Understanding User-Bot Interactions for Small-Scale Automation in Open-Source Development

Dongyu Liu[1]
MIT
Cambridge, MA, USA
dongyu@mit.edu

Micah J. Smith[1]
MIT
Cambridge, MA, USA
micahs@mit.edu

Kalyan Veeramachaneni
MIT
Cambridge, MA, USA
kalyanv@mit.edu

CHI '20 Extended Abstracts, April 25–30, 2020, Honolulu, HI, USA.

ACM ISBN 978-1-4503-6819-3/20/04.
http://dx.doi.org/10.1145/3334480.3382998

Abstract
Small-scale automation tools, or "bots," have been widely deployed in open-source software development to support manual project maintenance tasks. Though interactions between these bots and human developers can have significant effects on user experience, previous research has instead mostly focused on project outcomes. We reviewed existing small-scale bots in wide use on GitHub. After an in-depth qualitative and quantitative evaluation, we compiled several important design principles for human-bot interaction in this context. Following the requirements, we further propose a workflow to support bot developers.

Author Keywords
human-centered computing; HCI design and evaluation methods; software and its engineering; software creation and management

CCS Concepts
•Human-centered computing → **HCI design and evaluation methods; •Software and its engineering** → *Software creation and management;*

[1]Equal contribution

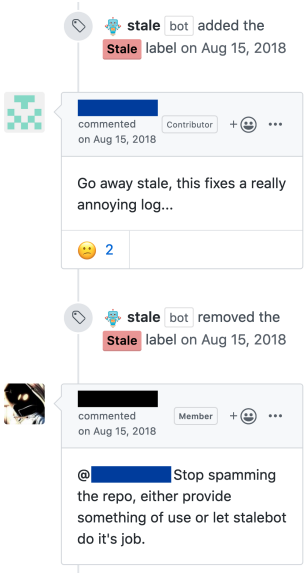

Figure 1: Contentious human-bot and contributor-maintainer interactions occur in a pull request review thread on GitHub. The `stale` bot applied a label after a period of inactivity causing the original contributor to "bump" the thread, in turn provoking a response from a project maintainer.

Introduction

As development of open-source software (OSS) repositories continues to grow, developers have increasingly turned to new approaches in order to reduce the demands of project maintenance and ease workflows for contributors. Automation tools, while once reserved for large tasks such as running test suites, are now also being used for small maintenance-related tasks such as welcoming new contributors or triaging bug reports. These *small-scale bots* are simple applications that perform small maintenance-related tasks on repositories, by using a code hosting platform's API to take action in response to events on the platform and interacting with developers through the platform itself.

While small-scale bots are deployed by project maintainers in order to enhance their workflow, these bots may have consequences on the user experience of existing and prospective project contributors. The preferences and needs of maintainers and contributors may even come into conflict. For example, `stale` bot automatically responds to contributor and maintainer actions on GitHub Issues to mark threads as "stale" or "not stale" based on periods of inactivity [7]. While the volume of open issues can be a burden for maintainers [8], and while new contributors who obtain a quick resolution to bug reports are more likely to become long-term contributors [20], user experience can be negatively affected by aggressive bot actions (i.e., [2], Figure 1).

Previous research in software development bots has largely ignored the study of *interactions* between bots, maintainers, and contributors and consequences for user experience, instead focusing on project outcomes such as issue response time and recommendation accuracy [18, 16]. Similarly, there is potential to more deliberately apply design lessons from chatbots and AI systems [1, 9].

In this paper, we study user experience in relation to interactions between bots and OSS developers. We extensively survey bots and compile key characteristics. We introduce a mental model for understanding opportunities for human-bot interaction design. From a thorough literature review and user study, we propose seven design principles for bot developers. Finally, we apply our approach to a case study.

Background

GitHub, like other code hosting providers, exposes a comprehensive API for actions on its platform that can be used by purpose-driven apps, or "bots", to automate and improve developer workflows. A bot can subscribe to events on a repository with fine granularity, such as pushed branches, new issues opened, and completed automated checks. When these events occur, a webhook with detailed information on the event is sent to the underlying application. The bot can respond as itself by taking action using the GitHub API, such as writing comments, committing code, reviewing pull requests, and creating software releases. Using bot development frameworks [15, 6, 4], bot developers can create small-scale tools by writing little more than a short script. Although the same tooling is also used to create "large-scale" apps, such as continuous integration providers that run a collection of test suites and report the results, these are not the focus of this work.

Wessel et al. [17] study the ways bots are currently used in OSS projects on GitHub and whether introducing bots changes certain project characteristics such as number of comments and PR close time. Erlenhov et al. [5] provide a faceted taxonomy of bot types with facets such as purpose, initiation, communication, and intelligence. Lebeuf et al. [10] provide a separate taxonomy based on the environ-

		Interactions								Configuration			
		Medium					Content			Store		Options	
Task	Bot	M	C	L	K	G	Text	Cmd	GUI	File	Oth.	Tem.	U
Acknowledge contributors	all-contributors-bot		✓			✓	✓	✓		✓		✓	
Manage ML submissions	ballet-bot		✓		✓	✓	✓			✓			
Document patch origins	dco				✓		✓	✓	✓	✓			
Onboard new contributors	first-timers		✓	✓		✓	✓			✓		✓	
Organize issues at scale	issue-label-bot		✓	✓			✓			✓	✓		
Find best PR reviewers	mention-bot	✓	✓				✓			✓		✓	✓
Ensure well-formed PRs	poppins-pr-checklist		✓				✓		✓	✓	✓		
Maintain civil discourse	sentiment-bot	✓	✓				✓			✓		✓	✓
Act on issues quickly	stale		✓	✓		✓	✓			✓		✓	
Track todo items	todo	✓	✓	✓		✓	✓			✓		✓	✓
Correct spelling	typot		✓			✓	✓		✓				

Table 1: OSS development tasks addressed by selected small-scale bots. Legend: M=mentions/notifications, C=comments, L=labels, K=checks, G=git actions, Cmd=commands or keywords, File=file-based configuration store, Oth.=other configuration store, Tem.=templates can be configured, U=green lists/red lists of users can be configured.

ment in which the bot operates, the intrinsic properties of the bot, and the bot's interaction with its environment.

Certain bots have been studied in detail from both quantitative and qualitative points of view. For example, `mention-bot`, which recommends reviewers for pull requests, was found to have a large impact on review response time in less active projects, but solicited mixed perceptions from users overall due to concerns such as insensitivity to context and unbalanced workload allocation [14]. Lessons learned from bots in other contexts such as chatbots include that users prefer to have a clear understanding of bot capabilities and have low tolerance for mistakes and bugs [16, 12, 11, 19, 9]. Cheng and Guo [3] analyze threads discussing usability and UX issues for three large open-source projects with implications for bots that interact with users in Issues.

Bot Characteristics

We surveyed existing small-scale bots. In this setting, we restrict our analysis to GitHub apps that (1) act exclusively through the GitHub API in response to webhooks (2) address project maintenance tasks (3) do not have external application components such as web services and (4) maintain limited or no state across actions. Starting with bots we were already familiar with, we also reviewed bots mentioned in the literature and searched for projects on GitHub tagged with `probot`, `github-app`, and related topics. We also randomly sampled public projects on GitHub

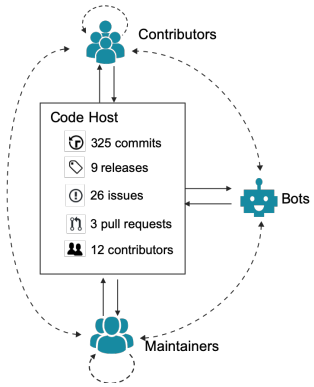

Figure 2: Human-bot interaction workflow mediated by code hosting platform. While we allow that agents can interact directly (dashed lines) — such as via email or chat, unmediated by the platform — it is not the focus for bot interaction designers.

that have installed these bots and manually analyzed human-bot interactions in discussion threads. A list of selected bots is shown in Table 1.

We observe that most bots interact with humans conversationally by creating text elements such as issue/PR bodies and comments; they render text from a simple template, which may be configured by project maintainers; and are configured in a file stored in the repository and owned by maintainers. Other interaction and configuration styles are rare.

We extract the following key characteristics of small-scale bots that have the potential to affect user experience.

- **Interaction medium.** Bot developers can take advantage of the rich set of interactions possible on platforms like GitHub. These include mentions and notifications, comments, reactions, review annotations, labels, checks, check reports, requested action buttons, and team discussions. For example, `issue-label-bot` allows developers to give emoji reaction feedback on issue labeling decisions.
- **Interaction content.** For each interaction, bot developers can consider representing the transmitted content in various ways. These include rendered text templates, structured text (keywords, commands, markup), media (images, speech, emoji), GUI elements, check statuses, and visualizations. For example, `typot` renders a multiple choice list of corrections to misspelled words in Markdown.
- **Configuration options.** Bot developers can customize human-bot interactions by providing flexible bot configuration. Aspects of interactions that can be configurable include notification and mention preferences, text templates, operational parameters, and error handling. For example, `mention-bot` allows maintainers to configure lists of allowed/disallowed users for the bot to "@mention" for review of PRs, though even more careful customization is needed to avoid overwhelming developers [14].
- **Configuration store.** Bot developers can store configuration settings as files in the repository, keywords/commands in user-generated content such as commit messages, files located in other repositories includes ones owned by contributors, and bot defaults. For example, `all-contributors-bot` respects configuration in the `.all-contributorsrc` file in the project repository allowing both bots and maintainers to update the gallery of project contributors.

Human-Bot Interaction Workflow

How should bot developers identify opportunities for enhancing interactions? We present a simple framework that can be a useful mental model for bot developers (Figure 2). Our framework consists of a workflow graph with agents comprised of contributors C, maintainers M, and bots B with interactions mediated by a code hosting platform P. Agents (nodes in the graph) can receive messages on channels (directed edges) causing some change in their internal state, update their state, and take actions in response.[2] Bot developers impose an existing software development maintenance task on this framework by tracing the path that is traversed by human-human interactions. They should then observe that bots can take action in response to every change of state on the platform and consider adding edges $P \rightarrow B$ and $B \rightarrow P$ at each such change.

Design Principles

We conducted a thorough literature review and an extensive user study to answer two research questions: **(RQ1)** *What*

[2]While we allow that agents can interact directly, it is not common in the small-scale tasks we study and we assume these interactions are not present in the design of small-scale bots.

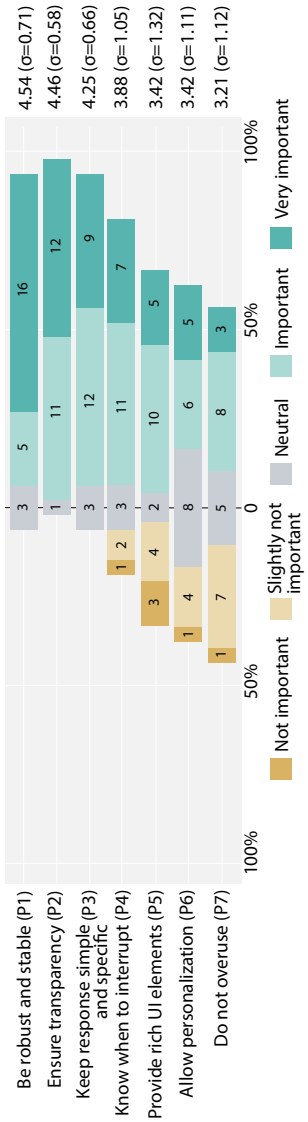

Figure 3: The ranking results of the importance of seven design principles.

are the key design principles for GitHub bots interacting with developers? **(RQ2)** *How do maintainers, contributors, and bot developers perceive these design principles?*

To answer the first question, we revisited bot-related research papers from various conferences/journals ranging from human-computer interaction to software engineering, such as CHI, CSCW, FSE, ASE, BotSE, and SEIP. We compiled a list of potential design principles that are helpful in improving human-bot interactions. We conducted semi-structured interviews with some stakeholders to further screen the principles and retain the key ones (Figure 3). Though they may recall general UI design principles [13], our principles are interpreted in a much different setting. The insights can specifically benefit GitHub bot designers and encourage effective use of bots.

For the second research question, we invited developers by email, in group lists, and by personal invitations to take an online survey (n=24, response rate $\sim$ 27%). Respondents had between 2 and 12 years of software development experience (μ=6.0, σ=2.83) and served different roles participating in projects. Ten respondents were familiar with developing with GitHub bots and two of them were bot developers. We asked the respondents to rate the importance of each design principle on a 5-point Likert scale with 1 being the least important. Figure 3 shows the ranking results of all the design principles.

Principle 1 *Be robust and stable* (μ=4.54, σ=0.71). Bots should take the same or similar actions when given the same or similar inputs. The principle is regarded as "very important" by most respondents (16/24). One commented, *"Stable and predictable behavior is an important part to eliminate the human-machine boundary."*

Principle 2 *Ensure transparency when bots take action* (μ=4.46, σ=0.58). On many occasions, bots have to take actions to facilitate development, such as accepting a pull request (`ballet-bot`), or assigning code reviewers (`mention-bot`). Transparent communication provides visibility and enables maintainers to correct mis-actions. For example, `mention-bot` and other similar bots have been observed to allocate code review assignments in an unbalanced manner. One respondent highlighted that transparency could be achieved using preset rules.

Principle 3 *Keep bot responses simple and specific* (μ=4.25, σ=0.66). On one hand, users should understand what to expect from the bots. One respondent commented, *"I feel it is meaningful and helpful to know how intelligent the bot is and what I can expect from it"*. On the other hand, bot responses should be bound to certain specified subjects and avoid creating complicated, off-topic paths.

Principle 4 *Know when bots can, and cannot, interrupt a human* (μ=3.88, σ=1.05). Developers, like other knowledge workers, require periods of concentration; minimizing or deferring interruptions can support developer productivity [21]. One respondent commented, *"Bots at the wrong time may interrupt my thoughts."*

Principle 5 *Incorporate rich UI elements as needed* (μ=3.42, σ=1.32). UI elements such as buttons, menus, images, and information charts can better engage users in communication. Well-incorporated UI elements can help developers significantly improve the efficiency to perform certain tasks (e.g., confirm pull request). However, this principle conflicts with principle #3 in some sense. The high standard deviation shows that respondents have divergent opinions. Therefore, we must be prudent to incorporate rich UI elements when needed.

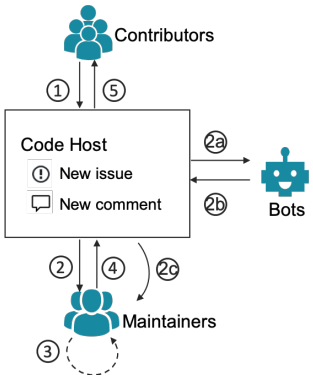

Figure 4: A workflow for Stale bot. A typical issue resolution workflow may be as follows: (1) $C \rightarrow P$ (read "platform") posts initial bug report (or follow-up comment), (2) $P \rightarrow M$ notifies of new comment, (3) $M \rightarrow M$ investigate issue among themselves, (4) $M \rightarrow P$ follows up with comment or git action, and (5) $P \rightarrow C$ notifies of new response. The workflow can then repeat.

Principle 6 *Allow personalizing bot behavior* (μ=3.42, σ=1.11). The benefits and risks of personalization and configuration must be carefully weighed. The distribution of scores indicates divergent respondent opinions. One respondent observed that considering that bot output is mostly text-based, how bots present the content can highly impact users' perceptions. Some respondents highlighted their desires to customize bots interactions such as when they are "@mentioned", even if they do not have maintainer privileges to the repository. Alternatively, several respondents pointed out that bot-behavior should be standardized for stability (Principle #1).

Principle 7 *Do not overuse bots* (μ=3.21, σ=1.12). Over-reliance on bots in communication can hinder creativity and productivity, because the adoption of bots reduces the opportunities for interpersonal communication. Surprisingly, this principle achieved the lowest importance score, as sometimes in OSS development, interpersonal communication can be a challenge rather than a benefit. One respondent observed, *"If a person tells another person to fix their code style, its pedantic, when a bot does it, it's expected."*

Example *Bot Task: Act to quickly resolve issues.* In this example, we walk through our proposed interaction workflow framework and demonstrate how our design principles can be applied to guiding bot design. Issue or PR threads with no activity can block progress of maintainers or discourage contributors [8, 20]. The bot developer then looks to improve this process (Figure 4). The `stale` bot is inserted in this workflow by following up on activity on P after some period of inactivity. For example, n days after $P \rightarrow B$ notifies the bot of a contributor comment (2a), the actual bot responds $B \rightarrow P$ by applying a stale label and rendering a comment template according to the `.github/stale.yml` configuration file (2b). This simple workflow demonstrates principle #1.

We might improve this bot by considering other interaction and configuration approaches. For example, we observe that the inability to control the amount of communication from the bot is a source of friction for OSS developers (Figure 1). Following principles #4, #5 and #6, the developers of stale bot could consider allowing the bot to be controlled per-thread using emoji feedback.

Threats to validity. As we note, some design principles can conflict with each other and respondents feel differently about their relative importance. This could be addressed by increasing the sample of our survey and analyzing responses from contributors, maintainers, and bot creators separately. We could also conduct a controlled user study to further understand and measure how the principles impact user experience.

Conclusion

In this paper, we have systematically studied the interaction behaviors between humans and bots in OSS development. We presented a unified human-bot interaction workflow that can be applied to any small-scale automation tool. Through an extensive literature review and user study, we summarized seven useful design principles that will significantly influence how users perceive a bot when interacting with it. We argue that these design principles could be used as a good guideline to evaluate the user experience of a bot with regard to its interaction behavior.

Acknowledgements

This work is supported under NSF award 1761812. We thank our survey respondents and anonymous reviewers for their comments.

REFERENCES

[1] Saleema Amershi, Kori Inkpen, Jaime Teevan, Ruth Kikin-Gil, Eric Horvitz, Dan Weld, Mihaela Vorvoreanu, Adam Fourney, Besmira Nushi, Penny Collisson, and et al. 2019. Guidelines for Human-AI Interaction. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems - CHI '19*. ACM Press, 1–13. DOI: http://dx.doi.org/10.1145/3290605.3300233

[2] Cargo Issue 6035. 2018. Tracking Issue for Stale Bot. (2018). https://github.com/rust-lang/cargo/issues/6035 Accessed 2019-11-30.

[3] Jinghui Cheng and Jin L.C. Guo. 2018. How Do the Open Source Communities Address Usability and UX Issues?: An Exploratory Study. In *Extended Abstracts of the 2018 CHI Conference on Human Factors in Computing Systems - CHI '18*. ACM Press, 1–6. DOI: http://dx.doi.org/10.1145/3170427.3188467

[4] Kyle Daigle. 2018. GitHub Actions: built by you, run by us. (2018). https://github.blog/2018-10-17-action-demos Accessed 2019-11-27.

[5] Linda Erlenhov, Francisco Gomes de Oliveira Neto, Riccardo Scandariato, and Philipp Leitner. 2019. Current and Future Bots in Software Development. In *2019 IEEE/ACM 1st International Workshop on Bots in Software Engineering (BotSE)*. IEEE, 7–11. DOI: http://dx.doi.org/10.1109/BotSE.2019.00009

[6] Brian T Ford. 2014. Mary Poppins. (2014). https://github.com/btford/mary-poppins Accessed 2019-11-30.

[7] Brandon Keepers. 2019. Stale | Close stale Issues and Pull Requests. (2019). https://github.com/probot/stale Accessed 2019-11-30.

[8] Andrew J. Ko and Parmit K. Chilana. 2010. How power users help and hinder open bug reporting. In *Proceedings of the 28th international conference on Human factors in computing systems - CHI '10*. ACM Press, 1665. DOI: http://dx.doi.org/10.1145/1753326.1753576

[9] Carlene Lebeuf, Margaret-Anne Storey, and Alexey Zagalsky. 2017. How Software Developers Mitigate Collaboration Friction with Chatbots. *arXiv:1702.07011 [cs]* (Feb 2017). http://arxiv.org/abs/1702.07011 arXiv: 1702.07011.

[10] Carlene Lebeuf, Alexey Zagalsky, Matthieu Foucault, and Margaret-Anne Storey. 2019. Defining and Classifying Software Bots: A Faceted Taxonomy. In *2019 IEEE/ACM 1st International Workshop on Bots in Software Engineering (BotSE)*. IEEE, 1–6. DOI: http://dx.doi.org/10.1109/BotSE.2019.00008

[11] Christoph Matthies, Franziska Dobrigkeit, and Guenter Hesse. 2019. An Additional Set of (Automated) Eyes: Chatbots for Agile Retrospectives. In *2019 IEEE/ACM 1st International Workshop on Bots in Software Engineering (BotSE)*. IEEE, 34–37. DOI: http://dx.doi.org/10.1109/BotSE.2019.00017

[12] 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 - CHI EA '16*. ACM Press, 1272–1279. DOI: http://dx.doi.org/10.1145/2851581.2892311

[13] Jakob Nielsen and Rolf Molich. 1990. Heuristic evaluation of user interfaces. In *Proceedings of the SIGCHI conference on Human factors in computing systems Empowering people - CHI '90*. ACM Press, 249–256. DOI: http://dx.doi.org/10.1145/97243.97281

[14] 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 on - ChineseCHI '18*. ACM Press, 152–155. DOI: http://dx.doi.org/10.1145/3202667.3202694

[15] Probot 2019. Probot | GitHub Apps to automate and improve your workflow. (2019). https://probot.github.io Accessed 2019-11-27.

[16] Simon Urli, Zhongxing Yu, Lionel Seinturier, and Martin Monperrus. 2018. How to Design a Program Repair Bot? Insights from the Repairnator Project. In *ICSE-SEIP '18: 40th International Conference on Software Engineering: Software Engineering in Practice Track*. ACM, New York, NY, USA, 10.

[17] 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 on Human-Computer Interaction* 2, CSCW (Nov 2018), 1–19. DOI:http://dx.doi.org/10.1145/3274451

[18] 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 *2019 IEEE/ACM 1st International Workshop on Bots in Software Engineering (BotSE)*. IEEE, 38–42. DOI: http://dx.doi.org/10.1109/BotSE.2019.00018

[19] Bowen Xu, Zhenchang Xing, Xin Xia, and David Lo. 2017. AnswerBot: Automated generation of answer summary to developers' technical questions. In *2017 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 706–716. DOI: http://dx.doi.org/10.1109/ASE.2017.8115681

[20] Minghui Zhou and Audris Mockus. 2012. What make long term contributors: Willingness and opportunity in OSS community. In *2012 34th International Conference on Software Engineering (ICSE)*. IEEE, 518–528. DOI: http://dx.doi.org/10.1109/ICSE.2012.6227164

[21] Manuela Züger, Will Snipes, Christopher Corley, André N. Meyer, Boyang Li, Thomas Fritz, David Shepherd, Vinay Augustine, Patrick Francis, and Nicholas Kraft. 2017. Reducing Interruptions at Work: A Large-Scale Field Study of FlowLight. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems - CHI '17*. ACM Press, 61–72. DOI: http://dx.doi.org/10.1145/3025453.3025662