Effects of Adopting Code Review Bots on Pull Requests to OSS Projects

Mairieli Wessel*, Alexander Serebrenik†, Igor Wiese‡, Igor Steinmacher‡§ and Marco A. Gerosa§
*University of São Paulo, Brazil
†Eindhoven University of Technology, The Netherlands
‡Universidade Tecnológica Federal do Paraná, Campo Mourão, Brazil
§Northern Arizona University, USA
mairieli@ime.usp.br, a.serebrenik@tue.nl, igor@utfpr.edu.br,
{igor.steinmacher, marco.gerosa}@nau.edu

***Abstract*—Software bots, which are widely adopted by Open Source Software (OSS) projects, support developers on several activities, including code review. However, as with any new technology adoption, bots may impact group dynamics. Since understanding and anticipating such effects is important for planning and management, 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. 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. Practitioners and maintainers may leverage our results to understand, or even predict, bot effects on their projects' social interactions.**

***Index Terms*—Software Bots, GitHub Bots, Code Review, Open Source Software, Software Engineering**

I. Introduction

Many Open Source Software (OSS) projects employ code review as an essential part of the development process [1]. Code review is a well-known practice for software quality assurance [2]. In the pull-based development model, project maintainers carefully inspect code changes and engage in discussion with the contributors to understand and improve the modifications before integrating them into the codebase [3]. The time maintainers spend reviewing pull requests is non-negligible and can affect, for example, the volume of new contributions [4] and the onboarding of newcomers [5].

In this context, software bots play a prominent role in the code review process [6] by serving as an interface between users and other tools [7] and reducing 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 [8]. Code review bots guide contributors to provide necessary information before maintainers triage the pull requests [6].

Notoriously, though, the adoption of new technology can bring consequences that differ from the expectations of the technology designers and adopters [9]. 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 [10]. Since, according to Mulder et al. [11], many effects are not directly caused by the new technology itself, but by the changes in human behavior that it provokes, it is important to assess and discuss the effects of new technology on group dynamics, and this is often neglected for software bots.

In the code review process, bots may affect existing project activities in several ways. For example, while project maintainers may direct their effort to other activities, the bot could provide poor feedback [6, 12] that leads to contributor drop-out—indeed, lack of feedback on pull requests is known to discourage further contributions [13].

We aim to understand how the dynamics of pull requests of GitHub projects change following the adoption of a code review bot. To understand what happens after the adoption of a bot, we used a *Regression Discontinuity Design* [14] to model the effects of code review bot adoption across $1,194$ OSS projects hosted on GitHub. Hence, our main research question is:

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

Extending the work of Wessel et al. [6], 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 the longitudinal effects of the bot adoption. 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.

Analyzing the statistical models, we found that more pull requests are merged into the codebase after the code review bot adoption, and there is less communication between contributors and maintainers. Considering non-merged pull requests, after bot adoption, projects have less monthly non-merged pull requests, and faster pull requests rejections.

In this paper, we make the following contributions: (i) identification of changes in project activity indicators after the adoption of a code review bot; and (ii) an elucidation of how the introduction of a bot can impact OSS projects.

These contributions aim to help practitioners and maintainers understand the 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.

II. 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 [15, 16] to formulate hypotheses to further investigate in our main study.

A. Case Study Method

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 couple of years: the Julia programming language project[1] and CakePHP,[2] 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 a code review bot named Codecov, which posted the first comments on pull requests to the Julia project in July 2016 and CakePHP in April 2016.

After selecting the projects, we analyzed data from one year before and one year after the bot adoption, using the data available at the GHTorrent dataset [17]. During this time frame, the only bot adopted by Julia and CakePHP was the Codecov bot. Similar to previous work [18], we exclude 30 days around the bot adoption to avoid the influence of the instability caused during this period. Afterward, we aggregated individual pull request data into monthly periods, considering 12 months before and after the bot introduction. We choose the month time frame based on previous literature [18, 19, 20]. 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. [6]:

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. We used the median because the distribution is skewed.

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. We used the median because the distribution is skewed.

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

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

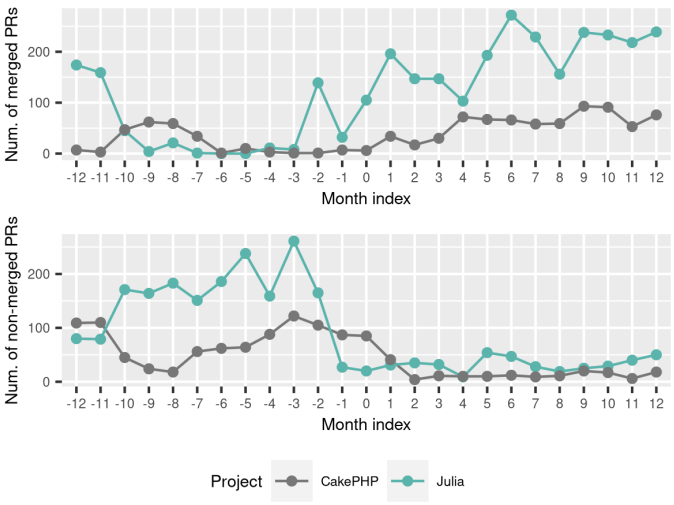

Fig. 1. Monthly merged and non-merged pull requests.

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. We use the median because the distribution is skewed.

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 introduction, we used the non-parametric Mann-Whitney-Wilcoxon test [21]. 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 [22] 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 a 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.

B. Case Study Results

The number of merged pull requests *increased* for both projects (Julia: p-value 0.0003, $\delta = -0.87$; CakePHP: p-value 0.001, $\delta = -0.76$) while 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 1 shows the monthly number of merged and non-merged pull requests, top and bottom respectively, before and after the 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.

Figure 2 shows the monthly median of comments on merged and non-merged pull requests, respectively. CakePHP showed

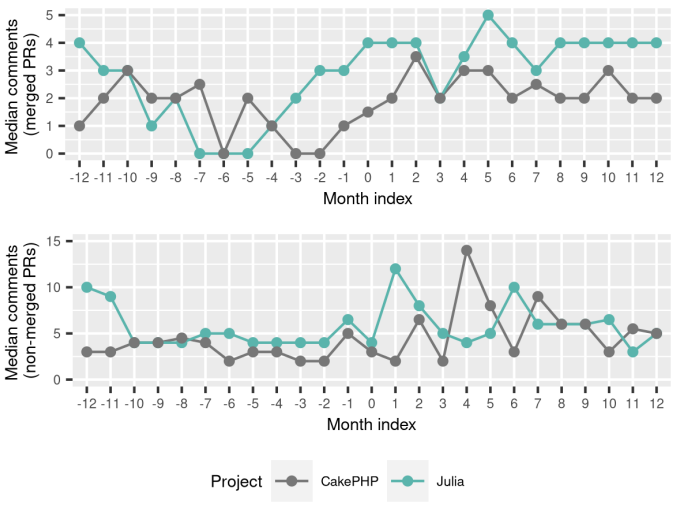

Fig. 2. Monthly comments on merged and non-merged pull requests.

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:

$H_{2.1}$ 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.

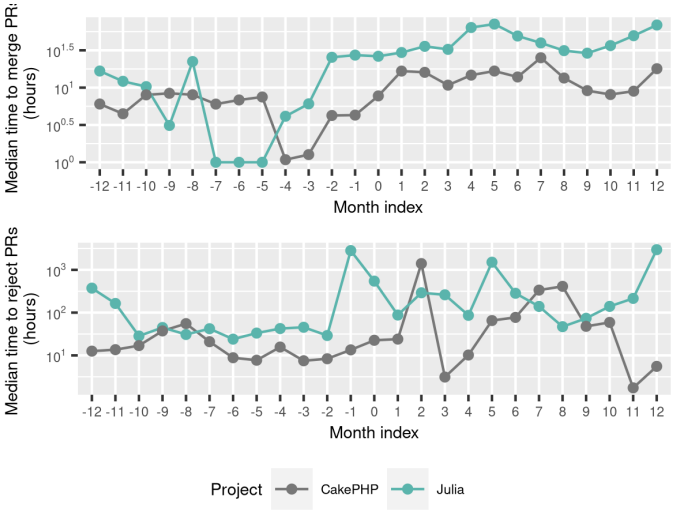

Fig. 3. Monthly median time to merge and reject pull requests.

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 Figure 3. 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.

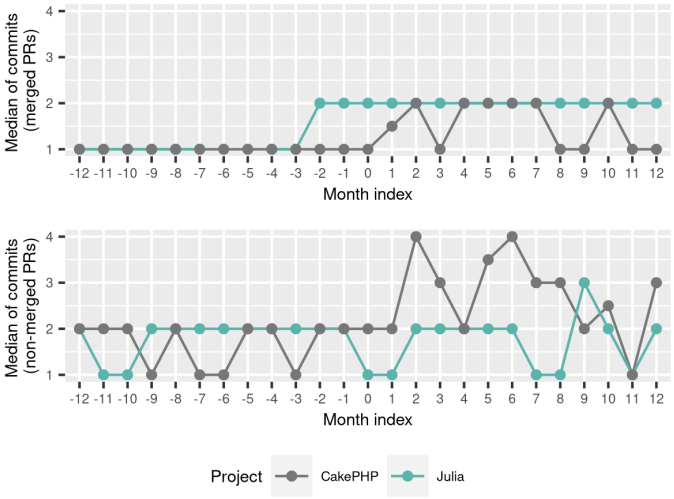

Fig. 4. Monthly commits on merged and non-merged pull requests.

Investigating the number of pull request commits per month (see Figure 4), 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.

In conclusion, differently than Wessel et al. [6], 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.

III. Main Study Design

In the following, we present the statistical approach and data collection procedures for the main study.

A. 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 the bot adoption. We employed Regression Discontinuity Design (RDD) [14, 23], that has been applied in the context of software engineering in the past [18, 20]. 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 [24]. The statistical model behind RDD is

$$y_i = \alpha + \beta \cdot \textit{time}_i + \gamma \cdot \textit{intervention}_i + \delta \cdot \textit{time\_after\_intervention}_i + \eta \cdot \text{controls}_i + \varepsilon_i$$

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 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 $\textit{controls}_i$ 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 $\beta + \delta$. 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 x 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 [25]. We implemented the RDD models as a mixed-effects linear regression using the R package *lmerTest* [26].

To capture project-to-project and language-to-language variability, we modeled *project name* and *programming language* as random effects [27]. By modeling these features as random effects, we can account for and explain different behaviors observed across projects or programming languages [18]. We evaluate the model fit using *marginal* (R^2_m) and *conditional* (R^2_c) scores, as described by Nakagawa and Schielzeth [28]. 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 [29]. We also account for multicollinearity, excluding any fixed effects for which the variance inflation factor (VIF) is higher than 5 [29].

B. Data Collection

1) 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 [17] 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. [6]. Following the method used by Zhao et al. [18] 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 code review bots. For each project, we collected data on all its merged and non-merged pull requests.

2) Aggregating projects variables: Similar to the exploratory case study (see Section II), 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 [17]:

Project name: the name of the project, used to identify the project on GitHub. We account for the fact that different projects can lead to different contribution patterns.

Programming language: the primary project programming language as automatically determined and provided by the GitHub. We consider that projects with different programming languages can lead to different activities and contribution patterns.

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.

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

Total number of commits: as a proxy for the activity level of a project, we compute the total number of commits.

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

TABLE I
AN OVERVIEW OF THE STUDIED BOTS

Bot name	GitHub user	Link	# of projects
Ansible's issue bot	ansibot	https://github.com/ansible/ansibullbot	1
Elastic Machine	elasticmachine	https://github.com/elasticmachine	3
Codecov	codecov-io	https://github.com/marketplace/codecov	460
Coveralls	coveralls	https://github.com/coveralls	730
			Total of 1,194 under study

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 [30]. 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. We excluded 272 projects that did not received any comments during the last four months. After applying all filters, 1,194 GitHub software projects remained. Table I shows the number of projects per bot. All of these four bots perform similar tasks on pull requests—provide comments on pull requests about code coverage.

IV. MAIN STUDY RESULTS

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

A. 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 RDD models, as described in Section III-A. For these models, the *number of merged/non-merged pull requests* per month is the dependent variable. Table II summarizes the results of these two RDD models. In addition to the model coefficients, the table also shows the sum of squares, with a variance explained for each variable.

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. We note an increasing trend before adoption; a statistically significant discontinuity at the adoption time; and a positive trend after adoption that indicates that the number of merged pull requests increased even faster.

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 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. We note a decreasing trend before adoption, a statistically significant discontinuity at the adoption time, and a slight acceleration after adoption in the decreasing time trend seen before adoption.

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

> Overall, there are more monthly merged pull requests and fewer monthly non-merged pull requests after adopting a code review bot.

B. Effects in 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 III 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

TABLE II
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	Sum of Squares	Coefficients	Sum of Squares
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

*** $p < 0.001$, ** $p < 0.01$, * $p < 0.05$

TABLE III
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	Sum of Squares	Coefficients	Sum of Squares
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

*** $p < 0.001$, ** $p < 0.01$, * $p < 0.05$

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 the 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 bots 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 suggests that communication during the pull request review is strongly associated with the time to reject the pull request. Table III 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 as time passes.

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, our model for non-merged pull requests did not show any bot effect as time passes, then we also reject $\mathbf{H}_{2.2}$.

> 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.

C. 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. Then, 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 IV.

Analyzing the results to 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. 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

TABLE IV
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	Sum of Squares	Coefficients	Sum of Squares
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

*** $p < 0.001$, ** $p < 0.01$, * $p < 0.05$

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 the adoption time. 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 variables ($R^2_c = 0.72$). 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 the bot adoption is reversed, toward a decrease after adoption.

Thus, based on regression results for merged and non-merged pull requests, we reject both $\mathbf{H}_{3.1}$ and $\mathbf{H}_{3.2}$.

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.

D. Effects in Commits

Finally, we studied whether code review bot adoption affects the number of commits made before and during the 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 V.

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 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 V, 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 V 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.

Therefore, based on models for merged and non-merged pull requests, we reject both $\mathbf{H}_{4.1}$ and $\mathbf{H}_{4.2}$.

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

V. DISCUSSION

Adding a code review bot to a project can represent the desire to enhance feedback to stakeholders, helping contributors and maintainers, and achieving improved interpersonal communication, as already discussed by Storey and Zagalsky [7]. Still, code review bots can guide contributors toward detecting change effects before maintainers triage the pull requests [6], ensuring high-quality standards. In this paper, following the study of Wessel et al. [6], we focused on monthly activity indicators that are not primarily related to bot adoption, but

TABLE V
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	Sum of Squares	Coefficients	Sum of Squares
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

*** $p < 0.001$, ** $p < 0.01$, * $p < 0.05$

might be impacted by it. We found that half of them have a statistically significant effect on the review process.

According to the regression results, the monthly number of merged pull requests continued increasing, even faster, after the code review bot adoption. This would indicate that contributors started to have faster and clearer feedback on what they need to do to have their contribution accepted. In addition, the number of non-merged pull requests continued to decrease, even faster, after bot adoption. Therefore, 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. On the one hand, bots helped maintainers to focus on the non-trivial review tasks. On the other hand, if a project did not have the workforce to handle these incoming contributions it could become a maintenance burden. These findings confirm the hypothesis we formulated based on the exploratory case study.

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. 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, likely caused by other project-specific questions.

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 the 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, this may indicate that, after the bot adoption, maintainers stopped to expend efforts on pull requests that are not likely to be integrated into the codebase.

As we found in the model of comments 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 passes. 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 IV-D, 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.

Indeed, the dynamics of pull requests of GitHub projects changed following the adoption of code review bots. This change in the pull request dynamics can directly affect contributors' and maintainers' work. Hence, understanding how the code review bot adoption affects a project is important for practitioners and open-source maintainers, mainly to avoid unexpected or even undesired effects. Awareness of unexpected bot effects can lead maintainers to take countermeasures and/or decide whether or not to use a code review bot.

VI. RELATED WORK

Software bots support activities in software engineering, such as communication and decision-making [7]. Bots are

particularly relevant in social-coding platforms [31], such as GitHub, where the pull-based model [30] offers several opportunities for community engagement, but at the same time increases the workload for maintainers [32, 33]. Thus, OSS communities have been adopting bots to reduce the workload by automating repetitive tasks on GitHub pull requests [6].

Bots are software applications that integrate their work with human tasks, serving as interfaces between users and other tools [34, 35], and providing additional value to the human users [36]. Software bots frequently reside on platforms where users work and interact with other users [37]. On GitHub, bots have user profiles to interact with the developers, executing well-defined tasks [6].

Storey et al. [7] and Paikari and van der Hoek [38] highlight that the potentially negative impact of task automation through bots is being overlooked. Storey et al. [7] 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 [8], we expected that bots would impact group dynamics in a way that differs from non-bot forms of automation.

Wessel et al. [6] 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 these bots into 13 categories according to the tasks they perform. The third most frequently used bots are code review bots. According to Wessel et al. [6], 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 serve to report the feedback of a third-party service into the GitHub platform. In a preliminary study, Wessel et al. [39] conducted a survey with 127 open source maintainers experienced in using code review bots. While maintainers report that bots satisfied 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, newcomers' dropout.

Prior work has also investigated the impact of CI and code review tools on GitHub projects [18, 19, 20] across time. While Zhao et al. [18] and Cassee et al. [20] focused on the impact of the Travis CI tool's introduction on development practices, Kavaler et al. [19] turned to the impact of linters, dependency managers, and coverage reporter tools. Our work extends this literature by providing a more in-depth investigation of the effects of code review bot adoption.

VII. Threats to Validity

While our results only apply to OSS projects hosted on GitHub, many relevant projects are currently hosted on this platform [40]. Our selection of projects also limits our results. Therefore, even though we considered a large number of projects and our results indicates 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.[3]

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. 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. [41] 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 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 or not the bot 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 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. [6].

VIII. 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.

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.

Practitioners and open-source maintainers may use our results to understand how group dynamics can be affected

[3] https://zenodo.org/record/3858029#.Xs15vilKhhE

by the introduction of a code review bot, designing countermeasurements to avoid undesired effects. Future work includes the qualitative investigation of the effects of adopting a bot and the expansion of our analysis for other types of bots, activity indicators, and social coding platforms.

ACKNOWLEDGMENTS

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).

REFERENCES

[1] O. Baysal, O. Kononenko, R. Holmes, and M. W. Godfrey, "Investigating technical and non-technical factors influencing modern code review," *Empirical Software Engineering*, vol. 21, no. 3, pp. 932–959, 2016.

[2] F. Ebert, F. Castor, N. Novielli, and A. Serebrenik, "Confusion in code reviews: Reasons, impacts, and coping strategies," in *2019 IEEE 26th International Conference on Software Analysis, Evolution and Reengineering (SANER)*. IEEE, 2019, pp. 49–60.

[3] S. McIntosh, Y. Kamei, B. Adams, and A. E. Hassan, "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*, 2014, pp. 192–201.

[4] Y. Yu, H. Wang, V. Filkov, P. Devanbu, and B. Vasilescu, "Wait for it: Determinants of pull request evaluation latency on GitHub," in *2015 IEEE/ACM 12th Working Conference on Mining Software Repositories*, May 2015, pp. 367–371.

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

[6] M. Wessel, B. M. de Souza, I. Steinmacher, I. S. Wiese, I. Polato, A. P. Chaves, and M. A. Gerosa, "The power of bots: Characterizing and understanding bots in OSS projects," *Proc. ACM Hum.-Comput. Interact.*, vol. 2, no. CSCW, pp. 182:1–182:19, Nov. 2018. [Online]. Available: http://doi.acm.org/10.1145/3274451

[7] M.-A. Storey and A. Zagalsky, "Disrupting developer productivity one bot at a time," in *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, ser. FSE 2016. New York, NY, USA: ACM, 2016, pp. 928–931. [Online]. Available: http://doi.acm.org/10.1145/2950290.2983989

[8] M. Monperrus, "Explainable software bot contributions: Case study of automated bug fixes," in *Proceedings of the 1st International Workshop on Bots in Software Engineering*, ser. BotSE '19. Piscataway, NJ, USA: IEEE Press, 2019, pp. 12–15. [Online]. Available: https://doi.org/10.1109/BotSE.2019.00010

[9] T. Healy, "The unanticipated consequences of technology," *Nanotechnology: ethical and social Implications*, pp. 155–173, 2012.

[10] D. D. Woods and E. S. Patterson, "How unexpected events produce an escalation of cognitive and coordinative demands," *PA Hancock, & PA Desmond, Stress, workload, and fatigue. Mahwah, NJ: L. Erlbaum*, 2001.

[11] K. Mulder, "Impact of new technologies: how to assess the intended and unintended effects of new technologies," *Handb. Sustain. Eng.(2013)*, 2013.

[12] M. Wessel and I. Steinmacher, "The inconvenient side of software bots on pull requests," in *2nd International Workshop on Bots in Software Engineering*, ser. BotSE '20, 2020.

[13] I. Steinmacher, G. Pinto, I. S. Wiese, and M. A. Gerosa, "Almost there: A study on quasi-contributors in open source software projects," in *Proceedings of the 40th International Conference on Software Engineering*, ser. ICSE '18. New York, NY, USA: ACM, 2018, pp. 256–266. [Online]. Available: http://doi.acm.org/10.1145/3180155.3180208

[14] D. L. Thistlethwaite and D. T. Campbell, "Regression-discontinuity analysis: An alternative to the ex post facto experiment." *Journal of Educational psychology*, vol. 51, no. 6, p. 309, 1960.

[15] P. Runeson and M. Höst, "Guidelines for conducting and reporting case study research in software engineering," *Empirical software engineering*, vol. 14, no. 2, p. 131, 2009.

[16] R. K. Yin, "Design and methods," *Case study research*, vol. 3, 2003.

[17] G. Gousios and D. Spinellis, "GHTorrent: GitHub's data from a firehose," in *2012 9th IEEE Working Conference on Mining Software Repositories (MSR)*. IEEE, 2012, pp. 12–21.

[18] Y. Zhao, A. Serebrenik, Y. Zhou, V. Filkov, and B. Vasilescu, "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, 2017, pp. 60–71.

[19] D. Kavaler, A. Trockman, B. Vasilescu, and V. Filkov, "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, 2019, pp. 476–487.

[20] N. Cassee, B. Vasilescu, and A. Serebrenik, "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, 2020.

[21] D. S. Wilks, *Statistical methods in the atmospheric sciences*. Academic press, 2011, vol. 100.

[22] J. Romano, J. D. Kromrey, J. Coraggio, and J. Skowronek, "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*, 2006, pp. 1–33.

[23] G. W. Imbens and T. Lemieux, "Regression discontinuity designs: A guide to practice," *Journal of econometrics*, vol. 142, no. 2, pp. 615–635, 2008.

[24] T. Cook and D. Campbell, *Quasi-Experimentation: Design and Analysis Issues for Field Settings*. Houghton Mifflin, 1979.

[25] Y. Benjamini and Y. Hochberg, "Controlling the false discovery rate: a practical and powerful approach to multiple testing," *Journal of the Royal statistical society: series B (Methodological)*, vol. 57, no. 1, pp. 289–300, 1995.

[26] A. Kuznetsova, P. B. Brockhoff, and R. H. B. Christensen, "lmertest package: tests in linear mixed effects models," *Journal of Statistical Software*, vol. 82, no. 13, 2017.

[27] A. Gałecki and T. Burzykowski, *Linear mixed-effects models using R: A step-by-step approach*. Springer Science & Business Media, 2013.

[28] S. Nakagawa and H. Schielzeth, "A general and simple method for obtaining r2 from generalized linear mixed-effects models," *Methods in ecology and evolution*, vol. 4, no. 2, pp. 133–142, 2013.

[29] S. Sheather, *A modern approach to regression with R*. Springer Science & Business Media, 2009.

[30] G. Gousios, M. Pinzger, and A. v. Deursen, "An exploratory study of the pull-based software development model," in *Proceedings of the 36th International Conference on Software Engineering*. ACM, 2014, pp. 345–355.

[31] L. Dabbish, C. Stuart, J. Tsay, and J. Herbsleb, "Social coding in GitHub: Transparency and collaboration in an open software repository," in *Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work*, ser. CSCW '12. New York, NY, USA: ACM, 2012, pp. 1277–1286. [Online]. Available: http://doi.acm.org/10.1145/2145204.2145396

[32] G. Gousios, M.-A. Storey, and A. Bacchelli, "Work practices and challenges in pull-based development: The contributor's perspective," in *Proceedings of the 38th International Conference on Software Engineering*, ser. ICSE '16. New York, NY, USA: ACM, 2016, pp. 285–296. [Online]. Available: http://doi.acm.org/10.1145/2884781.2884826

[33] G. Pinto, I. Steinmacher, and M. A. Gerosa, "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, 2016, pp. 112–123.

[34] M.-A. Storey, A. Zagalsky, F. F. Filho, L. Singer, and D. M. German, "How social and communication channels shape and challenge a participatory culture in software development," *IEEE Trans. Softw. Eng.*, vol. 43, no. 2, pp. 185–204, Feb. 2017. [Online]. Available: https://doi.org/10.1109/TSE.2016.2584053

[35] C. Lebeuf, M. D. Storey, and A. Zagalsky, "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*, ser. CSCW '17, 2017. [Online]. Available: http://arxiv.org/abs/1702.07011

[36] C. Lebeuf, A. Zagalsky, M. Foucault, and M.-A. Storey, "Defining and classifying software bots: A faceted taxonomy," in *Proceedings of the 1st International Workshop on Bots in Software Engineering*, ser. BotSE '19. Piscataway, NJ, USA: IEEE Press, 2019, pp. 1–6. [Online]. Available: https://doi.org/10.1109/BotSE.2019.00008

[37] C. Lebeuf, M.-A. Storey, and A. Zagalsky, "Software bots," *IEEE Software*, vol. 35, no. 1, pp. 18–23, 2018.

[38] E. Paikari and A. van der Hoek, "A framework for understanding chatbots and their future," in *Proceedings of the 11th International Workshop on Cooperative and Human Aspects of Software Engineering*, ser. CHASE '18. New York, NY, USA: ACM, 2018, pp. 13–16. [Online]. Available: http://doi.acm.org/10.1145/3195836.3195859

[39] M. Wessel, A. Serebrenik, I. Wiese, I. Steinmacher, and M. A. Gerosa, "What to expect from code review bots on GitHub? a survey with OSS maintainers," in *SBES 2020 - Ideias Inovadoras e Resultados Emergentes*, oct 2020.

[40] L. F. Dias, I. Steinmacher, G. Pinto, D. A. D. Costa, and M. Gerosa, "How does the shift to GitHub impact project collaboration?" in *2016 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, Oct 2016, pp. 473–477.

[41] E. Kalliamvakou, G. Gousios, K. Blincoe, L. Singer, D. M. German, and D. Damian, "The promises and perils of mining GitHub," in *Proceedings of the 11th Working Conference on Mining Software Repositories*, ser. MSR 2014. New York, NY, USA: ACM, 2014, pp. 92–101. [Online]. Available: http://doi.acm.org/10.1145/2597073.2597074