Turnover in Open-Source Projects: The Case of Core Developers

Fabio Ferreira
Center of Informatics
Federal Institute of the Southeast of Minas Gerais (IF Sudeste MG)
Barbacena, Brazil
fabio.ferreira@ifsudestemg.edu.br

Luciana Lourdes Silva
Department of Computing
Federal Institute of Minas Gerais (IFMG)
Ouro Branco, Brazil
luciana.lourdes.silva@ifmg.edu.br

Marco Tulio Valente
ASERG Group - Department of Computer Science
Federal University of Minas Gerais (UFMG)
Belo Horizonte, Brazil
mtov@dcc.ufmg.br

ABSTRACT

In order to know whether FLOSS projects are performing well, researchers have studied the ability of these projects to both attract and retain contributors, and consequently, their turnover, that is, the exit of some contributors and the entry of others. However, most contributors accounted for turnover in FLOSS projects are not active and make one or two contributions per year, which can lead to inaccurate results. In this paper, we compute the turnover of 174 FLOSS projects for each year between 2015 and 2018, considering only core developers. We analyze how the *Core Developers' Newcomers and Leavers* rates can influence the actionability of these projects, considering the time to fix issues and bugs, and the time to implement enhancements. We found out that 104 (59.7%) out of 174 projects have at least 30% turnover per year, 46 (26.4%) projects exceed 50%, and only 10 (5.7%) projects have less than 10% of annual turnover on average. We also found that projects owned by organizations have a higher *Core Developer Turnover (CDT)* rate than projects owned by individual users. The results show that the turnover of core developers increases with the size of teams, and it is notably higher in Ruby projects. Finally, we use the Core Developer Newcomers (CDN_{Rate}) and Leavers (CDL_{Rate}) rates to classify FLOSS projects as Attractive, Unattractive, Stable, Unstable. The results show that projects classified as *Unstable* (High Number of Core Developers Leavers and Newcomers) take a longer time to fix issues and bugs and to implement enhancements than other groups.

CCS CONCEPTS

• **Software and its engineering**; • **Software organization and properties**; • **Collaboration in software development**;

KEYWORDS

Core Developers, Turnover, GitHub, Free/Libre and Open Source Software, FLOSS

ACM Reference Format:
Fabio Ferreira, Luciana Lourdes Silva, and Marco Tulio Valente. 2020. Turnover in Open-Source Projects: The Case of Core Developers. In *34th Brazilian Symposium on Software Engineering (SBES '20), October 19–23, 2020, Brazil.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Nowadays, Free/Libre and Open Source Software (FLOSS) play an essential role for users and companies. As a substantial number of major software systems adopt FLOSS, there is a continuous concern about these projects' health [1]. Typically, during the evolution of a software system, team members leave or change their role in the project, and new contributors join the team [2]. Mockus [3] analyzed developer turnover—which is based on the number of developers who leave a project and are replaced by new ones—on a large software project. He found that the arrival of a new contributor to a project has no impact on it. In contrast, leaving members harms the project performance and quality due to the loss of knowledge and experience.

In the FLOSS projects context, researchers also have studied how projects attraction and retention of contributors directly impact developer turnover. For example, Yamashita *et al.* [4] apply two metrics traditionally used in migratory studies in the United States (*Magnet* and *Sticky*) to evaluate projects' ability to retain existing contributors as well as to attract new ones. They define *Magnet* projects as those that attract a large number of new contributors, while *Sticky* projects are those where a significant proportion of contributors keep making contributions over time (i.e., they are not a one-time-contributor [5]). Similarly, Foucault *et al.* [2] investigate whether turnover is frequent in FLOSS projects and how it impacts the overall quality of those projects. The authors correlate leavers (developers who left the projects) and newcomers (developers who are new contributors) with the density of fixed bugs to determine the effects of turnover on software quality. Their results reveal that the attraction of new contributors to a project is a positive event, but newcomers' activities negatively impact software quality.

However, these studies do not discard occasional contributors, i.e., developers who contribute with very few commits. According to Coelho *et al.* [6], most developers in FLOSS projects are not very active, and they usually contribute with one or two commits per year. Further, in some FLOSS projects, many developers have exactly one code contribution, called for Lee *et al.* [5] as One-Time code Contributor (OTC). Therefore, these occasional contributors may influence the findings of previous research on FLOSS turnover. For instance, studies that investigate the contributors turnover usually use only two equal periods ($T1 = T2$) to analyze the arrival and the departure of contributors in a project. They also define a newcomer developer on a FLOSS project as a contributor who had at least one accepted contribution in $T2$ and did not contribute in $T1$. On the other hand, a developer leaver is a contributor who had at least one accepted contribution in $T1$ and did not contribute in $T2$. Thus, these occasional contributors in a short period may interfere on previous study findings, and consequently, the result may not represent the reality of the project.

In this paper, to tackle these threats, we investigate the turnover of 174 FLOSS projects between 2015 and 2018. In addition, we only consider *core developers* [7, 8] (contributors who are responsible for most important tasks in a software project such as design, implementation, and maintenance). Specifically, we are interested in answer the following research questions:

(RQ1) *How common is core developer turnover in FLOSS projects?*

> Our results reveal that 104 (59.7%) out of 174 projects have at least 30% turnover per year, 46 (26.4%) projects exceed 50%, and only 10 (5.7%) projects have less than 10% of annual turnover rates on average.

(RQ2) *How do team size, programming language, and project owner impact the turnover of core developers?*

> We found that projects owned by organizations have higher *Core Developer Turnover (CDT) rate* than projects owned by individual GitHub users. Our results also show that the core developers' turnover increases along with the size of teams (it is notably higher in Ruby projects). The results do not show a statistically significant difference acceptance rate regarding older projects.

(RQ3) *How stable are FLOSS projects regarding their rates of core developers newcomers and leavers?*

To answer this RQ, we classified the projects into four *turnover status groups*: Attractive, Unattractive, Stable, and Unstable groups.

> We detected 21 projects (12.0%) that follow the Unattractive project behavior (high leavers rate and low newcomers rate) and 69 projects (39.6%) that follow the Unstable project behavior (high newcomers and leavers rates). In contrast, 18 projects (10.3%) show Attractive project behavior (low rate of leavers and high rate of newcomers), i.e., projects which tend to increase their core team size. Moreover, 66 projects (37.9%) follow Stable behavior (low newcomers and leavers rates).

(RQ4) *Does the core developers' turnover impact the issue resolution time?*

To answer this RQ, we analyze how each *turnover status group* (i.e., Attractive, Unattractive, Stable, and Unstable groups) affects the time to fix issues and bugs and to implement enhancements.

> We found that projects classified as Unstable take a longer time to fix issues compared to other groups, nearly 66 and 48 days, respectively. Moreover, the time required to fix bugs in Unstable projects (127.29 days) is nearly twice as large as the period spent by Attractive projects (66.85 days). Finally, Unstable projects require more time to implement enhancements than other projects, nearly 212 and 169 days, respectively.

The remainder of this paper is organized as follows. Section 2 presents the heuristic we used to identify core developers, how we computed core developer turnover, and the project selection process to build our dataset. Our results are present in Section 3 and the implications of these results are discussed in Section 4. In Sections 5 and 6, we describe related work and threats to validity, respectively. Finally, Section 7 concludes our work and presents ideas for future work.

2 STUDY DESIGN

2.1 Core Developers Computation

Large OSS projects have hundreds of contributors, but they usually rely on a few ones to survive, which are called *core developers* [9, 10]. Core developers are responsible for the most important tasks in a project, such as design, implementation, and maintenance. Essentially, they manage and guide software evolution. In this paper, we use a Commit-Based Heuristic [7], which is commonly used to detect core developers [6, 11]. Particularly, we adopt a variation [6] of this heuristic to define the core team of a project. Initially, both heuristics [6, 7] compute the number of commits for each developer and then select the set of contributors who produce 80% of the total number of commits. Further, the customized heuristic [6, 7] discards, from the contributors set identified in the previous step, contributors who have less than 5% of the overall amount of commits.

2.2 Core Developers Turnover

We define core developer turnover as the rate in which core contributors leave a project in a certain period and the average number of contributors in that period. Figure 1 illustrates the steps we followed to compute the turnover of core developers on each studied project. For each year, we apply the heuristic on commits to compute the core team, e.g., Set B represents the group of core contributors of the *n*-th year. Finally, we calculate *Core Developer Newcomers and Leavers rates.*

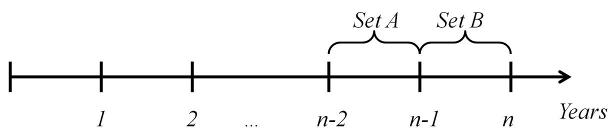
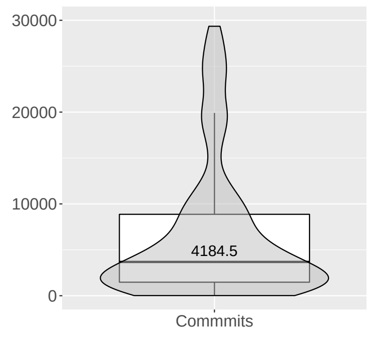
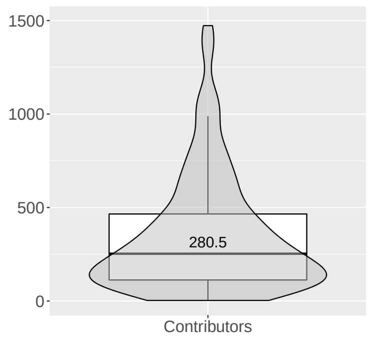


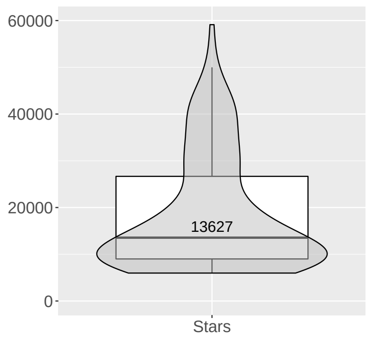
Figure 1: Set A = core developers computed regarding the commits between *(n-2)-th* and *(n-1)-th* years; Similarly, Set B = core developers computed regarding the commits between *(n-1)-th* and *n-th* years



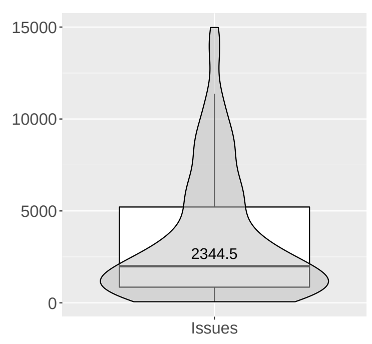
(a) Number of commits



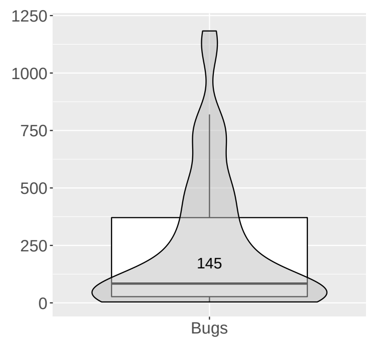
(b) Number of contributors



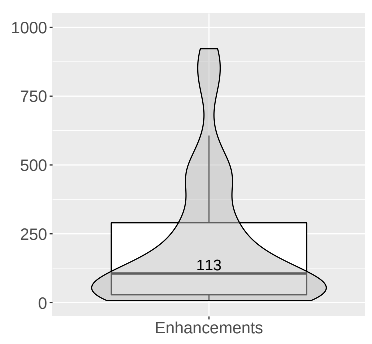
(c) Number of stars



(d) Number of Issues



(e) Number of Bugs



(f) Number of Enhancements

Figure 2: An overview of our dataset

Core Developers Newcomers Rate: Conceptually, new core developers are contributors who are not core members in a previous year (e.g., they are not in Set A) but who became core developers in the next year (e.g., they are in Set B), as seen in Figure 1. Then, the *Core Developer Newcomers (CDN rate)* for the *n*-th year is computed as follows:

$$CDN_{Rate}(n) = \frac{|CDN(n)|}{|Set\ B|} \times 100$$

where

$$CDN(n) = Set\ B - Set\ A$$

Core Developers Leavers Rate: Conceptually, core developers leavers are contributors who were core member in a previous year (e.g., they are in Set A) but they did not contribute enough to remain as core developer in the next year (e.g., they are not in Set B). Then, the *Core Developer Leavers (CDL rate)* for the *n*-th year is computed as follows:

$$CDL_{Rate}(n) = \frac{|CDL(n)|}{|Set\ A|} \times 100$$

where

$$CDL(n) = Set\ A - Set\ B$$

Finally, we rely on a well-known Human Resource Metric to compute the Core Developer Turnover (CDT) rate [12, 13]. To compute the monthly or annual turnover rate, the number of employees who left the company is divided by the average number of active employees in the evaluated period of time and multiplied by 100. The average number of employees is calculate by adding the number of active employees at the beginning of the period and the number of employees at the end of the period, and dividing the result by two. Similarly, to calculate the Core Developer Turnover (CDT) rate for the *n*-th year, we take the number of core developers in Set A and the number of core developers in Set B to compute the average number of core developers. Then, we divide the number of core developers in *CDL(n)* by the average number of active core developers as follows:

$$CDT_{Rate}(n) = \frac{|CDL(n)|}{avg(|Set\ A| + |Set\ B|)} \times 100$$

The Core Developers Newcomers help minimize the Core Developer Turnover rate since they are in Set B.

2.3 Dataset

First, we select five common programming languages on GitHub (Java, JavaScript, PHP, Python, and Ruby). Then, we select the top-50 most popular projects on GitHub for each language (total of 250 projects), ranked by stars. Finally, we applied some steps to discard projects from this initial selection:

(1) *Non-software projects:* First, we remove non-software projects, such as books (e.g., BITCOINBOOK/BITCOINBOOK), tutorials (e.g., WINTERBE/JAVA8-TUTORIAL), and awesome-lists (e.g., VINTA/AWESOME-PYTHON). We also discard two projects having more than 50% of non-code commits (i.e., text and comments commits).
(2) *New Projects*: Our goal is to analyze core developer turnovers from 2015 to 2018. Therefore, since we need the set of core developers of 2014 to calculate the Core Developers Newcomers rate of 2015, we discard 60 projects that started after 2014.
(3) *Sparse Commits*: To compute core developers' annual turnover, we rely on developers' activities. For this reason, we discard eight projects without commits in the last six months (non-active projects) and two projects with less than 100 commits in total.
(4) *Small Team*: We remove projects with less than ten contributors since our goal is to investigate large OSS projects. Therefore, we discard three projects from our dataset.

Our dataset ended up with 174 open source projects, including popular projects, such as ELASTICSEARCH, SELENIUM, NODE, REACT, ANGULAR, RAILS, JQUERY, DJANGO, and projects from popular companies, such as Google, Facebook, Netflix, and Alibaba. Figure 2 shows the distribution of the number of commits, contributors, stars, issues, bugs, and enhancements for the selected projects. The first quartiles are 1,670 for commits, 127 contributors, 9K stars, 936 issues, 31 bugs, and 34 enhancements, respectively. The outliers are omitted for better visualization. The main programming language on these projects are JavaScript (30 projects, 17.3%), PHP (44 projects, 25.5%), Python (28 projects, 16.2%), Ruby (41 projects, 23.7%), and Java (30 projects, 17.3%).

All data used in this study is available at:https://bit.ly/3cjnfOG

3 RESULTS

(RQ1) How common is core developer turnover in FLOSS projects?

To answer this RQ, we calculate annual developer turnover rates between 2015 and 2018. We decide not to adopt periods shorter than 12 months, as contributors may stop producing commits for some time (e.g., during their vacation or sick leave). Figure 3 shows ranges for *turnover*(P) values and the percentage of projects impacted by turnover on their team. As we can observe, most projects present a turnover rate between 30% up to 70%. Specifically, 104 out of 174 projects (59.7%) have at least 30% turnover per year, 46 projects (26.4%) exceed 50%, and only 10 projects (5.7%) have less than 10% of annual turnover rates on average.

According to the U.S. Bureau of Statistics[1], the average employee turnover rate in the U.S. is about 12% to 15% annually. In contrast, we can observe in Figure 3 that the annual core developers turnover in FLOSS projects is substantially higher than the turnover rate reported by U.S. Bureau. This significant difference may suggest that core developers loose their motivations over time. Interestingly, on a survey reported by Coelho et al. [6], core developers reveal

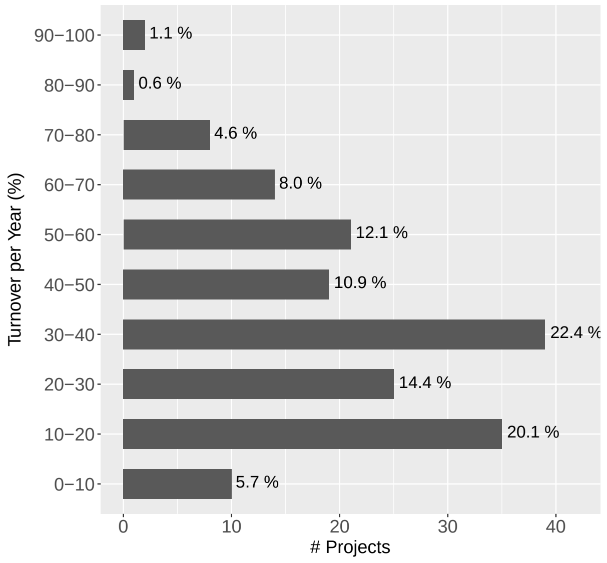


Figure 3: Number of projects by annual core developer turnover

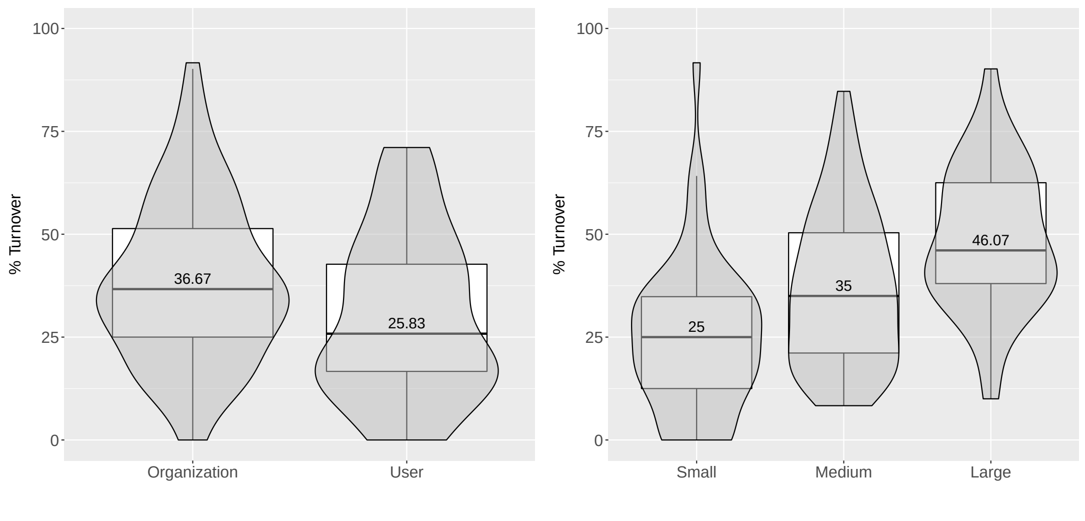
several barriers they face to contribute to a project (e.g., lack of time of the project leaders).

> Only a small percentage of projects (5.7%) have less than 10% of annual turnover rates. Most of them present high turnover rates per year, e.g., 59.7% and 26.4% of the projects have at least 30% and 50%, respectively.

(RQ2) How do team size, programming language, and project owner impact the turnover of core developers?

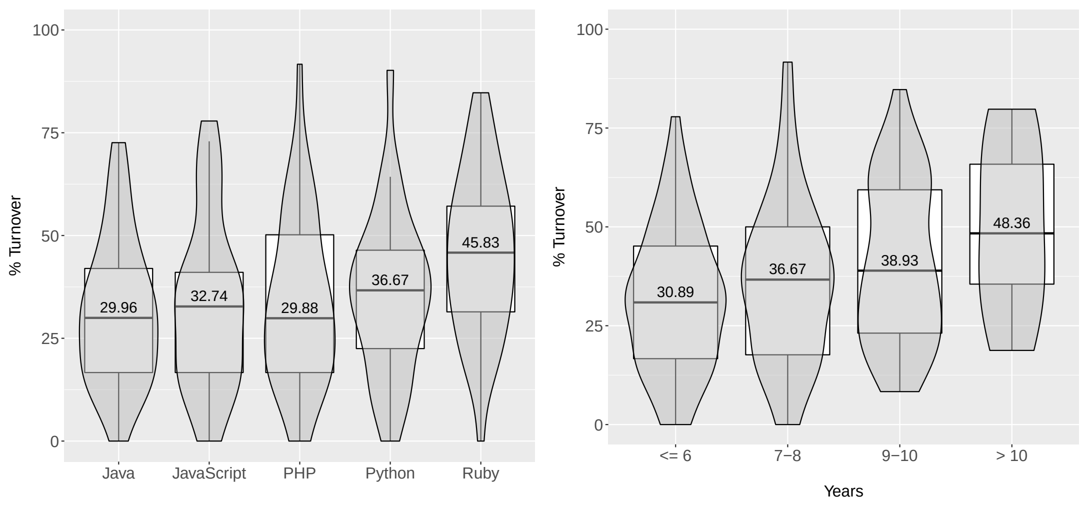
To answer this RQ, we explore the differences concerning distinct factors to analyze whether they can provide insights related to turnover. Figures 4 and 5 show Core Developer Turnover (CDT) rates computed according to project owner, programming language, team size, and project age dimensions. Regarding the team size factor, we intend to check the hypothesis that the bigger the team, the higher the chances of turnover. Regarding the project age factor, we intend to check whether older projects are more stable and therefore have less turnover. Finally, we use the project owner factor because projects owned by organizations might be more stable and have less turnover. The violin plot in Figure 4a shows CDT rates between projects whose owner is an organization and projects whose owner is an individual GitHub user. We compare the values of each factor between the selected projects. First, we evaluate the statistical significance of the difference between the two groups by applying the Mann-Whitney test. The violin plots in Figure 4a show a clear difference between organization and user projects. The CDT rate for organization projects is 36.67%, whereas user projects are 25.83%, considering the median values.

[1]https://www.bls.gov/jlt/



(a) CDT rate by Project Owner **(b) CDT rate by Team Size**

Figure 4: Core Developer Turnover (CDT) rates computed according to project owner and team size perspectives



(a) CDT by Programming Language **(b) CDT by Project Age**

Figure 5: Core Developer Turnover (CDT) rates computed according to programming language and project age perspectives

The third quartile values are 51.36% and 42.70% for organization and user projects, respectively. We use the one-sided version of the Mann-Whitney test (p-value < 0.02) to confirm these differences. A possible reason for this difference is that developers who work for projects owned by organizations can be paid to contribute, which is not common in projects owned by individual users [6]. That is, the motivation to contribute may cease when the developer no longer receives money for it.

Table 1 shows the average turnover per year of some well-known FLOSS projects owned by organizations. The mean turnover formula is: $avg(CDT_{2015} + CDT_{2016} + CDT_{2017} + CDT_{2018})$.

As we can confirm, the values are usually high, ranging from 29.1% (DJANGO/DJANGO) to 61.4% (ADOBE/BRACKETS). We found these numbers very interesting since they reveal that although many large software companies nowadays open source some of their internal projects, these projects are not immune to turnover. Indeed, turnover, in this case, is often higher than the one observed in individual projects, whose key developers tend, therefore, to keep contributing to the projects for a longer time.

Table 1: Mean Turnover Per Year of Some Well-Known FLOSS Projects

Repository	Mean Tunover
ADOBE/BRACKETS	61.4%
REACTIVEX/RXJAVA	55.9%
DOCKER/COMPOSE	46.4%
NODEJS/NODE	46.3%
RAILS/RAILS	39.5%
WORDPRESS/WORDPRESS	38.1%
ATOM/ATOM	39.0%
ANGULAR/ANGULAR.JS	36.8%
FACEBOOK/REACT	33.3%
DJANGO/DJANGO	29.1%

Next, we organized projects according to team size. We discard one-time-contributors and occasionally contributors as team members. For this reason, we do not include developers with at least 0.5% out of the total number of commits of the project. The percentage means that each developer needs to have at least ten commits (approximately), considering the first quartile of the total number of commits presented in Figure 2a. Similarly to Agrawal *et al.* [14], we adopt three team sizes:

- *Small teams:* number of developers < 10.
- *Medium teams:* number of developers ≥ 10 and < 20.
- *Large teams:* number of developers ≥ 20

The violin plots in Figure 4b show turnover distribution according to the size of teams. We apply Kruskal-Wallis test to analyze the statistical significance of the difference between the team size groups. We can observe higher turnover rates for larger core teams, i.e., the probability of a core developer to join or leave a project increases along with the team sizes. Figure 4b also shows that the median turnover is 25% for projects with less than 10 members, while for projects with more than 20 members is 46.07% (*Kruskal-Wallis chi-squared = 32.74, df = 2, p-value* < 0.01).

As shown in Figure 5a, we also evaluate the distribution of core developer turnover for five selected programming languages. We also use Kruskal-Wallis test to analyze the statistical significance of the difference between the groups of programming languages. We note that there is no significant difference concerning programming language, with exception for projects implemented in RUBY (Kruskal-Wallis chi-squared = 12.7, df = 4, *p*-value = 0.01).

Regarding the distribution of the turnover according to the projects' age, we analyzed projects with at least five years (i.e., we discard projects created after 2014). We organized projects into four groups: i) projects younger than seven years, ii) projects ranging between seven and eight years old, iii) projects ranging between nine and ten years old, and iv) projects older than ten years. We also analyze the statistical significance of the difference between the groups by applying the Kruskal-Wallis test. The violin plots in Figure 5b show an increase in core developer turnover rates for older projects. In other words, the probability of a core developer entering or leave a project increases along with the project's age. The median turnover is 30.89% for projects with less than or equal to 6 years, while for projects with more than 10 years is 48.36%. However, there is no statistical significance in this case (Kruskal-Wallis chi-squared = 7.1978, df = 3, *p*-value = 0.06).

Projects owned by organizations have higher turnover rates (36.67%) than projects owned by users (25.83%). Large teams also tend to have higher turnover rates (46.07% for projects with more than 10 members, in contrast to teams with less than 10 members with 25% of turnover). Turnover is notably higher in Ruby projects.

(RQ3) How stable are FLOSS projects regarding their rates of core developers newcomers and leavers?

We compute Core Developer Newcomers (CDN_{Rate}) and Leavers (CDL_{Rate}) rates for the 174 projects from 2015 to 2018. To visualize the relationship between core developers rates, we plot Core Developer Newcomers and Leavers rates for each project and divide the plot into four quadrants similar to Yamashita *et al.* [4]. We use the CDN_{Rate} and CDL_{Rate} medians as thresholds of the quadrants since the median is a more robust measure to outliers [15].

Figure 6 shows the distribution between rates of both core developers metrics. The quadrants are as follow:

- **Stable** (*Low Number of Newcomers and Leavers*): Projects which tend to maintain their core developers, i.e., they do not present high variation in the core team. Figure 6 shows these projects in the light green quadrant.
- **Attractive** (*Low Number of Leavers and High Newcomers*): Projects which tend to grow their core team. They keep most of their core developers but also attract new ones. Figure 6 shows these projects in the dark green quadrant.
- **Unattractive** (*High Number of Leavers but Low Newcomers*): Projects which tend to decrease their core team. They require special attention because they do not keep their core developers nor can attract new ones. Figure 6 shows these projects in the red quadrant.
- **Unstable** (*High Number of Leavers and Newcomers*): Although these projects successfully attract new core developers, they fail at keeping the existing ones. Figure 6 shows these projects which present an alert situation in the orange quadrant.

Figure 6 also shows that the median rates of newcomers and leavers core developers tend to be equivalent, 30.35%, and 31.25%, respectively. There are 21 projects (12.04%) in the red quadrant, in which the project is losing many core developers and is also having difficulties attracting new ones. Therefore, such projects are in an alert situation (e.g., we highlight CACHET, RXJAVA, and PHPEXCEL). On the other hand, only 18 projects (10.34%) are in the dark green quadrant, which is considered the best situation, where projects in addition to keeping their core developers are attracting new ones. Among them, we highlight the BREW and JEKYLL.

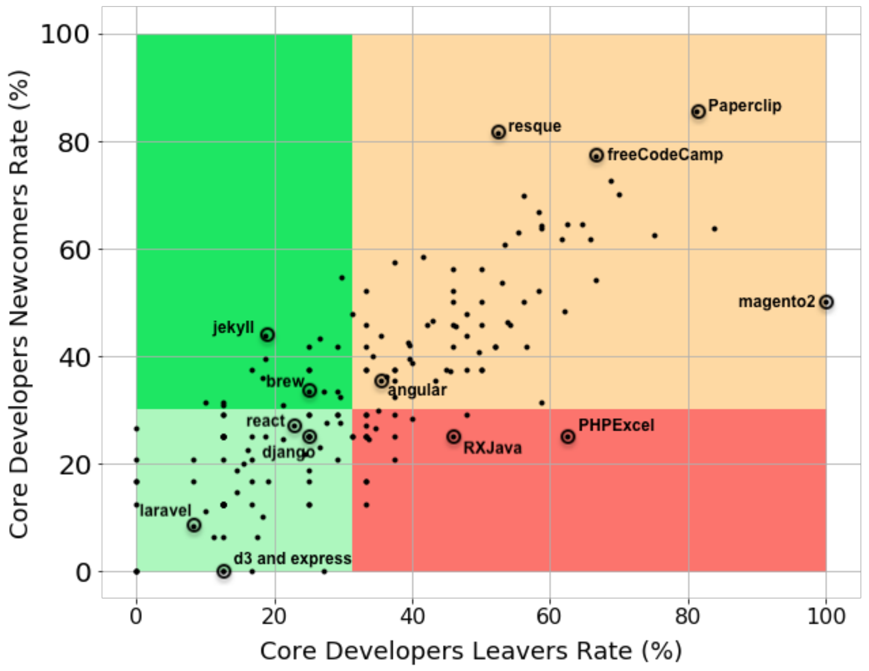


Figure 6: Distribution of core developer newcomers and leavers

The orange quadrant, in which the project is losing many core developers but also attracting new ones, concentrates the higher number of projects, 69 (39.65%). Although projects in this quadrant succeed in attracting new core developers, the high rate of leavers can cause loss of knowledge and experience [3]. For example, we highlight freeCodeCamp, Paperclip, and Resque.

Finally, the green light quadrant concentrates 66 (37.93%) projects, in which projects do not attract new core developers but can keep their existing ones (e.g., React, Django, and Laravel).

In summary, Attractive and Unattractive projects represent 10.3% and 12.0% of the projects, respectively. On the other hand, most of the studied projects are classified as Unstable (39.6%) and Stable (37.9%).

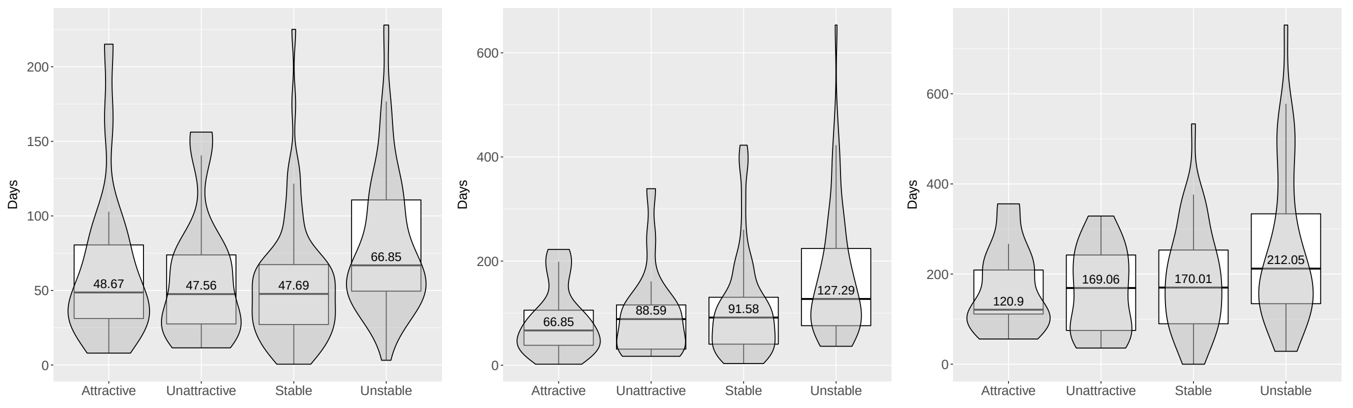
(RQ4) Does the core developers' turnover impact the issue resolution time?

To answer this RQ, we analyze projects by turnover status groups (Stable, Attractive, Unattractive, and Unstable) to detect whether a particular group impacts the time to fix issues in general, bugs, and the time to implement enhancements.

Elapsed Time to Resolve Issues: In this analysis, we collect all solved issues (including issues labeled as bug and enhancements) for each project without taking into account labels, i.e., we do not select a particular type of issues (e.g., bug). The median number of days to solve an issue, considering all the 174 projects and all types of issues, is 55.4 days. Figure 7a shows violin plots with the elapsed time in days to solve issues for each turnover status group. We can note that the elapsed median time to solve issues is approximately 48 days for Attractive, Unattractive, and Stable projects. In this analysis, only for Unstable projects we found a significant statistical difference according to Kruskal-Wallis test. The median time to resolve issues on these projects is 66.85 days (Kruskal-Wallis chi-squared = 16.243, df = 3, p-value < 0.01).

Elapsed Time to Fix Bugs: For each project, we selected all solved issues labeled as *bug* to conduct this analysis. Figure 7b shows violin plots with the elapsed time in days to fix bugs for each turnover status group. As we can observe, Attractive projects tend to fix bugs faster than others, its median time is 66.85 days. Unattractive and Stable projects take approximately 90 days to fix their bugs. Compared to the Attractive projects, Unstable projects take substantially more time to fix bugs (the median time is 127.29 days). They reveal a significant statistical difference according to Kruskal-Wallis test (chi-squared = 16.119, df = 3, p-value < 0.01)

Elapsed Time to Implement Enhancement Issues: For each project, we select all resolved issues labeled as *enhancement* to conduct this analysis. The violin plots in Figure 7c show the number of days by the turnover status group. Although the median time reveals that Attractive projects take a substantially shorter time to conclude enhancement issues (120.9 days) compared to Unstable projects (212.05 days), the number of Attractive projects is very low, out of 95 of 174 projects that classify problems as Enhancements, only 9 (out of 95) are Attractive, which is not representative. Therefore, we could not obtain a statistically significant difference between the groups by applying the Kruskal-Wallis tests (chi-squared



(a) Time to solve issues **(b) Time to solve bugs** **(c) Time to solve enhancements**

Figure 7: Time to solve issues and bugs

= 3.9827, df = 3, p-value = 0.2). However, when we isolate the Unstable group from the others, we could obtain a statistically significant difference according to the one-sided version of the Mann-Whitney test (p-value $\leq$ 0.05). Unstable projects take approximately 212 days to implement enhancements while the other projects take approximately 169 days.

Unstable projects take a longer time to fix issues in general, bugs, and to implement enhancements. For issues, Unstable projects take 66 days to solve them, while other groups take approximately 48 days. For enhancements, Unstable projects take 212 days to implement them while other projects take 169 days. Finally, for bugs, Attractive projects tend to fix them faster (66.85 days), while the time required by Unstable projects is nearly twice (127.29 days).

4 DISCUSSION

According to Sami M. Abbasi and Kenneth W. Hollman [16], turnover is one of the most significant causes of declining productivity and sagging morale in both the public and private sectors. Many studies have shown that turnover is usual in software projects, including FLOSS projects. However, we argue that FLOSS projects require an evaluation of turnover in a particular way. In FLOSS projects, considering all contributors while calculating the turnover might be unfair, and existing studies do not discard occasional contributors. On the contrary, they usually consider one-time-contributors [5]. Thus, the distinguishing design contribution of our study is to focus only on core-developers.

Another contribution is the fact that turnover is investigated over time and not only considering two periods of time. For each project, we identified the set of core developers, from 2014 to 2018, and analyzed how these sets evolved over the years. Moreover, our framework classifies FLOSS projects as Attractive, Unattractive, Stable, and Unstable. This classification has practical value for project maintainers and clients since it reveals, under self-explained categories, the "stability" of FLOSS projects' key developers.

Before starting this research, we hypothesized that the turnover of core developers was not a regular event in FLOSS projects since these projects generally have few core developers. However, RQ1 results show that the turnover of core developers is usual, especially in older projects and projects owned by organizations, as shown in RQ2. Compared to the average employee turnover rate in the U.S. (12% to 15% annually[2]), the core developer turnover rate of the studied projects is significantly higher (30% annually).

On the other hand, the arrival of new core developers can be crucial to oxygenating the team, and the departure can cause loss of knowledge and experience. Therefore, in RQ4, we analyze how these events can impact the actionability of FLOSS projects. For each project, we collected all issues from 2015 to 2018 as well as their respective labels. For each year, we analyzed the amount of opened and closed issues that allow identified that Unstable projects, with a high rate of leavers and newcomers, take more time time to fix issues and bugs and to implement enhancements than other projects.

A limitation of our study resides in how we identify bugs and enhancements. We automatically identified bugs and enhancements, considering all issues that included the label "bug" as bugs and all issues that included the label "enhancement" as enhancements. Therefore, it is possible to have projects that use other labels to denote bugs and enhancements. Besides, not every project labels issues as bug or enhancement out of the projects we analyzed, 137 (78.7%) out 174 projects have issues labeled as bug, while 95 (54.5%) out 174 projects have issues labeled as enhancement.

Another limitation is that we do not consider that a developer who is no longer a core developer may have continued to contribute to the project in other ways or have been just a bit less active. For example, by conducting management activities or even contributing with commits, but occasionally.

[2]https://www.bls.gov/jlt/

5 THREATS TO VALIDITY

In this section, we report threats to validity of this work, as follow:

- **External Validity:** The first concern is related to the generalization of our results. The study was applied on 174 active projects older than five years, implemented in different popular programming languages with hundreds of developers (the median number is 280.5 developers). Although we cover several variables that may impact our results, we may not generalize to other projects implemented in other languages, such as C and C++. However, our dataset is diversified in programming languages. In summary, 17.3% of the projects are implemented in JavaScript, 25.5% in PHP, 16.2% in Python, 23.7% in Ruby, and 17.3% in Java.
- **Internal Validity:** This threat relates to facets that may affect our experimental results. We assume that developers of a project are only those who apply changes in the source code. For this reason, we collected all commits and pull requests applied by them on the projects they contribute. However, developers may use other sources to contribute, such as bug-tracking systems and forums. These repositories could be considered in further work to detect core developers. Another threat refers to the thresholds to define the quadrants. As the mean is sensitive to the influence of outliers and skewed data, we use the median to tackle this threat.
- **Construct Validity:** A construct validity threat relates to the commit-based heuristic for core developer detection [6, 7, 11]. The customized metric [6], adopted in this study, assumes that the core team produces 80% of the total number of commits, and each core contributor should have more than 5% of the overall amount of commits. Therefore, we might not have included contributors who belong to the core team for having a number of contributions slightly different from our thresholds.

6 RELATED WORK

Employee turnover is a key concept in Human Resources (HR) metrics to measure talent loss in a company. This metric is centered on the number of employees who leave a company in a certain period of time and the average number of employees in that period. In the software industry context, companies are highly dependent on intellectual capital, and the negative impacts for each departure tend to be significantly high [17]. According to a turnover report in 2018 from LinkedIn, the software sector has the highest turnover rate at 13.2% on average (e.g., the computer games at 15.5% and user experience designers at 23.3%).[3]

6.1 Turnover in Software Industry

Chatzipetrou *et al.* [17] conducted a study in an offshore vendor company to investigate whether employees retention is related to their experience and whether there is a threshold associated with the employees tendency to leave the studied company. They concluded that there is an association between employee retention and the time employees remain in the company. Almost 90% of the employees leave the company within the first year, while in the second year is 50%, revealing that two years is the retention threshold for the investigated company.

Similarly, Bass *et al.* [18] conducted a study to find out why turnover in offshore outsourcing teams is higher than offshore in-sourcing. They interviewed members of three organizations: a large outsourcing, an offshore software development laboratory, and a company with globally distributed teams. Their findings suggest an effect on employee motivation. The outsourcing company reported negative experiences (e.g., anti-social working hours), while the others reported positive experiences (e.g., work-life balance). They concluded that offshore outsourcing providers could reduce member turnover if they improve work-life balance and adopt more family-friendly employment policies.

In an attempt to tackle the high developers turnover problem in non-open-source companies, Bao *et al.* [19] analyze monthly reports from two IT companies. Their goal is to predict whether a developer will leave the company in his/her first year after being hired. In summary, they detected three main factors that could impact on developer departure: i) the content of task report, ii) the standard deviation of working hours, and iii) the standard deviation of working hours of team members in the first month.

6.2 Turnover in FLOSS Projects

In the FLOSS context, several studies investigate different problems, such as: project attractiveness for newcomers [8, 20], contributors turnover [2, 21], reasons developers abandon open source projects [22, 23], and reasons projects become unmaintained [24–26]. For example, Meirelles et al. [20] investigate the existence of relationships between source code metrics and the ability of projects to attract users and developers. Their findings reveal a correlation between some source code metrics with attractiveness. While authors analyze projects written in the C language, we select projects implemented in five different programming languages (Java, JavaScript, PHP, Python, and Ruby). Moreover, we analyze projects attractiveness with core developers newcomers and leavers.

Particularly, turnover has drawn attention since its impact may be even higher than in the software industry for having many contributors as volunteers. For example, Yamashita *et al.* [4] use population migration metrics to investigate migratory trends of developers in FLOSS projects. Specifically, they evaluate projects' tendency to retain existing contributors and attract new ones. Their findings show that most subject projects tend to retain more contributors than to attract new ones.

Foucault *et al.* [2] investigate whether turnover is an important factor in OSS projects and how it can impact the quality of these projects. The authors define metrics to measure external and internal turnover. Based on these metrics, they analyze newcomers and leavers (external turnover) and developers role changes within the project (internal turnover). Their results reveal that the studied projects have a high turnover. They also find that external turnover impact negatively on the quality of projects modules. However, their study focuses on all contributors of FLOSS, whereas, in this paper, we study only the turnover of key developers, also known as core developers. On the other side, Sharma et al. [21] explore the impact of developer and project variables on developer turnover

[3]https://business.linkedin.com/talent-solutions/blog/trends-and-research/2018/the-3-industries-with-the-highest-turnover-rates

rates. They build a turnover behavior model to investigate factors that may lead developers to become inactive and may impact developer turnover rates. Their findings reveal that developer role (based on the type of commit changes applied in the project), past activity, project age, and project size are major predictors of turnover rates. In our work, we use developer role, project age, and past activities variables to investigate developer turnover. By contrast, we evaluate the developer role variable from a different perspective, i.e., we only consider developers who are core members in the project. Moreover, in our study, we include additional variables not evaluated by the authors [21], such as team size, project owner, and issue reports.

Nonetheless, most FLOSS contributors are not very active and they usually make one or two commits per year [6]. The aforementioned studies take into account all commits to compute turnover, which may affect their results, i.e., they may not represent the projects' reality. In contrast, our metrics are centered on core developers to analyze turnover in FLOSS projects.

7 CONCLUSIONS

In this paper, we select 174 FLOSS projects and computed its *Core Developer Turnover (CDT)* rate for each year between 2015 and 2018. We propose four research questions to evaluate *CDT* rates, factors that may impact CDT, a framework to classify FLOSS projects according to the turnover status group, Attractive, Unattractive, Stable, and Unstable, and how these turnover status groups impact the issue resolution time of projects. Our principal findings are as follows:

- Core developers' turnover is a frequent event in FLOSS projects, and it tends to be higher in Ruby projects.
- Projects owned by organizations have a higher *Core Developer Turnover (CDT)* rate than projects owned by individual users.
- Core developers' turnover increases for large teams.
- Attractive projects tend to fix bugs faster, while Unstable projects take a significantly longer time to fix them (around twice in number of days).

As future work, we plan to compare the workload of older Core Developers with Core Developers Newcomers. Is the Newcomers productivity equivalent to the existing Core Developers? We also intend to compare whether Core Developers Leavers continued to contribute to the project in other ways, such as in management activities or even contributing with commits, but occasionally. Finally, an interesting study is to predict when a core developer will leave a FLOSS project.

ACKNOWLEDGMENTS

This work is supported by CNPq and FAPEMIG.

REFERENCES

[1] D. A. Tamburri, F. Palomba, A. Serebrenik, and A. Zaidman, "Discovering community patterns in open-source: a systematic approach and its evaluation," *Empirical Software Engineering*, vol. 24, no. 3, pp. 1369–1417, 2019.
[2] M. Foucault, M. Palyart, X. Blanc, G. C. Murphy, and J.-R. Falleri, "Impact of developer turnover on quality in open-source software," in *10th Joint Meeting on Foundations of Software Engineering (FSE)*, 2015, pp. 829–841.
[3] A. Mockus, "Organizational volatility and its effects on software defects," in *18th International Symposium on the Foundations of Software Engineering (FSE)*, 2010, pp. 117–126.
[4] K. Yamashita, S. McIntosh, Y. Kamei, and N. Ubayashi, "Magnet or sticky? an oss project-by-project typology," in *11th Working Conference on Mining Software Repositories (MSR)*, 2014, pp. 344–347.
[5] A. Lee, J. C. Carver, and A. Bosu, "Understanding the impressions, motivations, and barriers of one time code contributors to floss projects: A survey," in *39th International Conference on Software Engineering (ICSE)*, 2017, pp. 187–197.
[6] J. Coelho, M. T. Valente, L. L. Silva, and A. Hora, "Why we engage in FLOSS: Answers from core developers," in *11th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)*, 2018, pp. 14–21.
[7] M. Joblin, S. Apel, C. Hunsen, and W. Mauerer, "Classifying developers into core and peripheral: An empirical study on count and network metrics," in *39th International Conference on Software Engineering (ICSE)*, 2017, pp. 164–174.
[8] A. Terceiro, L. R. Rios, and C. Chavez, "An empirical study on the structural complexity introduced by core and peripheral developers in free software projects," in *24th Brazilian Symposium on Software Engineering (SBES)*, 2010, pp. 21–29.
[9] G. Avelino, L. Passos, A. Hora, and M. T. Valente, "A novel approach for estimating truck factors," in *24th International Conference on Program Comprehension (ICPC)*, 2016, pp. 1–10.
[10] G. Avelino, L. Passos, A. Hora, and M. T. Valente, "Measuring and analyzing code authorship in 1 + 118 open source projects," *Science of Computer Programming*, vol. 176, no. 1, pp. 14 – 32, 2019.
[11] G. Robles, J. M. Gonzalez-Barahona, and I. Herraiz, "Evolution of the core team of developers in libre software projects," in *6th International Working Conference on Mining Software Repositories (MSR)*, 2009, pp. 167–170.
[12] J. B. Arthur, "Effects of human resource systems on manufacturing performance and turnover," *Academy of Management journal*, vol. 37, no. 3, pp. 670–687, 1994.
[13] G. Saridakis and C. Cooper, *Research handbook on employee turnover.* Edward Elgar Publishing, 2016.
[14] A. Agrawal, A. Rahman, R. Krishna, A. Sobran, and T. Menzies, "We don't need another hero?: the impact of heroes on software development," in *40th International Conference on Software Engineering: Software Engineering in Practice (ICSE:SEIP)*, 2018, pp. 245–253.
[15] C. Leys, C. Ley, O. Klein, P. Bernard, and L. Licata, "Detecting outliers: Do not use standard deviation around the mean, use absolute deviation around the median," *Journal of Experimental Social Psychology*, vol. 49, no. 4, pp. 764–766, 2013.
[16] S. M. Abbasi and K. W. Hollman, "Turnover: The real bottom line," *Public personnel management*, vol. 29, no. 3, pp. 333–342, 2000.
[17] P. Chatzipetrou, D. Šmite, and R. van Solingen, "When and who leaves matters: emerging results from an empirical study of employee turnover," in *12th International Symposium on Empirical Software Engineering and Measurement (ESEM)*, 2018, p. 53.
[18] J. M. Bass, B. Sarah, M. A. Razzak, and J. Noll, "Employee retention and turnover in global software development: comparing in-house offshoring and offshore outsourcing," in *13th International Conference on Global Software Engineering (ICGSE)*, 2018, pp. 77–86.
[19] L. Bao, Z. Xing, X. Xia, D. Lo, and S. Li, "Who will leave the company?: a large-scale industry study of developer turnover by mining monthly work report," in *14th International Conference on Mining Software Repositories (MSR)*, 2017, pp. 170–181.
[20] P. Meirelles, C. Santos Jr., J. Miranda, F. Kon, A. Terceiro, and C. Chavez, "A study of the relationships between source code metrics and attractiveness in free software projects," in *24th Brazilian Symposium on Software Engineering (SBES)*, 2010, pp. 11–20.
[21] P. N. Sharma, J. Hulland, and S. Daniel, "Examining turnover in open source software projects using logistic hierarchical linear modeling approach," in *8th International Conference on Open Source Systems (OSS)*, 2012, pp. 331–337.
[22] I. Steinmacher, I. Wiese, A. P. Chaves, and M. A. Gerosa, "Why do newcomers abandon open source software projects?" in *6th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)*, 2013, pp. 25–32.
[23] G. Avelino, E. Constantinou, M. T. Valente, and A. Serebrenik, "On the abandonment and survival of open source projects: An empirical investigation," in *13th International Symposium on Empirical Software Engineering and Measurement (ESEM)*, 2019, pp. 1–11.
[24] J. Coelho, M. T. Valente, L. L. Silva, and E. Shihab, "Identifying unmaintained projects in GitHub," in *12th International Symposium on Empirical Software Engineering and Measurement (ESEM)*, 2018, pp. 1–10.
[25] J. Coelho, M. T. Valente, L. Milen, and L. L. Silva, "Is this GitHub project maintained? measuring the level of maintenance activity of open-source projects," *Information and Software Technology*, vol. 1, no. 1, pp. 1–35, 2020.
[26] J. Coelho and M. T. Valente, "Why modern open source projects fail," in *25th International Symposium on the Foundations of Software Engineering (FSE)*, 2017, pp. 186–196.