Guidelines for Developing Bots for GitHub

Wessel, Mairieli; Zaidman, Andy; Gerosa, Marco Aurélio; Steinmacher, Igor

DOI
10.1109/MS.2022.3224813

Publication date
2023

Document Version
Final published version

Published in
IEEE Software

Citation (APA)

Important note
To cite this publication, please use the final published version (if applicable). Please check the document version above.

Copyright
Other than for strictly personal use, it is not permitted to download, forward or distribute the text or part of it, without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license such as Creative Commons.

Takedown policy
Please contact us and provide details if you believe this document breaches copyrights. We will remove access to the work immediately and investigate your claim.
Green Open Access added to TU Delft Institutional Repository

'You share, we take care!' - Taverne project

https://www.openaccess.nl/en/you-share-we-take-care

Otherwise as indicated in the copyright section: the publisher is the copyright holder of this work and the author uses the Dutch legislation to make this work public.
Guidelines for Developing Bots for GitHub

Mairieli Wessel, Radboud University
Andy Zaidman, Delft University of Technology
Marco A. Gerosa and Igor Steinmacher, Northern Arizona University

// Projects on GitHub rely on the automation provided by software development bots. Nevertheless, the presence of bots can be annoying and disruptive to the community. Backed by multiple studies with practitioners, this article provides guidelines for developing and maintaining software bots. //
Backed by the observations gathered from these studies, this article 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 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. The developers’ most recurrent complaints are related to annoying bot behaviors. Those behaviors include the case in which bots provide comments “in the middle of the pull request,” frequently overusing visual elements, and the case in which bots perform repetitive

---

**Phase I - Identification of Bot Challenges**

Survey the Open Source Community

Interview Open Source Practitioners

Findings: Challenges Categorisation

- Interaction Challenges
  - Ethical Issues
  - Communication Issues
  - Expectation Breakdowns
- Adoption Challenges
  - Discoverability Issues
  - Configuration Issues
  - Technical Complexity
- Development Challenges
  - Platform Limitations

**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.** The methodology employed to identify challenges, build a prototype, and create guidelines. The result from Phase I was published at CSCW, and the result from Phase II was published at ICSE 2022. We added a graphical mark (¶) to identify the challenges related to noisiness, which crosscut the three categories of challenges.
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.

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.

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 the 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 (that is, 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 (that is, 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 (that is, passing, falling, or 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.*

**Guideline 4 (G#4)**
*Avoid overly humanized bot messages.*

**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 5 (G#5)**
*Make bots’ purpose clear.*

**Guideline 6 (G#6)**
*Provide options to configure bot notifications.*

**Interaction Challenges.** By automating and providing feedback on time-consuming tasks (for example, 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.

**Adoption Challenges.** The study conducted to codesign 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 with better configuration control over bot actions, rather than just turning off bot comments.

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 section on a website or in a 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 sociotechnical 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.

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 [application programming interface].” 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 (that is, maintainer, casual contributor, or 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 (for example, Slack and 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 (that is, a metabot). 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.

FIGURE 2. The 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) and platform recommendations (R#1, R#2). The interactive version of the prototype is publicly available on Zenodo.10
FEATURE: REDESIGNING THE HUMAN–BOT INTERACTIONS

with (Sn). Firstly, we split our prototype into two different versions: 1) the experts’ pull request interface designed to support maintainers and experienced contributors and 2) the newcomers’ pull request interface. We designed a dedicated place for all information and events regarding bots in the pull request (S1; platform recommendation R2). The mediator bot creates a summary with the most important information about each bot and then groups them into categories (for example, “warnings” and “information”) (S3; G1–2).

To avoid inflating the pull requests with several comments from the mediator bot, one suggested strategy is to keep the most recent information (S2; G1). 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 R1).

We implemented an interactive process of displaying bots’ information (S6). The mediator bot guides 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 a part of this guidance, the mediator bot also refers to contribution guidelines to assist newcomers and present a concise and direct welcoming message.

We envision that our guidelines will help developers to produce bots that better automate tasks and further guide developers in collaborative software development.

Acknowledgment
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 Grants 1815503 and 1900903; and the Dutch Science Foundation (NOW) through the Vici “TestShift” project (Grant VIC.182.032). We also thank the developers who spent their time participating in our research.

References

The mediator bot guides newcomers by showing the information from other bots “step by step.”

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, 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. In addition 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.

Authorized licensed use limited to: TU Delft Library. Downloaded on May 11,2023 at 06:47:25 UTC from IEEE Xplore. Restrictions apply.


