The Inconvenient Side of Software Bots on Pull Requests

Mairieli Wessel
University of Sao Paulo
Sao Paulo, Brazil
mairieli@ime.usp.br

Igor Steinmacher
Northern Arizona University
Flagstaff, USA
igor.steinmacher@nau.edu

ABSTRACT

Software bots are applications that integrate their work with humans' tasks, serving as conduits between users and other tools. Due to their ability to automate tasks, bots have been widely adopted by Open Source Software (OSS) projects hosted on GitHub. Commonly, OSS projects use bots to automate a variety of routine tasks to save time from maintainers and contributors. Although bots can be useful for supporting maintainers' work, sometimes their comments are seen as spams, and are quickly ignored by contributors. In fact, the way that these bots interact on pull requests can be disruptive and perceived as unwelcoming. In this paper, we propose the concept of a meta-bot to deal with current problems on the human-bot interaction on pull requests. Besides providing additional value to this interaction, meta-bot will reduce interruptions and help maintainers and contributors stay aware of important information.

CCS CONCEPTS

• **Human-centered computing → Open source software**.

KEYWORDS

software bots, bots, meta-bot, pull-based model, open source software

ACM Reference Format:
Mairieli Wessel and Igor Steinmacher. 2020. The Inconvenient Side of Software Bots on Pull Requests. In *IEEE/ACM 42nd International Conference on Software Engineering Workshops (ICSEW'20), May 23–29, 2020, Seoul, Republic of Korea.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3387940.3391504

1 INTRODUCTION

Software bots have become particularly relevant for Open Source Software (OSS) projects hosted on GitHub, due to the intensive integration workload inherent to the pull request model [9]. Bots are software applications that integrate their work with humans' tasks [11]. Basically, a bot serve as an interface between developers and other tools [27]. The human-bot integration implies partnership, which means that they complement each others' activities [8].

ICSEW'20, May 23–29, 2020, Seoul, Republic of Korea

ACM ISBN 978-1-4503-7963-2/20/05…$15.00
https://doi.org/10.1145/3387940.3391504

On GitHub, bots are *task-oriented*, responsible for automating well-defined tasks to reduce the workload of maintainers and contributors [30]. Particularly, *GitHub bots* are usually adopted to automate a variety of predefined tasks, such as ensuring license agreement signing, reporting continuous integration failures, reviewing code and pull requests [15, 22, 30], triaging issues [31], and refactoring the source code [32].

Similarly to human developers, *GitHub bots* have their own user profile and interact through comments, becoming new voices on the pull request conversation [16]. The interaction of these new "team members", however, can be disruptive and may affect the pull request communication and dynamic in unexpected ways. Consequently, their comments are perceived as spams, and sometimes are quickly ignored by contributors. For example, while project maintainers may direct their effort to other activities, the bot could mistakenly close pull requests or provide poor feedback [30] leading to contributors to stop contributing since lack of feedback on pull requests may discourage further contributions [25].

Considering the problems aforementioned, related to human-bot interaction, in this paper we present our vision of a meta-bot. Basically, this meta-bot will act as a middleman between the developers and the existing bots interacting on the pull requests. It might avoid, for example, overwhelming developers with massive information or unnecessary notifications. Furthermore, it gives the developer a centralized way to control the dynamic of the interaction.

In this paper, we make the following contributions: (i) an understanding of bots' integration on the pull-based model and their definition; (ii) highlighting of some current problems in the human-bot interaction on pull requests; and (iii) a vision of a meta-bot to overcome those current problems.

2 BOTS SUPPORTING PULL REQUESTS

Open Source Software (OSS) development is inherently collaborative, frequently involving a community of geographically dispersed developers [26]. These developers commonly work on social coding platforms, such as GitHub, that provide features for collaborating and sharing [6]. To receive external contributions, repositories are shared by *fork* (i.e., clone), and modified by pull requests.

The pull-based model offers new opportunities for community engagement, especially to OSS community, but at the same time increases the workload for maintainers to communicate, review code, deal with license issues, explain project guidelines, run tests, and merge pull requests [10]. Due to this intensive integration workload, bots have been adopted to automate a variety of predefined tasks around pull requests [30].

Behaving as human users, *GitHub bots* have their own user profile and can open, close, or comment on pull requests and issues. Additionally, some bots have an official integration with GitHub

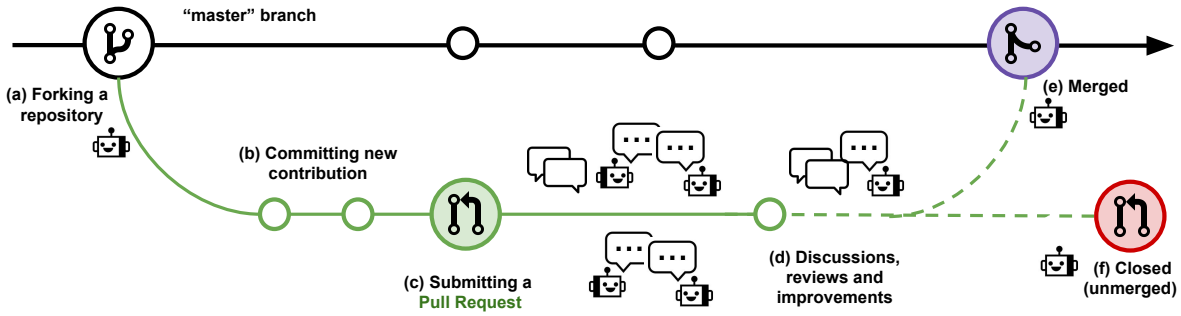

Figure 1: Pull-based model workflow integrated with bots

and are available at the GitHub marketplace[1] or Probot[2]. These official bots are properly tagged in the pull request messages, such as *Dependabot*[3].

Bots have spread across all pull-based workflow, as shown in Figure 1. Bots can be found submitting pull requests, interacting through comments, and merging or closing them. Some bots automatically check the project's source code, searching for broken dependencies, vulnerabilities, or bugs; and then submits a pull request for fixing these issues as shown in Fig. 1(a-c). For example, *Dependabot* automatically creates pull requests to keep project dependencies up-to-date. The *Refactoring-Bot*, proposed by Wyrich and Bogner [32], creates pull requests to refactor the code, removing code smells.

There are also bots designed to support contributors and maintainers after pull requests have been submitted, which aims to facilitate discussions and reviews (Fig. 1(d)). *Git Enforcer*[4], comments on pull requests that do not satisfy specific rules defined by the project maintainers. Moreover, it is also possible to communicate with some bots through comments in the pull requests. *Reminders*[5] bot, for example, sets a reminder on specific pull request when the maintainer request. Other bots help maintainers closing or merging pull requests, as shown in both Fig. 1(e) and Fig. 1(f). *Stale bot*, studied in our previous work [31], automatically closes abandoned pull requests. *Mergify*[6] automatically merges a pull request based on rules predefined by project maintainers.

2.1 GitHub bot definition

Despite its increasing popularity, analyzing and understanding bots is a major challenge. The terminology used to describe bots is vast, diverse, and often inconsistent [12]. Consequently, this hinders a better understanding of the term *bot* and its usage. In this paper, we focus on a specific category of software bot: *GitHub bots*.

Our definition reflects how a bot works on GitHub. As aforementioned, *GitHub bots* have their own user profile and can behave as a developer: opening, closing or commenting on pull requests and issues. Playing a role within the development team, *GitHub bots* execute well-defined tasks that complement other developers' work. They also serve as an interface between developers and external services, for example, reporting the results of Continuous Integration (CI) tools in a pull request comment.

> Essentially, we define *GitHub bots* as **task-oriented bots** that behave on the GitHub environment like a human user. A *GitHub bot* is an application that integrates its work with human tasks [8], serving as a interface between users and services [27]. They provide new forms of interactions with already existing tools [2], automating predefined tasks and binding services together.

3 RELATED WORK

In terms of understanding the practical implications of bot adoption, Storey and Zagalsky [27] describe a cognitive framework to explain how bots support software development productivity. Storey and Zagalsky [27] claim that bots are often used to avoid interruptions to developers' work, but they may lead to other, less obvious distractions.

In order to save developers' time, previous works have focused on designing bots to be integrated into the pull request workflow to perform a variety of tasks, such as repairing bugs [16, 17, 28, 29], refactoring the code [32], recommending tools [3], detecting duplicated development [23], updating dependencies [15], and mentioning reviewers [21, 22]. Additionally, bots on GitHub are different from those found supporting development activities in general. While *GitHub bots* automate specific tasks on pull requests, interacting with developers through comments, software engineering bots focus on answering developers' questions [1, 4, 5, 19, 20, 33] and are generally integrated into communication platforms [5, 13, 14, 20].

Although some studies focus on not interrupting developers' workflow during the bot interaction, they do not draw attention to potential problems introduced by these bots at large. Furthermore, little has been done to evaluate and improve the state of practice. Considering this, we focus on problems that contributors and maintainers face while interacting with software bots on pull requests of OSS projects. With a more in-depth understanding of these problems, researchers and practitioners can invest their efforts

[1] https://github.com/marketplace
[2] https://probot.github.io/apps/
[3] https://dependabot.com
[4] https://github.com/Schachte/Git-Enforcer
[5] https://probot.github.io/apps/reminders/
[6] https://github.com/marketplace/mergify

in designing or improving bots, ultimately supporting maintainers and contributors on reviewing and submitting pull requests.

4 HUMAN-BOT INTERACTION PROBLEMS

GitHub bots have become new voices on the pull request conversation [16]. According to Brown and Parnin [3], the human-bot interaction on GitHub can be inconvenient and lead to negative feedback from maintainers and contributors, due to poor bots' design. Consequently, while bots can be useful for automating maintainers' work, the way they interact may lead to unexpected effects on the communication and dynamic of the pull request. Mirhosseini and Parnin [15], for example, reported that maintainers are overwhelmed by bots notification, which interrupts their workflow.

A similar problem on the human-bot interaction is explored on Wikipedia community [18, 34]. Wikipedia bots change the overall ecosystem by interacting with their operators, managers, and human editors, as well as other bots. Zheng et al. [34] describe that while editors acknowledge bots for streamlining knowledge production, they complain that bots not only solves problems, but creates them as well.

In our previous work [30], we openly asked contributors and maintainers about the "problems/challenges of using bots" on pull requests. Several contributors complained about the way the bots interact, saying that the *bots provide non-comprehensive/poor feedback*. Other respondents also referred to communication issues such as *bots introducing communication noise* and *lack of information on how to interact with the bot*. To go a step further in this investigation, we gathered some anecdotal evidence of problems in the human-bot interaction from the state-of-the-practice.

Based on a initial list of *GitHub bots* from our previous study [30], we used the GitHub advanced search[7] to find potential projects. We queried the tool, searching for projects that received pull requests or comments from any of the bots, and projects in which any of the bots were mentioned by other developers.

For each selected project, we have manually analyzed the pull requests looking for (i) human users mentioning bots, and (ii) bots' interactions—such as opening, merging or commenting on pull requests. During this process, we noticed the interaction of other bots and also considered them in the analysis. We used the challenges previously identified [30] as a seed to code the issues we identified in the pull requests, and thus defined the problems faced in the state-of-the-practice. We present the main categories of problems in the following.

Bots introducing communication noise — This category represents the problems that disturb human-bot communication, especially when the bot comments are seen as spam by contributors and maintainers. In some cases, bots introduce communication noise inflating the pull request with annoying comments that are rapidly ignored. For example, the *commitlint* bot is responsible for running a tool, called *commitlint*[8], over all pull requests and then reporting possible problems as comments. Since this bot adds a comment to each commit of the pull request, maintainers and contributors are overwhelmed by notifications. Other example of spamming was reported by a contributor of the *atom*[9] project. The contributor asks project maintainers to remove the *lock bot*: "*Any chance you guys turn down the lock bot spam, I have a lot of issues subscribed and I get about 20 notifications per day just from those bot actions and there does not seem to be away to ignore them*".

Lack of information about the bot — This issue relates to the way the bots are designed to communicate, which leads them to be misused. In our previous work [30], we described that maintainers and contributors complained of the lack of information on how to interact with bots. Additionally, we found pieces of evidence that contributors would like to know what is the bot's role in the project. For example, after interaction of the *React Community Bot* on a specific pull request of the project *React*[10], one contributor asked for more information about that bot: "*Hey guys, how is the reactjs-bot?*."

Bots taking wrong actions — It was the third most reported problem by contributors to our previous work [30]. As described by our respondents, we also found in the state-of-the-practice *stale bot* mistakenly closing active pull requests. In some cases, the rule-based nature of these bots lead them to take wrong actions in certain situations. In the *Concourse* project[11], for example, after the bot has marked an active pull request as inactive for three times, a maintainer commented: "Lemme just slap a label on here to calm the stale bot down. :P Seems like [the label] is useful if not just as an aggregator."

5 THE CONCEPT OF A META-BOT

We envision a meta-bot as a promising approach to deal with current problems related to human-bot interaction. Our meta-bot was inspired by Sadeddin et al. [24] work. In order to deal with several responses from different bots, Sadeddin et al. [24] showed that a meta-bot would obtain product information from several shopping bots, and then summarize and present it to the user. The concept of meta-bot is also present in the literature of software agents. Generalist agents are also referred to as super bots or meta-bots [7]. This is because they often combine multiple tasks and functionalities of specialist agents into a single agent.

Essentially, the meta-bot way to solve interaction problems is by mediating the action of other bots used on pull requests. It will operate as an middleman between human developers and the bots in a repository. Different from other *GitHub bots*, the meta-bot will not handle specific tasks on pull requests. Instead, the meta-bot will provide additional value to the interaction of already existing bots.

Our envisioned meta-bot will provide more flexibility to developers, allowing them to configure the dynamic of the interaction. Moreover, once the meta-bot is integrated into a GitHub repository, it will be aware of the task-oriented bots adopted to handle the pull requests. By providing a centralized control, meta-bot will be capable of integrating and orchestrating those bots.

Following, we provide an overview of how the meta-bot will mitigate the human-bot interaction problems presented in Section 4.

[7] https://github.com/search/advanced
[8] https://commitlint.js.org
[9] https://github.com/atom/atom
[10] https://github.com/facebook/react
[11] https://github.com/concourse/concourse

Avoiding massive information — A way to mitigate communication noise is by restricting inconvenient bots to interact directly on pull requests. As the meaning of inconvenience can vary from project to project, the meta-bot will provide an interface that each project can previously configure bots' restrictions. Therefore, once a new pull request is opened, the meta-bot will handle bots' response based on the restrictions: (i) allowing the bot interaction (e.g. commenting, labeling, assigning a reviewer) or (ii) summarizing the response of bots not allowed to interact on that pull request.

Supporting developers' questions and requests — Different from most common *GitHub bots*, the meta-bot is a conversational bot. Maintainers and contributors will be able to interact with the meta-bot any time via comments on the pull request, asking the meta-bot for help or clarifications on the existing bots. To answer both developers' questions and requests, the meta-bot will collect and learn from information publicly available on different sources (e.g. GitHub marketplace, GitHub API, Probot). In response to a developers' request, for example, the meta-bot will be able to ask a specific bot to run.

Providing customizable feedback — In addition to the two previous functionalities, the meta-bot will allow the customization of the content of its responses in order to attend the specific interests and needs of maintainers and contributors. As aforementioned, the meta-bot will provide summarized feedback based on the other bots outcome. Developers can manage the information that our bot will show, requesting it to include or exclude some information.

Handling bot exceptions — If needed, the meta-bot will help developers to deal with the consequences of one or more bots taking wrong actions on pull requests. By monitoring the bots' activity, the meta-bot learns from previous interactions and suspend those that attempt to disturb the developers workflow. For example, if the meta-bot identify bot comments as spam, it will block the bot to prevent the same action on other pull requests. Additionally, the meta-bot will request the intervention of a developer.

Figure 2 shows an example of the interaction between a human developer and the envisioned meta-bot, called *Dashbot*, on a pull request. This project adopted four *GitHub bots* to handle specific tasks on pull requests:

- ***trafico-bot***[12] – a bot that automatically adds labels on pull requests according to their status (e.g. *work in progress*, *ready for review*)
- ***commitlint*** – a bot that checks for conventional commit rules (see more details in section 4).
- ***request-info***[13] – a bot that request more info on pull requests with default title or empty description.
- ***tester-bot*** – a bot that runs the tests. This bot was developed by the maintainers of the project.

A maintainer has created a pull request to integrate an important feature to the project (Figure 2). In order to mitigate human-bot interaction problems, *Dashbot* is responsible for controlling when and to what extent the other four bots can interact on pull requests. However, maintainers had previously configured the meta-bot to allow the interaction of the *trafico* bot on every new pull request.

[12]https://github.com/marketplace/trafico-pull-request-labeler
[13]https://probot.github.io/apps/request-info/

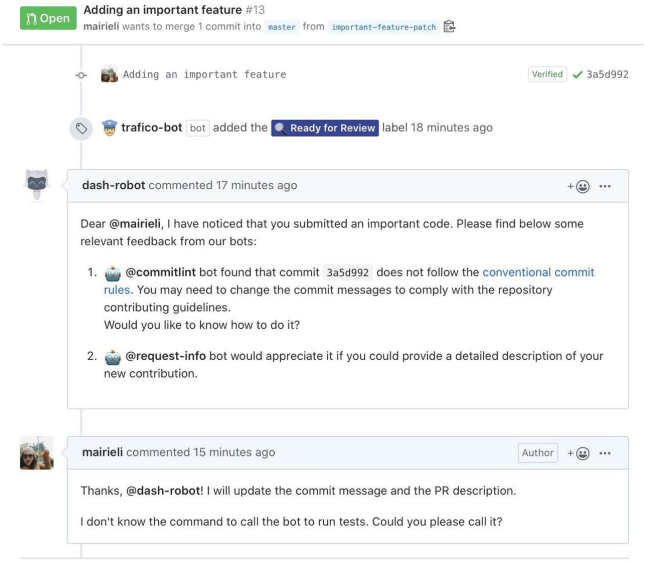

Figure 2: Interface mockup of the interaction between a contributor/maintainer and the meta-bot on a pull request.

Therefore, the *trafico* bot added a label to mark the pull request as *ready for review*.

Afterwards, the meta-bot received the feedback from two bots: *commitlint* and *request-info*. Instead of have these two bots inflating the pull request with comments, the meta-bot proactively summarized their feedback on a single comment to notify the maintainer. At this point, the lack of information on how to interact directly with a specific bot leads the maintainer asking the meta-bot to call it.

6 CONCLUSION

The emergence of bot activity all over the OSS community on GitHub is an indication of the growing importance of these new team members for automating activities around pull requests. Although this widespread adoption, bots are bothering both contributors and maintainers. Considering this, we propose the concept of a meta-bot to mitigate some of the interaction problems introduced by bots around pull requests.

ACKNOWLEDGMENTS

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Finance Code 001, CNPq (# 141222/2018-2) and NSF (# 1815503).

REFERENCES

[1] Ahmad Abdellatif and Emad Shihab. 2019. MSRBot: Using Bots to Answer Questions from Software Repositories. *arXiv preprint arXiv:1905.06991* (2019).
[2] Nick C. Bradley, Thomas Fritz, and Reid Holmes. 2018. Context-aware Conversational Developer Assistants. In *Proceedings of the 40th International Conference on Software Engineering* (Gothenburg, Sweden) *(ICSE '18)*. ACM, New York, NY, USA, 993–1003. https://doi.org/10.1145/3180155.3180238
[3] 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

[4] Liang Cai, Haoye Wang, Bowen Xu, Qiao Huang, Xin Xia, David Lo, and Zhenchang Xing. 2019. AnswerBot: an answer summary generation tool based on stack overflow. In *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. ACM, 1134–1138.

[5] Jhonny Cerezo, Juraj Kubelka, Romain Robbes, and Alexandre Bergel. 2019. Building an Expert Recommender Chatbot. In *Proceedings of the 1st International Workshop on Bots in Software Engineering* (Montreal, Quebec, Canada) *(BotSE '19)*. IEEE Press, Piscataway, NJ, USA, 59–63. https://doi.org/10.1109/BotSE.2019.00022

[6] 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.

[7] Meric Dagli. 2019. *Designing for Trust.* Ph.D. Dissertation. figshare.

[8] Umer Farooq and Jonathan Grudin. 2016. Human-computer integration. *interactions* 23, 6 (2016), 27–32.

[9] Georgios Gousios, Margaret-Anne Storey, and Alberto Bacchelli. 2016. Work Practices and Challenges in Pull-based Development: The Contributor's Perspective. In *Proceedings of the 38th International Conference on Software Engineering* (Austin, Texas) *(ICSE '16)*. ACM, New York, NY, USA, 285–296. https://doi.org/10.1145/2884781.2884826

[10] Georgios Gousios, Margaret-Anne Storey, and Alberto Bacchelli. 2016. Work Practices and Challenges in Pull-based Development: The Contributor's Perspective. In *Proceedings of the 38th International Conference on Software Engineering* (Austin, Texas) *(ICSE '16)*. ACM, New York, NY, USA, 285–296. https://doi.org/10.1145/2884781.2884826

[11] Carlene Lebeuf, Margaret-Anne Storey, and Alexey Zagalsky. 2018. Software Bots. *IEEE Software* 35, 1 (2018), 18–23.

[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] Christoph Matthies, Franziska Dobrigkeit, and Guenter Hesse. 2019. An Additional Set of (Automated) Eyes: Chatbots for Agile Retrospectives. In *Proceedings of the 1st International Workshop on Bots in Software Engineering* (Montreal, Quebec, Canada) *(BotSE '19)*. IEEE Press, Piscataway, NJ, USA, 34–37. https://doi.org/10.1109/BotSE.2019.00017

[15] 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

[16] 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

[17] Martin Monperrus, Simon Urli, Thomas Durieux, Matias Martinez, Benoit Baudry, and Lionel Seinturier. 2018. Human-competitive Patches in Automatic Program Repair with Repairnator. *CoRR* abs/1810.05806 (2018). arXiv:1810.05806 http://arxiv.org/abs/1810.05806

[18] Claudia Müller-Birn, Leonhard Dobusch, and James D Herbsleb. 2013. Work-to-rule: the emergence of algorithmic governance in Wikipedia. In *Proceedings of the 6th International Conference on Communities and Technologies*. ACM, 80–89.

[19] Alessandro Murgia, Daan Janssens, Serge Demeyer, and Bogdan Vasilescu. 2016. Among the machines: Human-bot interaction on social q&a websites. In *Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems*. ACM, 1272–1279.

[20] Elahe Paikari, JaeEun Choi, SeonKyu Kim, Sooyoung Baek, MyeongSoo Kim, SeungEon Lee, ChaeYeon Han, YoungJae Kim, KaHye Ahn, Chan Cheong, and Andre van der Hoek. 2019. A Chatbot for Conflict Detection and Resolution. In *Proceedings of the 1st International Workshop on Bots in Software Engineering* (Montreal, Quebec, Canada) *(BotSE '19)*. IEEE Press, Piscataway, NJ, USA, 29–33. https://doi.org/10.1109/BotSE.2019.00016

[21] Zhenhui Peng and Xiaojuan Ma. 2019. Exploring how software developers work with mention bot in GitHub. *CCF Transactions on Pervasive Computing and Interaction* 1, 3 (01 Nov 2019), 190–203. https://doi.org/10.1007/s42486-019-00013-2

[22] Zhenhui Peng, Jeehoon Yoo, Meng Xia, Sunghun Kim, and Xiaojuan Ma. 2018. Exploring How Software Developers Work with Mention Bot in GitHub. In *Proceedings of the Sixth International Symposium of Chinese CHI* (Montreal, QC, Canada) *(ChineseCHI '18)*. ACM, New York, NY, USA, 152–155. https://doi.org/10.1145/3202667.3202694

[23] 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.

[24] Khaled W Sadeddin, Alexander Serenko, and James Hayes. 2007. Online shopping bots for electronic commerce: the comparison of functionality and performance. *International Journal of Electronic Business* 5, 6 (2007), 576.

[25] 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

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

[27] 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

[28] 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*. ACM, 95–104.

[29] 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

[30] 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. *Proc. ACM Hum.-Comput. Interact.* 2, CSCW, Article 182 (Nov. 2018), 19 pages. https://doi.org/10.1145/3274451

[31] Mairieli Wessel, Igor Steinmacher, Igor Wiese, and Marco A. Gerosa. 2019. Should I Stale or Should I Close?: An Analysis of a Bot That Closes Abandoned Issues and Pull Requests. In *Proceedings of the 1st International Workshop on Bots in Software Engineering* (Montreal, Quebec, Canada) *(BotSE '19)*. IEEE Press, Piscataway, NJ, USA, 38–42. https://doi.org/10.1109/BotSE.2019.00018

[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] Bowen Xu, Zhenchang Xing, Xin Xia, and David Lo. 2017. AnswerBot: automated generation of answer summary to developersź technical questions. In *Proceedings of the 32nd IEEE/ACM International Conference on Automated Software Engineering*. IEEE Press, 706–716.

[34] L Zheng, Christopher M Albano, and Jeffrey V Nickerson. 2018. Steps toward Understanding the Design and Evaluation Spaces of Bot and Human Knowledge Production Systems. In *Wiki Workshop'19*.