Perception of Software Bots on Pull Requests on Social Coding Environments

Mairieli Wessel[1,2], **Igor Steinmacher**[3], **Marco A. Gerosa**[1,3]

[1] University of São Paulo, Brazil

[2]Radboud University, The Netherlands

[3]Northern Arizona University, USA

mairieli.wessel@ru.nl, {igor.steinmacher,marco.gerosa}@nau.edu

Abstract. *Software bots connect users and tools, streamlining the pull request review process in social coding platforms. Nevertheless, the presence of bots can be annoying and disruptive to the community. Interviewing open-source developers, we created a theory of how human developers perceive annoying bot behaviors as noise. To identify strategies to overcome such problems, we applied Design Fiction as a participatory method with practitioners. Our findings may help practitioners understand the effects of adopting a bot, and researchers and tool designers may leverage our results to better support human-bot interaction on social coding platforms.*

Resumo. *Os software bots conectam usuários e ferramentas, simplificando o processo de revisão de pull requests em plataformas sociais de codificação. No entanto, a presença destes bots pode ser prejudicial para a comunidade. Entrevistando desenvolvedores de código aberto, criamos uma teoria de como desenvolvedores humanos percebem comportamentos irritantes de bots como ruído. Para identificar estratégias para superar tais problemas, aplicamos o Design Fiction como método participativo com desenvolvedores e pesquisadores. Nossas descobertas podem ajudar os profissionais a entender os efeitos da adoção de um bot, e pesquisadores e designers de ferramentas podem utilizar nossos resultados para apoiar a interação humano-bot em plataformas sociais de codificação.*

1. Introduction

Open-source software development is inherently collaborative, frequently involving geographically dispersed contributors. OSS projects often are hosted in social coding platforms, such as GitHub and GitLab, which provide features that aid collaboration and sharing, such as pull requests [Tsay et al. 2014]. Pull requests facilitate interaction among developers to review and integrate code contributions. In the pull-based development model, project maintainers carefully inspect code changes and engage in discussion with contributors to understand and improve the modifications before integrating them into the codebase [McIntosh et al. 2014]. The time maintainers spend reviewing pull requests is non-negligible and can affect, for example, the volume of new contributions [Yu et al. 2015] and the onboarding of newcomers [Steinmacher et al. 2013].

Software bots play a prominent role in the pull request review process [Wessel et al. 2018]. These automation tools serve as an interface between users

and other tools [Storey and Zagalsky 2016] and reduce the workload of maintainers and contributors. Accomplishing tasks that were previously performed solely by human developers, and interacting in the same communication channels as their human counterparts, bots have become new voices in the pull request conversation [Monperrus 2019]. Throughout comments on pull requests, bots guide contributors to provide necessary information before maintainers triage the pull requests [Wessel et al. 2018]. To alleviate their workload [Gousios et al. 2016], project maintainers often rely on software bots to check whether the code builds, the tests pass, and the contribution conforms to a defined style guide [Vasilescu et al. 2015, Kavaler et al. 2019]. More complex tasks include repairing bugs [Urli et al. 2018, Monperrus 2019], refactoring source code [Wyrich and Bogner 2019], recommending tools [Brown and Parnin 2019], updating dependencies [Mirhosseini and Parnin 2017], and fixing static analysis violations [Carvalho et al. 2020].

The introduction of bots aims to save cost, effort, and time [Storey and Zagalsky 2016], allowing maintainers to focus on development and review tasks. However, new technology often brings consequences that counter designers' and adopters' expectations [Healy 2012]. Developers who *a priori* expect technological developments to lead to performance improvements can be caught off-guard by *a posteriori* unanticipated operational complexities and collateral effects [Woods and Patterson 2001]. For example, we have shown that although the number of human comments decreases after the introduction of bots [Wessel et al. 2020b], many developers do not perceive this decrease [Wessel et al. 2020a]. These collateral effects and the misalignment between the preferences and needs of project maintainers and bot developers can cause expectation breakdowns, as illustrated by a developer complaining on social media: "*Whoever wrote [bot-name] fundamentally does not understand software development.*"[1] Moreover, as bots have become new voices in developers' conversation [Monperrus 2019], they may overburden developers who already suffer from information overload when communicating online [Nematzadeh et al. 2016]. On an abandoned pull request, a maintainer noted the frequency of actions of a bot: "*@bot-name seems pretty active here [...].*"[2] As the introduction of a technology may provoke changes in human behavior [Mulder 2013], it is important to understand how bots affect the group dynamics; yet, this is often neglected [Storey and Zagalsky 2016, Paikari and van der Hoek 2018].

Considering developers' perspectives on the overall effects of introducing bots, designers can revisit their bots to better support the interactions in the development workflow and account for collateral effects. So far, the literature presents scarce evidence, and only as secondary results, of the challenges incurred when adopting bots. According to [Brown and Parnin 2019], the human-bot interaction on pull requests can be inconvenient, leading developers to abandon their contributions due to poor bots' design. This problem may be especially relevant for newcomers, who require special support during the onboarding process due to the barriers they face [Steinmacher et al. 2016]. Newcomers can perceive bots' complex answers as discouraging, since bots often provide a long list of critical contribution feedback (e.g., style guidelines, failed tests), rather than supportive

[1] https://twitter.com/mojavelinux/status/1125077242822836228
[2] https://github.com/facebook/react/pull/12457\#issuecomment-413429168

assistance.

To make bots more effective at communicating to developers, design problems need to be solved to avoid repetitive notifications, provide consistency in the tasks being done, and make bots adaptive [Storey et al. 2020, Liu et al. 2020]. Designers should envision bots as socio-technical rather than purely technical applications, considering human interaction, developers' collaboration, and ethical concerns [Storey and Zagalsky 2016]. The adoption of bots in open-source projects is a recent trend and the literature lacks design strategies that include the end-users' perspective to enhance the bots interaction on social coding platforms. Considering this context, in this study we aim at identifying the challenges incurred by bots on pull requests and interventions to mitigate them.

The main contributions of this paper are:

1. Open-source developers' perspective on the importance and support of bots.
2. The identification of changes in project activity indicators after the adoption of a bot.
3. A theory of how human developers perceive annoying bot behaviors as noise on social coding platforms.
4. Design strategies and interventions that could enhance current bot interaction and reduce noise.

With a more in-depth understanding of the challenges incurred by the bots' interaction and promising ways to enhance the human-bot interaction, researchers and practitioners can invest their efforts in designing or improving bots, ultimately supporting developers in submitting and reviewing pull requests.

2. Related Work

According to [Storey and Zagalsky 2016], a software development bot is *"a conduit or an interface between users and services, typically through a conversational user interface"*. In the following, we provide more details about the existing literature related to the human-bot interaction, especially on social coding platforms.

2.1. Bots on social coding platforms

Software development bots assist developers in their daily tasks such as generating bug fixes [Serban et al. 2021], answering developers' technical questions [Abdellatif et al. 2020], and performing code refactoring [Wyrich and Bogner 2019]. Certain bots have been studied in detail, revealing challenges and limitations of their interventions in pull requests. For example, while analyzing the *tool-recommender-bot*, [Brown and Parnin 2019] report that bots still need to overcome problems such as notification workload. [Mirhosseini and Parnin 2017] analyzed the *greenkeeper* bot and found that maintainers were often overwhelmed by notifications and only a third of the bots' pull requests were merged into the codebase. [Peng and Ma 2019] conducted a case study on how developers perceive and work with *mention bot*. The results show that this bot has saved developers' efforts; however, it may not meet the diverse needs of all users. For example, while project owners require simplicity and stability, contributors require transparency, and reviewers require selectivity. Despite its potential benefits, results also show that developers can be bothered by frequent review notifications when dealing with a heavy workload.

2.2. Mediator bots

Mediator bots have been proposed in the literature in different domains. For example, Sadeddin *et al.* [Sadeddin et al. 2007] showed that a meta-bot could obtain product information from several shopping bots and summarize the information before presenting it to users. Previous research also investigated the user experience when interacting with single- vs. multi-bot conversational systems. In a Wizard-of-Oz study, Chaves and Gerosa [Chaves and Gerosa 2018] showed that participants reported more confusion in a multi-bot scenario than when using a meta-bot. The meta-bot concept also appears in the literature on software agents. Generalist agents are usually referred to as Super Bots or mediator bots [Dagli 2019], since they often combine multiple tasks and functionalities of specialist agents into a single agent. Given this preliminary evidence from other domains, we hypothesize that *a mediator bot can organize information and reduce information overload created by other bots around pull requests.*

3. Research Design

As a warm-up study, we started our investigation by characterizing the bots that support pull requests on GitHub. We sampled 351 popular projects from GitHub and surveyed 228 open-source developers. Our results indicate that bot adoption is widespread in open-source software projects and is used to perform a variety of tasks on pull requests. Developers also report some challenges of using bots on pull requests. Several contributors complained about the way the bots interact, saying that the bots provide non-comprehensive or poor feedback. In contrast, others mentioned that bots introduce communication noise and that there is a lack of information on how to interact with the bot. To further understand the challenges related to bot adoption and how to deal with them, we devised a study[3] split into three phases to answer the following research questions:

Research Question 1

How do **pull request activities change** after a bot is adopted in a project?

Since bots may bring unexpected impacts to group dynamics, as frequently occurs with new technology adoption, understanding and anticipating such effects is important for planning and management. In this RQ, we investigate whether project activity indicators, such as the number of pull requests merged and non-merged, number of comments, the time to close pull requests, and number of commits change after bot adoption. We used a *Regression Discontinuity Design* [Thistlethwaite and Campbell 1960] to model the effect of bot adoption across 1,194 projects that had adopted bots for at least one year. Afterward, to further shed light on our results, we conducted 12 semi-structured interviews with practitioners, including open-source project maintainers and contributors experienced with bots. Our results indicate that the adoption of bots, in fact, changes the dynamics of pull request activities. These results motivated us to investigate further developers' perspectives on the overall effects of introducing bots to open-source projects.

Research Question 2

What **interaction challenges** do bots introduce when supporting pull requests?

[3]Our institutional review board approved the research protocol

To understand the impact of bots interaction in-depth, we then focused on investigating the challenges incurred by bots interaction on pull requests. We identified several challenges caused by bots in pull request interactions by interviewing 21 practitioners, including project maintainers, contributors, and bot developers. In particular, our findings indicate noise as a recurrent and central problem. Noise affects both human communication and development workflow by overwhelming and distracting developers. The main contribution for this research question is a theory of how human developers perceive annoying bot behaviors as noise on social coding platforms.

Research Question 2

What design strategies can potentially **reduce the noise created by bots** on pull requests?

As noise emerged as a central interaction challenge from our empirical analysis, we have further investigated how to overcome it. We created two interventions: (1) a mediator bot that organizes existing bot information in a pull request, and (2) a separate interface for the bot interaction in the pull request. To design and implement the interventions, we applied Design Fiction [Blythe 2014], a technique that has been broadly used in the Human-Computer Interaction field to explore and critique future technologies. We presented to 32 open-source maintainers, contributors, bot developers, and bot researchers a fictional story of a mediator bot capable of better supporting developers' interactions on pull requests and operating as a mediator between developers and the existing bots. During synchronous design fiction sessions, participants answered questions to complete the end of the fictional story, discussing the design strategies for the mediator bot and raising concerns about the use of bots.

4. Findings

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

4.1. Effects of adopting a bot on pull requests (RQ1)

We start by investigating the effects of bot adoption on the number of merged and non-merged pull requests, and the number of pull requests merged and non-merged, number of comments, the time to close pull requests, and number of commits on both merged and non-merged pull requests. We fit eight mixed-effect regression discontinuity design models for those variables. Analyzing the models, we found that after bot adoption more pull requests are merged into the codebase, and communication decreases between contributors and maintainers. Considering non-merged pull requests, after bot adoption projects have fewer monthly non-merged pull requests, and faster pull request rejections.

We then presented to open-source developers the main findings of our statistical approach: "*After adopting a code review bot there are more merged pull requests, less communication between developers, fewer rejected pull requests, and faster rejections*." We asked them to conjecture on the possible reasons for each of these results. Project maintainers and contributors reported several reasons for more merged pull requests, fewer comments, and fewer and faster rejections. Most of the participants claimed that when a project has bots that provide detailed information on code quality metrics, especially in the sense of coverage metrics, both maintainers and contributors can more

quickly gain a general idea of the quality of the contributions. According to the participants, one of the reasons for more pull requests being merged after the code review bot introduction is that these bots act as quality gatekeepers. Although less recurrent, participants mentioned that in some cases bot comments might be perceived as noise by developers, which disrupts the conversation in the pull request.

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

Research Question 1

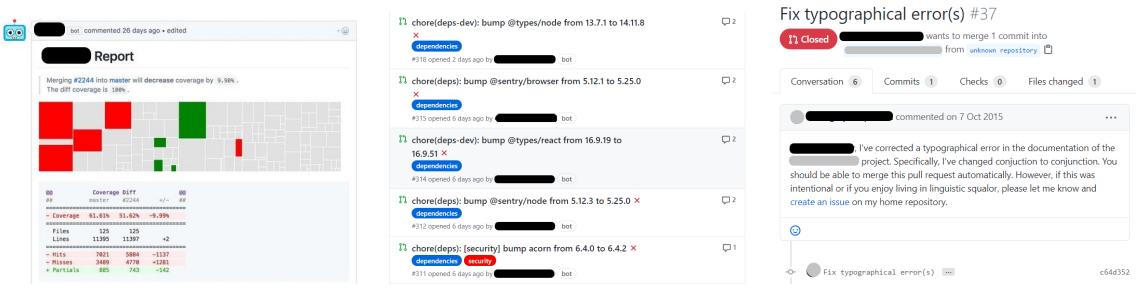
How do **pull request activities change** after a bot is adopted in a project?

We have found that the adoption of bots helps developers merge more pull requests and reduces the need for communication between developers. According to them, bot comments help them to understand the state and quality of the contribution, making maintainers more confident to merge pull requests, which also changes the focus of developer discussions.

4.2. Challenges of bots on pull requests (RQ2)

Interviewing practitioners, we have found three categories of reported challenges: interaction, adoption, and development challenges. The most recurrent and central problem reported by our interviewees was the introduction of noise into the developers' communication channel. This problem was a crosscutting concern related to bots' development, adoption, and interaction in open-source software projects. Some developers complained about annoying bot behaviors. Those behaviors include the case in which bots provide comments with dense information "*in the middle of the pull request*", frequently overusing visual elements and the case in which bots perform repetitive actions, such as creating numerous pull requests and leaving dozens of comments in a row. These behaviors are often perceived as *noise*, which can lead to information and notification overload, which disrupts both human communication and development workflow. To handle the challenges provoked by noise, developers rely on countermeasures, such as re-configuring or re-designing the bot.

To illustrate the described behaviors, we highlighted some examples cited by our participants and described in the state-of-the-practice. Figure 1a shows the case of a verbose comment, which included a lot of information and many graphical elements, inserted by a bot in the middle of a human conversation. In Figure 1b, we show a bot overloading a single repository with many pull requests, even if there were opened pull requests by the same bot. Finally, Figure 1c depicts a bot spamming a repository with an unsolicited pull request.



(a) Verbosity (b) High frequent actions (c) Unsolicited actions

Figura 1. Examples of annoying behaviors from the state-of-the-practice

Research Question 2

What **interaction challenges** do bots introduce when supporting pull requests?

Noise is a central challenge in bots' interactions on pull requests. In short, we found that the noise introduced by bots leads to information overload, which interferes with how humans communicate, work, and collaborate on social coding platforms.

4.3. Mediator bot and separate interface for bots (RQ3)

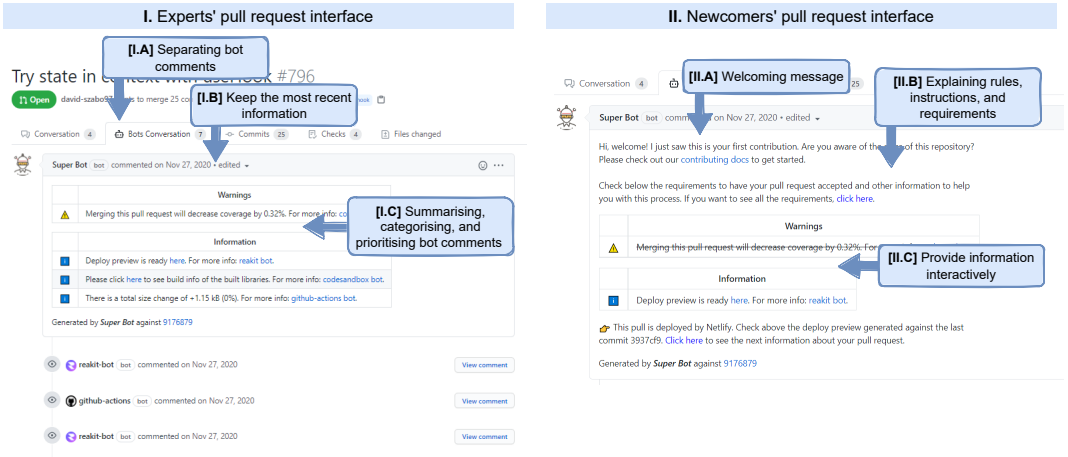


Figura 2. Prototype of the interventions in a real-world scenario on GitHub.

One recommendation that appeared frequently in the design fiction sessions was to offer different views for experienced developers and newcomers. Therefore, we split our prototype into two different versions: the experts' pull request interface (see Figure 2-I), designed to support maintainers and experienced contributors; and the newcomers' pull request interface (see Figure 2-II). We designed a specific place for all information and events regarding bots in the pull request (Figure 2-I.A). The mediator bot creates a *summary* with the most important information about each bot, then groups them into *categories* (e.g., "warnings", "information") (Figure 2-I.C).

To avoid inflating the pull requests with several comments from the mediator bot, one suggested strategy is to *keep the most recent information*. We include the latest information from each bot in the summary (Figure2-I.B). Reakit bot, for example, posted two comments in the timeline of bot events; however, only one entry is displayed in the

summarized table for that bot. In addition, in the timeline of bot events, it is possible to expand all bot comments to see the complete messages.

An important distinction between the two versions is the way the mediator bot displays the information for newcomers versus experts. In Figure 2-II.C, we present the interactive process of displaying bots' information. The mediator bot guide newcomers by showing the information from other bots "*step by step*." Study participants deemed this strategy a potential solution to reduce the impact of receiving several different bot notifications at once. As part of this guidance, the mediator bot also refers to contribution guidelines to assist the newcomers and present a concise and direct welcoming message.

To assess the developers' perception of the designed interface, we interviewed fifteen practitioners. The interviewees provided their feedback on particular aspects of our findings, expressing their preferences about the elements of the designed prototype. We also applied the Technology Acceptance Model (TAM) [Davis 1989], administering a questionnaire immediately after concluding each interview.

The participants who gave feedback were, overall, positive about the prototype. They perceived the designed interface as useful and easy to use, and would potentially use it in the future, indicating the suitability of the design strategies. For instance, an experienced open-source maintainer reported: "*I'm very resistant to bots; however, I liked it a lot for a couple of reasons.*" He explained that he appreciated the creation of a specific place for bots in the pull request, and the "*compressed information*" displayed by the mediator bot, since he does not "*need to open a CI page to know what happened*". According to another participant, when bot comments appear in between human comments, it is easy to miss a piece of interesting information and our approach would help to avoid that. In addition to the positive comments, we found that some design elements needed improvements, including adding references to bot events in the main pull request timeline and providing interactive feedback as an opt-out feature.

Research Question 2

What design strategies can potentially **reduce the noise created by bots** on pull requests?

As a result of the design fiction methodology, we identified a series of design strategies for information management, newcomers' assistance, notification management, spam/failure managements, and platform support. Participants perceived our prototype as a useful and easy to use, and would potentially use it in the future, indicating the suitability of the design strategies.

5. Limitations

As any empirical research, our research presents some limitations and potential threats to validity. In this section, we discuss them, their potential impact on the results, and how we have mitigated these limitations.

Generalizability of the results Since we recruited practitioners experienced with bots on the GitHub platform, our findings may not necessarily apply to other social coding platforms, such as GitLab and Bitbucket. Although we do not anticipate big differences

in these platforms, additional research is necessary to investigate the transferability of the results.

Data representativeness Although we conducted our study with a substantial number of practitioners, we likely did not discover all possible challenges, strategies or provide full explanations of them. We are aware that each bot as well as each project has its singularities and that the open-source universe is expansive. Our strategies to keep collecting data until reaching information saturation and to consider different practitioner profiles and identify recurrent mentions of challenges and design strategies from multiple perspectives aimed to alleviate this issue. Anyway, our findings reflect the perspective of practitioners experienced with bots. Therefore, we acknowledge that additional research is necessary to consider the perspective of those who do not have any experience with bots on social coding platforms.

Information saturation We continued recruiting participants and conducting interviews until we came to an agreement that no new significant information was found. As posed by Strauss and Corbin [Strauss and Corbin 1997], sampling may be discontinued once the collected data is considered sufficiently dense and data collection no longer generates new information. As previously mentioned, we also made sure to interview different groups with different perspectives on bots before deciding whether saturation had been reached.

Reliability of results To improve the reliability of our findings, we employed a constant comparison method [Glaser and Strauss 2017]. In this method, each interpretation is constantly compared with existing findings as it emerges from the qualitative analysis. In addition, we also developed a prototype and collected feedback from the participants. To check the reliability of the TAM instrument, we performed a reliability check on the questionnaire items.

6. Conclusion

Motivated by the growing importance of software bots that act upon the pull-based development model, we have investigated the challenges incurred by bots. We found three categories of reported challenges: interaction, adoption, and development challenges. Bot noisiness has appeared as a crosscutting concern in all three categories. Noisiness often leads to communication issues and expectation breakdowns. Developers often complain about bot verbose messages, timing, and high frequency of actions, which might be caused by platform limitations or bot configuration issues. Backed by the results of our empirical studies, we have investigated strategies to mitigate noise and deal with some of the identified challenges. Compared to the previous literature, our findings provide a comprehensive understanding and exploration of design ideas to enhance the integration between bots, humans, and social coding platforms.

Referências

Abdellatif, A., Badran, K., and Shihab, E. (2020). Msrbot: Using bots to answer questions from software repositories. *Empirical Software Engineering*, 25(3):1834–1863.

Blythe, M. (2014). Research through design fiction: narrative in real and imaginary abstracts. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 703–712. ACM.

Brown, C. and Parnin, C. (2019). Sorry to bother you: Designing bots for effective recommendations. In *Proceedings of the 1st International Workshop on Bots in Software Engineering*, BotSE '19, pages 54–58, Piscataway, NJ, USA. IEEE Press.

Carvalho, A., Luz, W., Marcilio, D., Bonifacio, R., Pinto, G., and Canedo, E. D. (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)*, pages 161–171. IEEE.

Chaves, A. P. and Gerosa, M. A. (2018). Single or multiple conversational agents? an interactional coherence comparison. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, pages 1–13.

Dagli, M. (2019). *Designing for Trust*. PhD thesis, figshare.

Davis, F. D. (1989). Perceived usefulness, perceived ease of use, and user acceptance of information technology. *MIS quarterly*, pages 319–340.

Glaser, B. G. and Strauss, A. L. (2017). *Discovery of grounded theory: Strategies for qualitative research*. Routledge.

Gousios, G., Storey, M.-A., and Bacchelli, A. (2016). Work practices and challenges in pull-based development: The contributor's perspective. In *Proceedings of the 38th International Conference on Software Engineering*, ICSE '16, pages 285–296, New York, NY, USA. ACM.

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

Kavaler, D., Trockman, A., Vasilescu, B., and Filkov, V. (2019). Tool choice matters: Javascript quality assurance tools and usage outcomes in github projects. In *Proceedings of the 2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*, pages 476–487.

Liu, D., Smith, M. J., and Veeramachaneni, K. (2020). Understanding user-bot interactions for small-scale automation in open-source development. In *Proceedings of the Extended Abstracts of the 2020 CHI Conference on Human Factors in Computing Systems*, CHI EA '20, page 1–8, New York, NY, USA. Association for Computing Machinery.

McIntosh, S., Kamei, Y., Adams, B., and Hassan, A. E. (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*, pages 192–201.

Mirhosseini, S. and Parnin, C. (2017). Can automated pull requests encourage software developers to upgrade out-of-date dependencies? In *Proceedings of the 32nd IEEE/ACM International Conference on Automated Software Engineering*, ASE 2017, pages 84–94, Piscataway, NJ, USA. IEEE Press.

Monperrus, M. (2019). Explainable software bot contributions: Case study of automated bug fixes. In *Proceedings of the 1st International Workshop on Bots in Software Engineering*, BotSE '19, pages 12–15, Piscataway, NJ, USA. IEEE Press.

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

Nematzadeh, A., Ciampaglia, G. L., Ahn, Y.-Y., and Flammini, A. (2016). Information overload in group communication: From conversation to cacophony in the twitch chat. *Royal Society open science*, 6(10):191412.

Paikari, E. and van der Hoek, A. (2018). A framework for understanding chatbots and their future. In *Proceedings of the 11th International Workshop on Cooperative and Human Aspects of Software Engineering*, CHASE '18, pages 13–16, New York, NY, USA. ACM.

Peng, Z. and Ma, X. (2019). Exploring how software developers work with mention bot in github. *CCF Transactions on Pervasive Computing and Interaction*, 1(3):190–203.

Sadeddin, K. W., Serenko, A., and Hayes, J. (2007). Online shopping bots for electronic commerce: the comparison of functionality and performance. *International Journal of Electronic Business*, 5(6):576.

Serban, D., Golsteijn, B., Holdorp, R., and Serebrenik, A. (2021). Saw-bot: Proposing fixes for static analysis warnings with github suggestions. In *Workshop on Bots in Software Engineering*, United States. IEEE Computer Society.

Steinmacher, I., Conte, T. U., Treude, C., and Gerosa, M. A. (2016). Overcoming open source project entry barriers with a portal for newcomers. In *Proceedings of the 38th International Conference on Software Engineering*, ICSE '16, pages 273–284, New York, NY, USA. ACM.

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

Storey, M.-A., Serebrenik, A., Rosé, C. P., Zimmermann, T., and Herbsleb, J. D. (2020). BOTse: Bots in Software Engineering (Dagstuhl Seminar 19471). *Dagstuhl Reports*, 9(11):84–96.

Storey, M.-A. and Zagalsky, A. (2016). Disrupting developer productivity one bot at a time. In *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, FSE 2016, pages 928–931, New York, NY, USA. ACM.

Strauss, A. and Corbin, J. M. (1997). *Grounded theory in practice*. Sage.

Thistlethwaite, D. L. and Campbell, D. T. (1960). Regression-discontinuity analysis: An alternative to the ex post facto experiment. *Journal of Educational psychology*, 51(6):309.

Tsay, J., Dabbish, L., and Herbsleb, J. (2014). Let's talk about it: Evaluating contributions through discussion in github. In *Proceedings of the 22Nd ACM SIGSOFT International*

Symposium on Foundations of Software Engineering, FSE 2014, pages 144–154, New York, NY, USA. ACM.

Urli, S., Yu, Z., Seinturier, L., and Monperrus, M. (2018). How to design a program repair bot?: insights from the repairnator project. In *Proceedings of the 40th International Conference on Software Engineering: Software Engineering in Practice*, pages 95–104. ACM.

Vasilescu, B., Yu, Y., Wang, H., Devanbu, P., and Filkov, V. (2015). Quality and productivity outcomes relating to continuous integration in github. In *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*, ESEC/FSE 2015, page 805–816, New York, NY, USA. Association for Computing Machinery.

Wessel, M., de Souza, B. M., Steinmacher, I., Wiese, I. S., Polato, I., Chaves, A. P., and Gerosa, M. A. (2018). The power of bots: Characterizing and understanding bots in oss projects. *Proc. ACM Hum.-Comput. Interact.*, 2(CSCW):182:1–182:19.

Wessel, M., Serebrenik, A., Wiese, I., Steinmacher, I., and Gerosa, M. A. (2020a). What to expect from code review bots on GitHub? a survey with OSS maintainers. In *Proceedings of the SBES 2020 - Ideias Inovadoras e Resultados Emergentes*.

Wessel, M., Serebrenik, A., Wiese, I. S., Steinmacher, I., and Gerosa, M. A. (2020b). Effects of adopting code review bots on pull requests to oss projects. In *Proceedings of the IEEE International Conference on Software Maintenance and Evolution*. IEEE Computer Society.

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

Wyrich, M. and Bogner, J. (2019). Towards an autonomous bot for automatic source code refactoring. In *Proceedings of the 1st International Workshop on Bots in Software Engineering*, BotSE '19, pages 24–28, Piscataway, NJ, USA. IEEE Press.

Yu, Y., Wang, H., Filkov, V., Devanbu, P., and Vasilescu, B. (2015). Wait for it: Determinants of pull request evaluation latency on GitHub. In *2015 IEEE/ACM 12th Working Conference on Mining Software Repositories*, pages 367–371.