Department: Head
Editor: Name, xxxx@email

Guidelines for Developing Bots for GitHub

Mairieli Wessel
Radboud University, The Netherlands

Andy Zaidman
Delft University of Technology, The Netherlands

Marco A. Gerosa
Northern Arizona University, USA

Igor Steinmacher
Northern Arizona University, USA

***Abstract*—Projects on GitHub rely on the automation provided by software development bots to uphold quality and alleviate developers' workload. Nevertheless, the presence of bots can be annoying and disruptive to the community. Backed by multiple studies with practitioners, this paper provides guidelines for developing and maintaining software bots. These guidelines aim to support the development of future and current bots and social coding platforms.**

■ **BRIDGING THE GAP** between human collaborative software development and automated processes, bots are used to alleviate the software development workload, improve productivity, and enable use cases for which humans are not realistically suitable [1]. On social coding platforms, such as GitHub, a bot acts autonomously to some extent, has a user account, and plays a role within the development team, executing tasks that complement the developers' work [2].

Automating simple, time-consuming, or tedious tasks and collecting dispersed information are some ways that bots support software projects. In previous work, we have found that the adoption of bots helps developers merge more pull requests, and reduces the need for communication between developers [3]. However, while bots are useful for automating a variety of tasks related to software development, prior research has shown that they have the potential side-effect of disrupting developers in their work [4].

Surveying and interviewing practitioners, we have found three categories of reported challenges: interaction, adoption, and development challenges [5], [4] (see Figure 1). 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 a bot's verbose messages, timing, and high frequency of actions, which might be caused by platform limitations or bot configuration issues.

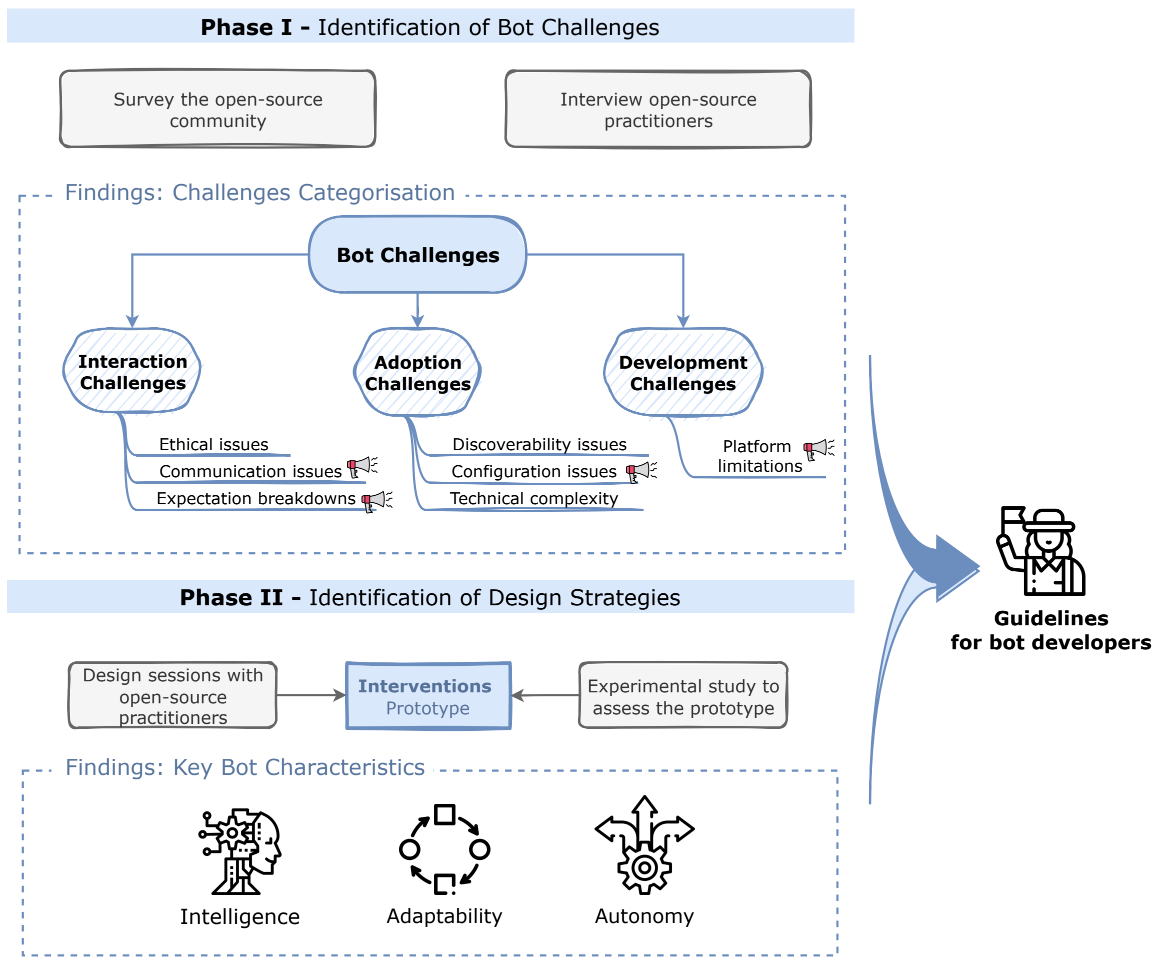

Phase I - Identification of Bot Challenges
Survey the open-source community
Interview open-source practitioners
Findings: Challenges Categorisation
Bot Challenges
Interaction Challenges
Adoption Challenges
Development Challenges
Ethical issues
Communication issues
Expectation breakdowns
Discoverability issues
Configuration issues
Technical complexity
Platform limitations
Guidelines for bot developers
Phase II - Identification of Design Strategies
Design sessions with open-source practitioners
Interventions Prototype
Experimental study to assess the prototype
Findings: Key Bot Characteristics
Intelligence
Adaptability
Autonomy

Figure 1. Methodology employed to identify challenges, build a prototype, and create guidelines. The result from Phase I was published at CSCW ([5], [4]), and Phase II at ICSE 2022 [6]. We added a graphical mark (📢) to identify the challenges related to noisiness, which crosscut the three categories of challenges.

Backed by the results of our empirical studies, we have investigated interventions/strategies to mitigate noise and deal with some of the identified challenges [6]. In line with Erlenhov et al. [1]'s results, our results indicate that a combination of three different characteristics appears to be relevant for a bot: intelligence, adaptability, and autonomy. Although intelligence and adaptability recurrently appear in the literature as a desired bot characteristic [1], [7], they are not yet widely present in bots that work on GitHub [5].

Backed by the observations gathered from these studies, this paper presents a set of guidelines to help develop software bots for GitHub and (re-)design the human-bot interaction on social coding platforms. We expect that the advances in bot creation frameworks will provide better support for the fulfillment of the guidelines in the future.

Research overview

We have collected evidence of bot noisiness throughout multiple empirical studies as presented in Figure 1. First, we surveyed 205 open source contributors and 23 maintainers [5] and openly asked them about the challenges of using and interacting with bots. To deepen our understanding of these challenges, we interviewed 21 practitioners experienced with bots, including project maintainers, contributors, and bot developers [4]. The developers' most recurrent complaints are related to 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.

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 [6]. To design and implement the interventions, we applied Design Fiction [8], 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.

Building on the findings of our empirical investigation, we propose a set of guidelines for both bot developers and tool builders. All the guidelines are backed by the evidence previously collected and supported by the literature.

Guidelines for developing bots

To make bots more effective at accomplishing their tasks, design problems need to be solved to avoid repetitive notifications, provide consistency in the tasks being done, make bots adaptive, and provide clear and contextualized feedback to project contributors and maintainers [4], [6]. To better design the next generation of bots, we provide a set of guidelines along with three main categories: designing bot interaction, facilitating bot adoption, and overcoming platform limitations.

Designing bot interaction

One of the essential aspects of bot interaction is communication. However, the existing bots might fail to provide meaningful information to developers. The most recurrent and central challenge is the introduction of noise into the developers' communication channels. Developers complained about *annoying bot behaviors* such as verbosity, high frequency and timing of actions, and unsolicited actions. Therefore, we present a set of guidelines to support tool builders and bot developers in designing bot interactions.

Guideline 1 (G#1)

Provide clear, concise, and well-organized information.

Interaction challenges: We evidenced the need for *background knowledge to interact* with and understand the messages of bots on GitHub. Combined with *lack of context*, it might be extremely difficult for humans to extract meaningful guidance from bots' feedback. In these cases, when a bot message is not clear enough, developers "*[...] need to go and ask a human for clarity*", which may generate more work for both contributors and maintainers.

What should bot developers do: To reduce the cognitive effort to process bot feedback, it is preferable to prioritize conciseness over completeness. For example, a bot that informs developers whether the changes in a pull request affected the code coverage (i.e., a code coverage bot) should focus on reporting the overall result, and pointing to sources of additional information.

Guideline 2 (G#2)

Focus on an appropriate way to show information.

Interaction challenges: Another important aspect of bot interaction is the way bots should display information to developers. Developers frequently do not like it when "*[...] bots put a bunch of information that they try to convey in comments instead of [providing] status hooks or a link somewhere.*"

What should bot developers do: Bot developers should identify the best way to convey the information. On GitHub, this can be achieved by exploring possible ways to show information on the platform, which can be either status information or comments. For example, a bot that looks over the code in a pull request and catches quality issues (i.e., a code quality bot) can comment on a pull request to report a list of code formatting issues found. In cases where only an overall status (i.e., passing, falling, blocked) is needed, it is preferable to use status information and avoid overloading pull requests with additional comments.

Guideline 3 (G#3)

Provide actionable changes to developers.

Interaction challenges: Another recurrent complaint from our survey and interview participants is that bots *do not provide actionable changes* for developers. Some of the messages and outcomes from bots are so strict that they do not guide developers on what they should do next to accomplish their tasks: "*it is great to see 'yes' or 'no,' but if it is not actionable, then it is not useful [...]".*
What should bot developers do: Bot outcomes should be accompanied by actionable and technically sound recommendations by default for the decision-making of developers. For example, a pull request comment from a code coverage bot informing that the coverage decreased is not actionable. However, a comment accompanied by suggested changes is highly actionable because it helps developers to figure out the next steps.

Guideline 4 (G#4)

Avoid overly humanized bot messages.

Interaction challenges: Previous studies on human-chatbot interaction have already shown that human users can hold higher expectations with overly humanized bots (e.g., bots that say "*thank you*") which can lead to frustration [9]. Our study results underscore that some developers feel uncomfortable interacting with a bot, as mentioned by one participant: " '*for some people, it is still quite strange, and they are quite surprised by it.*" Also, receiving "*thanks*" from a non-human feels less sincere.
What should bot developers do: Although developers envision the bot mediator interacting with users through natural language, more direct, and non-humanized bot messages are appreciated. For instance, developers suggested avoiding sentences that do not add to the bot's feedback, such as "*Hey, I'm here to help you [...].*"

Guideline 5 (G#5)

Make bots' purpose clear.

Interaction challenges: By automating and providing feedback on time-consuming tasks (e.g., checking code style or calculating code coverage), bots are intended to reduce the workload of project maintainers and inform project contributors. Nevertheless, maintainers reported that a challenge they see is that "*contributors don't understand the value of bots for maintainers.*" We also found that developers with different profiles and backgrounds have different expectations with regard to bot interaction. Bots, for example, *enforce predefined cultural rules* of a community, causing expectation breakdowns for outsiders.
What should bot developers do: It is essential to make the purpose of each bot clear, avoiding expectation breakdowns from both sides. This may be implemented, for example, by including a footnote descriptive sentence or a link to further information about the bot in the bot comment.

Facilitating bot adoption

If maintainers find an appropriate bot, they then have to deal with configuration challenges. Thus, we present advice on how to facilitate the bot adoption process in addition to the guidelines for designing the human-bot interaction.

Guideline 6 (G#6)

Provide options to configure bot notification.

Adoption challenges: The study conducted to co-design the mediator bot prototype showed that open-source developers would like to customize aspects of the bot interaction, including notification frequency and timing. Therefore, it is important for bot developers to design a highly customizable bot, providing project maintainers better configuration control over bot actions, rather than just turning off bot comments.

What should bot developers do: In the mediator bot design sessions, developers suggested *scheduling of bot notifications*, so that the bot would avoid notifying developers according to (customizable) timeframes indicating when they do not want interruptions. This may be implemented, for example, using a "do not disturb" mode. Another option is *not to notify maintainers until the condition is satisfied*. In this case, the bot would notify the developers only when the predefined conditions are met: "*I want to be notified about new pull requests after all my tests have passed. And after the bots commented, and if everything is green, then I want to be notified.* The recommendation is that these mechanisms are explicitly announced during bot adoption (e.g., noiseless configuration, preset levels of information).

Guideline 7 (G#7)

Include documentation of alternative installation settings to accommodate different types of users.

Adoption challenges: It is *difficult to tailor the bot configuration* to fit the needs of a project. Even after maintainers spend the time needed to configure the bot, there is sometimes no way to predict what the bot will do once installed. According to developers' experience, it is "*easy to install the bot with the basic configuration. However, it is not easy to adjust the configuration to your needs*".

What should bot developers do: Bot developers should document the bot installation, giving concrete examples of the bot outcomes and possible effects of each configuration choice, and keep it updated. This can also be implemented by creating a FAQ (Frequently Asked Questions) section in a website or repository where the bot code is stored. This is also an opportunity for lowering the entry barrier for new project maintainers, who need to be aware of how each bot works on the project.

Guidelines' takeaway

Bot developers should envision bots as *socio-technical* rather than *technical* applications, which must be designed to consider human interaction, developers' collaboration, and other ethical concerns.

Recommendations to platform builders

To complement our guidelines, we also explored the platform restrictions since they might limit both the extent of the bots' actions and the way bots communicate. We, therefore, present a set of recommendations for platform builders that would aid bot developers.

Recommendation 1 (R#1): Enable multiple interaction mechanisms.

Platform limitations: Our empirical investigation of bot challenges revealed some limitations imposed by the GitHub platform that restrict the design of bots. As mentioned by one participant: "*There are still a few things that just cannot be done with the [GitHub] API.*." The platform restrictions might limit both the extent of the bots' actions and the way bots communicate.

What should platform builders do: It is essential to provide alternative ways for bots to interact on the platform. A developer stated that the platform ideally would provide additional mechanisms since bots interact only through comments. In other environments, such as Slack, developers can interact more flexibly with (chat)bots. On GitHub, this might be achieved by enabling distinct views of the same bot output depending on the developer's role (i.e., maintainer, casual contributor, newcomer) and enabling developers to filter and hide specific bot information.

Recommendation 2 (R#2): Consider creating a dedicated communication channel for bots.

Platform limitations: The interviews we conducted with developers have shown that dealing with bots providing comments with dense information "*in the middle of the pull request*" can be "*[...] a lot more distracting than it is helpful.*" Bots may overburden developers who already suffer from information overload when communicating online.

What should platform builders do: To reduce information overload, participants suggested removing bot interactions from the main conversation interface and creating a dedicated place for them. We prototyped this strategy by designing a new tab in the pull request interface; this idea can be leveraged to reshape the interface and better display bot interactions. There is also room for integrating GitHub bots into other developer communication platforms (e.g., Slack, Discord).

The mediator bot

To alleviate the concern of bot noisiness in pull requests, we have investigated the concept of a bot that operates as a mediator between developers and the existing bots (i.e., a meta-bot). This section presents our mediator bot prototype and how it connects to the proposed guidelines. Figure 2 provides an overview of the mediator bot design strategies, which we mark throughout the text with (*S#n*). Firstly, we split our prototype into two different versions: the experts' pull request interface designed to support maintainers and experienced contributors; and the newcomers' pull request interface. We designed a dedicated place for all information and events regarding bots in the pull request (*S#1*; platform recommendation *R#2*). The mediator bot creates a *summary* with the most important information about each bot, then groups them into *categories* (e.g., "warnings", "information") (*S#3*; *G#1-2*).

To avoid inflating the pull requests with several comments from the mediator bot, one suggested strategy is to *keep the most recent information* (*S#2*; *G#1*). We include the latest information from each bot in the summary. 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 (platform recommendation *R#1*).

In the newcomer's interface, we added a text-based message to fulfill the requirement of welcoming newcomers (*S#4*; *G#4*). Beyond presenting a welcoming message, the mediator bot also points the contributor to other sources that can contain information about rules, instructions, and requirements (*S#5*; *G#1*) of the project. Thus, we included a link to Reakit's contributing guidelines.

Another important distinction between the two versions is how the mediator bot displays the information for newcomers versus experts (platform recommendation *R#1*). We implemented an interactive process of displaying bots' information (*S#6*). 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 simultaneously. As part of this guidance, the mediator bot also refers to contribution guidelines to assist newcomers and present a concise and direct welcoming message.

Conclusion

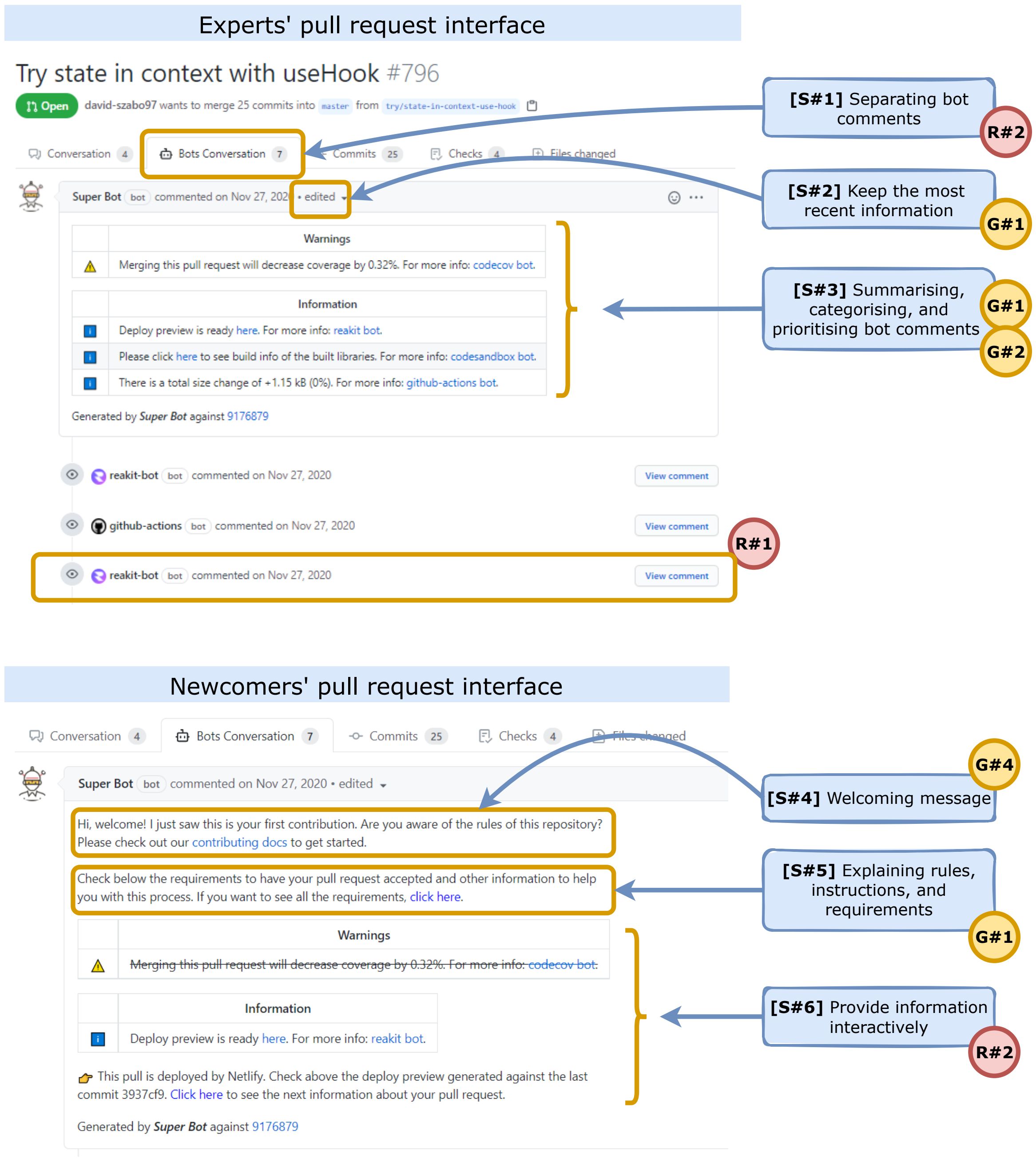
Motivated by the growing importance of software bots that act upon the pull-based development model, we have proposed guidelines on how to improve the next generation of bots, considering interaction, adoption, and development challenges identified in prior work. These guidelines can serve bot developers and contributors and maintainers of GitHub projects that use bots in two dimensions: understanding how bots are perceived and how they can be leveraged to support development tasks. Orthogonal to the guidelines, we have also explored the concept of a mediator bot, to alleviate the growing concern of noisiness among bot users. We envision that our guidelines will help developers to produce bots that better automate tasks and further guide developers in collaborative software development.

Acknowledgments

This work was partially supported by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Finance Code 001; CNPq grants 141222/2018-2, 314174/2020-6, and 313067/2020-1; the National Science Foundation under Grant numbers 1815503 and 1900903; and the Dutch science foundation NWO through the Vici "TestShift" project (No. VI.C.182.032). We also thank the developers who spent their time participating in our research.

■ REFERENCES

1. L. Erlenhov, F. G. d. O. Neto, and P. Leitner, "An empirical study of bots in software development–characteristics and challenges from a practitioner's per-


Experts' pull request interface
[S#1] Separating bot comments
R#2
[S#2] Keep the most recent information
G#1
[S#3] Summarising, categorising, and prioritising bot comments
G#1
G#2
R#1
Newcomers' pull request interface
G#4
[S#4] Welcoming message
[S#5] Explaining rules, instructions, and requirements
G#1
[S#6] Provide information interactively
R#2

Try state in context with useHook #796

Open david-szabo97 wants to merge 25 commits into `master` from `try/state-in-context-use-hook`

Conversation 4 | Bots Conversation 7 | Commits 25 | Checks 4 | Files changed

Super Bot bot commented on Nov 27, 2020 • edited

	Warnings
⚠	Merging this pull request will decrease coverage by 0.32%. For more info: codecov bot.

	Information
ℹ	Deploy preview is ready here. For more info: reakit bot.
ℹ	Please click here to see build info of the built libraries. For more info: codesandbox bot.
ℹ	There is a total size change of +1.15 kB (0%). For more info: github-actions bot.

Generated by *Super Bot* against 9176879

reakit-bot bot commented on Nov 27, 2020 View comment

github-actions bot commented on Nov 27, 2020 View comment

reakit-bot bot commented on Nov 27, 2020 View comment

Conversation 4 | Bots Conversation 7 | Commits 25 | Checks 4 | Files changed

Super Bot bot commented on Nov 27, 2020 • edited

Hi, welcome! I just saw this is your first contribution. Are you aware of the rules of this repository? Please check out our contributing docs to get started.

Check below the requirements to have your pull request accepted and other information to help you with this process. If you want to see all the requirements, click here.

	Warnings
⚠	~~Merging this pull request will decrease coverage by 0.32%. For more info: codecov bot.~~

	Information
ℹ	Deploy preview is ready here. For more info: reakit bot.

This pull is deployed by Netlify. Check above the deploy preview generated against the last commit 3937cf9. Click here to see the next information about your pull request.

Generated by *Super Bot* against 9176879

Figure 2. Prototype of the interventions in a real-world scenario on GitHub. It shows the relationship between the design strategies for the mediator bot derived from Phase II (S#1-6) and our proposed guidelines (G#1, G#2, G#4)/platform recommendations (R#1, R#2). The interactive version of the prototype is publicly available on Zenodo [10].

spective," in *Proceedings of the 2020 28th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, ser. FSE 2020, 2016.

2. M. Wessel and I. Steinmacher, "The inconvenient side of software bots on pull requests," in *Proceedings of the 2nd International Workshop on Bots in Software Engineering*, ser. BotSE, 2020.
3. M. Wessel, A. Serebrenik, I. S. Wiese, I. Steinmacher, and M. A. Gerosa, "Effects of adopting code review bots on pull requests to oss projects," in *IEEE International Conference on Software Maintenance and Evolution*. IEEE Computer Society, 2020.
4. M. Wessel, I. Wiese, I. Steinmacher, and M. A. Gerosa, "Don't disturb me: Challenges of interacting with software bots on open source software projects," *Proceedings of ACM Human-Computer Interaction*, no. CSCW, 2021.
5. M. Wessel, B. M. de Souza, I. Steinmacher, I. S. Wiese, I. Polato, A. P. Chaves, and M. A. Gerosa, "The power of bots: Characterizing and understanding bots in oss projects," *Proceedings of the ACM Conference on Computer Supported Cooperative Work Social Computing*, vol. 2, no. CSCW, pp. 182:1–182:19, Nov. 2018. [Online]. Available: http://doi.acm.org/10.1145/3274451
6. M. Wessel, A. Abdelattif, I. Wiese, T. Conte, E. Shihab, M. A. Gerosa, and I. Steinmacher, "Bots for pull requests: The good, the bad, and the promising," *Proceedings of IEEE/ACM International Conference on Software Engineering*, no. ICSE, 2022.
7. C. Lebeuf, A. Zagalsky, M. Foucault, and M.-A. Storey, "Defining and classifying software bots: A faceted taxonomy," in *Proceedings of the 1st International Workshop on Bots in Software Engineering*, ser. BotSE '19. Piscataway, NJ, USA: IEEE Press, 2019, pp. 1–6. [Online]. Available: https://doi.org/10.1109/BotSE.2019.00008
8. M. Blythe, "Research through design fiction: narrative in real and imaginary abstracts," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 2014, pp. 703–712.
9. U. Gnewuch, S. Morana, and A. Maedche, "Towards designing cooperative and social conversational agents for customer service," in *International Conference on Information Systems (ICIS)*. AIS, 2017.
10. M. Wessel, A. Abdellatif, I. Wiese, T. Conte, E. Shihab, M. A. Gerosa, and I. Steinmacher, "Supplementary Material for ICSE'22 paper "Bots for Pull Requests: The Good, the Bad, and the Promising"," Sep. 2021. [Online]. Available: https://doi.org/10.5281/zenodo.5675702

Mairieli Wessel is an Assistant Professor in the Department of Software Science (SwS) at Radboud University, The Netherlands. She obtained her Ph.D. in Computer Science from the University of São Paulo, Brazil. Her main research interest is in software engineering (SE) and computer-supported cooperative work (CSCW), focused on software bots and open-source development. Her research goal is to design intelligent support for developers by leveraging bots' capabilities. For more information, visit http://www.mairieli.com

Andy Zaidman, is a Full Professor in software engineering at Delft University of Technology, The Netherlands. He received his M.Sc. and Ph.D. degrees in Computer Science from the University of Antwerp, Belgium, in 2002 and 2006, respectively. His main research interests include software evolution, program comprehension, mining software repositories, software quality, and software testing. He is an active member of the research community and involved in the organization of numerous conferences (WCRE'08, WCRE'09, VISSOFT'14 and MSR'18). In 2013 he was the laureate of a prestigious Vidi midcareer grant, while in 2019 he received the most prestigious Vici career grant from the Dutch science foundation NWO.

Marco A. Gerosa, is a Full Professor at the Northern Arizona University, USA, and a Ph.D. advisor at the University of São Paulo, Brazil. He received his Ph.D. from the Catholic University of Rio de Janeiro in 2006. He researches Software Engineering and CSCW, and recent projects include the development of tools and strategies to support developers' onboarding to open-source software communities and the design of bots and chatbots. He published more than 200 papers and serves on the program committee (PC) of top-tier conferences, such as ICSE, FSE, MSR, ICSME, and CSCW. For more information, visit http://www.marcoagerosa.com

Igor Steinmacher, is an Assistant Professor in the School of Informatics, Computing, and Cyber Systems at the Northern Arizona University (NAU), and was previously at the Federal University of Technology Paraná (UTFPR), Brazil. He received a Ph.D. in Computer Science from the University of São Paulo (USP — Brazil). He researches the intersections of Software Engineering (SE) and Computer Supported Cooperative Work (CSCW). Currently, his research focuses on the behavior of developers in

Open Source Communities, including the support of newcomers, the impact of Bots in the community, and gender bias in Open Source Software. His interests include Open Source Software, Human Aspects of Software Engineering, Empirical Software Engineering, and Mining Software Repositories techniques.