What to Expect from Code Review Bots on GitHub? A Survey with OSS Maintainers

Mairieli Wessel
mairieli@ime.usp.br
University of São Paulo

Alexander Serebrenik
a.serebrenik@tue.nl
Eindhoven University of Technology

Igor Wiese
igor@utfpr.edu.br
Universidade Tecnológica Federal do Paraná

Igor Steinmacher
igor.steinmacher@nau.edu
Northern Arizona University

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

ABSTRACT
Software bots are used by Open Source Software (OSS) projects to streamline the code review process. Interfacing between developers and automated services, code review bots report continuous integration failures, code quality checks, and code coverage. However, the impact of such bots on maintenance tasks is still neglected. In this paper, we study how project maintainers experience code review bots. We surveyed 127 maintainers and asked about their expectations and perception of changes incurred by code review bots. Our findings reveal that the most frequent expectations include enhancing the feedback bots provide to developers, reducing the maintenance burden for developers, and enforcing code coverage. While maintainers report that bots satisfied their expectations, they also perceived unexpected effects, such as communication noise and newcomers’ dropout. Based on these results, we provide a series of implications for bot developers, as well as insights for future research.

CCS CONCEPTS
• Human-centered computing → Open source software. • Software and its engineering → Software creation and management.

KEYWORDS
software bots, pull-based model, open source software, code review

1 INTRODUCTION
Code review is a software quality assurance practice [8] common in Open Source Software (OSS) projects [3]. Since open source development involves a community of geographically dispersed developers [23], projects are often hosted on social coding platforms, such as GitHub [7]. To receive external contributions, repositories are shared by fork, and modified by pull requests. In the pull-based development model, project maintainers spend a non-negligible time inspecting code changes and engaging in discussion with contributors to understand and improve the modifications before integrating them into the codebase [15, 33].

Open source software communities use software bots to assist and streamline the code review process [9, 29]. In short, bots are software applications that integrate with human tasks, serving as interfaces that connect developers and other tools [26], and providing additional value to human users [12]. 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 [17]. According to Wessel et al. [29], code review bots differ from other bots by guiding contributors to provide necessary information before maintainers review the pull requests. On GitHub, these bots are responsible for leaving comments on pull requests, reporting continuous integration failures, code quality checks, and code coverage.

In theory, the automation provided by these bots should save maintainers effort and time [25], and lead them to focus on higher priority aspects of code review [2]. Nevertheless, the adoption of a code review bot, similar to any technological adoption, can bring unexpected consequences. Since, according to Mulder et al. [18], 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. In the case of the effect of software bots on project maintainers, this is often neglected.

In this paper, we aim to understand why open source maintainers integrate code review bots into the pull request workflow and how they perceive the changes these bots induce. In short, we answer the following research questions:

RQ1. What motivates maintainers to adopt code review bots?

RQ2. How do maintainers perceive the changes code review bots introduce to the software process?

To achieve our goal, we conducted a survey with 127 maintainers of OSS projects hosted on GitHub that adopted code review bots. We investigate the maintainers’ perceptions on whether project activity indicators change after bot adoption, such as the number of pull
requests received, merged, and non-merged, number of comments, and the time to close pull requests.

Analyzing the survey results, we found that maintainers were predominantly motivated by reducing their effort on tedious tasks to allow them to focus on more interesting ones, and enhancing the feedback communicated to developers. Regarding the changes introduced by the bot, we noted that less manual effort was required after adoption, a high-quality code was enforced, and pull request review sped up. However, four maintainers also reported unexpected aspects of bot adoption, including communication noise, more time spent on tests, newcomers’ dropout, and bots impersonating maintainers, which stressed out contributors.

Our contributions are twofold: (i) a set of maintainers’ motivations for using a bot to assist the code review process; and (ii) a discussion of how maintainers see the impact of bot introduction and support. These contributions may help maintainers anticipate bots’ effects on a project, and guide bot developers to consider the implications of new bots as they design them. Our findings, while preliminary, can suggest research hypotheses on the impact of code review bots on the code review process in open source projects, which follow-up studies can support or refute.

2 BACKGROUND AND RELATED WORK

Software bots have been designed to assist with the technical and social aspects of software development activities [13], including communication and decision-making [25]. Basically, these bots act as a conduit between software developers and other tools [25]. Wessel et al. have shown that bot adoption is indeed widespread in OSS projects hosted on GitHub [29]. GitHub bots have been developed to be integrated into the pull request workflow to perform a variety of tasks beyond code review support [31]. These tasks include repairing bugs [17, 27, 28], refactoring the code [32], recommending tools [4], detecting duplicated development [20], updating dependencies [16], and fixing static analysis violations [5].

Despite their increasing popularity, understanding the effects of bots is a major challenge. Storey and Zagalsky [25] and Paikari and van der Hoek [19] highlight that the potential negative impact of task automation through bot technology is still neglected. While bots are often used to avoid interruptions to developers’ work, they may lead to other, less obvious distractions [25]. Additionally, Liu et al. [14] claim that bots may have negative impacts on the user experience of open source contributors, since the needs and preferences of maintainers and contributors are not the same. While previous studies provide recommendations on how to evaluate bots’ capabilities and performance [1, 4], they do not draw attention to the impact of bot adoption on software development or on how software engineers perceive the bots’ effects.

Wessel et al. [29] 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. Wessel et al. [30] also employed a regression discontinuity design on OSS projects, revealing that the bot adoption increases the number of monthly merged pull requests, decreases monthly non-merged pull requests, and decreases communication among developers.

Prior work has also investigated the impact of continuous integration (CI) and code review tools on GitHub projects [6, 11, 34]. While Zhao et al. [34] and Cassee et al. [6] investigated the impact of the Travis CI tool’s introduction on development practices, Kavaler et al. [11] turned to the impact of linters, dependency managers, and coverage reporter tools. Our work extends the literature by providing an understanding of why code review bots are being adopted and the effects of such adoption, focusing on the perceptions of open source maintainers.

3 STUDY METHODOLOGY

We conducted a survey to obtain insights on how open source maintainers perceive the impact of using code review bots on pull requests and the effects of these bots on the project activities.

3.1 Survey Design

We first identified OSS projects hosted on GitHub that at some point had adopted at least one code review bot [29]. To find these projects, we queried the GHTorrent dataset [10], searching for projects that had received comments on pull requests from any of the code review bots identified by Wessel et al. [29]. For each project, we determined when a bot was introduced based on the date of the bot’s first comment. Afterwards, we contacted maintainers who merged more than one pull request before and after the bot adoption. To avoid duplicate invitations, we kept only the first record of maintainers who appeared in more than one project. Our initial target population comprised 1,960 maintainers of projects that adopted code review bots and made their e-mail addresses publicly available via the GitHub API.

To increase survey participation, we followed the best practices described by Smith et al. [21], such as sending personalized invitations and allowing participants to remain anonymous. The survey was set up as an online questionnaire, and it was sent on September 18, 2019. We received answers for 3 months and sent a reminder on October 2019. Participation was voluntary, and the estimated time to complete the survey was 10 minutes. We received answers from 127 maintainers, while the delivery of 26 messages failed. For this survey, we had a response rate of \( \approx 6.55\% \), which is consistent with other studies in software engineering [22].

Our maintainers’ survey had three main questions, which we made publicly available.1 In summary, we asked maintainers about their expectations and perception of changes caused by the adoption of a code review bot. Regarding the changes in the software process level, we asked maintainers about the same activity indicators studied by Wessel et al. [29]: the number of opened, merged, and non-merged pull requests, number of comments, and the time to close pull requests.

3.2 Data analysis

We used a card sorting approach [35] to qualitatively analyze the answers to the open-ended questions Q1 and Q3. Two researchers conducted card sorting in two steps. In the first step, each researcher analyzed the answers (cards) independently and applied codes to each answer, sorting them into meaningful groups. This step was followed by a discussion meeting until reaching a consensus on the

1https://zenodo.org/record/3992379#.Xz1_iSlKg3E
We asked maintainers what made them decide to start using bots.

Table 1: Reasons for adoption of code review bots

<table>
<thead>
<tr>
<th>Reasons</th>
<th># of answers (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enhance feedback to developers</td>
<td>31 (24.4%)</td>
</tr>
<tr>
<td>Reduce maintainers effort</td>
<td>30 (23.6%)</td>
</tr>
<tr>
<td>Enforce high code coverage</td>
<td>22 (17.3%)</td>
</tr>
<tr>
<td>Automate routine tasks</td>
<td>20 (15.7%)</td>
</tr>
<tr>
<td>Ensure high-quality standards</td>
<td>20 (15.7%)</td>
</tr>
<tr>
<td>Detect change effects</td>
<td>7 (5.5%)</td>
</tr>
<tr>
<td>Curiosity</td>
<td>5 (3.9%)</td>
</tr>
<tr>
<td>Improve interpersonal communication</td>
<td>5 (3.9%)</td>
</tr>
<tr>
<td>Lack of available tools</td>
<td>5 (3.9%)</td>
</tr>
<tr>
<td>Outside contributor’s suggestion</td>
<td>2 (1.6%)</td>
</tr>
</tbody>
</table>

From the maintainers’ perspective, the most recurrent motivation relates to enhancing the feedback to developers (31 mentions). This category includes cases in which the respondents’ desired to see both code review metrics and additional information “in a pretty and automated fashion” and “without having to go to another tool.” Several respondents recognized the value of bot feedback for both reviewers and contributors: “bots write useful information as comments and you can analyze it without switching the context.” In addition, other respondents pointed out the importance of “giving uniform feedback to all contributors” and “let[ting] contributors see how they affect the code.” Another two respondents mentioned that this kind of feedback might also increase contributors’ public accountability, giving reviewers “confidence that the author cares about testing” and about the quality of the code contribution.

Another recurrent reason regards reducing maintainers’ effort (30 mentions). Several maintainers were motivated by the necessity to save time and reduce their own effort during the code review process. Most of them said that reducing maintainers’ effort on trivial tasks, such as finding syntax errors and checking code style and coverage requirements, allows them to “spend more time on the important parts.” Moreover, the feedback provided by a code review bot helps maintainers avoid “repeating the same comments for each pull request.”

With 22 mentions, enforcing high code coverage during the code review process was the third most common reason. In general, respondents mentioned that code review bots were adopted to help detect and prevent reduction in code coverage. They also mentioned that these bots “ensure good coverage to allow changes on the code base with high confidence that the project will continue to function as expected” since they “don’t want to drop (significantly) in coverage.” Respondents (20) also reported another related reason: ensure high-quality standards. Respondents said that using code review bots for “automating repetitive tasks ensures they get done, increasing code quality” and “reduce[s] the risk of bugs being missed by reviewers.”

Several maintainers (20) were also motivated by automating routine tasks that previously were manually performed. Respondents mentioned the desire to automate routine tasks in order to structure the process of code review and “make the process more repeatable.” The routine tasks include tracking the coverage and “automatically upload[ing] code coverage results to a 3rd-party service.” Others provided more generic answers, briefly mentioning “automation.”

Maintainers were also motivated by curiosity to test a new technological tool and by a suggestion of an outside contributor. In the other five cases, our respondents were motivated by improving interpersonal communication, since “an automatic answer by a bot isn’t taken personally” and “it is a friendly way to ensure quality.” Moreover, a code review bot “improves interpersonal communication on pull requests and thus may reduce the chance a pull request is abandoned by the author.”

### 4 RESULTS

In this section, we report our main findings.

#### 4.1 Maintainers’ Motivations to Adopt a Code Review Bot

We asked maintainers what made them decide to start using bots to support code review activities. Four participants (3.15%) did not report any reason. The other answers were grouped into 10 categories, as can be seen in Table 1.

From the maintainers’ perspective, the most recurrent motivation relates to enhancing the feedback to developers (31 mentions). This category includes cases in which the respondents’ desired to see both code review metrics and additional information “in a pretty and automated fashion” and “without having to go to another tool.” Several respondents recognized the value of bot feedback for both reviewers and contributors: “bots write useful information as comments and you can analyze it without switching the context.” In addition, other respondents pointed out the importance of “giving uniform feedback to all contributors” and “let[ting] contributors see how they affect the code.” Another two respondents mentioned that this kind of feedback might also increase contributors’ public accountability, giving reviewers “confidence that the author cares about testing” and about the quality of the code contribution.

Another recurrent reason regards reducing maintainers’ effort (30 mentions). Several maintainers were motivated by the necessity to save time and reduce their own effort during the code review process. Most of them said that reducing maintainers’ effort on trivial tasks, such as finding syntax errors and checking code style and coverage requirements, allows them to “spend more time on the important parts.” Moreover, the feedback provided by a code review bot helps maintainers avoid “repeating the same comments for each pull request.”

With 22 mentions, enforcing high code coverage during the code review process was the third most common reason. In general, respondents mentioned that code review bots were adopted to help detect and prevent reduction in code coverage. They also mentioned that these bots “ensure good coverage to allow changes on the code base with high confidence that the project will continue to function as expected” since they “don’t want to drop (significantly) in coverage.” Respondents (20) also reported another related reason: ensure high-quality standards. Respondents said that using code review bots for “automating repetitive tasks ensures they get done, increasing code quality” and “reduce[s] the risk of bugs being missed by reviewers.”

Several maintainers (20) were also motivated by automating routine tasks that previously were manually performed. Respondents mentioned the desire to automate routine tasks in order to structure the process of code review and “make the process more repeatable”. The routine tasks include tracking the coverage and “automatically upload[ing] code coverage results to a 3rd-party service.” Others provided more generic answers, briefly mentioning “automation.”

Maintainers were also motivated by curiosity to test a new technological tool and by a suggestion of an outside contributor. In the other five cases, our respondents were motivated by improving interpersonal communication, since “an automatic answer by a bot isn’t taken personally” and “it is a friendly way to ensure quality.” Moreover, a code review bot “improves interpersonal communication on pull requests and thus may reduce the chance a pull request is abandoned by the author.”

#### 4.2 Maintainers’ Perceptions of Bots Effects

We also asked maintainers about their perspective on the potential changes to their projects that the code review bot introduced. The answers followed a 5-point Likert scale with neutral, ranging from “Strongly disagree” to “Strongly agree.” In Figure 1, we observe that most of the respondents did not agree with the expected impact of bot adoption on pull requests, considering the five studied activities indicators: number of pull requests received, merged, and non-merged; number of comments; and the time to close pull requests.

Most of the respondents claimed that there is no relation between the number of pull requests and the presence of the bot; they stated that the amount of opened pull requests “depends on bugs or features for the software.” However, one respondent claimed that it could lead to an increase in the number of pull requests, and “a better experience for everyone involved (which might eventually lead to repeat contributors).” Regarding merged and non-merged pull requests, maintainers claimed that these trends are typically “human factors” unrelated to bot adoption. One maintainer believed that the ability to filter out contributions that reduce code quality also reduces the merge rates of pull requests.
Respondents (36%) perceived an increase in the number of comments made to pull requests after bot adoption. One respondent claimed that this increase occurs because contributions that drastically reduce the coverage stimulate the exchange of comments between maintainers and contributors. Another maintainer explained that the number of comments increased because maintainers and "contributors started discussing how to best test something."

Maintainers believe (41% of them) that the code review bot helped decrease the time-to-close pull requests. One respondent did not agree with the statement, and left a comment telling us that the code review bot actually increased the time to merge pull requests, due to the need for additional time to write tests and obtain a stable code. Another maintainer commented that the bot increases the time to merge the contributions, though to them "it is not perceived as a bad thing."

We also openly asked maintainers about the changes introduced by the adoption of code review bots on the maintenance process and in the project itself. Twenty-three participants (18.1%) did not report any change. The other responses were grouped into 13 categories, as can be seen in Table 2.

The most recurrent reported change is that the adoption of code review bots requires less manual labor from maintainers (33 mentions). In general, respondents mentioned that the maintenance process is easier when they have fewer manual tasks to perform, because they "need to spend less time on it." The maintainers also suggest that bots could help reduce the number of human resources necessary to complete a task, which makes "it easier by reducing the number of review comments, general feedback and manual quality assurance required for a successful merge." Nevertheless, maintainers are also aware of the implications that "automation like this is always prone to non-fatal error."

Several maintainers (20) noticed changes in the quality of the contributions received, reporting that the bot helps to enforce high-quality code. In one example, a respondent mentioned that "the introduction of bots increased the quality of the code seen by maintainers in the initial review since contributors got timely (a few minutes) feedback about parts that failed basic quality standards such as missing tests, missing documentation, incorrect style, or broken functionality." Another 6 respondents also realized positive effects on the quality of the code review process, which "translate in a more efficient code review and more robust codebase in the long term."

Since one of the most common reasons to adopt a code review bot is to enforce code coverage, unsurprisingly, 16 respondents mentioned the increase in the code coverage after adoption. Most of the respondents reported that these bots help to "encourage to add more tests" when "the coverage is not good enough." One respondent stated the importance of the awareness of code coverage: "the effects are visible to the contributors, and they will generally resolve any decreased coverage in the pull request." Additionally, one respondent claimed that the bot feedback also "spurred further pull requests to increase coverage."

Another bot adoption effect is that reviewing pull requests became faster, which was reported by 16 maintainers. Three respondents mentioned that faster reviews lead to faster merging. A respondent stated that high-quality pull requests were more quickly identified since "the human review step was always started with a
Adding a code review bot to a project can represent the desire to bet-
as a feature maintainers desire. In our survey, maintainers claim it
support for newcomer onboarding both in terms of challenges and
communication noise.
Our study also reports four unexpected and negative effects of
issues, and coverage. situational awareness provided that contributors' confidence increased when a code review bot
do to have their contribution reviewed. Maintainers also noted
provides an immediate and clear sense of what contributors need
to focus on more important aspects of code review. Further-
maintainers started to spend less effort on trivial tasks, allowing
maintenance are in line with the reported motivations. Indeed,
high code coverage.
In fact, our results reveal that the predominant reason for using a
bot to support the code review process needs to consider the impact the
bot may have on both technical and social contexts. Based on our
results, further bot improvements can be envisioned. For example,
in order to prevent bots from introducing communication noise, bot
developers should know when and to what extent the bot should interrupt a human [14, 24].
Improving bots’ design. Anyone who wants to develop a bot
to support the code review process needs to consider the impact the
bot may have on both technical and social contexts. Based on our
results, further bot improvements can be envisioned. For example,
in order to prevent bots from introducing communication noise, bot
developers should know when and to what extent the bot should interrupt a human [14, 24].
Improving newcomers support. As aforementioned, previous
literature on bots already mentioned a lack of support for newcom-
er [29]. It is reasonable to expect that newcomers who receive
friendly feedback will have a higher engagement level and thus sus-
tain their participation on the project. Hence, future research can
help bot designers by providing guidelines and insights to support
new contributors.
6 THREATS TO VALIDITY
Since we leverage qualitative research methods to categorize the
open-ended questions asked in our survey, we may have introduced
categorization bias. To mitigate this bias, we conducted this process
in pairs and carefully discussed categorization among the authors.
Regarding our survey, the order that we presented the questions
to the respondents may have influenced the way they answered
them. In addition, we cannot guarantee that maintainers correctly
in the questions’ context.
RQ2). Based on these preliminary findings, future
research can focus on better supporting and understanding bots’
influences on social interactions in the context of OSS projects.
Moreover, future work can investigate the effects of adopting a bot and the expansion of our analysis for other types of bots, activity indicators, and social coding platforms.

**ACKNOWLEDGMENTS**

We thank all the participants of this study, who volunteered to support our research. 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**


