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

ACM Reference Format:
Mairieli Wessel, Alexander Serebrenik, Igor Wiese, Igor Steinmacher, and Marco A. Gerosa. 2020. What to Expect from Code Review Bots on GitHub? A Survey with OSS Maintainers. In *34th Brazilian Symposium on Software Engineering (SBES '20), October 21–23, 2020, Natal, Brazil.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3422392.3422459

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

SBES '20, October 21–23, 2020, Natal, Brazil

ACM ISBN 978-1-4503-8753-8/20/09…$15.00
https://doi.org/10.1145/3422392.3422459

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 $\simeq 6.55\%$, 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

[1] https://zenodo.org/record/3992379#.Xz1_iSlKg3E

Table 1: Reasons for adoption of code review bots

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

code names and categorization of each item. At the end of this process, the answers were sorted into high-level groups. In the second step, the researchers analyzed the categories, aiming to refine the classification and group-related codes into more significant, higher-level categories and themes. We used *open* card sorting, meaning we had no predefined codes or groups; the codes emerged and evolved during the analysis process. In addition, we quantitatively analyzed closed-ended question (Q2) to understand developers' perceptions of the impact of bots on pull requests.

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.*"

Answer to RQ1. Maintainers reported 10 reasons for using code review bots. We found that several maintainers were motivated by enhancing the feedback to developers (24.4%), reducing their own efforts (23.6%), and enforcing high code coverage (17.3%).

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.

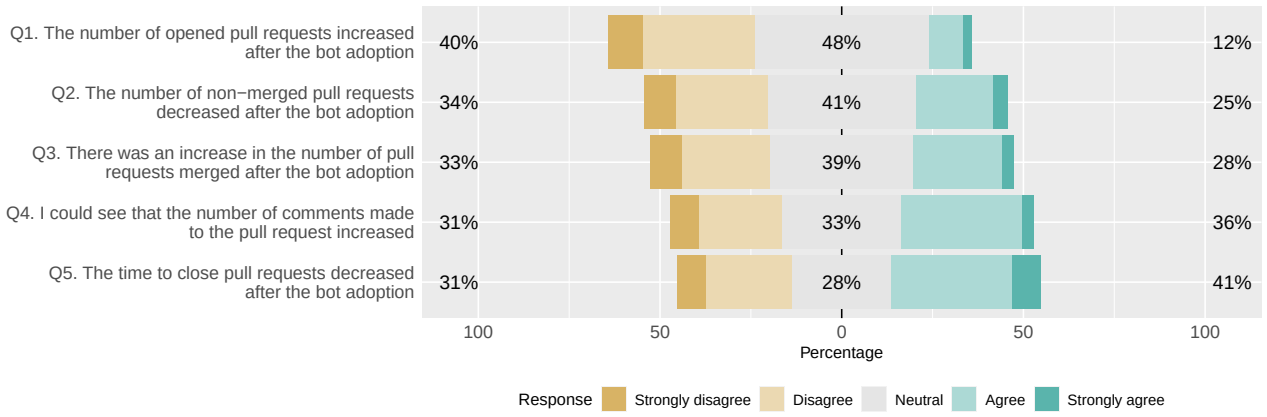

Figure 1: Maintainers' perspectives on the effects of bot adoption on pull requests activities

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.*"

Table 2: Changes to the maintenance process introduced by code review bot adoption

Changes	# of answers (%)
Less manual labor required from maintainers	33 (25.9%)
Bots enforce high-quality code	20 (15.7%)
Bots increase code coverage	16 (12.6%)
Faster pull request review	16 (12.6%)
Bots help identify missing tests	10 (7.8%)
Bots help with standardization	8 (6.3%)
Bots improve the quality of code review	7 (5.5%)
Bots provide consistent and immediate feedback	7 (5.5%)
Bots reduce communication barriers	5 (3.9%)
Newcomers feel intimidated	2 (1.5%)
Bots impersonate human developers	1 (0.7%)
Bots introduce communication noise	1 (0.7%)
Testing starts to require more time than development	1 (0.7%)

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*

baseline level of quality" and thus merged faster. In addition, another maintainer reinforced the efficiency of this process: "*some of the bots do it so well, that we can merge pull requests immediately after opening it.*" In addition, 7 maintainers also reported that the ***quality of the code review process improved***.

Other categories, although less recurrent, called our attention to the negative effects reportedly caused by bot adoption. One respondent said that ***bots intimidate newcomers***, since some newcomers close their pull requests after a bot comment. Another believes that, for a newcomer, receiving an assessment "*you let coverage go down,*" instead of a "*thanks for your contribution,*" "*can be a little daunting.*" Respondents also mentioned that after adoption ***testing started to require more time than development*** and the ***bot's comments introduced noise***. Another respondent said that a bot can ***impersonate human developers*** due to bots' strict rules, which stressed out contributors.

Answer to RQ2. Among the positive changes incurred from code review bots, maintainers reported that less manual labor was required after bot adoption (25.9%) and bots enforced high-quality code (15.7%). The negative effects include communication noise, more time spent with tests, newcomers' dropout, and bots impersonating maintainers.

5 DISCUSSION AND IMPLICATIONS

Adding a code review bot to a project can represent the desire to better communicate with developers, helping contributors and maintainers be more effective, and achieving improved interpersonal communication, as already discussed by Storey and Zagalsky [25]. In fact, our results reveal that the predominant reason for using a code review bot is to improve the feedback communicated to developers. Moreover, maintainers are also interested in automating code review tasks to reduce the maintenance burden and enforce high code coverage.

Most of the maintainers' perceptions of how bots impact on maintenance are in line with the reported motivations. Indeed, maintainers started to spend less effort on trivial tasks, allowing them to focus on more important aspects of code review. Furthermore, code review bots guide contributors toward detecting change effects before maintainers triage the pull requests [29], ensuring high-quality standards and a faster code review. Bots' feedback provides an immediate and clear sense of what contributors need to do to have their contribution reviewed. Maintainers also noted that contributors' confidence increased when a code review bot provided *situational awareness* [25], indicating *standards, language issues, and coverage* to contributors.

On the one hand, adopting a bot save maintainers' costs, time, and effort during the code review activities. On the other hand, our study also reports four unexpected and negative effects of adopting a bot to assist the code review process. Such effects include *communication noise*, *more time spent with tests*, *newcomers' dropout*, and *bots impersonating maintainers*. Although less recurrent, these effects are non-negligible to the OSS community.

Previous work by Wessel et al. [29] has already mentioned the support for newcomer onboarding both in terms of challenges and as a feature maintainers desire. In our survey, maintainers claim it is easier for newcomers to submit a high-quality pull request with only the intervention of bots. However, another maintainer pointed out that when newcomers and casual contributors receive feedback from the bot, it can lead to rework, discussions, and ultimately dropping out from contributing.

Our study suggests practical implications for practitioners as well as insights and suggestions for researchers.

Awareness of bot effects. Indeed, the maintenance activities changed following the adoption of code review bots. This change can directly affect contributors' and maintainers' work. Hence, understanding how the code review bot adoption affects a project is important for practitioners, 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.

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 newcomers [29]. 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 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 understood sentences 4 and 5. We tried to order the questions based on the natural sequence of actions to help respondents understand the questions' context.

7 FINAL CONSIDERATIONS

In this work, we conducted a preliminary investigation into maintainers' perceptions of the effects of adopting bots to support the code review process on pull requests. The most frequently mentioned motivations for using bots including automating repetitive tasks, improving tools' feedback to developers and reducing maintenance effort (**RQ1**). Moreover, maintainers cite several benefits of bots, such as decreasing the time to close pull requests and reducing the workload with laborious and repetitive tasks. However, maintainers also stated negative effects, including the introduction of noise and (**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

[1] Ahmad Abdellatif and Emad Shihab. 2020. MSRBot: Using Bots to Answer Questions from Software Repositories. *Empirical Software Engineering (EMSE)* 25 (2020), 1834–1863. https://doi.org/10.1007/s10664-019-09788-5

[2] Alberto Bacchelli and Christian Bird. 2013. Expectations, outcomes, and challenges of modern code review. In *2013 35th International Conference on Software Engineering (ICSE)*. IEEE, 712–721.

[3] Olga Baysal, Oleksii Kononenko, Reid Holmes, and Michael W Godfrey. 2016. Investigating technical and non-technical factors influencing modern code review. *Empirical Software Engineering* 21, 3 (2016), 932–959.

[4] Chris Brown and Chris Parnin. 2019. Sorry to Bother You: Designing Bots for Effective Recommendations. In *Proceedings of the 1st International Workshop on Bots in Software Engineering* (Montreal, Quebec, Canada) *(BotSE '19)*. IEEE Press, Piscataway, NJ, USA, 54–58. https://doi.org/10.1109/BotSE.2019.00021

[5] A. Carvalho, W. Luz, D. Marcílio, R. Bonifácio, G. Pinto, and E. Dias Canedo. 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)*. 161–171.

[6] Nathan Cassee, Bogdan Vasilescu, and Alexander Serebrenik. 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.

[7] Laura Dabbish, Colleen Stuart, Jason Tsay, and Jim Herbsleb. 2012. Social Coding in GitHub: Transparency and Collaboration in an Open Software Repository. In *CSCW*. ACM, New York, NY, USA, 1277–1286.

[8] Felipe Ebert, Fernando Castor, Nicole Novielli, and Alexander Serebrenik. 2019. Confusion in code reviews: Reasons, impacts, and coping strategies. In *2019 IEEE 26th International Conference on Software Analysis, Evolution and Reengineering (SANER)*. IEEE, 49–60.

[9] Linda Erlenhov, Francisco Gomes de Oliveira Neto, Riccardo Scandariato, and Philipp Leitner. 2019. Current and Future Bots in Software Development. In *Proceedings of the 1st International Workshop on Bots in Software Engineering* (Montreal, Quebec, Canada) *(BotSE '19)*. IEEE Press, Piscataway, NJ, USA, 7–11. https://doi.org/10.1109/BotSE.2019.00009

[10] Georgios Gousios and Diomidis Spinellis. 2012. GHTorrent: GitHub's data from a firehose. In *2012 9th IEEE Working Conference on Mining Software Repositories (MSR)*. IEEE, 12–21.

[11] David Kavaler, Asher Trockman, Bogdan Vasilescu, and Vladimir Filkov. 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, 476–487.

[12] Carlene Lebeuf, Alexey Zagalsky, Matthieu Foucault, and Margaret-Anne Storey. 2019. Defining and Classifying Software Bots: A Faceted Taxonomy. In *Proceedings of the 1st International Workshop on Bots in Software Engineering* (Montreal, Quebec, Canada) *(BotSE '19)*. IEEE Press, Piscataway, NJ, USA, 1–6. https://doi.org/10.1109/BotSE.2019.00008

[13] Bin Lin, Alexey Zagalsky, Margaret-Anne Storey, and Alexander Serebrenik. 2016. Why developers are slacking off: Understanding how software teams use slack. In *Proceedings of the 19th ACM Conference on Computer Supported Cooperative Work and Social Computing Companion*. ACM, 333–336.

[14] Dongyu Liu, Micah J. Smith, and Kalyan Veeramachaneni. 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* (Honolulu, HI, USA) *(CHI EA '20)*. Association for Computing Machinery, New York, NY, USA, 1–8. https://doi.org/10.1145/3334480.3382998

[15] Shane McIntosh, Yasutaka Kamei, Bram Adams, and Ahmed E Hassan. 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*. 192–201.

[16] Samim Mirhosseini and Chris Parnin. 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* (Urbana-Champaign, IL, USA) *(ASE 2017)*. IEEE Press, Piscataway, NJ, USA, 84–94. http://dl.acm.org/citation.cfm?id=3155562.3155577

[17] Martin Monperrus. 2019. Explainable Software Bot Contributions: Case Study of Automated Bug Fixes. In *Proceedings of the 1st International Workshop on Bots in Software Engineering* (Montreal, Quebec, Canada) *(BotSE '19)*. IEEE Press, Piscataway, NJ, USA, 12–15. https://doi.org/10.1109/BotSE.2019.00010

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

[19] Elahe Paikari and André van der Hoek. 2018. A Framework for Understanding Chatbots and Their Future. In *Proceedings of the 11th International Workshop on Cooperative and Human Aspects of Software Engineering* (Gothenburg, Sweden) *(CHASE '18)*. ACM, New York, NY, USA, 13–16. https://doi.org/10.1145/3195836.3195859

[20] Luyao Ren, Shurui Zhou, Christian Kästner, and Andrzej Wąsowski. 2019. Identifying Redundancies in Fork-based Development. In *2019 IEEE 26th International Conference on Software Analysis, Evolution and Reengineering (SANER)*. IEEE, 230–241.

[21] Edward Smith, Robert Loftin, Emerson Murphy-Hill, Christian Bird, and Thomas Zimmermann. 2013. Improving developer participation rates in surveys. In *2013 6th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)*. IEEE, 89–92.

[22] Igor Steinmacher, Gustavo Pinto, Igor Scaliante Wiese, and Marco A. Gerosa. 2018. Almost There: A Study on Quasi-contributors in Open Source Software Projects. In *Proceedings of the 40th International Conference on Software Engineering* (Gothenburg, Sweden) *(ICSE '18)*. ACM, New York, NY, USA, 256–266. https://doi.org/10.1145/3180155.3180208

[23] Igor Fábio Steinmacher. 2015. *Supporting newcomers to overcome the barriers to contribute to open source software projects.* Ph.D. Dissertation. Universidade de São Paulo.

[24] Margaret-Anne Storey, Alexander Serebrenik, Carolyn Penstein Rosé, Thomas Zimmermann, and James D. Herbsleb. 2020. BOTse: Bots in Software Engineering (Dagstuhl Seminar 19471). *Dagstuhl Reports* 9, 11 (2020), 84–96.

[25] Margaret-Anne Storey and Alexey Zagalsky. 2016. Disrupting Developer Productivity One Bot at a Time. In *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering* (Seattle, WA, USA) *(FSE 2016)*. ACM, New York, NY, USA, 928–931. https://doi.org/10.1145/2950290.2983989

[26] Margaret-Anne Storey, Alexey Zagalsky, Fernando Figueira Filho, Leif Singer, and Daniel M. German. 2017. How Social and Communication Channels Shape and Challenge a Participatory Culture in Software Development. *IEEE Trans. Softw. Eng.* 43, 2 (Feb. 2017), 185–204. https://doi.org/10.1109/TSE.2016.2584053

[27] Simon Urli, Zhongxing Yu, Lionel Seinturier, and Martin Monperrus. 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* (Gothenburg, Sweden) *(ICSE-SEIP '18)*. ACM, New York, NY, USA, 95–104. https://doi.org/10.1145/3183519.3183540

[28] Rijnard van Tonder and Claire Le Goues. 2019. Towards s/Engineer/Bot: Principles for Program Repair Bots. In *Proceedings of the 1st International Workshop on Bots in Software Engineering* (Montreal, Quebec, Canada) *(BotSE '19)*. IEEE Press, Piscataway, NJ, USA, 43–47. https://doi.org/10.1109/BotSE.2019.00019

[29] Mairieli Wessel, Bruno Mendes de Souza, Igor Steinmacher, Igor S. Wiese, Ivanilton Polato, Ana Paula Chaves, and Marco A. Gerosa. 2018. The Power of Bots: Characterizing and Understanding Bots in OSS Projects. *Proceedings of the ACM Conference on Computer Supported Cooperative Work Social Computing* 2, CSCW, Article 182 (Nov. 2018), 19 pages. https://doi.org/10.1145/3274451

[30] Mairieli Wessel, Alexander Serebrenik, Igor Scaliante Wiese, Igor Steinmacher, and Marco Aurelio Gerosa. 2020. Effects of Adopting Code Review Bots on Pull Requests to OSS Projects. In *IEEE International Conference on Software Maintenance and Evolution*. IEEE Computer Society.

[31] Mairieli Wessel and Igor Steinmacher. 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

[32] Marvin Wyrich and Justus Bogner. 2019. Towards an Autonomous Bot for Automatic Source Code Refactoring. In *Proceedings of the 1st International Workshop on Bots in Software Engineering* (Montreal, Quebec, Canada) *(BotSE '19)*. IEEE Press, Piscataway, NJ, USA, 24–28. https://doi.org/10.1109/BotSE.2019.00015

[33] Yue Yu, Huaimin Wang, Vladimir Filkov, Premkumar Devanbu, and Bogdan Vasilescu. 2015. Wait for It: Determinants of Pull Request Evaluation Latency on GitHub. In *2015 IEEE/ACM 12th Working Conference on Mining Software Repositories*. 367–371. https://doi.org/10.1109/MSR.2015.42

[34] Yangyang Zhao, Alexander Serebrenik, Yuming Zhou, Vladimir Filkov, and Bogdan Vasilescu. 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, 60–71.

[35] Thomas Zimmermann. 2016. Card-sorting: From text to themes. In *Perspectives on Data Science for Software Engineering*. Elsevier, 137–141.