Quality Gatekeepers
Investigating the Effects of Code Review Bots on Pull Request Activities

Wessel, Mairieli; Serebrenik, Alexander; Wiese, Igor Scaliante; Steinmacher, Igor; Gerosa, Marco Aurélio

DOI
10.1007/s10664-022-10130-9

Publication date
2022

Document Version
Final published version

Published in
Empirical Software Engineering

Citation (APA)
Wessel, M., Serebrenik, A., Wiese, I. S., Steinmacher, I., & Gerosa, M. A. (2022). Quality Gatekeepers: Investigating the Effects of Code Review Bots on Pull Request Activities. *Empirical Software Engineering*, *27*(5), Article 108. https://doi.org/10.1007/s10664-022-10130-9

Empirical Software Engineering (2022) 27:108
https://doi.org/10.1007/s10664-022-10130-9

Quality gatekeepers: investigating the effects of code review bots on pull request activities

Mairieli Wessel[1,2] · Alexander Serebrenik[3] · Igor Wiese[4] · Igor Steinmacher[4] · Marco A. Gerosa[5]

Accepted: 2 February 2022

Abstract

Software bots have been facilitating several development activities in Open Source Software (OSS) projects, including code review. However, these bots may bring unexpected impacts to group dynamics, as frequently occurs with new technology adoption. Understanding and anticipating such effects is important for planning and management. To analyze these effects, we investigate how several activity indicators change after the adoption of a code review bot. We employed a regression discontinuity design on 1,194 software projects from GitHub. We also interviewed 12 practitioners, including open-source maintainers and contributors. Our results indicate that the adoption of code review bots increases the number of monthly merged pull requests, decreases monthly non-merged pull requests, and decreases communication among developers. From the developers' perspective, these effects are explained by the transparency and confidence the bot comments introduce, in addition to the changes in the discussion focused on pull requests. Practitioners and maintainers may leverage our results to understand, or even predict, bot effects on their projects.

Keywords Software bots · GitHub bots · Code review · Automation · Open source software · Software engineering

1 Introduction

Open Source Software (OSS) projects frequently employ code review in the development process (Baysal et al. 2016), as it is a well-known practice for software quality assurance (Ebert et al. 2019). In the pull-based development model, project maintainers carefully inspect code changes and engage in discussion wdraftrulesith contributors to understand

Communicated by: Zhenchang Xing and Kelly Blincoe

This article belongs to the Topical Collection: *Software Maintenance and Evolution (ICSME)*

The work was conducted while the author (Mairieli Wessel) was affiliated to University of São Paulo.

✉ Mairieli Wessel
m.wessel@tudelft.nl; mairieli@ime.usp.br

Extended author information available on the last page of the article.

Published online: 30 May 2022

and improve the modifications before integrating them into the codebase (McIntosh et al. 2014). The time maintainers spend reviewing pull requests is non-negligible and can affect, for example, the volume of new contributions (Yu et al. 2015) and the onboarding of newcomers (Steinmacher et al. 2013).

Software bots play a prominent role in the code review process (Wessel et al. 2018). These automation tools serve as an interface between users and other tools (Storey and Zagalsky 2016) and reduce the workload of maintainers and contributors. Accomplishing tasks that were previously performed solely by human developers, and interacting in the same communication channels as their human counterparts, bots have become new voices in the code review conversation (Monperrus 2019). Throughout comments on pull requests, code review bots guide contributors to provide necessary information before maintainers triage the pull requests (Wessel et al. 2018).

Notoriously, though, the adoption of new technology can bring consequences that counter the expectations of the technology designers and adopters (Healy 2012). Many systems intended to serve the user ultimately add new burdens. Developers who a priori expect technological developments to produce significant performance improvements can be caught off-guard by a posteriori unanticipated operational complexities (Woods and Patterson 2001). According to Mulder (2013), many effects are not directly caused by the new technology itself, but by the changes in human behavior that it provokes. Therefore, it is important to assess and discuss the effects of a new technology on group dynamics; yet, this is often neglected when it comes to software bots (Storey and Zagalsky 2016; Paikari and van der Hoek 2018).

In the code review process, bots may affect existing project activities in several ways. For example, bots can provide poor feedback (Wessel et al. 2018; Wessel and Steinmacher 2020), as illustrated by a developer: "*the comments of @<bot-name> should contain more description on how to read the information contained and what one actually [understand] from it. For a newcomer its not obvious at all.*"[1] In turn, this poor feedback may lead to contributor drop-out—indeed, poor feedback on pull requests is known to discourage further contributions (Steinmacher et al. 2018; Balali et al. 2018).

In this paper, we aim to understand how the dynamics of GitHub project pull requests change following the adoption of code review bots. To understand what happens after the adoption of a bot, we used a mixed-methods approach (Easterbrook et al. 2008) with a sequential explanatory strategy (Creswell 2003), combining data analysis of GitHub data with semi-structured interviews conducted with open-source developers. We used a *Regression Discontinuity Design* (RDD) (Thistlethwaite and Campbell 1960) to model the effects of code review bot adoption across 1,194 OSS projects hosted on GitHub. We used RDD since it can assess how much an intervention changed an outcome of interest, immediately and over time, and also evaluate whether the change could be attributed to other factors rather than the intervention. Afterward, to further shed light on our results, we conducted semi-structured interviews with practitioners, including open-source project maintainers and contributors experienced with code review bots.

We found that, after code review bot adoption, more pull requests are merged into the codebase, and communication decreases between contributors and maintainers. Considering non-merged pull requests, after bot adoption projects have fewer monthly non-merged pull requests, and faster pull request rejections. From the practitioners' perspective, the bot comments make it easier to understand the state and quality of the contributions and increase

[1] https://twitter.com/markusstaab/status/1048503185361555457

maintainers' confidence in merging pull requests. According to them, contributors are likely to make changes in the code without interacting with other maintainers, which also helps to change the focus of developers' discussions.

The main contributions of this paper are:

1. The identification of changes in project activity indicators after the adoption of a code review bot.
2. The elucidation of how the introduction of a bot can impact OSS projects.
3. Open-source developers' perspective on the impacts of code reviews bots.

These contributions aim to help practitioners and maintainers understand bots' effects on a project, especially to avoid the ones that they consider undesirable. Additionally, our findings may guide developers to consider the implications of new bots as they design them.

This paper extends our ICSME 2020 paper entitled "Effects of Adopting Code Review Bots on Pull Requests to OSS Projects" (Wessel et al. 2020b). In this extended version, we further investigate the reasons for the change incurred by code review bot adoption, considering the practical perspective of open-source developers. To do so, we adjusted the methodology, results, and discussion, including a new research question (i.e., RQ2), which is based on the qualitative analysis of interviews with 12 open-source developers. We also present a more extensive related work section, where we discuss empirical works that use Regression Discontinuity Design to model the effects of a variety of interventions on development activities.

2 Exploratory Case Study

As little is known about the effects of code review bots' adoption in the dynamics of pull requests, we conducted an exploratory case study (Runeson and Höst 2009; Yin 2003) to formulate hypotheses to further investigate in our main study. Figure 1 shows an overview of the research design employed in this exploratory case study.

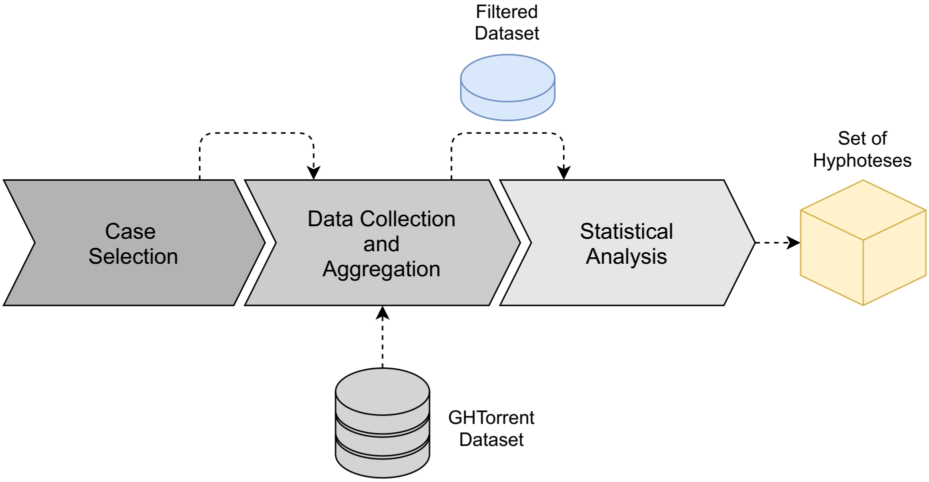


Fig. 1 Case study design overview

2.1 Code Review Bot on Pull Requests

According to Wessel et al. (2018), code review bots are software bots that analyze code style, test coverage, code quality, and smells. As an interface between human developers and other tools, code review bots generally report the feedback of a third-party service on the GitHub platform. Thus, these bots are designed to support contributors and maintainers after pull requests have been submitted, aiming to facilitate discussions and assist code reviews. One example of a code review bot is the Codecov bot.[2] This bot reports the code coverage on every new pull request right after all tests have passed. As shown in Fig. 2, Codecov bot leaves highly detailed comments, including the percentage of increased or decreased coverage, and the impacted files.‘

2.2 Case Selection

To carry out our exploratory case study, we selected two projects that we were aware of that used code review bots for at least a one year: the Julia programming language project[3] and CakePHP,[4] a web development framework for PHP. Both projects have popular and active repositories—Julia has more than 26.1k stars, 3.8k forks, 17k pull requests, and 46.4k commits; while CakePHP has more than 8.1k stars, 3.4k forks, 8.6k pull requests, 40.9k commits, and is used by 10k projects. Both projects adopt Codecov bot, which posted the first comments on pull requests to the Julia project in July 2016 and CakePHP in April 2016.

2.3 Data Collection and Aggregation

After selecting the projects, we analyzed data from one year before and one year after bot adoption, using the data available in the GHTorrent dataset (Gousios and Spinellis 2012). During this time frame, the only bot adopted by Julia and CakePHP was the Codecov bot. Similar to previous work (Zhao et al. 2017), we exclude 30 days around the bot's adoption to avoid the influence of instability caused during this period. Afterward, we aggregated individual pull request data into monthly periods, considering 12 months before and after the bot's introduction. We choose the month time frame based on previous literature (Zhao et al. 2017; Kavaler et al. 2019; Cassee et al. 2020). All metrics were aggregated based on the month of the pull request being closed/merged.

We considered the same activity indicators used in the previous work by Wessel et al. (2018):

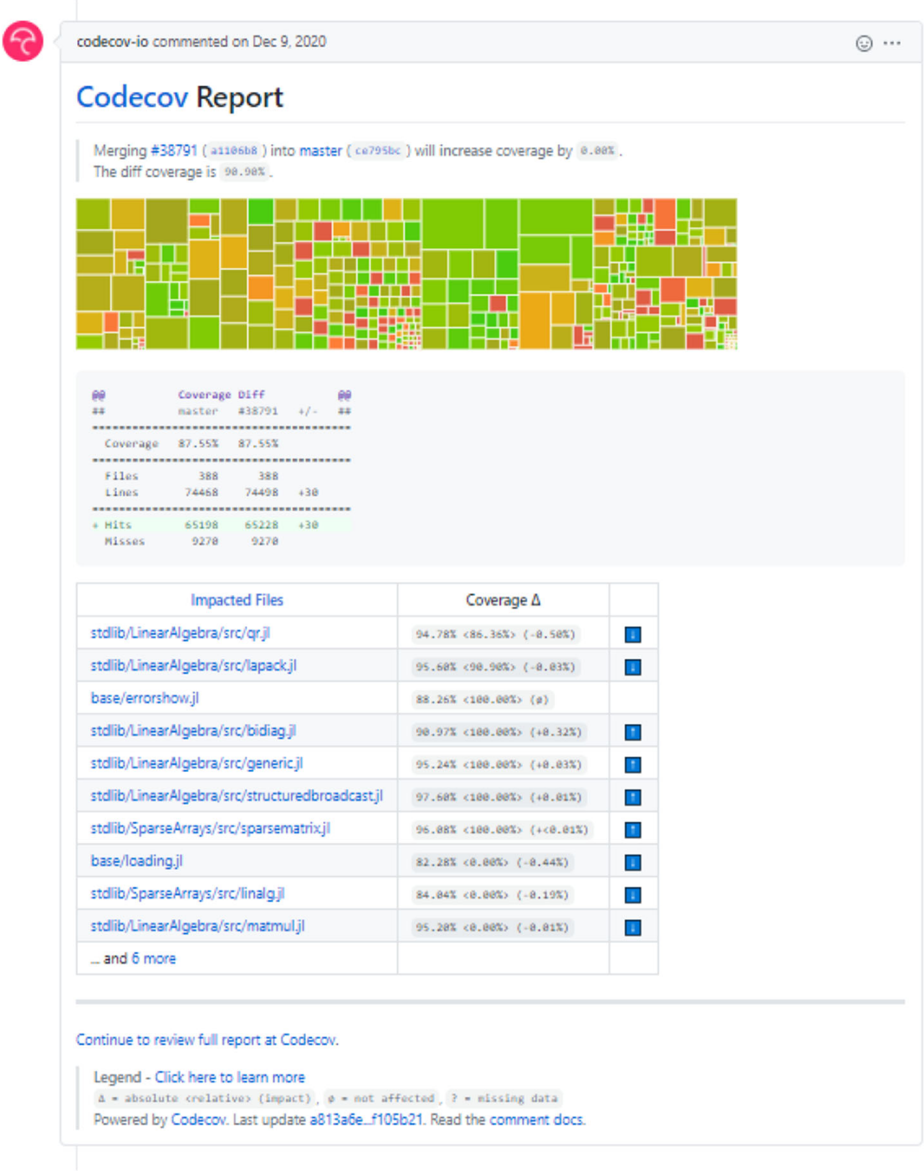
Merged/non-merged pull requests: the number of monthly contributions (pull requests) that have been merged, or closed but not merged into the project, computed over all closed pull requests in each time frame.

Comments on merged/non-merged pull requests:the median number of monthly comments—excluding bot comments—computed over all merged and non-merged pull requests in each time frame.

[2] https://github.com/marketplace/codecov

[3] https://github.com/JuliaLang/julia

[4] https://github.com/cakephp/cakephp



codecov-io commented on Dec 9, 2020

Codecov Report

> Merging #38791 (a1106b8) into master (ce795bc) will increase coverage by 0.00%.
> The diff coverage is 90.90%.

```
@@            Coverage Diff           @@
##           master   #38791   +/-   ##
=======================================
  Coverage   87.55%   87.55%
=======================================
  Files         388      388
  Lines       74468    74498   +30
=======================================
+ Hits        65198    65228   +30
  Misses       9270     9270
```

Impacted Files	Coverage Δ	
stdlib/LinearAlgebra/src/qr.jl	94.78% <86.36%> (-0.50%)	↓
stdlib/LinearAlgebra/src/lapack.jl	95.60% <90.90%> (-0.03%)	↓
base/errorshow.jl	88.26% <100.00%> (ø)	
stdlib/LinearAlgebra/src/bidiag.jl	90.97% <100.00%> (+0.32%)	↑
stdlib/LinearAlgebra/src/generic.jl	95.24% <100.00%> (+0.03%)	↑
stdlib/LinearAlgebra/src/structuredbroadcast.jl	97.60% <100.00%> (+0.01%)	↑
stdlib/SparseArrays/src/sparsematrix.jl	96.08% <100.00%> (+<0.01%)	↑
base/loading.jl	82.28% <0.00%> (-0.44%)	↓
stdlib/SparseArrays/src/linalg.jl	84.04% <0.00%> (-0.19%)	↓
stdlib/LinearAlgebra/src/matmul.jl	95.20% <0.00%> (-0.01%)	↓
... and 6 more		

Continue to review full report at Codecov.

> Legend - Click here to learn more
> Δ = absolute <relative> (impact), ø = not affected, ? = missing data
> Powered by Codecov. Last update a813a6e...f105b21. Read the comment docs.

Fig. 2 Codecov bot comment example

Time-to-merge/time-to-close pull requests: the median of monthly pull request latency (in hours), computed as the difference between the time when the pull request was closed and the time when it was opened. The median is computed using all merged and non-merged pull requests in each time frame.

Commits of merged/non-merged pull requests: the median of monthly commits computed over all merged and non-merged pull requests in each time frame.

For all activity indicators we use the median because their distribution is skewed.

2.4 Statistical Analysis

We ran statistical tests to compare the activity indicators distributions before and after the bot adoption. As the sample is small, and there is no critical mass of data points around the bot's introduction, we used the non-parametric Mann-Whitney-Wilcoxon test (Wilks 2011). In this context, the null hypothesis (H_0) is that the distributions pre- and post-adoption are the same, and the alternative hypothesis (H_1) is that these distributions differ. We also used Cliff's Delta (Romano et al. 2006) to quantify the difference between these groups of observations beyond p-value interpretation. Moreover, we inspected the monthly distribution of each metric to search for indications of change.

As aforementioned, the case studies helped us to formulate hypotheses for the main study, which comprised more than one thousand projects. We formulated hypotheses whenever we observed changes in the indicators for at least one of the two projects we analyzed in the case study.

2.5 Case Study Results

In the following, we discuss the trends in project activities after bot adoption. We report the results considering the studied pull request activities: number of merged and non-merged pull requests, median of pull request comments, time-to-merge and time-to-close pull requests, and median of pull request commits.

2.5.1 Trends in the Number of Merged and Non-merged Pull Requests

The number of merged pull requests *increased* for both projects (Julia: p-value 0.0003, $\delta = -0.87$; CakePHP: p-value 0.001, $\delta = -0.76$), whereas the non-merged pull requests *decreased* for both projects (Julia: p-value 0.00007, $\delta = 0.87$; CakePHP: p-value 0.00008, $\delta = 0.95$). Figure 3 shows the monthly number of merged and non-merged pull requests, top and bottom respectively, before and after bot adoption for both projects. Based on these findings, we hypothesize that:

$H_{1.1}$ The number of monthly merged pull requests increases after the introduction of a code review bot.

$H_{1.2}$ The number of monthly non-merged pull requests decreases after the introduction of a code review bot.

2.5.2 Trends in the Median of Pull Request Comments

Figure 4 shows the monthly median of comments on merged and non-merged pull requests, respectively. CakePHP showed statistically significant differences between pre- and post-adoption distributions. The number of comments *increased* for merged pull requests (p-value = 0.01, $\delta = -0.56$) and also for non-merged ones (p-value = 0.03, $\delta = -0.50$) with a large effect size. Thus, we hypothesize that:

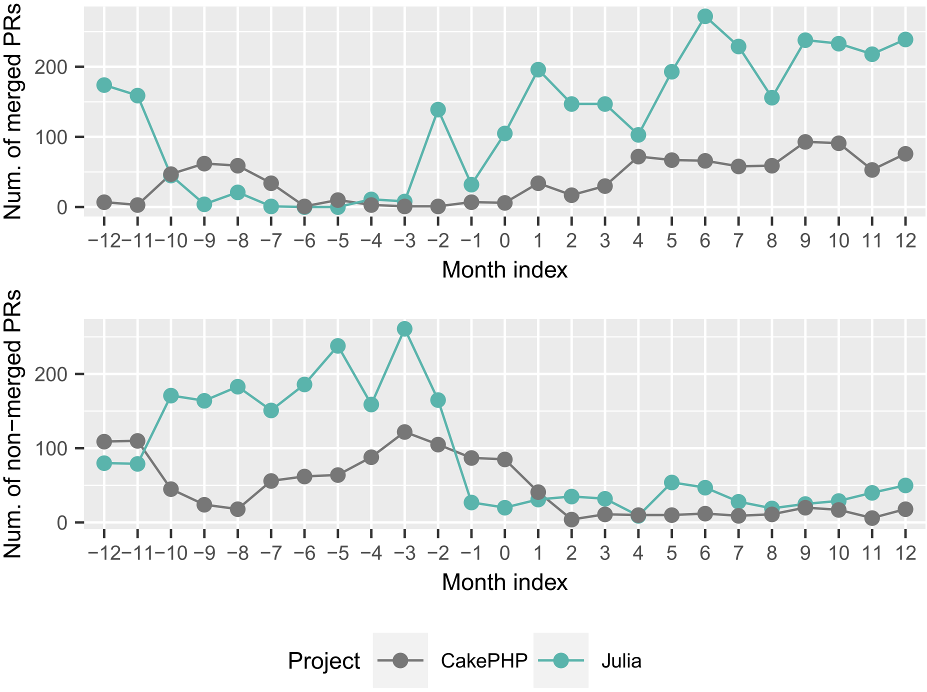


Fig. 3 Monthly merged and non-merged pull requests

> ***$H_{2.1}$* The adoption of code review bots is associated with an increase in the monthly number of comments for merged pull requests.**

> ***$H_{2.2}$* The number of monthly comments on non-merged pull requests increases after the adoption of a code review bot.**

2.5.3 Trends in the Time to Close Pull Request Comments

The median time to merge pull requests *increased* for both projects (Julia: p-value 0.0003, $\delta = -1.00$; CakePHP: p-value 0.000001, $\delta = -0.98$). Considering non-merged pull requests, the difference between pre- and post-adoption is statistically significant only for Julia. For this project, the median time to close pull requests *increased* (p-value 0.00007) with a large effect size ($\delta = -0.65$). The distribution can be seen in Fig. 5. Therefore, we hypothesize that:

> ***$H_{3.1}$* There is an increase in the monthly time to merge pull requests after the introduction of code review bots.**

> ***$H_{3.2}$* There is an increase in the monthly time to reject pull requests after the adoption of a code review bot.**

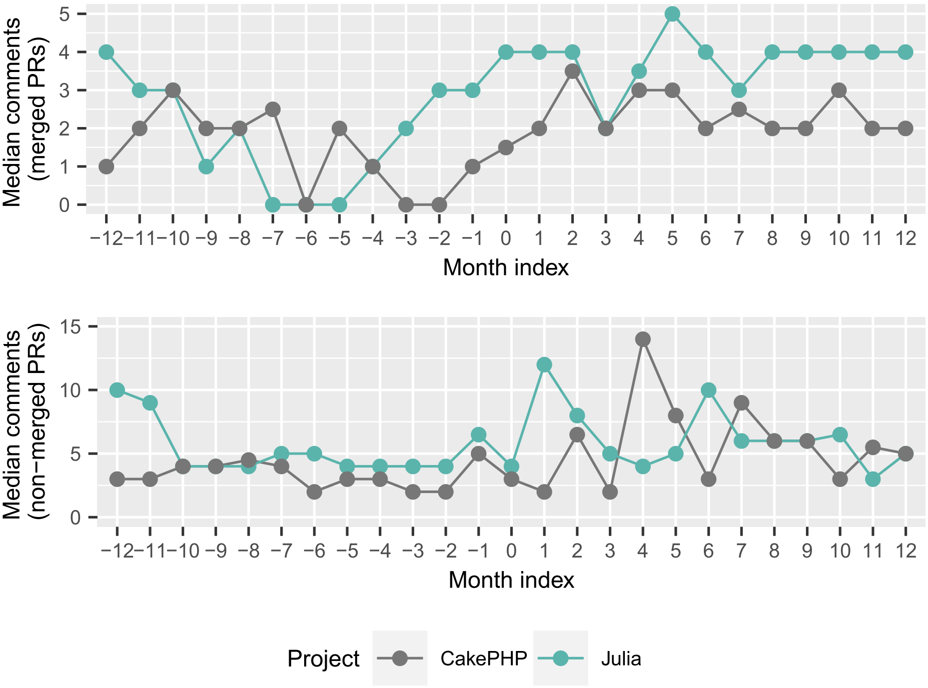


Fig. 4 Monthly comments on merged and non-merged pull requests

2.5.4 Trends in the Median of Pull Request Commits

Investigating the number of pull request commits per month (see Fig. 6), we note that the medians before the adoption are quite stable, especially for merged pull requests. In comparison, after adoption we observe more variance. The difference is statistically significant only for CakePHP, for which the number of pull request commits increased for merged pull requests (p-value = 0.002, $\delta = -0.58$) and for non-merged pull requests (p-value = 0.002, $\delta = -0.69$) with a large effect size. Based on this, we posit:

$H_{4.1}$ ***There is an increase in the monthly number of commits for merged pull requests after code review bot adoption.***

$H_{4.2}$ ***There is an increase in the monthly number of commits for non-merged pull requests after code review bot adoption.***

Summary of the Case Study. Unlike Wessel et al. (2018), we observe statistically significant differences for all four activity indicators we investigated in at least one of the two projects. Based on these observations, we formulated hypotheses to be further investigated in our main study, comprising a large number of projects, and employed the regression discontinuity design.

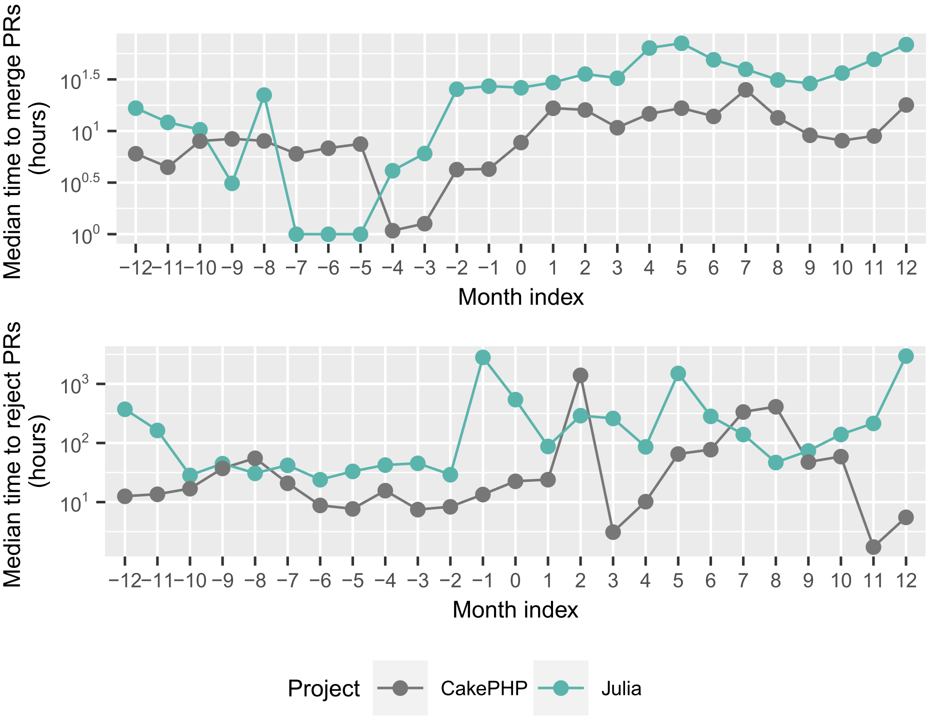


Fig. 5 Monthly median time to merge and reject pull requests

3 Main Study Design

In this section, we describe our research questions (Section 3.1), the statistical approach and data collection procedures (Section 3.2), and the qualitative approach (Section 3.3).

3.1 Research Questions

The main goal of this study is to investigate how and for what reasons, if any, the adoption of code review bots affects the dynamics of GitHub project pull requests. To achieve this goal, we investigated the following research questions:

RQ1. *How do pull request activities change after a code review bot is adopted in a project?*

Extending the work of Wessel et al. (2018), we investigate changes in project activity indicators, such as the number of pull requests merged and non-merged, number of comments, the time to close pull requests, and the number of commits per pull request. Using time series analysis, we account for how the bot adoption has impacted these project activity indicators over time. We also go one step further, exploring a large sample of open-source projects and focusing on understanding the effects of a specific bot category.

RQ2. *How could the change in pull request activities be explained?*

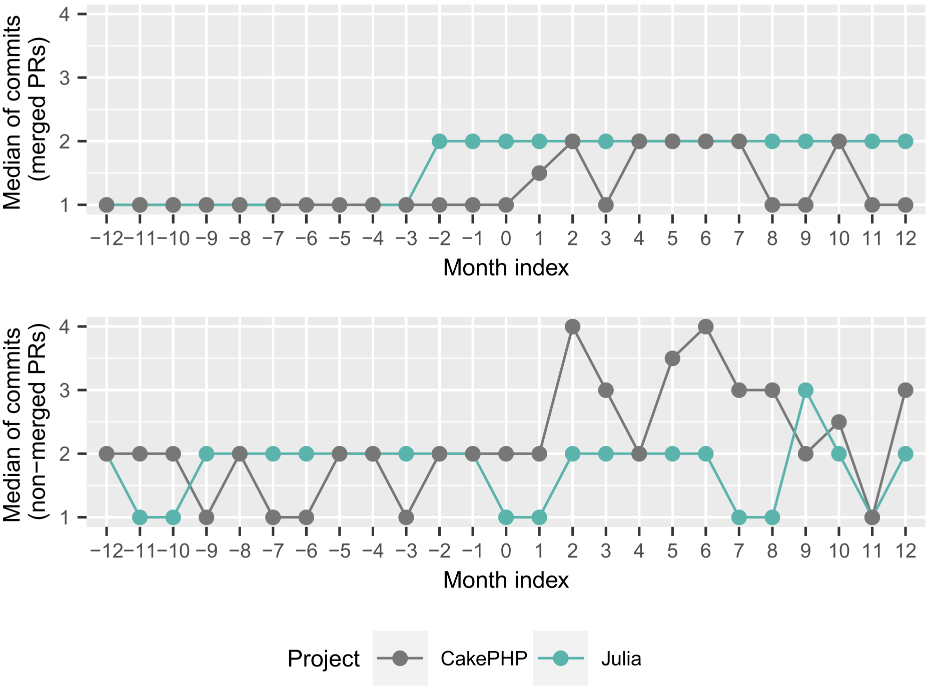


Fig. 6 Monthly commits on merged and non-merged pull requests

Besides understanding the change incurred by bot adoption, we explore why it happens. To do so, we interviewed a set of open-source developers who actually have been using these bots.

Figure 7 illustrates an overview of the steps taken to address the research questions. Next, we explain each step in order to justify the study design decisions.

3.2 Stage 1—Statistical Approach

Considering the hypotheses formulated in the case study, in our main study we employed time series analysis to account for the longitudinal effects of bot adoption. We employed Regression Discontinuity Design (RDD) (Thistlethwaite and Campbell 1960; Imbens and Lemieux 2008), which has been applied in the context of software engineering in the past (Zhao et al. 2017; Cassee et al. 2020). RDD is a technique used to model the extent of a discontinuity at the moment of intervention and long after the intervention. The technique is based on the assumption that if the intervention does not affect the outcome, there would be no discontinuity, and the outcome would be continuous over time (Cook and Campbell 1979). The statistical model behind RDD is

$$\begin{aligned} y_i \ = \ & \alpha + \beta \cdot time_i + \gamma \cdot intervention_i \\ & + \delta \cdot time\_after\_intervention_i \ + \eta \cdot controls_i + \varepsilon_i \end{aligned}$$

where i indicates the observations for a given project. To model the passage of time as well as the bot introduction, we include three additional variables: *time*, *time after intervention*, and *intervention*. The *time* variable is measured as months at the time j from the start to the

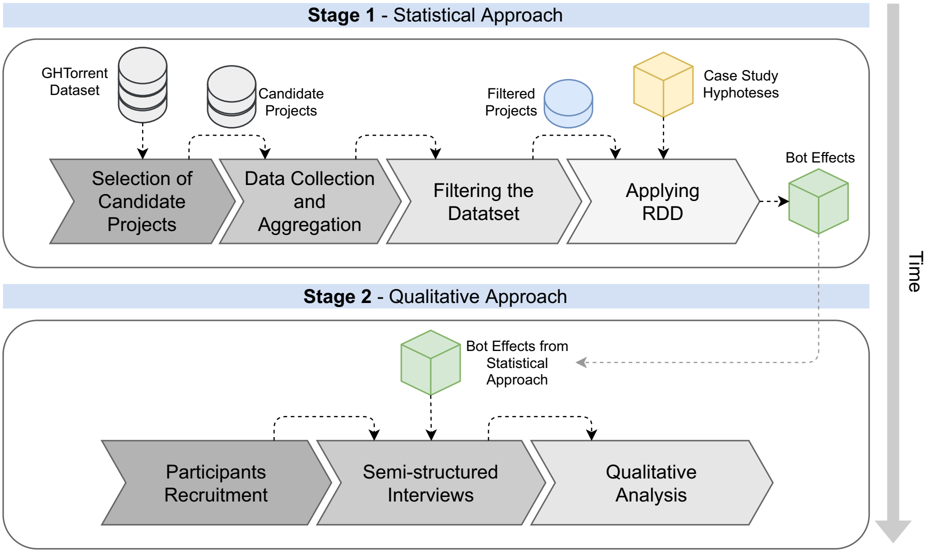

Stage 1 - Statistical Approach
GHTorrent Dataset
Candidate Projects
Filtered Projects
Case Study Hyphoteses
Bot Effects
Selection of Candidate Projects
Data Collection and Aggregation
Filtering the Datatset
Applying RDD
Stage 2 - Qualitative Approach
Bot Effects from Statistical Approach
Participants Recruitment
Semi-structured Interviews
Qualitative Analysis
Time

Fig. 7 Main research design overview

end of our observation period for each project (24 months). The *intervention* variable is a binary value used to indicate whether the time j occurs before ($intervention = 0$) or after ($intervention = 1$) adoption event. The *time_after_intervention* variable counts the number of months at time j since the bot adoption, and the variable is set up to 0 before adoption.

The $controls_i$ variables enable the analysis of bot adoption effects, rather than confounding the effects that influence the dependent variables. For observations before the intervention, holding controls constant, the resulting regression line has a slope of β, and after the intervention it has an slop of $\beta + \delta$. Further, the size of the intervention effect is measured as the difference equal to γ between the two regression values of y_i at the moment of the intervention.

Considering that we are interested in the effects of code review bots on the monthly trend of the number of pull requests, number of comments, time-to-close pull requests, and number of commits over a pull request, and all these for both merged and non-merged pull requests, we fitted eight models (2 cases × 4 variables). To balance false-positives and false-negatives, we report the corrected p-values after applying multiple corrections using the method of Benjamini and Hochberg (1995). We implemented the RDD models as a mixed-effects linear regression using the R package *lmerTest* (Kuznetsova et al. 2017).

To capture project-to-project and language-to-language variability, we modeled *project name* and *programming language* as random effects (Gałecki and Burzykowski 2013). By modeling these features as random effects, we can account for and explain different behaviors observed across projects or programming languages (Zhao et al. 2017). We evaluate the model fit using *marginal* (R^2_m) and *conditional* (R^2_c) scores, as described by Nakagawa and Schielzeth (2013). The R^2_m can be interpreted as the variance explained by the fixed effects alone, and R^2_c as the variance explained by the fixed and random effects together.

In mixed-effects regression, the variables used to model the intervention along with the other fixed effects are aggregated across all projects, resulting in coefficients useful for interpretation. The interpretation of these regression coefficients supports the discussion of the intervention and its effects, if any. Thus, we report the significant coefficients ($p < 0.05$)

in the regression as well as their variance, obtained using ANOVA. In addition, we *log* transform the fixed effects and dependent variables that have high variance (Sheather 2009). We also account for multicollinearity, excluding any fixed effects for which the variance inflation factor (VIF) is higher than 5 (Sheather 2009).

3.2.1 Selection of Candidate Projects

To identify open-source software projects hosted on GitHub that at some point had adopted a code review bot, we queried the GHTorrent dataset (Gousios and Spinellis 2012) and filtered projects in which at least one pull request comment was made by one of the code review bots identified by Wessel et al. (2018). Following the method used by Zhao et al. (2017) to assemble a time series, we considered only those projects that had been active for at least one year before and one year after the bot adoption. We found 4,767 projects that adopted at least one of the four code review bots identified by Wessel et al. (2018) (ansibot, elasticmachine, codecov-io, coveralls). For each project, we collected data on all its merged and non-merged pull requests. By analyzing these projects we noticed that 220 of them adopted both codecov-io and coveralls, while the other 4,547 adopted only one of the code reviews bots (coveralls: 3,269; codecov-io: 1,270; elasticmachine: 5; ansibot: 3).

3.2.2 Data Collection and Aggregation

Similar to the exploratory case study (see Section 2), we aggregated the project data in monthly time frames and collected the four variables we expected to be influenced by the introduction of the bot: number of merged and non-merged pull requests, median number of comments, median time-to-close pull requests, and median number of commits. All these variables were computed over pull requests that have been merged and non-merged in a time frame.

We also collected six control variables, using the GHTorrent dataset (Gousios and Spinellis 2012):

Project name: the name of the project, used to identify the project on GitHub. We accounted for the fact that different projects can lead to different contribution patterns. We used the project name as a random effect.

Programming language: the primary project programming language as automatically determined and provided by GitHub. We considered that projects with different programming languages can lead to different activities and contribution patterns (Zhao et al. 2017; Cassee et al. 2020). We used programming language as a random effect.

Time since the first pull request: in months, computed since the earliest recorded pull request in the entire project history. We use this to capture the difference in adopting the bot earlier or later in the project life cycle, after the projects started to use pull requests (Zhao et al. 2017; Cassee et al. 2020).

Total number of pull request authors: as a proxy for the size of the project community, we counted how many contributors submitted pull requests to the project.

Total number of commits: as a proxy for the activity level of a project, we computed the total number of commits since the earliest recorded commit in the entire project history.

Number of pull requests opened: the number of contributions (pull requests) received per month by the project. We expected that projects with a high number of contributions also observe a high number of comments, latency, commits, and merged and non-merged contributions.

3.2.3 Filtering the Final Dataset

After excluding the period of instability (30 days around the adoption), we inspected the dataset and found 223 projects with no comments authored by any of the studied bots. We manually checked 30% of these cases and concluded that some projects only added the bot for a testing period and then disabled it. We removed these 223 projects from our dataset.

We also checked the activity level of the candidate projects, since many projects on GitHub are inactive (Gousios et al. 2014). We excluded from our dataset projects without at least a six month period of consistent pull request activity during the one-year period before and after bot adoption. After applying this filter, a set of 1,740 GitHub software projects remained. To ensure that we observed the effects of each bot separately, we also excluded from our dataset 78 projects that adopted more than one of the studied bots and 196 projects that used non-code review bots. In addition, we checked the activity level of the bots on the candidate projects to remove projects that disabled the bot during the analyzed period. We then excluded 272 projects that had not received any comments during the previous four months. After applying all filters, 1,194 GitHub software projects remained. Table 1 shows the number of projects per bot. All of these four bots perform similar tasks on pull requests—providing comments on pull requests about code coverage.

3.3 Stage 2—Qualitative Approach

As aforementioned, we also applied a qualitative approach aimed to understand the effects evidenced by the statistical approach from the practitioners' perspective. In the following, we describe the participants recruitment, semi-structured interview procedures, and the qualitative analysis.

3.3.1 Participants Recruitment

In this study, we employed several strategies to recruit participants. First, we advertised the interview on social media platforms frequently used by developers (Singer et al. 2014; Storey et al. 2010; Aniche et al. 2018), including Twitter, Facebook, and Reddit. We also manually searched the projects that were part of the statistical analysis for pull requests explicitly installing or (re)configuring the analyzed bots. We added a comment on some of these pull requests to invite the pull request author to the interview. We also sent emails to personal contacts who we knew had experience with these bots. In addition, we asked participants to refer us to other qualified participants.

We continued recruiting participants till we came to an agreement that the last three interviews had not provided any new findings. According to Strauss and Corbin (1997), sampling can be discontinued once the data collection no longer unveils new information. Additionally, the size of our participant set is in line with the anthropology literature, which

Table 1 An overview of the studied bots

Bot name	GitHub user	# of projects
Ansible's issue bot	ansibot	1
Elastic Machine	elasticmachine	3
Codecov	codecov-io	460
Coveralls	coveralls	730
Total of 1,194 under study		

Table 2 Demographics of interviewees

P1	4–5	✓		North America	Man
P2	Over 10	✓	✓	North America	Man
P3	4–5	✓	✓	Europe	Man
P4	3	✓	✓	Europe	Man
P5	4–5	✓	✓	Europe	Woman
P6	Over 10	✓	✓	North America	Man
P7	5–10	✓	✓	Europe	Man
P8	4-5	✓	✓	Europe	Man
P9	1		✓	Europe	Man
P10	4–5	✓	✓	South America	Man
P11	4–5	✓		South America	Man
P12	Over 10		✓	South America	Man

mentions that a set of 10-20 knowledgeable people is sufficient to uncover and understand the core categories in any study of lived experience (Bernard 2017).

3.3.2 Participants Demographics

In total, we interviewed 12 open-source developers experienced with code review bots—identified here as P1–P12. Out of these twelve participants, one is an open-source maintainer, two are contributors, and the other nine are both maintainers and contributors. In addition, participants are geographically distributed across Europe (≃50%), North America (≃25%), and South America (≃25%). Snowballing was the origin of five of our participants. Personal contacts was the origin of four of our participants. The advertisements on social media were the origin of the other three interviews. Table 2 presents the demographic information of the interviewees.

3.3.3 Semi-structured Interviews

We conducted *semi-structured* interviews, comprising open- and closed-ended questions designed to elicit foreseen and unexpected information and enable interviewers to explore interesting topics that emerged during the interview (Hove and Anda 2005). Before each interview, we shared a consent form with the participants asking for their agreement. By participants' requests, one interview (P11) was conducted via email. The other eleven interviews were conducted via video calls. The participants received a 25-dollar gift card as a token of appreciation for their time.

We started the interviews with a short explanation of the research objectives and guidelines, followed by demographic questions to capture the familiarity of the interviewees with open-source development and code review bots. We then described to the interviewee the study we conducted and the main findings from the statistical approach and asked the developers to conjecture about the reasons for the effects we observed:

Q1. After adopting a code review bot there are more merged pull requests, less communication between developers, fewer rejected pull requests, and faster rejections. We are intrigued about these effects and would like to hear thoughts from developers who actually use these bots. Could you conjecture the reasons why this happens?

We follow-up this question with more specific questions when participants have not mentioned reasons for any of the four observed effects. Afterwards, we asked two additional questions:

Q2. Have you observed these effects in your own project?
Q3. What other effects did you observe in your project and attribute to the introduction of the code review bot?

The detailed interview script is publicly available.[5] Each interview was conducted remotely by the first author of this paper and lasted, on average, 35 min.

3.3.4 Qualitative Analysis of Interviews

Each interview recording was transcribed by the first author of this paper. We then analyzed the interview transcripts by applying open and axial coding procedures (Strauss and Corbin 1998; Stol et al. 2016) throughout multiple rounds of analysis. We started by applying open coding, whereby we identified the reasons for bots' effects. To do so, the first author of this paper conducted a preliminary analysis, identifying the main codes. Then, the first author discussed with fourth and fifth authors the coding in weekly hands-on meetings. These discussions aimed to increase the reliability of the results and mitigate bias (Strauss and Corbin 2007; Patton 2014). Afterwards, the first author further analyzed and revised the interviews to identify relationships between concepts that emerged from the open coding analysis (axial coding). During this process, we employed a constant comparison method (Glaser and Strauss 2017), wherein we continuously compared the results from one interview with those obtained from the previous ones. The axial coding resulted on grouping the participants' answers into five categories.

For confidentiality reasons, we do not share the interview transcripts. However, we made our complete code book publicly available. The code book includes the all code names, descriptions, and examples of quotes.

4 Main Study Results

In the following, we report the results of our study by research question.

4.1 Effects of Code Review Bot Adoption (RQ1)

In this section, we discuss the effects of code review bot adoption on project activities along four dimensions: (i) accepted and rejected pull requests, (ii) communication, (iii) pull request resolution efficiency, and (iv) modification effort.

4.1.1 Effects in Merged and Non-merged Pull Requests

We start by investigating the effects of bot adoption on the number of merged and non-merged pull requests. From the exploratory case study, we hypothesized that the use of code review bots is associated with an increase in the number of monthly merged pull requests and a decrease in the number of monthly non-merged pull requests. We fit two mixed-effect

[5] https://doi.org/10.5281/zenodo.4618498

Table 3 The effects of code review bots on PRs. The response is **log(number of merged/non-merged PRs)** per month

	Merged pull requests		Non-merged pull requests	
	Coefficients	SS	Coefficients	SS
Intercept	−0.262***		−0.574***	
TimeSinceFirstPullRequest	0.00004**	4.3	−0.0001***	2.4
log(TotalPullRequestAuthors)	−0.094***	171.8	0.086***	775.7
log(TotalCommits)	0.042***	484.0	0.068***	428.6
log(OpenedPullRequests)	0.494***	8227.1	0.388***	4958.5
log(PullRequestComments)	0.433***	2954.3	0.389***	2341.0
log(PullRequestCommits)	0.272***	721.0	0.165***	255.5
time	0.004***	203.2	−0.004***	376.1
interventionTrue	0.095***	16.8	−0.163***	48.4
time_after_intervention	0.004**	1.7	−0.004**	1.6
Marginal R^2	0.68			0.67
Conditional R^2	0.75			0.74

*** $p < 0.001$, ** $p < 0.01$, * $p < 0.05$. SS stands for "Sum of Squares"

Time series predictors in **bold**

RDD models, as described in Section 3.2. For these models, the *number of merged/non-merged pull requests* per month is the dependent variable. Table 3 summarizes the results of these two RDD models. In addition to the model coefficients, the table also shows the SS, with a variance explained for each variable. We also highlighted the time series predictors *time*, *time after intervention*, and *intervention* in **bold**.

Analyzing the model for merged pull requests, we found that the fixed-effects part fits the data well ($R^2_m = 0.68$). However, considering $R^2_c = 0.75$, variability also appears from project-to-project and language-to-language. Among the fixed effects, we observe that the number of monthly pull requests explains most of the variability in the model. As expected, this indicates that projects receiving more contributions tend to have more merged pull requests, with other variables held constant.

Furthermore, the statistical significance of the time series predictors indicates that the adoption of code review bots affected the trend in the number of merged pull requests. Observing the *time* coefficient, we note an increasing trend before adoption. There is a statistically significant discontinuity at adoption, since the coefficient for *intervention* is statistically significant. Further, there is a positive trend after adoption (see *time after intervention*) and the sum of the coefficients for *time* and *time after intervention* is positive; thus, indicating that the number of merged pull requests increased even faster after bot adoption.

Similar to the previous model, the fixed-effect part of the non-merged pull requests model fits the data well ($R^2_m = 0.67$), even though a considerable amount of variability is explained by random effects ($R^2_c = 0.74$). We note similar results on fixed effects: projects receiving more contributions tend to have more non-merged pull requests. All the three time-series predictors for this model are statistically significant, showing a measurable effect of the code review bot's adoption on the time to review and accept a pull request. The *time* coefficient shows a decreasing trend before adoption, *intervention* coefficient reports a statistically significant discontinuity at the adoption time, and there is a slight acceleration

Table 4 The effect of code review bots on pull request comments. The response is **log(median of comments)** per month

	Merged pull requests		Non-merged pull requests	
	Coefficients	SS	Coefficients	SS
Intercept	−0.096***		−0.123***	
TimeSinceFirstPullRequest	0.00000	20.0	−0.00002*	24.4
log(TotalPullRequestAuthors)	0.053***	163.6	0.069***	621.1
log(TotalCommits)	−0.014***	36.6	−0.009**	106.0
log(OpenedPullRequests)	0.079***	1002.8	0.072***	1362.9
log(TimeToClosePullRequests)	0.093***	3239.7	0.101***	4615.5
log(PullRequestCommits)	0.093***	55.0	0.123***	119.4
time	−0.001	1.0	−0.001	7.2
interventionTrue	0.023**	0.8	−0.025***	1.1
time_after_intervention	−0.002*	0.5	0.0001	0.0
Marginal R^2	0.50			0.66
Conditional R^2	0.56			0.70

*** $p < 0.001$, ** $p < 0.01$, * $p < 0.05$. SS stands for "Sum of Squares"

Time series predictors in **bold**

after adoption in the decreasing time trend seen before adoption observed since the sum of the coefficients for *time* and *time after intervention* is negative.

Therefore, based on models for merged and non-merged pull requests, we confirm both $\mathbf{H}_{1.1}$ and $\mathbf{H}_{1.2}$.

Effects in Merged and Non-merged Pull Requests. Overall, there are more monthly merged pull requests and fewer monthly non-merged pull requests after adopting a code review bot.

4.1.2 Effects on Developers' Communication

In the exploratory case study, we hypothesized that bot adoption increases monthly human communication on pull requests for both merged and non-merged pull requests. To statistically investigate this, we fit one model to merged pull requests and another to non-merged ones. The *median of pull request comments* per month is the dependent variable, while *number of monthly pull requests*, *median of time-to-close pull requests*, and *median of pull request commits* are independent variables. Table 4 shows the results of the fitted models.

Considering the model of comments on merged pull requests, we found that the model taking into account only fixed effects ($R^2_m = 0.50$) fits the data well. However, there is also variability from the random effects ($R^2_c = 0.56$). We observe that *time-to-close pull requests explains the largest amount of variability in the model*, indicating that communication during the pull request review is strongly associated with the time to merge it. Regarding the bot effects, there is a discontinuity at adoption time, followed by a statistically significant decrease after the bot's introduction.

As above, the model of non-merged pull requests fits the data well ($R^2_m = 0.66$) and there is also variability explained by the random variables ($R^2_c = 0.70$). This model also

Table 5 The Effect of Code Review bots on time-to-close PRs. The response is **log(median of time-to-close PRs)** per month

	Merged pull requests		Non-merged pull requests	
	Coefficients	SS	Coefficients	SS
Intercept	0.377**		0.221	
TimeSinceFirstPullRequest	0.0002**	452	0.00001	891
log(TotalPullRequestAuthors)	0.208***	2186	0.166***	21320
log(TotalCommits)	−0.145***	824	−0.057**	4770
log(OpenedPullRequests)	0.120***	34444	0.240***	50376
log(PullRequestComments)	2.472***	117571	3.326***	176312
log(PullRequestCommits)	2.275***	47117	1.721***	26733
time	0.027***	3007	0.012**	56
interventionTrue	0.256***	128	−0.056	9
time_after_intervention	0.009	6	−0.028***	66
Marginal R^2	0.61			0.69
Conditional R^2	0.67			0.72

*** $p < 0.001$, ** $p < 0.01$, * $p < 0.05$. SS stands for "Sum of Squares"

Time series predictors in **bold**

suggests that communication during the pull request review is strongly associated with the time to reject the pull request. Table 4 shows that the effect of bot adoption on non-merged pull requests differs from the effect on merged ones. The statistical significance of the *intervention* coefficient indicates that the adoption of code review bots slightly affected communication; however, there is no bot effect in the long run.

Since our model for merged pull requests shows a decrease in the number of comments after bot adoption, we rejected $\mathbf{H}_{2.1}$. Still, given that our model for non-merged pull requests could not observe any statistically significant bot effect as time passes, we cannot accept $\mathbf{H}_{2.2}$.

Effects in Communication. On average, there is less monthly communication on merged pull requests after adopting a code review bot. However, the monthly communication on non-merged pull requests does not change as time passes.

4.1.3 Effects in Pull Request Resolution Efficiency

In the exploratory case study, we found that the monthly time to close pull requests increased after bot adoption. Next, we fitted two RDD models, for both merged and non-merged pull requests, where *median of time to close pull requests* per month is the dependent variable. The results are shown in Table 5.

Analyzing the results of the effect of code review bots on the latency to merge pull requests, we found that combined fixed-and-random effects fit the data better than the fixed effects only ($R_c^2 = 0.67$ vs $R_m^2 = 0.61$). Although several variables affect the trends of pull request latency, communication during the pull requests is responsible for most of the variability in the data. This indicates the expected results: the more effort contributors expend discussing the contribution, the more time the contribution takes to merge. The number of commits also explains the amount of data variability, since a project with many changes needs more time to review and merge them. Moreover, we observe an increasing trend before adoption, followed by a statistically significant discontinuity at adoption. After adoption, however, there is no bot effect on the time to merge pull requests since the *time_after_intervention* coefficient is not statistically significant.

Turning to the model of non-merged pull requests, we note that it fits the data well ($R^2_m = 0.69$), and there is also a variability explained by the random effects ($R^2_c = 0.72$). As above, communication during the pull requests is responsible for most of the variability encountered in the results. In this model, the number of received contributions is important to explain variability in the data—projects with many contributions need more time to review and reject them. The effect of bot adoption on the time spent to reject pull requests differs from the previous model. Regarding the time series predictors, the model did not detect any discontinuity at adoption time. However, the positive trend in the latency to reject pull requests before bot adoption is reversed toward a decrease after adoption.

Thus, since we could not observe statistically significant bot effects as time passes, we cannot confirm $\mathbf{H}_{3.1}$. Further, as the model of non-merged pull requests shows a decrease in the monthly time to close pull requests, we reject $\mathbf{H}_{3.2}$.

Effects in PR Resolution Efficiency. After adopting the code review bot, on average less time is required from maintainers to review and reject pull requests. However, the time required to review and accept a pull request does not change after code review bot adoption.

4.1.4 Effects in Commits

Finally, we studied whether code review bot adoption affects the number of commits made before and during pull request review. Our hypothesis is that the monthly number of commits increases with the introduction of code review bots. Again, we fitted two models for merged and non-merged pull requests, where the *median of pull request commits* per month is the dependent variable. The results are shown in Table 6.

Analyzing the model of commits on merged pull requests, we found that the combined fixed-and-random effects ($R^2_c = 0.48$) fit the data better than the fixed effects ($R^2_m = 0.34$), showing that most of the explained variability in the data is associated with project-to-project and language-to-language variability, rather than with the fixed effects. The statistical significance of the *intervention* coefficient indicates that the adoption of code review bots affected the number of commits only at the moment of adoption. Additionally, from Table 6, we can also observe that the number of pull request comments per month explains most of the variability in the result. This result suggests that the more comments there are, the more commits there will be, as discussed above.

Investigating the results of the non-merged pull request model, we found that the model fits the data well and that the random effects are again important in this regard. We also observe from Table 6 that the adoption of a bot is not associated with the number of commits on non-merged pull requests, since *intervention* and *time_after_intervention* coefficients are not statistically significant.

Based on models for merged and non-merged pull requests, we could not observe statistically significant effects of bot adoption. Therefore, we cannot confirm both $\mathbf{H}_{4.1}$ and $\mathbf{H}_{4.2}$.

Effects in Commits. After adopting a code review bot, the monthly trend in the median of pull request commits does not change for both merged and non-merged pull requests.

4.2 Developers' Perspective on the Reasons for the Observed Effects (RQ2)

As explained in Section 3.3, we presented to open-source developers the main findings of our statistical approach: "*After adopting a code review bot there are more merged pull requests, less communication between developers, fewer rejected pull requests, and faster rejections*." We asked them to conjecture on the possible reasons for each of these results.

We grouped the participants' answers into 5 categories, as can be seen in Table 7. We associate one of the effects with its correspondent reasons whenever participants explicitly mentioned this relationship. We also added a mark (✓) to highlight which effects are explained by each one of the reasons, according to the participants' responses.

More visibility and transparency of the contribution state. Most of the participants claimed that when a project has bots that provide detailed information on code quality metrics, especially in the sense of coverage metrics, both maintainers and contributors can more quickly gain a general idea of the quality of the contributions. As stated by P6: "*bots are able to raise visibility, both for the contributor and for the maintainer. They can make it more clear more quickly the state of that contribution*." More than obtaining clarity on the quality of the code, it is also easy for maintainers to verify whether the pull request contributors will improve their contribution toward achieving acceptance.

Table 6 The effect of code review bots on pull request commits. The response is **log(median of Pull Request commits)** per month

	Merged pull requests		Non-merged pull requests	
	Coefficients	SS	Coefficients	SS
Intercept	0.358***		0.063	
TimeSinceFirstPullRequest	0.0001***	0.30	0.00002	5.7
log(TotalPullRequestAuthors)	−0.144***	0.02	−0.058***	202.2
log(TotalCommits)	0.017***	74.04	0.028***	171.9
log(OpenedPullRequests)	0.163***	1513.60	0.125***	1502.9
log(PullRequestComments)	0.520***	2375.74	0.600***	3306.3
time	0.001	138.60	−0.003**	8.7
interventionTrue	0.137***	33.57	0.003	0.0
time_after_intervention	0.001	0.05	0.001	0.1
Marginal R^2	0.34			0.42
Conditional R^2	0.48			0.50

*** $p < 0.001$, ** $p < 0.01$, * $p < 0.05$. SS stands for "Sum of Squares"

Time series predictors in **bold**

Thus, they conjecture that all bot effects we found during the statistical analysis might be explained by this enhanced feedback given by the bot.

As soon as contributors submit their pull requests, the code review bot posts a detailed comment regarding the code coverage. In the P7 experience, the "*immediate feedback of the quality of [code coverage] on the pull request*" is closely related to the increasing acceptance rate. If the pull request does not affect code coverage in a negative way, then maintainers are able to "*much more quickly judge whether or not it's a reasonable request*" (P4). On the other side of the spectrum, if the pull requests fail the tests and decreases the coverage, then the maintainers "*will not bother with that pull request at all, and just reject it*" (P4). Also maintainers "*are more inclined to directly reject the pull request*" since it does not respect the rules imposed by the project. In some cases, maintainers expect that the contributor will take an action based on the bot comments, as explained by P6: "*if [contributors] are not following up and resolving the issue, it makes it more clear to the maintainer that it's not an acceptable contribution.*"

Participants also recognize that these bots are usually "*pretty good at explaining very precisely*" (P2) and not merely stating that "*[maintainers] will not accept the pull request*" (P2) without further explanation. For example, if the coverage decreased, the bot will post "*your pull request dropped the test coverage from 95 to 94%. And these are the lines you edit that are not covered. So, please add tests to cover these specific lines.*" (P2), which according to P2 is extremely useful for a contributor. According to P1, for example, the visibility of the bot comments helps maintainers to make sure contributors understand why the pull request has been rejected without the necessity of engaging in a long discussion: "*now the maintainer can just point at it and be like 'you didn't pass the status check, because you didn't write tests.' It is more obvious*".

More confidence in the process in place. According to the participants, one of the reasons for more pull requests being merged after the code review bot introduction is that these bots act as quality gatekeepers. For example, P1 mentioned that *"by having other metrics, like code coverage, to be able to say 'Great! I know that at least a test has been written for that line of code', there is some sort of gatekeeping."* Besides the effect of merging more pull requests, participants also mentioned another effect: "*accepting code*

Table 7 Main reasons for the findings from the RDD models

Reason	#	Explains			
		More merged PRs	Fewer comments	Fewer rejected PRs	Faster rejections
More visibility and transparency of the contribution state	8	✓	✓	✓	✓
More confidence in the process in place	8	✓	✓		✓
Bot feedback changes developers' discussion focus	8		✓		
Bot feedback pushes contributors to take an action	5	✓		✓	
Bot feedback perceived as noise	2		✓		

contributions can be much, much faster" (P2). Basically, code review bots are used as a way to achieve "*automatic verification*" (P7). According to P7, if the bots confirm that the change is correct, then "*the developer is more convinced that the change is useful and valid.*" In the opposite way, if the bots shows that the change is incorrect, the pull request will be rejected faster, as it does not require "*human interaction to arrive at this conclusion*" (P7), which implies less communication between developers. Furthermore, P4 also relates the confidence in the bot as one of the reasons for less communication between developers: *the fact that there is less communication between the contributors and maintainers might be an effect that we can get a bit overdependent on bots, in the sense you trust them too much.*" Therefore, since maintainers trust the bots' feedback, they "*ask fewer questions*" (P2).

Bot feedback changes developers' discussion focus. Participants recurrently mentioned that bot comments enabled them to focus on other high-priority discussions, which led to a decrease in the communication between the project maintainers and contributors on pull requests. To some extent this decrease occurs since *it's not necessary anymore, because a lot of that [comments] are [already] handled automatically by the bots*" (P4). In P3's experience, maintainers "*talk more for new developers, to text them usually things like 'Add new test please' and then [maintainers] don't have to [make] that kind of comment[] anymore. That's why there's less communication.*"

Moreover, when receiving non-human feedback, contributors are less likely to start a broader discussion about the viability or necessity of software testing, as explained by P2: "*once you have set up the bots, and it is automated, people are less likely to argue about it, which is just a nice effect of bots. Especially for bots that kind of point out failures. I think it's good to have that from bots, and not from people.*" There are some exceptions, however, when contributors experienced an increase in communication incurred by the bot comments, especially when they do not understand how they might increase the coverage rate. As posed by P9: *In my experience, it causes a longer discussion, because then I have to talk to the engineers like 'hey, what kind of a test should I add such as coveralls passes?"'*

Bot feedback pushes contributors to take an action. Also related to the transparency introduced by the bot comments, and in line with the idea of code review bots as quality gatekeepers, these bots lead developers to take an action: "*It gives me clear instructions on what I have to do to resolve it. So, I'm very likely to act on it*" (P2). These bots protect developers from reducing the code's coverage. Therefore, developers would consider either closing the pull request, if it is not worth their time, or following up with the necessary changes: *you have this systematic check that says 'okay, that's not good.' And then the developer is saying, 'okay, it won't be accepted if I don't provide the test'* "(P3).

Bot feedback perceived as noise. Although less recurrent, participants mentioned that in some cases bot comments might be perceived as noise by developers, which disrupts the conversation in the pull request. On the one hand, "*comments from code coverage bots tend to give you more visibility and provide more context[]*" (P6). On the other hand, developers complain about the noise these comments introduce to the communication channel. According to P7, the repetitive comments of code coverage bots are "*disrupting the conversation*", since "*if you have to develop a certain conversation and you have a bot message, this could have a negative impact on the conversation.*" One of the consequences of this noise incurred by the repetitive bot comments is that "*[developers] pay less attention to it*" (P7), impacting the developers communication.

We also asked developers whether they have seen the observed effects on their own projects, and what are the other effects they attribute to the code review bot adoption. The most recurrent (8) observed effect was less communication. As stated by P10: "*I remember one of the maintainers saying 'the tests are missing here.' She always had to post that comment. Then, we adopted the bot to comment on the coverage and had no need for her to comment anymore.*" Also, 6 participants observed fewer pull requests rejections and faster rejections, and 5 participants have observed more merged pull requests. Finally, developers did not attribute any other effect to the bot introduction.

Summary of reasons. Project maintainers and contributors reported several reasons for more merged pull requests, fewer comments, and fewer and faster rejections. According to them, bot comments help them to understand the state and quality of the contribution, making maintainers more confident to merge pull requests, which also changes the focus of developer discussions.

5 Discussion

Adding a code review bot to a project often represents the desire to enhance feedback about the contributions, helping contributors and maintainers, and achieving improved interpersonal communication, as already discussed by Storey and Zagalsky (2016). Additionally, code review bots can guide contributors toward detecting change effects before maintainers triage the pull requests (Wessel et al. 2018), ensuring high-quality standards. In this paper, following the study of Wessel et al. (2018), we focused on monthly activity indicators that are not primarily related to bot adoption, but might be impacted by it. We found that the bot adoption has a statistically significant effect on a variety of activity indicators.

According to the regression results, the monthly number of merged pull requests increased, even faster, after the code review bot adoption. In addition, the number of non-merged pull requests continued to decrease, even faster, after bot adoption. These models showed that after adopting the bot, maintainers started to deal with an increasing influx of contributions ready to be further reviewed and integrated into the codebase. Also, these findings confirm the hypothesis we formulated based on the exploratory case study. According to our participants, the increase in the monthly number of merged pull requests, as well as the decrease in the monthly number of non-merged one, are explained by the transparency introduced by the bot feedback. Contributors started to have faster and clearer feedback on what they needed to do to have their contribution accepted. Further, participants also mentioned that contributors have been pushed to enhance their pull requests based on bot feedback.

In addition, we noticed that just after the adoption of the code review bot the median number of comments slightly increased for merged pull requests. The number of comments on these pull requests could increase due to contributions that drastically reduced the coverage, stimulating discussions between maintainers and contributors. This can happen especially at the beginning of bot adoption, since contributors might be unfamiliar with bot feedback. After that initial period, we found that the median number of comments on merged pull requests decreased each month. According to our participants, less communication could be explained by the transparency and confidence developers gain from bot feedback. Also, developers mentioned that after bot adoption the focus of the developers discussion changed, since there is no need for certain discussions related to coverage. Considering non-merged pull requests, there is no significant change in the number of comments as time passes. These results differ from the case study results, indicating that individual projects reveal different results, which are likely caused by other project-specific characteristics.

From the regression results, we also noticed an increase in the time spent to merge pull requests just after bot adoption. It makes sense from the contributors' side, since the bot introduces a secondary evaluation step. Especially at the beginning of the adoption, the code review bot might increase the time to merge pull requests due to the need to learn how to meet all bot requirements and obtain a stable code. Maintainers might also deal with an increase in the volume of contributions ready to review and merge, impacting the time spent to review all of them. Further, the regression model shows a decrease in the time spent to review and reject pull requests. Overall, according with our participants it indicates that after the bot adoption maintainers stopped expending effort on pull requests that were not likely to be integrated into the codebase.

As we found in the model of commits on merged pull requests, just after the adoption of the bot the median number of pull request commits increased. The bot provides immediate feedback in terms of proof of failure, which can lead contributors to submit code modifications to change the bot feedback and have their contribution accepted. Overall, the regression models reveal that the monthly number of commits did not change for both merged and non-merged pull requests as time passed. These results differ from the case study results. Nevertheless, even if there is an increase in the number of commits reported in the case study, overall the monthly number of commits are quite stable. For example, for CakePHP it varies from 1 to 2 for merged pull requests, and 1 to 4 for non-merged pull requests. Additionally, in the main study, we account for control variables, rather than analyzing the monthly number of commits interdependently. As presented in Section 4.1.4, for example, the number of comments on pull requests explains the largest amount of variability in these models, indicating that the number of commits is strongly associated with the communication during the pull request review.

6 Implications and Future Work

In the following, we discuss implications and future work for researchers and practitioners in light of our results and related literature.

6.1 Implications for Project Members

Projects need to make informed decisions on whether to adopt code review bots (or software bots in general) and how to use them effectively. We found that the dynamics of pull requests changed following the adoption of code review bots. Hence, besides understanding the effects on code quality, practitioners and open-source developers should become aware of other consequences of bot adoption and take countermeasures to avoid the undesired ones. For example, our statistical findings show a decrease in the amount of discussion between humans after the bot's introduction. According to developers, this effect is likely to be explained by more visibility and transparency, or the changes in the focus of the discussions. However, developers might also perceive bot comments as noise, which disrupts the conversation in the pull request. Thus, project members should be aware of these possible side effects since noise is a recurrent problem when adopting bots on pull requests (Wessel et al. 2021). For instance, they might consider re-configuring the bot to avoid some behaviors, such as high frequency of actions—bots performing repetitive actions, such as creating numerous pull requests and leaving dozen of comments in a row—and comments verbosity—bots providing comments with dense information.

6.2 Implications for Researchers

For researchers interested in software bots, it is important to understand the role of code review bots in the bot landscape. It is important to understand how such bots affect the interplay of developers in their effort to develop software, and our study provides the first step in this direction. Considering that bot output is mostly text-based, how bots present content can highly impact developers' perceptions (Liu et al. 2020; Chaves and Gerosa 2020). Additional effort is necessary to investigate how the developers' cognitive styles (Vorvoreanu et al. 2019; Mendez et al. 2018) might influence the way developers interpret the bot comments' content. In this way, future research can investigate how people with different cognitive styles handle bot messages and learn from them. Other social characteristics of the bots can also be investigated in this context (Chaves and Gerosa 2020). Future research can lead to a set of guidelines on how to design effective messages for different cognitive styles and developer profiles. Further, developers complain about the information overload caused by repetitive bot behavior on pull requests, which has received some attention from the research community (Wessel et al. 2018; Wessel and Steinmacher 2020; Erlenhov et al. 2016), but remains a challenging problem. In fact, there is room for improvement on human-bot collaboration on social coding platforms. When they are overloaded with information, teams must adapt and change their communication behavior (Ellwart et al. 2015). Therefore, there is also an opportunity to investigate changes in developers' behavior imposed by the effects of information overload. Additional research can also investigate how to use code reviews bots to support the training of new software engineers (Pinto et al. 2017).

Previous work by Wessel et al. (2018) has already mentioned that bot support for newcomers is both challenging and desirable. In a subsequent study, Wessel et al. (2020a) reported that although bots could make it easier for some newcomers to submit a high-quality pull request, bots can also provide newcomers with information that can lead to rework, discussion, and ultimately dropping out from contributing. It is reasonable to expect that newcomers who receive friendly feedback will have a higher engagement level and thus sustain their participation on the project. Hence, future research can help bot designers by providing guidelines and insights on how to support new contributors. Additional effort is also necessary to investigate the impact of code review bots' feedback for newcomers, who already face a variety of barriers (Balali et al. 2018; Steinmacher et al. 2015).

6.3 Implications for Code Review Bots

To avoid side effects of using code review bots, such as noise, bots should provide mechanisms to enable better configurable control over their actions, rather than just turn off bot comments. It is important to have easy mechanisms so project maintainers can turn off or pause a bot at any time. Further, these mechanisms need to be explicitly announced during bot adoption (e.g., noiseless configuration, preset levels of information). It is essential to provide a more flexible way for bots to interact, incorporating rich user interface elements to better engage users.

7 Related Work

In this section, we describe the studies related to the usage and impact of software bots. Further, we summarize works that employed regression discontinuity design (RDD) to account for the intervention effects on software development activities on GitHub.

7.1 Software Bots on Social Coding Platforms

Software bots are software applications that integrate their work with human tasks, serving as interfaces between users and other tools (Storey et al. 2017; Lebeuf et al. 2017), and providing additional value to human users (Lebeuf et al. 2019). Software bots frequently reside on platforms where users work and interact with other users (Lebeuf et al. 2018). On the GitHub platform, bots have user profiles to interact with the developers, executing well-defined tasks (Wessel et al. 2018).

Bots support social and technical activities in software engineering, including communication and decision-making (Storey and Zagalsky 2016). Bots are particularly relevant in social-coding platforms (Dabbish et al. 2012), such as GitHub, where the pull-based model (Gousios et al. 2014) offers several opportunities for community engagement, but at the same time increases the workload for maintainers (Gousios et al. 2016; Pinto et al. 2016). Open-source communities have been adopting bots to reduce the workload with a variety of automated repetitive tasks on GitHub pull requests (Wessel et al. 2018), including repairing bugs (Urli et al. 2018; Monperrus 2019), refactoring source code (Wyrich and Bogner 2019), recommending tools (Brown and Parnin 2019), updating dependencies (Mirhosseini and Parnin 2017), fixing static analysis violations (Carvalho et al. 2020; Serban et al. 2021), suggesting code improvements (Phan-udom et al. 2020), and predicting defects (Khanan et al. 2020).

Storey and Zagalsky (2016) and Paikari and van der Hoek (2018) highlight that the potentially negative impact of task automation through bots is being overlooked. Storey and Zagalsky (2016) claim that bots are often used to avoid interruptions to developers' work, but may lead to other, less obvious distractions. While previous studies provide recommendations on how to develop bots and evaluate bots' capabilities and performance, they do not draw attention to the impact of bot adoption on software development or how software engineers perceive the bots' impact. Since bots are seen as new team members (Monperrus 2019), we expected that bots would impact group dynamics in a way that differs from non-bot forms of automation.

Wessel et al. (2018) investigated the usage and impact of software bots to support contributors and maintainers with pull requests. After identifying bots on popular GitHub repositories, the authors classified them into 13 categories according to their tasks. Unlike Wessel et al. (2018), we focused on understanding the effects of a specific bot type, which is the most frequently used category of bots. In a preliminary study, Wessel et al. (2020a) surveyed 127 open source maintainers experienced in using code review bots. While maintainers report that bots satisfy their expectations regarding enhancing developers' feedback, reducing maintenance burden, and enforcing code coverage, they also perceived unexpected effects of having a bot, including communication noise, more time spent with tests, and newcomers' dropout. Our work extends this preliminary investigation by combining analysis of GitHub data with semi-structured interviews conducted with open-source developers. This study looks at how bots change the pull request dynamics and its reasons from practitioners' perspectives.

7.2 Using RDD to Access the Effects of Interventions on Software Development

In the software engineering domain, several researchers have been applying *Regression Discontinuity Design* (RDD) to model the effects of a variety of interventions on development activities over time. To understand the similarities between those studies, we conducted an extensive search for empirical works that employed RDD to investigate interventions in software development on GitHub in general. In Table 8 we summarize these studies, presenting an overview of what interventions have been used (e.g., bots, CI), what dependent variables have been studied, and what results have been obtained.

Zhao et al. (2017) introduced the RDD usage to study software development activities. Zhao et al. (2017) focused on the impact of the Continuous Integration (CI) tool's introduction on development practices. Conducting the statistical analysis on GitHub repositories, they found that adopting Travis CI leads to an increase in the number of merge commits, number of closed pull requests, and in pull request latency. With these results they also confirm earlier results about the benefits of CI, such as a better adherence to best practices. Meanwhile, Cassee et al. (2020) studied the effects of Travis CI on conserving developers' efforts during code review. Analyzing the pull requests' general comments and the review comments, which are associated with specific lines of code on the pull request, they found that the communication decreased after the CI adoption. At the same time, the trends in the commits after the creation of the pull requests remained unaffected. Also regarding CI, Guo and Leitner (2019) investigated the impact of its adoption on the delivery time of pull requests. They find no evidence of CI affecting the pull request delivery time in the studied projects.

In addition to the studies of CI, prior work has also investigated the impact of other automation tools designed to support developers during code review or while performing other repetitive tasks on pull requests. Kavaler et al. (2019), for example, investigated the impact of linters, dependency managers, and coverage reporter tools on GitHub projects across time. The results of applying RDD showed that tools are associated with a decrease in the monthly number of opened issues. Trockman et al. (2018) explored the impacts of the usage badges on GitHub repositories. They found that badges displaying the build status, test coverage, and up-to-dateness of dependencies are associated with more tests, more quality pull requests, and fresher dependencies. Kinsman et al. (2021) studied the effect of GitHub Action adoption by GitHub projects. The results revealed that introducing a GitHub Action leads to an increase in the number of rejected pull requests and a decrease in the commits in the merged pull requests. This differs from our results, which might be explained by the variety of tasks performed by the GitHub Actions in the study, and consequently their impacts on pull request activities.

Other studies have been investigating interventions that are not related to a tool adoption. For example, Zimmermann and Artís (2019) investigated the impact of switching from one bug tracker to another. They found that the switch induces an increase in issue reporting, particularly by the project core developers. Moreover, when moving from Bugzilla to GitHub, the communication between maintainers and contributors in the issues also increased. Moldon et al. (2020) focused on how developers' behavior was impacted by the removal of the daily activity streak counters from the user profile. The results show that the developer activity decreased on weekends compared to weekdays. According to the authors, the activity counters were influencing developers to contribute on days they would have otherwise rested. Walden (2020) employed RDD to assess the impact of a major security event

Table 8 Literature review related to Software development and RDD on GitHub

Study	Intervention	Common variables								
		Comments		Commits		Issues	PR latency		Pull requests	
Zhao et al. (2017)	Travis CI (adoption)			🟩 Merge commits			🟩 Time to close PRs		🟩 Closed PRs	
Cassee et al. (2020)	Travis CI (adoption)	🟥 PR comments	🟥 PR review comments	🟨 Commits after create the PR						
Guo and Leitner (2019)	Travis CI (adoption)						🟨 Time to deliver PRs			
Kavaler et al. (2019)	Quality assurance tools (adoption)					🟥 Opened				
Wessel et al. (2020b)	Code review bots (adoption)	🟥 Merged PRs	🟨 Non-merged PRs	🟨 Merged PRs	🟨 Non-merged PRs		🟥 Time to reject PRs	🟨 Time to merge PRs	🟩 Merged PR	🟥 Non-merged PRs
Kinsman et al. (2021)	Code review bots (adoption)	🟨 Merged PRs	🟨 Non-merged PRs	🟥 Merged PRs	🟨 Non-merged PRs		🟨 Time to reject PRs	🟨 Time to merge PRs	🟨 Merged PRs	🟩 Non-merged PRs
Trockman et al. (2018)	Repository badges (adoption)									
Zimmermann and Artís (2019)	Bug tracker (move from Bugzilla to GitHub)	🟩 Bug tracker/ Issue comments				🟩 Opened				
Moldon et al. (2020)	Gamification mechanisms (removing from GitHub)									
Walden (2020)	Major Security Event (Heartbleed)			🟩 Merge commits						

🟩 Increase

🟥 Decrease

🟨 Does not change

on the evolution of a specific project called OpenSSL. As a result of the intervention, the number of monthly commits increased and the code complexity decreased.

In short, we showed an overview of how RDD have been used in empirical software engineering studies. As described in the Table 8, previous works investigated distinct variables. Even selecting related variables as "comments", each study focused on different types of comments (e.g. general pull requests comments, review comments, issue comments), or comments applied to different scenarios (e.g. comments on merged and non-merged pull requests). Our work extends this literature by providing a more in-depth investigation of the effects of a specific type of automation, namely code review bot adoption.

8 Limitations and Threats to Validity

In this section, we discuss the limitations and potential threats to validity of our study, their potential impact on the results, and how we have mitigated them (Wohlin et al. 2012).

External Validity: While our results only apply to OSS projects hosted on GitHub, many relevant projects are currently hosted on this platform (Dias et al. 2016). Our selection of projects also limits our results. Therefore, even though we considered a large number of projects and our results indicate general trends, we recommend running segmented analyses when applying our results to a given project. For replication purposes, we made our data and source code publicly available.[6]

Construct Validity: One of the constructs in our study is the "first bot comment on a pull request" as a proxy to the "time of bot adoption" on a project. A more precise definition of this adoption time would have involved the integration date, which is not provided by the GitHub API. Moreover, recent studies have observed 'mixed' GitHub accounts, i.e., accounts shared by a human and a bot Golzadeh et al. (2021) and Cassee et al. (2021), e.g., exhibiting user name and avatar and posting both human-written and bot-generated comments. A more precise definition of bot adoption should consider activity of the 'mixed' accounts as well. Hence, the validity of the "time of bot adoption" construct might have been threatened by the definition. We reduce this threat by excluding the period of 15 days immediately before and after adoption from all analyses. Moreover, Kalliamvakou et al. (2014) stated that many merged pull requests appear non-merged, which could also affect the construct validity of our study, since we consider the number of merged pull requests. To increase construct validity and improve the reliability of our qualitative findings, we employed a constant comparison method (Glaser and Strauss 2017). In this method, each interpretation is constantly compared with existing findings as it emerges from the qualitative analysis.

Internal Validity: To reduce internal threats, we applied multiple data filtering steps to the statistical models. To confirm the robustness of our models, we varied the data filtering criteria, for example, by filtering projects that did not receive pull requests in all months, instead of at least 6 months, and observed similar phenomena. Projects that disabled the bot during the period we considered might be a threat. However, detecting whether a project disabled the bot or not is challenging. The GitHub API does not provide this information. We reduce this threat by removing from our dataset projects without bot comments during the last four months of analysis. Additionally, we added several

[6] https://doi.org/10.5281/zenodo.4618498

controls that might influence the independent variables to reduce confounding factors. However, in addition to the already identified dependent variables, there might be other factors that influence the activities related to pull requests. These factors could include the adoption of other code review bots, or even other types of bots and non-bot automation. To treat this, we removed projects that adopted more than one bot, based on the list of bots provided by Wessel et al. (2018). To ensure information saturation, we continued recruiting participants and conducting interviews until we came to an agreement that no new significant information was found. As posed by Strauss and Corbin (1997), sampling should be discontinued once the collected data is considered sufficiently dense and data collection no longer generates new information.

9 Conclusion

In this work, we conducted an exploratory empirical investigation of the effects of adopting bots to support the code review process on pull requests. While several code review bots have been proposed and adopted by the OSS community, relatively little has been done to evaluate the state of practice. To understand the impact on practice, we statistically analyzed data from 1,194 open source projects hosted on GitHub. Further, we had a deep investigation into the reasons of the identified impacts. We interviewed 12 project maintainers and contributors experienced with code review bots.

By modeling the data around the introduction of a code review bot, we notice different results from merged pull requests and non-merged ones. We see that the monthly number of merged pull requests of a project increases after the adoption of a code review bot, requiring less communication between maintainers and contributors. At the same time, code review bots can lead projects to reject fewer pull requests. Afterwards, when interviewing developers we found a comprehensive set of reasons for these effects. First of all, bot comments help contributors and maintainers to be aware the state and quality of the contribution, making maintainers more confident to merge pull requests, which also changes the focus of developers' discussions.

Practitioners and open-source maintainers may use our results to understand how group dynamics can be affected by the introduction of a code review bot, and to design countermeasurements to avoid undesired effects. Future work includes a qualitative investigation of the effects of adopting a bot and the expansion of our analysis for other types of bots, activity indicators, social coding platforms, and statistical approaches, such as counterfactual time series (Murphy-Hill et al. 2019).

Acknowledgements This work was partially supported by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Finance Code 001, CNPq (grant 141222/2018-2), and National Science Foundation (grants 1815503 and 1900903). We also thank the Open Source developers who spent their time to participate in our research.

Funding This work is partially supported by the National Science Foundation under Grant numbers 1815503, 1900903, CNPq grant # 313067/2020-1.

Availability of Data and Material We provided the supplementary material https://zenodo.org/record/4618498

Declarations

Conflict of Interest The authors has no conflict of interest

References

Aniche M, Treude C, Steinmacher I, Wiese I, Pinto G, Storey MA, Gerosa MA (2018) How modern news aggregators help development communities shape and share knowledge. In: ICSE'18, pp 499–510

Balali S, Steinmacher I, Annamalai U, Sarma A, Gerosa MA (2018) Newcomers' barriers... is that all? An analysis of mentors' and newcomers' barriers in oss projects. Computer Supported Cooperative Work (CSCW) 27(3):679–714

Baysal O, Kononenko O, Holmes R, Godfrey MW (2016) Investigating technical and non-technical factors influencing modern code review. Empir Softw Eng 21(3):932–959

Benjamini Y, Hochberg Y (1995) Controlling the false discovery rate: a practical and powerful approach to multiple testing. J R Stat Soc: Ser B (Methodol) 57(1):289–300

Bernard HR (2017) Research methods in anthropology: qualitative and quantitative approaches. Rowman & Littlefield, Laham, Maryland

Brown C, Parnin C (2019) Sorry to bother you: designing bots for effective recommendations. In: Proceedings of the 1st international workshop on bots in software engineering, BotSE

Carvalho A, Luz W, Marcílio D, Bonifácio R, Pinto G, Dias Canedo E (2020) c-3pr: a bot for fixing static analysis violations via pull requests. In: 2020 IEEE 27th International conference on software analysis, evolution and reengineering (SANER), pp 161–171

Cassee N, Vasilescu B, Serebrenik A (2020) The silent helper: the impact of continuous integration on code reviews. In: 27th IEEE international conference on software analysis, evolution and reengineering. IEEE Computer Society

Cassee N, Kitsanelis C, Constantinou E, Serebrenik A (2021) Human, bot or both? A study on the capabilities of classification models on mixed accounts. In: 37th IEEE international conference on software maintenance and evolution. IEEE, pp xx–xx

Chaves AP, Gerosa MA (2020) How should my chatbot interact? A survey on social characteristics in human–chatbot interaction design. Int J Hum–Comput Interact 37:729–758

Cook T, Campbell D (1979) Quasi-experimentation: design and analysis issues for field settings. Houghton Mifflin, Chicago

Creswell J (2003) Mixed methods procedures. Research design: qualitative, quantitative, and mixed methods approaches 3:203–240

Dabbish L, Stuart C, Tsay J, Herbsleb J (2012) Social coding in GitHub: transparency and collaboration in an open software repository. In: Proceedings of the ACM 2012 conference on computer supported cooperative work, CSCW '12. ACM, New York, pp 1277–1286. https://doi.org/10.1145/2145204.2145396. http://doi.acm.org/10.1145/2145204.2145396

Dias LF, Steinmacher I, Pinto G, Costa DAD, Gerosa M (2016) How does the shift to GitHub impact project collaboration? In: 2016 IEEE international conference on software maintenance and evolution (ICSME), pp 473–477. https://doi.org/10.1109/ICSME.2016.78

Easterbrook S, Singer J, Storey MA, Damian D (2008) Selecting empirical methods for software engineering research. In: Guide to advanced empirical software engineering. Springer, pp 285–311

Ebert F, Castor F, Novielli N, Serebrenik A (2019) Confusion in code reviews: reasons, impacts, and coping strategies. In: 2019 IEEE 26th international conference on software analysis, evolution and reengineering (SANER). IEEE, pp 49–60

Ellwart T, Happ C, Gurtner A, Rack O (2015) Managing information overload in virtual teams: effects of a structured online team adaptation on cognition and performance. Eur J Work Org Psychol 24(5):812–826

Erlenhov L, Gomes de Oliveira Neto F, Leitner P (2016) An empirical study of bots in software development–characteristics and challenges from a practitioner's perspective. In: Proceedings of the 2020 28th ACM SIGSOFT international symposium on foundations of software engineering, FSE 2020

Gałecki A, Burzykowski T (2013) Linear mixed-effects models using R: a step-by-step approach. Springer Science & Business Media

Glaser BG, Strauss AL (2017) Discovery of grounded theory: strategies for qualitative research. Routledge, New York

Golzadeh M, Decan A, Constantinou E, Mens T (2021) Identifying bot activity in github pull request and issue comments. In: 3rd IEEE/ACM international workshop on bots in software engineering, botSE@ICSE 2021, Madrid, Spain, June 4, 2021. IEEE, pp 21–25. https://doi.org/10.1109/BotSE52550.2021.00012

Gousios G, Spinellis D (2012) GHTOrrent: GitHub's data from a firehose. In: 2012 9th IEEE working conference on mining software repositories (MSR). IEEE, pp 12–21

Gousios G, Pinzger M, van Deursen A (2014) An exploratory study of the pull-based software development model. In: Proceedings of the 36th international conference on software engineering. ACM, pp 345–355

Gousios G, Storey MA, Bacchelli A (2016) Work practices and challenges in pull-based development: the contributor's perspective. In: Proceedings of the 38th international conference on software engineering, ICSE '16. ACM, New York, pp 285–296. https://doi.org/10.1145/2884781.2884826. http://doi.acm.org/10.1145/2884781.2884826

Guo Y, Leitner P (2019) Studying the impact of ci on pull request delivery time in open source projects—a conceptual replication. PeerJ Computer Science 5:e245

Healy T (2012) The unanticipated consequences of technology. Nanotechnology: ethical and social Implications 155–173

Hove SE, Anda B (2005) Experiences from conducting semi-structured interviews in empirical software engineering research. In: 11th IEEE international software metrics symposium (METRICS'05). IEEE, p 10

Imbens GW, Lemieux T (2008) Regression discontinuity designs: a guide to practice. J Econ 142(2):615–635

Kalliamvakou E, Gousios G, Blincoe K, Singer L, German DM, Damian D (2014) The promises and perils of mining GitHub. In: Proceedings of the 11th working conference on mining software repositories, MSR 2014. ACM, New York, pp 92–101. https://doi.org/10.1145/2597073.2597074.http://doi.acm.org/10.1145/2597073.2597074

Kavaler D, Trockman A, Vasilescu B, Filkov V (2019) Tool choice matters: JavaScript quality assurance tools and usage outcomes in GitHub projects. In: Proceedings of the 41st international conference on software engineering. IEEE Press, pp 476–487

Khanan C, Luewichana W, Pruktharathikoon K, Jiarpakdee J, Tantithamthavorn C, Choetkiertikul M, Ragkhitwetsagul C, Sunetnanta T (2020) Jitbot: an explainable just-in-time defect prediction bot. In: 2020 35th IEEE/ACM international conference on automated software engineering (ASE), pp 1336–1339

Kinsman T, Wessel M, Gerosa M, Treude C (2021) How do software developers use github actions to automate their workflows? In: Mining software repositories conference (MSR). IEEE

Kuznetsova A, Brockhoff PB, Christensen RHB (2017) lmertest package: tests in linear mixed effects models. J Stat Softw 82(13):1–26

Lebeuf C, Storey MD, Zagalsky A (2017) How software developers mitigate collaboration friction with chatbots. In: Talking with conversational agents in collaborative action workshop at the 20th ACM conference on computer-supported cooperative work and social computing, CSCW '17. http://arxiv.org/abs/1702.07011

Lebeuf C, Storey MA, Zagalsky A (2018) Software bots. IEEE Softw 35(1):18–23

Lebeuf C, Zagalsky A, Foucault M, Storey MA (2019) Defining and classifying software bots: a faceted taxonomy. In: Proceedings of the 1st international workshop on bots in software engineering, BotSE '19. IEEE Press, Piscataway, pp 1–6. https://doi.org/10.1109/BotSE.2019.00008

Liu D, Smith MJ, Veeramachaneni K (2020) Understanding user-bot interactions for small-scale automation in open-source development. In: Extended abstracts of the 2020 CHI conference on human factors in computing systems, CHI EA '20. Association for Computing Machinery, New York, pp 1–8. https://doi.org/10.1145/3334480.3382998

McIntosh S, Kamei Y, Adams B, Hassan AE (2014) The impact of code review coverage and code review participation on software quality: a case study of the qt, vtk, and itk projects. In: Proceedings of the 11th working conference on mining software repositories, pp 192–201

Mendez C, Padala HS, Steine-Hanson Z, Hildebrand C, Horvath A, Hill C, Simpson L, Patil N, Sarma A, Burnett M (2018) Open source barriers to entry, revisited: a sociotechnical perspective. In: 2018 IEEE/ACM 40th international conference on software engineering (ICSE), pp 1004–1015

Mirhosseini S, Parnin C (2017) Can automated pull requests encourage software developers to upgrade out-of-date dependencies? In: Proceedings of the 32nd IEEE/ACM international conference on automated software engineering, ASE 2017. IEEE Press, Piscataway, pp 84–94. http://dl.acm.org/citation.cfm?id=3155562.3155577

Moldon L, Strohmaier M, Wachs J (2020) How gamification affects software developers: cautionary evidence from a quasi-experiment on github. arXiv:200602371

Monperrus M (2019) Explainable software bot contributions: case study of automated bug fixes. In: Proceedings of the 1st international workshop on bots in software engineering, BotSE '19. IEEE Press, Piscataway, pp 12–15. https://doi.org/10.1109/BotSE.2019.00010

Mulder K (2013) Impact of new technologies: how to assess the intended and unintended effects of new technologies. Handb Sustain Eng

Murphy-Hill E, Smith EK, Sadowski C, Jaspan C, Winter C, Jorde M, Knight A, Trenk A, Gross S (2019) Do developers discover new tools on the toilet? In: Proceedings of the 41st international conference on software engineering, ICSE '19. IEEE Press, pp 465–475. https://doi.org/10.1109/ICSE.2019.00059

Nakagawa S, Schielzeth H (2013) A general and simple method for obtaining r2 from generalized linear mixed-effects models. Methods Ecol Evol 4(2):133–142

Paikari E, van der Hoek A (2018) A framework for understanding chatbots and their future. In: Proceedings of the 11th international workshop on cooperative and human aspects of software engineering, CHASE '18. ACM, New York, pp 13–16. https://doi.org/10.1145/3195836.3195859. http://doi.acm.org/10.1145/3195836.3195859

Patton MQ (2014) Qualitative research & evaluation methods: integrating theory and practice. Sage Publications, Los Angeles

Phan-udom P, Wattanakul N, Sakulniwat T, Ragkhitwetsagul C, Sunetnanta T, Choetkiertikul M, Kula RG (2020) Teddy: automatic recommendation of pythonic idiom usage for pull-based software projects. In: 2020 IEEE International conference on software maintenance and evolution (ICSME). IEEE, pp 806–809

Pinto G, Steinmacher I, Gerosa MA (2016) More common than you think: an in-depth study of casual contributors. In: 2016 IEEE 23rd international conference on software analysis, evolution, and reengineering (SANER), vol 1. IEEE, pp 112–123

Pinto GHL, Figueira Filho F, Steinmacher I, Gerosa MA (2017) Training software engineers using open-source software: the professors' perspective. In: 2017 IEEE 30th conference on software engineering education and training (CSEE&T). IEEE, pp 117–121

Romano J, Kromrey JD, Coraggio J, Skowronek J (2006) Appropriate statistics for ordinal level data: should we really be using t-test and cohen'sd for evaluating group differences on the nsse and other surveys. In: Annual meeting of the Florida Association of Institutional Research, pp 1–33

Runeson P, Höst M (2009) Guidelines for conducting and reporting case study research in software engineering. Empir Softw Eng 14(2):131

Serban D, Golsteijn B, Holdorp R, Serebrenik A (2021) Saw-bot: proposing fixes for static analysis warnings with github suggestions. In: Workshop on bots in software engineering. IEEE Computer Society

Sheather S (2009) A modern approach to regression with R. Springer Science & Business Media

Singer L, Figueira Filho F, Storey MA (2014) Software engineering at the speed of light: how developers stay current using Twitter. In: 36th ICSE, pp 211–221

Steinmacher I, Wiese I, Chaves AP, Gerosa MA (2013) Why do newcomers abandon open source software projects? In: 2013 6th international workshop on cooperative and human aspects of software engineering (CHASE). IEEE, pp 25–32

Steinmacher I, Conte T, Gerosa MA, Redmiles D (2015) Social barriers faced by newcomers placing their first contribution in open source software projects. In: Proceedings of the 18th ACM conference on Computer supported cooperative work & social computing, pp 1379–1392

Steinmacher I, Pinto G, Wiese IS, Gerosa MA (2018) Almost there: a study on quasi-contributors in open source software projects. In: Proceedings of the 40th international conference on software engineering, ICSE '18. ACM, New York, pp 256–266. https://doi.org/10.1145/3180155.3180208. http://doi.acm.org/10.1145/3180155.3180208

Stol KJ, Ralph P, Fitzgerald B (2016) Grounded theory in software engineering research: a critical review and guidelines. In: Proceedings of the 38th international conference on software engineering, pp 120–131

Storey MA, Zagalsky A (2016) Disrupting developer productivity one bot at a time. In: Proceedings of the 2016 24th ACM SIGSOFT international symposium on foundations of software engineering, FSE 2016. ACM, New York, pp 928–931. https://doi.org/10.1145/2950290.2983989

Storey MA, Treude C, van Deursen A, Cheng LT (2010) The impact of social media on software engineering practices and tools. In: FSE/SDP workshop on future of software engineering research, pp 359–364

Storey MA, Zagalsky A, Filho FF, Singer L, German DM (2017) How social and communication channels shape and challenge a participatory culture in software development. IEEE Trans Softw Eng 43(2):185–204. https://doi.org/10.1109/TSE.2016.2584053

Strauss A, Corbin JM (1997) Grounded theory in practice. Sage, Los Angeles

Strauss AL, Corbin J (1998) Basics of qualitative research: techniques and procedures for developing grounded theory sage publications. SAGE Publications, Los Angeles

Strauss A, Corbin JM (2007) Basics of qualitative research : techniques and procedures for developing grounded theory, 3rd edn. SAGE Publications, Los Angeles

Thistlethwaite DL, Campbell DT (1960) Regression-discontinuity analysis: an alternative to the ex post facto experiment. J Educ Psychol 51(6):309

Trockman A, Zhou S, Kästner C, Vasilescu B (2018) Adding sparkle to social coding: an empirical study of repository badges in the npm ecosystem. In: Proceedings of the 40th international conference on software engineering, pp 511–522

Urli S, Yu Z, Seinturier L, Monperrus M (2018) How to design a program repair bot?: insights from the repairnator project. In: Proceedings of the 40th international conference on software engineering: software engineering in practice, ICSE-SEIP '18. ACM, New York, pp 95–104. https://doi.org/10.1145/3183519.3183540. http://doi.acm.org/10.1145/3183519.3183540

Vorvoreanu M, Zhang L, Huang YH, Hilderbrand C, Steine-Hanson Z, Burnett M (2019) From gender biases to gender-inclusive design: an empirical investigation. In: Proceedings of the 2019 CHI conference on human factors in computing systems, CHI '19. Association for Computing Machinery, New York, pp 1–14. https://doi.org/10.1145/3290605.3300283

Walden J (2020) The impact of a major security event on an open source project: the case of openssl. arXiv:200514242

Wessel M, Steinmacher I (2020) The inconvenient side of software bots on pull requests. In: Proceedings of the 2nd international workshop on bots in software engineering, BotSE. https://doi.org/10.1145/3387940.3391504

Wessel M, de Souza BM, Steinmacher I, Wiese IS, Polato I, Chaves AP, Gerosa MA (2018) The power of bots: characterizing and understanding bots in OSS projects. Proc ACM Hum-Comput Interact 2(CSCW):182:1–182:19. https://doi.org/10.1145/3274451. http://doi.acm.org/10.1145/3274451

Wessel M, Serebrenik A, Wiese I, Steinmacher I, Gerosa MA (2020a) What to expect from code review bots on GitHub? a survey with OSS maintainers. In: SBES 2020—Ideias inovadoras e resultados emergentes

Wessel M, Serebrenik A, Wiese IS, Steinmacher I, Gerosa MA (2020b) Effects of adopting code review bots on pull requests to oss projects. In: IEEE International conference on software maintenance and evolution. IEEE Computer Society

Wessel M, Wiese I, Steinmacher I, Gerosa M (2021) Don't disturb me: challenges of interacting with software bots on open source software projects. In: Proceedings of ACM human-computer interaction (CSCW)

Wilks DS (2011) Statistical methods in the atmospheric sciences, vol 100. Academic Press, San Diego, California

Wohlin C, Runeson P, Höst M, Ohlsson MC, Regnell B, Wesslén A (2012) Experimentation in software engineering. Springer Science & Business Media

Woods DD, Patterson ES (2001) How unexpected events produce an escalation of cognitive and coordinative demands. In: Hancock PA, Desmond PA (eds) Stress, workload, and fatigue. L Erlbaum, Mahwah

Wyrich M, Bogner J (2019) Towards an autonomous bot for automatic source code refactoring. In: Proceedings of the 1st international workshop on bots in software engineering, BotSE '19. IEEE Press, Piscataway, pp 24–28. https://doi.org/10.1109/BotSE.2019.00015

Yin RK (2003) Design and methods. Case Study Research 3

Yu Y, Wang H, Filkov V, Devanbu P, Vasilescu B (2015) Wait for it: determinants of pull request evaluation latency on GitHub. In: 2015 IEEE/ACM 12th working conference on mining software repositories, pp 367–371. https://doi.org/10.1109/MSR.2015.42

Zhao Y, Serebrenik A, Zhou Y, Filkov V, Vasilescu B (2017) The impact of continuous integration on other software development practices: a large-scale empirical study. In: Proceedings of the 32nd IEEE/ACM international conference on automated software engineering. IEEE Press, pp 60–71

Zimmermann T, Artís AC (2019) Impact of switching bug trackers: a case study on a medium-sized open source project. In: 2019 IEEE international conference on software maintenance and evolution (ICSME). IEEE, pp 13–23

Mairieli Wessel is a Postdoctoral Associate at the Software Engineering Research Group (SERG) of Delft University of Technology (TU Delft). She obtained her Ph.D. in Computer Science from the University of São Paulo, Brazil. Her main research interest is in software engineering (SE) and computer-supported cooperative work (CSCW), focused on software bots and open-source development. Her research goal is to design intelligent support for developers by leveraging bots' capabilities.

Alexander Serebrenik is a Full Professor of Social Software Engineering at the Software Engineering and Technology cluster of Eindhoven University of Technology (TU/e). His research goal is to facilitate the evolution of software by taking into account social aspects of software development. His work tends to involve theories and methods both from within computer science (e.g., theory of socio-technical coordination; methods from natural language processing, machine learning) and from outside of computer science (e.g., organizational psychology). The underlying idea of his work is that of empiricism, i.e., that addressing software engineering challenges should be grounded in observation and experimentation, and requires a combination of the social and the technical perspectives. Prof. Serebrenik has co-authored a book, Evolving Software Systems (Springer Verlag, 2014), and more than 100 scientific papers and articles.

Igor Wiese is an Associate Professor in the Department of Computing at the Federal University of Technology—Parana, Brazil. He isinterested in Mining Software Repositories, Human Aspects of Software Engineering, and related topics. Wiese holds a PhD degree in Computer Science from the University of São Paulo. More information is available at www.igorwiese.com.

Igor Steinmacher is an Assistant Professor in the School of Informatics, Computing, and Cyber Systems at the Northern Arizona University (NAU), and was previously at the Federal University of Technology Paraná (UTFPR), Brazil. He received a Ph.D. in Computer Science from the University of São Paulo (USP - Brazil). He researches the intersections of Software Engineering (SE) and Computer Supported Cooperative Work (CSCW). Currently, his research focuses on the behavior of developers in Open Source Communities, including support of newcomers, the impact of Bots in the community, and gender bias in Open Source Software. His interests include Open Source Software, Human Aspects of Software Engineering, Empirical Software Engineering, and Mining Software Repositories techniques.

Marco A. Gerosa is an Associate Professor at the Northern Arizona University, USA and PhD advisor at the University of São Paulo, Brazil. He researches Software Engineering and CSCW. Recent projects include the development of tools and strategies to support newcomers onboarding to open source software communities and the design of bots and chatbots. He published more than 200 papers and serves on the program committee (PC) of top-tier conferences, such as FSE, MSR, and CSCW. For more information, visit http://www.marcoagerosa.com.

Affiliations

Mairieli Wessel[1,2] · Alexander Serebrenik[3] · Igor Wiese[4] · Igor Steinmacher[4] · Marco A. Gerosa[5]

Alexander Serebrenik
a.serebrenik@tue.nl

Igor Wiese
igor@utfpr.edu.br

Igor Steinmacher
igorfs@utfpr.edu.br

Marco A. Gerosa
marco.gerosa@nau.edu

[1] Delft University of Technology, Delft, The Netherlands

[2] University of São Paulo, São Paulo, Brazil

[3] Eindhoven University of Technology, Eindhoven, The Netherlands

[4] Universidade Tecnológica Federal do Paraná, Curitiba, Brazil

[5] Northern Arizona University, Flagstaff, AZ, USA