Enhancing Developers' Support on Pull Requests Activities with Software Bots

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

ABSTRACT

Software bots are employed to support developers' activities, serving as conduits between developers and other tools. Due to their focus on task automation, bots have become particularly relevant for Open Source Software (OSS) projects hosted on GitHub. While bots are adopted to save development cost, time, and effort, the bots' presence can be disruptive to the community. My research goal is two-fold: (i) identify problems caused by bots that interact in pull requests, and (ii) help bot designers enhance existing bots. Toward this end, we are interviewing maintainers, contributors, and bot developers to understand the problems in the human-bot interaction and how they affect the collaboration in a project. Afterward, we will employ Design Fiction to capture the developers' vision of bots' capabilities, in order to define guidelines for the design of bots on social coding platforms, and derive requirements for a meta-bot to deal with the problems. This work contributes more broadly to the design and use of software bots to enhance developers' collaboration and interaction.

CCS CONCEPTS

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

KEYWORDS

Software Bots, GitHub Bots, Open-source Software

ACM Reference Format:
Mairieli Wessel. 2020. Enhancing Developers' Support on Pull Requests Activities with Software Bots. In *Proceedings of the 28th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE '20), November 8–13, 2020, Virtual Event, USA.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3368089.3418539

1 INTRODUCTION

Social coding platforms, such as GitHub, are broadly used to host Open Source Software (OSS) projects. While providing features that aid collaboration and sharing [22], such as pull requests, these platforms increase the workload of maintainers to communicate and review contributions [9]. To help with the pull request review, maintainers often rely on automation tools to check whether the code builds, the tests pass, and the contribution conforms to a defined style guide [10].

Recently, OSS projects have started to use software bots to reduce the pull request review burden. As an interface that integrates humans and services, bots play a prominent role in social coding platforms [21]. These bots help reduce the intensive workload inherent to the pull request model by automating routine tasks and interacting with human developers. Such bots work on different tasks and are usually created as GitHub users that can submit code contributions, interact through comments, and merge or close pull requests [24]. By executing tasks that were previously only performed by human developers, and interacting in the same communication channels as developers, bots have become important voices in the pull request conversation [14].

In theory, the automation provided by these bots should save maintainers effort and time [21], leading them to focus on more priority development and review tasks. Nevertheless, the integration of these bots can be disruptive, giving rise to problems that interfere in development workflows. Mirhosseini and Parnin [13], for example, reported that maintainers are overwhelmed by bots' notifications on pull requests, which interrupt their workflow. According to Brown et al. [4], the human-bot interaction on pull requests can be inconvenient, leading developers to leave negative feedback or even abandon their contributions. This problem may be especially relevant for newcomers, who require special support during the onboarding process due to the barriers they face [19]. Newcomers can experience bots' complex answers as an additional and discouraging barrier since bots can provide a long list of critical contribution feedback (e.g., style guidelines, failed tests), rather than supportive assistance.

Although some studies focus on bots' development [1, 14], little has been done to investigate the potential problems introduced by bots at large. In fact, there are important aspects that must be considered by bot developers to increase the acceptance of bots by developers. During a recent Dagstuhl seminar on bots in Software Engineering that I attended [20], some key themes of bot interaction that emerged are avoiding repetitive notifications, providing consistency in the tasks being done, and making bots adaptive. Thus, it is critical to understand that software bots are socio-technical rather than technical applications, and must be designed to consider human interaction, developers' collaboration, and ethical concerns.

Considering this context, this research focuses on *designing and evaluating strategies to mitigate problems related to the interaction with software bots on social coding platforms, thereby assisting developers in communicating and accomplishing their tasks.* Thus, my

ESEC/FSE '20, November 8–13, 2020, Virtual Event, USA

ACM ISBN 978-1-4503-7043-1/20/11…$15.00
https://doi.org/10.1145/3368089.3418539

research aims to answer: **How can bots better support OSS developers' work on pull requests?** To answer this question, this research employs mixed-methods analysis of surveys, interviews and participatory design fiction, and quasi-experimental research studies.

2 RELATED WORK

Software bots are extensively proposed and analyzed in the literature of software engineering. Bots have been proposed to support technical and social aspects of software development activities [12], such as communication and decision-making [21]. These bots are an interface between software developers and other services [21]. According to Lebeuf et al. [11], this interface usually provides additional value on top of the software service's basic capabilities.

On GitHub, bots are integrated into the pull request workflow to perform a variety of tasks, including repairing bugs [14], refactoring the source code [25], recommending tools to help developers [4], detecting duplicated development [17], updating outdated dependencies [13], and fixing static analysis violations [5]. Similarly to human users, *GitHub bots* have their user profile and can open, close, or leave comments on pull requests and issues. *GitHub bots* execute well-defined tasks that complement other developers' work, playing a role within the development team.

Understanding how bots' interaction affects human developers is a major challenge. Storey and Zagalsky [21] and Paikari and van der Hoek [15] highlight that the potentially negative impact of task automation through bots is being overlooked. For example, Brown and Parnin [4], proposed the *tool-recommender-bot*, a bot that provides tool recommendations to software developers. This bot automatically submits a pull request that configures a tool and describes how it works. Brown and Parnin [4] applied *tool-recommender-bot* in real projects for evaluation purposes. Only two pull requests out of 52 recommendations were accepted. According to the authors, bots still need to overcome problems such as notification workload.

Mirhosseini and Parnin [13] analyzed 7,470 GitHub projects to understand whether automated pull requests submitted by the *greenkeeper* bot[1] actually help maintainers to update outdated dependencies. The results suggest that, on average, projects that used *greenkeeper* updated 1.6 times more than projects that did not use any tools. Although the bot is useful, maintainers are often overwhelmed by notifications: only a third of pull requests were merged into the codebase. Peng and Ma [16] conducted a case study on how developers perceive and work with *mention bot*. The results show that even *mention bot* has saved developers' efforts; different user groups have different requirements for the bot. For example, project owners require simplicity and stability, contributors require transparency, and reviewers require selectivity. Additionally, results show that developers are bothered with several review notifications during heavy workload.

Beschastnikh et al. [2] envisioned a bot platform to help researchers integrate their new techniques into software development. The aim of *Mediam* is to help researchers upload their bots to the platform, and allow multiple developers to run it in GitHub, which will generate reports for feedback. Beschastnikh et al. [2] envision bots being easily developed and deployed, allowing quick access to new methods developed by researchers. To avoid overwhelming developers, this platform is responsible for deciding which notifications will be sent to GitHub.

We go a step further previous research, explicitly investigating problems on the human-bot interaction on pull requests, and how to mitigate them. Our approach to mitigate these problems was inspired by Sadeddin et al. [18] work. To deal with several responses from different bots, Sadeddin et al. [18] showed that a meta-bot would obtain product information from several shopping bots, and then summarize and present it to the user.

3 PROPOSED RESEARCH

Based on the work of Sadeddin et al. [18], we hypothesize that *a meta-bot can mitigate human-bot interaction problems around pull requests.* The concept of meta-bot is also present in the literature of software agents [8] and chatbots [6]. Generalist agents are also referred to as meta-bots [8], as they often combine multiple tasks and functionalities of specialist agents into a single agent.

To evaluate this hypothesis, my research seeks to address the following research questions:

RQ1. What interaction problems do bots introduce when supporting pull requests?
RQ2. What features do maintainers, contributors, and bot developers envision for a bot to mitigate current problems?
RQ3. How might these envisioned features mitigate interaction problems and leverage bots to better support developers' work?

The research design comprises three phases and complementary studies, as presented in Figure 1. The next subsections detail the research methodology and results achieved so far.

3.1 Preliminary Work — Characterization of GitHub Bots and Their Impacts

This phase consists of studies conducted during the definition of this thesis's scope.

3.1.1 Characterization of Bots Supporting Pull Requests. We conducted a preliminary study to characterize the bots that support pull requests on GitHub. Our results indicate that bots' adoption is indeed widespread in OSS projects hosted on GitHub. Bots perform several tasks, including ensuring license agreement signing, reporting continuous integration failures, reviewing code and pull requests, triaging issues, and refactoring source code [23]. We also openly asked contributors and maintainers about the "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*. The respondents deemed the current bots as *not smart enough* and provided insights into the bots' potential new features, such as *improving notification and awareness*, *enhancing user interaction*, *improving communicability*, and *answering specific questions*. Our results suggest that *GitHub bots* serve as a useful way to access services and automate tasks; however, in terms of supporting developers' interaction, they are not as evolved as in other domains (e.g., education, customer service). These limitations

[1] https://greenkeeper.io

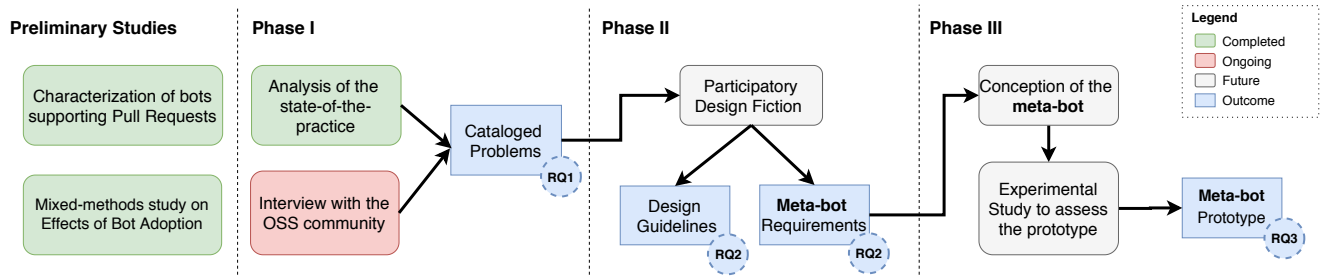

Figure 1: Research Overview

may have influenced developers' perceptions when they reported that bots should be smarter, have better ways to interact, or provide information that can be reasoned from previous interactions.

3.1.2 Effects of Bot adoption during Code Review. To understand the effect of bot adoption, we conducted an exploratory empirical investigation of the effects of adopting bots to support the code review process on pull requests. First, we statistically analyzed data from 1,194 open source projects hosted on GitHub. Analyzing the statistical models, we found that more pull requests are merged into the codebase after the bot adoption, and there is less communication between contributors and maintainers. Considering non-merged pull requests, after bot adoption, projects have less monthly non-merged pull requests, and faster pull requests rejections. To understand their perspective on the effects of bot adoption, we surveyed 127 open-source project maintainers, who reported 15 changes in the maintenance process. Some maintainers reported negative effects caused by bot adoption. Bots are *impersonating human developers*, *introducing noise*, and *intimidating newcomers*.

3.2 Phase I — Identification of Human-bot Interaction Problems on Pull Requests

Understanding how developers and bots interact on GitHub is the first step towards designing strategies to improve the human-bot interaction. To promote this investigation, we gathered some anecdotal evidence of problems from the state-of-the-practice. We manually analyzed pull requests, looking for (i) human users mentioning bots, and (ii) bots' interactions—such as opening, merging, or commenting on pull requests. We noticed that the bots used in pull requests indeed (i) overwhelm developers' communication with notifications and feedback, (ii) perform wrong actions, and (iii) are misused due to their poor documentation [24].

We are currently interviewing open source developers to understand their perspectives about the problems encountered in the state-of-the-practice, and to identify new issues. We conducted 21 semi-structured interviews with participants recruited by (i) advertisements on Twitter, (ii) direct messages, and (iii) e-mails. Participants were expected to have experience contributing to or maintaining projects that use bots to support pull request activities. Their experience with software development ranges from 3 to 25 years ($\simeq$ 11 years on average). Participants also belong to different ethnic groups (living in South America, North America, and Europe). Emergent problems include technical, social, and cultural issues. Further, our preliminary analysis also reveals the introduction of noise as a primary problem caused by bots' adoption. This noise might disrupt both human communication and development workflow. In short, this analysis provides the foundation for our catalog of interaction problems.

3.3 Phase II — Designing Strategies to Support Developers' Work on Pull Requests

After identifying the human-bot interaction problems, we will design strategies to better support developers' work on pull requests. By capturing the expectations of developers who interact with bots, we will: (i) define guidelines for the design of future bots, and (ii) elicit the features to a meta-bot. Therefore, we will use Design Fiction [3] as a participatory method to explore ways to overcome a subset of the most relevant problems evidenced.

Design Fiction uses speculative products, prototypes, and narratives to anticipate future trends or to propose visionary solutions, reflecting upon the present world. The speculative nature of this technique amplifies critical views of current social and technological developments, creating a fictional context narrated through designed artifacts. In the HCI community, for example, many researchers have used design fiction to anticipate users' needs [7].

Using Design Fiction, we will present to participants a fictional history of a meta-bot capable of better supporting developers' interactions on pull requests, operating as a middleman between developers and the existing bots. Participants will act as storytellers, answering questions to ground the end of the fictional history to raise social and ethical concerns around the use of bots and the requirements of the meta-bot.

3.4 Phase III — Transforming Design Strategies to Bot Prototypes

Grounded on the participatory design fiction, we will design and implement a preliminary prototype of the meta-bot. Essentially, we envision the meta-bot as a promising approach to assist developers in performing their tasks more efficiently and help OSS projects attract, engage, and retain contributors. Our envisioned meta-bot will provide more flexibility to developers, enabling them to configure the dynamics of the interaction. In summary, the meta-bot mediates the action of other bots used on pull requests to mitigate previously identified interaction problems. Compared to other GitHub bots, the meta-bot will provide additional value to the interaction of already existing bots through these key features: (i)

summarizing other bots' outcomes to avoid information overload; (ii) supporting developers' questions and requests; (iii) providing configurable feedback; and (iv) helping developers deal with bots' exceptions [24]. The specifics meta-bot requirement will be further defined after conclusion of Phase II (see section 3.3).

To evaluate the bot prototype, we will conduct a study with OSS project specialists. Since software bots are typically evaluated using factors that are related to both accuracy and usability [1, 4], we will evaluate whether the implemented features mitigate interaction problems. The results will enable us to improve the bots' requirements and the prototype according to the feedback received. In addition to the study involving specialists, the meta-bot will be integrated to a real OSS project. The goal of this integration is to receive feedback from contributors and maintainers, and understand to what extent the adopted strategies support the developers' collaboration. In this case, we will conduct debrief sessions with the developers that interacted with the meta-bot and quantitatively analyze this data.

4 CLAIMED CONTRIBUTIONS

My proposal contains three main novel contributions, which map onto the research questions and are related to the artifacts generated in each phase: (i) an empirical catalog of human-bot interaction problems in social coding platforms, with focus on the pull request process; (ii) guidelines for designiinto *bots* to support software development tasks; and (iii) a meta-bot to support developers' interactions. In the long-term, this research will provide a more nuanced view of the human-bot interaction in social coding platforms. With a more in-depth understanding of this interaction, researchers and practitioners can invest their efforts in designing or improving bots, ultimately supporting developers on submitting and reviewing pull requests.

ACKNOWLEDGMENT

The author is advised by Prof. Marco A. Gerosa and co-advised by Prof. Igor Steinmacher from Northern Arizona University. This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Finance Code 001 and CNPq (# 141222/2018-2).

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] Ivan Beschastnikh, Mircea F. Lungu, and Yanyan Zhuang. 2017. Accelerating Software Engineering Research Adoption with Analysis Bots. In *Proceedings of the 39th International Conference on Software Engineering: New Ideas and Emerging Results Track* (Buenos Aires, Argentina) *(ICSE-NIER '17)*. IEEE Press, Piscataway, NJ, USA, 35–38. https://doi.org/10.1109/ICSE-NIER.2017.17

[3] Mark Blythe. 2014. Research through design fiction: narrative in real and imaginary abstracts. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 703–712.

[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 (BotSE)*.

[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] Ana Paula Chaves and Marco Aurelio Gerosa. 2018. Single or Multiple Conversational Agents? An Interactional Coherence Comparison. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. 1–13.

[7] EunJeong Cheon and Norman Makoto Su. 2017. Configuring the User: "Robots Have Needs Too". In *Proceedings of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing* (Portland, Oregon, USA) *(CSCW '17)*. ACM, New York, NY, USA, 191–206. https://doi.org/10.1145/2998181.2998329

[8] Meric Dagli. 2018. *Designing for Trust: Exploring Trust and Collaboration in Conversational Agents for E-commerce*. Master's thesis. School of Design, Carnegie Mellon University.

[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] D. Kavaler, A. Trockman, B. Vasilescu, and V. Filkov. 2019. Tool Choice Matters: JavaScript Quality Assurance Tools and Usage Outcomes in GitHub Projects. In *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*. 476–487.

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

[12] 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.

[13] 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

[14] 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

[15] 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

[16] 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

[17] 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.

[18] 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.

[19] Igor Steinmacher, Tayana Uchoa Conte, Christoph Treude, and Marco Aurelio Gerosa. 2016. Overcoming Open Source Project Entry Barriers with a Portal for Newcomers. In *Proceedings of the 38th International Conference on Software Engineering (ICSE)*.

[20] 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.

[21] 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

[22] Jason Tsay, Laura Dabbish, and James Herbsleb. 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* (Hong Kong, China) *(FSE 2014)*. ACM, New York, NY, USA, 144–154. https://doi.org/10.1145/2635868.2635882

[23] 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

[24] 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

[25] 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