students-with-experience have long previous contribution intervals in their assigned projects (Q_1=187.2; Q_3=639). By further analyzing these cases, we found that they mostly consisted of GSoC former students (47). In Figure 7 (c), we can see that the newcomers did not typically keep committing to their GSoC projects (~45%). So, some OSS projects benefited from the newcomers' contributions even after the official program end.

In Figure 7 (d), as with newcomers, we can see that the students-with-experience did not keep committing to the repository. The long contribution interval of those who did refers mostly to participation in subsequent GSoC editions, which we consider a different, but valid, type of retention. Thus, the contribution of the students-with-experience to GSoC projects typically ranged from 114 (Q_1) to 596.5 (Q_3) days after the program. In addition, we found 13 students-with-experience (21.6%) who continued contributing regularly, which indicates that some participants remained tied to their GSoC project and participated in more than one edition.

The analysis of the students' contribution intervals before and after GSoC only shows one facet of contributions out of GSoC's timeframe, as it does not inform anything about the contributions' frequency. Figure 8 presents a relationship between contribution intervals (CI) and the number of distinct contribution days (CD) in scatter plots.

In Figure 8 (a), we can observe that although many newcomers started contributing after Google's announcement of accepted mentoring organizations, CDs are mostly less than 10. Only 14 (6%) of newcomers contributed more than 10 CDs.

In Figure 8 (b), we can observe that the students-with-experience's CIs are considerably higher than the newcomers' (who are limited to a 84-day limit of previous CIs by our definition). However, we can see that most students have less than 50 CDs, with the distribution median being 16 CDs before GSoC.

In Figure 8 (c), we can see that, after GSoC, newcomers' CI increased considerably, reaching in many cases to CIs higher than 500 days. However, with the exception of a few cases, CDs did not increase proportionally. For instance, we observed a median of 5 CDs for the newcomers who contributed longer than a month, 9 CDs for the ones who contributed longer than six months, and 14 CDs for the ones who contributed longer than a year.

In Figure 8 (d), we can see that, after GSoC, some of the students-with-experience had CDs comparable to the newcomers who contributed before knowing they would be accepted. In addition, we observed a median of 22 CDs for the students-with-experience who contributed more than a month, 38.5 for the ones who contributed longer than six months, and 41 for the ones who contributed longer than a year.

RQ2b. Is previous contribution associated with students' retention?

Many works in literature have correlated developers' contribution to OSS projects with numerous variables, trying to predict early-on the ones who will continue contributing to the OSS community (*e.g.*, [12], [20], [29]). We correlated the data we collected on the students to study the variables' predictive strength, especially in different participating periods, as shown in Table IV. The variables are presented in the main diagonal of the correlation matrix, preceded and followed by a letter A-N. The entries in the upper triangular refer to the newcomers' correlations, while the entries in the lower triangular refer to the students-with-experience correlations. Participation periods are highlighted in boxes in the lower and upper triangular of the matrix.

In Table IV, for newcomers, we can see that the correlations did not show any predictive strength, as variables are mostly weakly correlated (<0.5). For students-with-experience, we can see that: the *number of commits before* (B) and *code churn before* (E) are strongly correlated (>0.7) with how much code is written *during* (variables F and H) and *after* (variables K, M, and N) GSoC. In addition, the variables B and E are strongly correlated with how long students stay after the end of GSoC (variables G, J, and L). Similarly, the amount of code students wrote during the program showed to be good predictors regarding how much code students wrote and how long they stayed after GSoC. Thus, our findings complements the results of Schilling and colleagues [12] that prior development experience in the project are associated with higher levels of retention.

TABLE IV. CORRELATIONS ON NEWCOMERS AND STUDENTS-WITH-EXPERIENCE CONTRIBUTION.

Newcomers' spearman correlations																
					Before GSoC				During GSoC				After GSoC			
					B	C	D	E	F	G	H	I	J	K	L	M	N
A	**contrib interval before**			**A**	.926	.796	.955	.860	.293	.152	.203	.166	.113	.058	.087	.050	.023
.112	**B**	**# of commits before**			**B**	.852	.967	.922	.473	.334	.364	.335	.022	.221	.058	.191	.122
.004	.469	**C**	**merged commits before**			**C**	.817	.806	.360	.277	.301	.414	.009	.165	.022	.161	.159
.315	.679	.325	**D**	**# of contrib days before**			**D**	.911	.389	.246	.291	.243	.057	.135	.021	.117	.040
.120	.923	.499	.677	**E**	**code churn before**			**E**	.459	.318	.389	.319	.025	.216	.067	.204	.119
.076	**.940**	.381	.481	**.883**	**F**	**# of commits during**			**F**	.905	.883	.799	.189	.561	.321	.509	.373
.026	**.825**	.341	.409	**.753**	.897	**G**	**# contrib days during**			**G**	.830	.787	.053	.403	.160	.352	.268
.075	**.911**	.376	.481	**.886**	.974	.873	**H**	**code churn during**			**H**	.753	.121	.465	.247	.476	.295
.137	.598	**.714**	.313	.622	.653	.660	.660	**I**	**merged commits dur**			**I**	.139	.427	.240	.376	.457
.019	**.773**	.342	.320	**.726**	**.818**	.641	**.787**	.535	**J**	**contrib interval after**			**J**	.829	.920	.774	.735
.025	**.877**	.359	.441	**.837**	**.913**	**.750**	**.881**	.538	.920	**K**	**# of commits after**			**K**	.908	.914	.797
.027	**.815**	.299	.371	**.766**	**.853**	.673	**.806**	.497	.957	.943	**L**	**contrib days after**			**L**	.835	.778
.055	**.884**	.384	.473	**.857**	**.912**	**.729**	**.915**	.555	.894	.959	.890	**M**	**code churn after**			**M**	.750
.102	**.702**	.537	.369	**.712**	**.733**	.608	**.743**	.710	.836	.802	.811	.797	**N**	**merged commits # after**			
Before GSoC					**During GSoC**				**After GSoC**								
Students-with-experience's spearman correlations																	

V. DISCUSSION

One question that may arise for some OSS communities is how much return on their mentoring investment they can expect in terms of code contribution and new volunteers to OSS projects. Indeed, some communities aim to retain students as new contributors, as evidenced by the following excerpt:

> "*(...) Participating [in] GSoC will increase the visibility of Pharo project efforts (...) We expect also to bring more people into our community [by participating in GSoC]*"
> Pharo. Source: http://bit.ly/2mtN0Xr

However, especially for mentors, participation in CCEs involves a trade-off between the effort invested in mentoring students and the mentors' ability to simultaneously address the OSS project tasks, which made the Debian community decide not to participate in GSoC17:

> *"Due to the lower amount of general motivation, and most notably the weakness of our projects page during the Google review (...) Debian will not be part of [GSoC] this year. Some of our recurring mentors have shown some signs of 'GSoC fatigue', (...) let's have a summer to ourselves to recover (...) and come back next year"* Debian. Source: http://bit.ly/2nT0h99

CCEs should benefit OSS communities and students. Students should acquire experience, branding, and learning by joining the OSS communities' workforce, while the OSS communities providing mentoring should achieve project tasks accomplished during, and possibly after, the engagements.

Even though we leave the task of investigating students' actual learning or yet the nature of project tasks to future research, our results suggest that some OSS communities achieve project tasks accomplishment, especially the communities that selected students-with-experience. This is understandable, since it is usually hard on newcomers to go from the learning to contribute to the development of meaningful contributions.

> "*GSoC is an important program, because it provides a possibility to mentor students intensively over a relative long period of time. The Student gets more experience, while the project [gets] tasks done, that [otherwise] would be harder to do [by] pure volunteers.*" LibreOffice. Source: http://bit.ly/2n1xt1u

One possible implication of our results is that when OSS communities select newcomers for participating in GSoC, instead of students-with-experience, they need to be prepared to invest in the newcomers' mentoring, without expectations of long-term commitment, as it can be seen in Figure 7.

Not surprisingly, the period with higher contribution was *during* GSoC (*sponsored period*), as depicted, for example, in Figure 6. Our results showed that ~64% of the students did not stay later than a month after the program. Based on this finding, **we suggest that communities would come up with a strategy for handling the disappearing students**, which could be as simple as maintaining contact through email. Future research could investigate alternative ways to prevent students from abandoning OSS projects.

Our results also suggest that CCEs provide OSS communities with applicants' contribution before kickoff, possibly due to the competitive nature of the engagements. **OSS communities would offer a pre-CCE program to engage applicants.** The community would take advantage of applicants' contribution before the program, offering a formal opportunity for applicants to show their skills and interact with the community. As a result, the project would receive more contributions, and have the opportunity to showcase the community. This organization scheme could potentially mitigate the mentors' selection and ranking load, as they would have more data on the applicants. This strategy could also work for the BioPython OSS community, which experienced similar problems, as reported by Trainer and colleagues [10].

OSS communities and CCEs organization would also offer opportunities for those who were not selected to receive stipends to participate voluntarily. In this case, the participants would be awarded with participation certificates. Thus, even non-sponsored participants would have the chance to acquired knowledge, experience, and branding.

There is another interesting facet related to retention that we would like to highlight. Our results suggest that finding top contributors, though rare, could yield large dividends for the community, considering the number of (merged) com-

mits. The aforementioned findings—higher visibility, contributing as a strategy to increase acceptance odds, merged code during the program, and finding top contributors—may explain why the number of OSS communities interested in entering GSoC has increased throughout the years.

CCEs seem to be a channel of contribution to OSS projects that not only have mitigated barriers for the students who wanted to become volunteer contributors (see Steinmacher *et al*. [16] for an overview of the barriers that newcomers usually face), but also have taken advantage of who would never contributed otherwise.

VI. THREATS TO VALIDITY

This research has several limitations, which we here both acknowledge and report how we aimed to mitigate. First, our sample may not be representative of the whole GSoC students' population, despite our efforts to collect a representative random sample. This means that it may be possible to reach to other conclusions with a different set of students.

A major threat is the misidentification of the students and their projects in the SCMs, and the students' IDs in the local database. For instance, in some cases the student IDs, both in the SCMs and in the local database, were actually composed of the students' name initials (or combinations). Although the students' IDs were double checked by two different researchers and we excluded the cases that we were uncertain about, it is still possible that we incurred some misidentification.

In some cases, the same student used multiple IDs to perform the commits. In this case, the threats are that we could have: incorrectly grouped IDs from different students; not identified all the IDs used by a student; and/or identified the IDs used in a different GSoC edition than the one under consideration. Even though we closely inspected every student in our sample, it is still possible that these threats weakened our results.

In addition, we used the students' and projects' names as matching criteria to determine whether the students participated as mentors in other editions. In the case of students who share common names working for the same project, we might have wrongly counted them as the same student. We mitigated this threat by closely inspecting if the year of participation as student was before the participation as mentor for the same project. As we did not personally contact any mentor, it may be the case that students delivered their final code after the official GSoC's end date, which by our method would be wrongly counted as retention.

Finally, our conclusions may be biased toward the number of merged commits. We do not control potentially important variables, such as programming language, code complexity, or how important the merged commits were to the communities. It may be the case that the students who had only one merged commit contributed more—in terms of aggregated value—than those with more merged commits.

VII. CONCLUSION

There is evidence that OSS communities expect that community code engagements (CCE), such as Summers of Code, may be an effective channel not only for the attraction and retention of newcomers, but also for the code contributions made during participation [5], [9]–[11]. In this paper, we investigated the Google Summer of Code (GSoC), providing empirical evidence on different aspects of the students' contribution, such as how much GSoC fostered contributions (*i.e.*, commits, merged commits, and code churn) and how long did students contribute to the assigned projects before and after the engagements.

For analyzing RQ1 (How much code do CCE students contribute to codebase?), for each period, we counted how many of the students' commits were merged in codebase. We estimated how much code the students added by calculating the code churn (*i.e.*, lines added + lines removed) for each commit. We found that code merges occurred before, during, and after GSoC, including for newcomers. Most merged commits occurred *during* GSoC, although many OSS projects merged in other periods. We also could obtain the magnitude of students' code contributions by analyzing code churn' medians: ~11.5M (*before*); ~82M (*during*); and ~19M (*after*).

For analyzing RQ2 (How long do students contribute before and after CCEs?), we start by differentiating newcomers from students-with-experience. Then, we investigated contribution intervals, contributions, and the distinct contribution days' count (*i.e.*, distinct commit dates). We found that ~23% of newcomers contributed to GSoC project before knowing they would be accepted. After GSoC, contribution decreased from ~43% newcomers who kept contributing longer than a month to ~16% of them who kept contributing longer than a year. Students-with-experience started contributing more than a year earlier than kickoff, while a year later ~23% of them were still contributing. Regardless of experience time in the GSoC project, the number of distinct contribution days was not proportional to longer contribution intervals.

We conclude this work highlighting that OSS communities that need achieve projects tasks accomplishment should consider prioritizing students-with-experience, as they are already familiar with the projects' contribution norms and they possibly have a lower learning curve. For the students who kept contributing after GSoC, contributions tended to slowly diminish, which can signal to OSS communities that they should use their strategy for handling these students. This can be as simple as sending email explaining the importance of the students' contribution. In addition, OSS communities can establish a recommended period before CCEs for applicants start contributing and interacting with the community. Thus, applicants who start earlier and contribute more would have more acceptance chances.

ACKNOWLEDGEMENTS

The authors thank CNPq (process 430642/2016-4), NAU and FAPESP (process 2015/07399-1) for their financial support.

REFERENCES

[1] P. Resnick and R. E. Kraut, *Building Successful Online Communities: Evidence-Based Social Design*, no. May. The MIT Press, 2009.

[2] Y. Fang and D. Neufeld, "Understanding Sustained Participation in

Open Source Software Projects," *J. Manag. Inf. Syst.*, vol. 25, no. 4, pp. 9–50, 2009.

[3] G. Pinto, I. Steinmacher, and M. A. Gerosa, "More Common Than You Think: An In-depth Study of Casual Contributors," *2016 IEEE 23rd Int. Conf. Softw. Anal. Evol. Reengineering*, vol. 1, no. 1, pp. 112–123, 2016.

[4] K. Crowston, H. Annabi, and J. Howison, "Defining Open Source Software Project Success," in *Proceedings of the 24th International Conference on Information Systems (ICIS)*, 2003, pp. 1–14.

[5] E. H. Trainer, C. Chaihirunkarn, A. Kalyanasundaram, and J. D. Herbsleb, "Community code engagements: Summer of Code & hackathons for community building in scientific software," in *Proceedings of the International ACM SIGGROUP Conference on Supporting Group Work*, 2014, pp. 111–121.

[6] F. J. García-Peñalvo *et al.*, "Developing Win-win Solutions for Virtual Placements in Informatics: The VALS Case," in *Proceedings of the Second International Conference on Technological Ecosystems for Enhancing Multiculturality*, 2014, pp. 733–738.

[7] I.-H. Hann, J. Roberts, S. Slaughter, and R. Fielding, "Economic Incentives for Participating Open Source Software Projects," *Twenty-Third Int. Conf. Inf. Syst.*, vol. ICIS 2002, 2002.

[8] J. Tirole and J. Lerner, "Some Simple Economics of Open Source," *J. Ind. Econ.*, vol. 50, no. 2, pp. 197–234, 2002.

[9] L. Christopherson, R. Idaszak, and S. Ahalt, "Developing Scientific Software through the Open Community Engagement Process," in *First Workshop on Sustainable Software Science: Practice and Experiences*, 2013.

[10] E. H. Trainer, C. Chaihirunkarn, and J. D. Herbsleb, "The Big Effects of Short-term Efforts: Mentorship and Code Integration in Open Source Scientific Software," *J. Open Res. Softw.*, vol. 2, no. 1, p. Art. e18, 2014.

[11] E. H. Trainer, A. Kalyanasundaram, C. Chaihirunkarn, and J. D. Herbsleb, "How to Hackathon: Socio-technical Tradeoffs in Brief, Intensive Collocation," in *Proceedings of the 19th ACM Conference on Computer-Supported Cooperative Work & Social Computing - CSCW '16*, 2016, pp. 1116–1128.

[12] A. Schilling, S. Laumer, and T. Weitzel, "Who will remain? - An evaluation of actual Person-Job and Person-Team fit to predict developer retention in FLOSS projects," *Proc. Annu. Hawaii Int. Conf. Syst. Sci.*, pp. 3446–3455, 2011.

[13] K. R. Lakhani and R. G. Wolf, "Why Hackers Do What They Do: Understanding Motivation and Effort in Free/Open Source Software Projects," in *Perspectives on Free and Open Source Software*, Cambridge: MIT Press, 2005.

[14] A. Hars and S. Shaosong Ou, "Working for free? Motivations of participating in open source projects," in *Proceedings of the 34th Annual Hawaii International Conference on System Sciences*, 2001, vol. 7, p. 9.

[15] J. a. Roberts, I.-H. Hann, and S. a. Slaughter, "Understanding the Motivations, Participation, and Performance of Open Source Software Developers: A Longitudinal Study of the Apache Projects," *Manage. Sci.*, vol. 52, no. 7, pp. 984–999, 2006.

[16] I. Steinmacher, M. A. Gerosa, D. F. Redmiles, T. Conte, M. A. Gerosa, and D. F. Redmiles, "Social Barriers Faced by Newcomers Placing Their First Contribution in Open Source Software Projects," *Proc. 18th ACM Conf. Comput. Support. Coop. Work Soc. Comput. - CSCW '15*, pp. 1379–1392, 2015.

[17] I. Steinmacher, I. S. Wiese, T. Conte, M. A. Gerosa, and D. Redmiles, "The hard life of open source software project newcomers," *Proc. 7th Int. Work. Coop. Hum. Asp. Softw. Eng. - CHASE 2014*, pp. 72–78, 2014.

[18] F. Fagerholm, A. S. Guinea, J. Münch, and J. Borenstein, "The role of mentoring and project characteristics for onboarding in open source software projects," *ESEM conf.*, pp. 1–10, 2014.

[19] P. Meirelles, C. Santos, J. Miranda, F. Kon, A. Terceiro, and C. Chavez, "A Study of the Relationships between Source Code Metrics and Attractiveness in Free Software Projects."

[20] J. Colazo and Y. Fang, "Impact of license choice on open source software development activity," *J. Am. Soc. Inf. Sci. Technol.*, 2009.

[21] C. Santos, G. Kuk, F. Kon, and J. Pearson, "The attraction of contributors in free and open source software projects," *J. Strateg. Inf. Syst.*, vol. 22, no. 1, pp. 26–45, 2013.

[22] K. J. Stewart and S. Gosain, "The impact of ideology on effectiveness in open source software development teams," *MIS Q.*, vol. 30, no. 2, pp. 291–314, 2006.

[23] S. Krishnamurthy, S. Ou, and A. K. Tripathi, "Acceptance of monetary rewards in open source software development," *Res. Policy*, vol. 43, no. 4, pp. 632–644, 2014.

[24] M. Zhou and A. Mockus, "What make long term contributors: Willingness and opportunity in OSS community," in *34th International Conference on Software Engineering*, 2012, pp. 518–528.

[25] J. Lave and E. Wenger, *Situated learning: Legitimate Peripheral Participation*. Cambridge University Press, 1991.

[26] I. S. Wiese, J. T. da Silva, I. Steinmacher, C. Treude, and M. A. Gerosa, "Who is Who in the Mailing List? Comparing Six Disambiguation Heuristics to Identify Multiple Addresses of a Participant," *2016 IEEE Int. Conf. Softw. Maint. Evol.*, pp. 345–355, 2016.

[27] R. J. Grissom and J. J. Kim, *Effect Sizes for Research: A Broad Practical Approach*. Lawrence Erlbaum Associates, 2005.

[28] P. Daniel, "Want to be selected for Google Summer of Code 2016?," 2015. [Online]. Available: http://danielpocock.com/getting-selected-for-google-summer-of-code-2016. [Accessed: 10-Feb-2017].

[29] S. Dejean and N. Jullien, "Big from the beginning: Assessing online contributors' behavior by their first contribution," *Res. Policy*, vol. 44, no. 6, pp. 1226–1239, 2015.