Bots for Pull Requests: The Good, the Bad, and the Promising

Mairieli Wessel
m.wessel@tudelft.nl
Delft University of Technology, Netherlands

Ahmad Abdellatif
a_bdella@encs.concordia.ca
Concordia University, Canada

Igor Wiese
igor@utfpr.edu.br
Universidade Tecnológica Federal do Paraná, Brazil

Tayana Conte
tayana@icomp.ufam.edu.br
Federal University of Amazonas, Brazil

Emad Shihab
eshihab@encs.concordia.ca
Concordia University, Canada

Marco A. Gerosa
marco.gerosa@nau.edu
Northern Arizona University, USA

Igor Steinmacher
igorfs@utfpr.edu.br
Universidade Tecnológica Federal do Paraná, Brazil
Northern Arizona University, USA

ABSTRACT

Software bots automate tasks within Open Source Software (OSS) projects' pull requests and save reviewing time and effort ("the good"). However, their interactions can be disruptive and noisy and lead to information overload ("the bad"). To identify strategies to overcome such problems, we applied Design Fiction as a participatory method with 32 practitioners. We elicited 22 design strategies for a bot mediator or the pull request user interface ("the promising"). Participants envisioned a separate place in the pull request interface for bot interactions and a bot mediator that can summarize and customize other bots' actions to mitigate noise. We also collected participants' perceptions about a prototype implementing the envisioned strategies. Our design strategies can guide the development of future bots and social coding platforms.

CCS CONCEPTS

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

KEYWORDS

Software Bots, GitHub Bots, Human-bot Interaction, Open Source Software, Automation, Collaborative Development, Design Fiction

ACM Reference Format:
Mairieli Wessel, Ahmad Abdellatif, Igor Wiese, Tayana Conte, Emad Shihab, Marco A. Gerosa, and Igor Steinmacher. 2022. Bots for Pull Requests: The Good, the Bad, and the Promising. In *44th International Conference on Software Engineering (ICSE '22), May 21–29, 2022, Pittsburgh, PA, USA.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3510003.3512765

ICSE '22, May 21–29, 2022, Pittsburgh, PA, USA

ACM ISBN 978-1-4503-9221-1/22/05.
https://doi.org/10.1145/3510003.3512765

1 INTRODUCTION

Software development bots play a prominent role in the pull request review process [66] by serving as an interface between users and other tools [60] and reducing the workload of maintainers and contributors. Accomplishing tasks previously solely performed by human developers, and interacting in the same communication channels as their human counterparts, bots have become new voices in the pull request review conversation [42]. Bots even perform more activities than humans in some projects [19].

Nevertheless, the introduction of bots can interfere with the community dynamics. According to Brown and Parnin [7], the bot interactions on pull requests can be inconvenient, leading developers to leave negative feedback or even abandon their contributions. Indeed, the literature has shown that bots change the dynamics of collaboration [67] and that OSS developers often perceive their intervention as disruptive and noisy [68]. On an issue, a developer complained about the frequency of a bot's actions: "*[..] I get about 20 notifications per day just from those bot actions and there does not seem to be a way to ignore them.*"[1] Therefore, bots may overburden developers who already suffer from information overload when communicating online [48].

To make bots more effective at communicating to developers, design problems need to be solved to avoid repetitive notifications, provide consistency in the tasks being done, and make bots adaptive [37, 58]. Designers should envision software bots as socio-technical rather than purely technical applications, considering human interaction, developers' collaboration, and ethical concerns [60]. The adoption of bots in OSS projects is a recent trend and the literature lacks design strategies that include the end-users' perspective to enhance the bots interaction on social coding platforms.

In other domains, as it is hard to change third-party bots, researchers have proposed meta-bots to integrate and moderate the interactions of multiple bots [9, 17, 53]. We envision a meta-bot as a promising approach to mitigate the information overload from existing GitHub bots. Considering this context, the main goal of this study is to elicit design strategies to overcome the information

[1] https://github.com/atom/atom/issues/18736#issue-402497317

overload caused by bots on pull requests by means of a meta-bot. Specifically, our work investigates the following research questions:

Research Question 1

What design strategies can potentially **reduce the noise created by bots** on pull requests?

Research Question 2

How do **participants perceive** the design strategies to reduce the noise created by bots on pull requests?

To answer RQ1, we applied Design Fiction [5] as a participatory design method. This technique is used to probe, explore, and critique future technologies [5]. We recruited practitioners, including open-source maintainers, contributors, bot developers, and bot researchers, to act as designers in the early stages of the envisioned meta-bot conception. We presented to participants afi ctional story of a meta-bot capable of better supporting developers' interactions on pull requests, and operating as a mediator between developers and the existing bots. Participants answered questions to complete the end of thefi ctional story, raising concerns about the use of bots and discussing the design strategies for the meta-bot.

For the RQ2, we used the emerged design strategies to prototype a meta-bot and collect feedback from the practitioners, which resulted in a set of improvements to the strategies. Overall, our participants perceive the meta-bot as a layer between other bots and the users. It should summarize and customize messages according to the author's context—whether the author is a new contributor or an experienced maintainer. Participants also envision a separate place in the pull request interface for bot interactions.

Our work contributes to the state-of-the-art by empirically identifying a set of design strategies to enhance how bots present information on pull requests and avoid noise. Researchers and tool designers may leverage our results, obtained from the perspective of expert practitioners, to enhance bots' communication design, thereby supporting human-bot interaction on social coding platforms. In particular, the design strategies can benefit the larger software engineering research community as aspects related to the design of bots for social coding platforms are still under-explored. We are also among thefi rst to introduce participatory designfiction to software engineering studies.

2 RELATED WORK

Software bots frequently reside on platforms where users work and interact with other users [34], such as Twitter, Telegram, and StackOverflow. In the software engineering domain, bots support social and technical activities, including communication and decision-making [60]. Open-source communities have been adopting bots to automate a variety of repetitive tasks, including repairing bugs [42, 64], refactoring source code [69],fi xing static analysis violations [11, 54], suggesting code improvements [52], and predicting defects [29]. On the GitHub platform, bots have user profiles to interact with the developers, executing well-defined tasks [66].

Despite the widespread adoption of bots on social coding platforms, the interaction between bots and humans is still challenging [66, 68]. Analyzing the *tool-recommender-bot*, Brown and Parnin [7] report that bots still need to overcome problems such as notification overload. Mirhosseini and Parnin [41] analyzed the *greenkeeper* bot and found that maintainers were often overwhelmed by notifications and only a third of the bots' pull requests were merged into the codebase. Peng and Ma [51] studied how developers perceive and work with *mention bot* and concluded that it does not meet the diverse needs of users. For example, while project owners require simplicity and stability, contributors require transparency, and reviewers require selectivity. Results also show that developers are bothered by frequent review notifications when dealing with a heavy workload. These results are in line with the study conducted by Wessel et al. [68], which indicates that noise is a central problem. Noise affects both human communication and the development workflow by overwhelming and distracting developers. Complementing the previous literature, we provide afi rst step towards enhancing bots communication design, thereby overcoming noise.

In other contexts, researchers have proposed the use of a meta-bot to integrate and moderate the interactions of multiple bots. Sadeddin et al. [53] showed that a meta-bot could obtain product information from several shopping bots and summarize the information before presenting it to users. Previous research also investigated the user experience of single- vs. multi-bot conversational systems. In a Wizard-of-Oz study, Chaves and Gerosa [12] found that participants report more confusion in a multi-bot scenario than when using a meta-bot. The concept of the meta-bot also appears in the literature on software agents. Generalist agents are usually referred to as Super Bots or meta-bots [17] since they often combine multiple tasks and functionalities of specialist agents into a single agent. Given this preliminary evidence obtained in other domains, we hypothesize that *a meta-bot can mitigate the information overload created by other bots around pull requests.* As designing a meta-bot in this complex socio-technical environment is an open problem, we conducted a participatory design study to elicit tailored design strategies.

3 RESEARCH DESIGN

We devised a study[2] split into two phases, as depicted in Figure 1. We started by conducting a series of Design Fiction sessions with practitioners experienced with bots, aiming to explore strategies to overcome the information overload that bots can cause. In Phase II, we prototyped a set of emerging design strategies and collected feedback from practitioners. In the following subsections, we focus on the presentation of the participatory designfi ction methodology (Phase I). We describe the method and results from Phase II in Section 5.

3.1 Phase I: Research Approach

We applied Design Fiction method [5, 57], which has been broadly used in the Human-Computer Interactionfi eld [6, 20, 43]. The Design Fiction method wasfi rst defined by Sterling [57] as "*the deliberate use of diegetic prototypes to suspend disbelief about change.*" Designfi ction can be described as making use of practices such as prototyping and narrative elements to envision and explain plausible futures, while reflecting upon the present world [5, 20, 26, 35, 36, 38, 43]. Researchers have been employed this method in an

[2]The research protocol was approved by our institutional review board

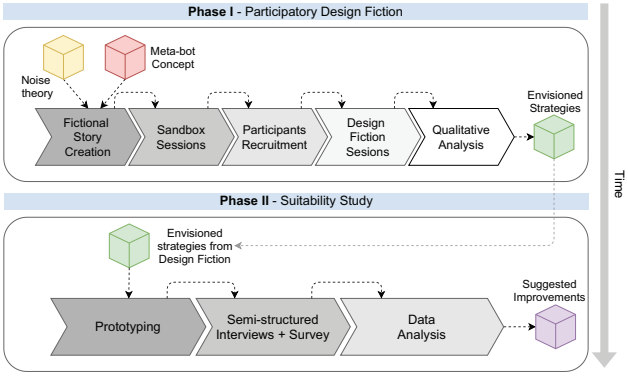

Figure 1: Overview of the Research Design

empirical way to elicit information from participants [8, 49] and communicate their insights [23, 31]. The speculative nature of this technique amplifies critical views of current social and technological developments, creating afi ctional context narrated through designed artifacts [16]. This approach facilitates exploring boundless thoughts and open discussions on a particular subject [6]. For instance, many researchers use designfi ction to anticipate issues [6], while others focus on values related to new technologies [15, 44] and anticipate users' needs [14, 20, 49].

Past studies applied Design Fiction as a participatory method to unveil design strategies for development technologies in a narrative format [8, 22, 46]. According to Muller [45], narratives in participatory work may be told by users as part of their contribution to specifying what products or services should do. Candello et al. [8], for example, applied Design Fiction to explore the expectations of science museum guides when teaching robots how to answer museum visitors questions. Candello et al. [8] crafted afi ctional story describing the dilemma of buying such robots to work as guides and participants answered questions about their expectations about these futuristic robots. Following Candello et al. [8]'s approach, we used narratives and follow-up questions to speculate on the design of the meta-bot for social coding platforms.

Design Fiction distinguishes itself in the way the designed artifacts bring context-specific meaning and social relevance [4] to the envisioned technology [30]. In this work, the use of Design Fiction enables the practitioners to envision a bot mediator and its environment, rather than focusing on the current technical limitations.

3.2 Phase I: Method

In the following, we describe thefi ctional story we used and how we conducted sandboxing, recruiting, and analysis for Phase I.

3.2.1 The Fictional Story. The story description follows the key idea raised by the noise theory of Wessel et al. [68]: information overload generated by the bots' interaction on pull requests disrupts both human communication and development workflow. The story describes the experience of an open-source maintainer who adopted bots to reduce her workload on pull request activities. After adopting a few bots, the information overload generated by the bots' noise became evident to other team members. At that point, her team brainstormed and decided to apply some countermeasures to overcome the noise. Their idea was to implement a meta-bot to act as a mediator between the existing bots and human developers. We told participants that thefi ctional story takes place approximately ten years in the future to let them be less constrained by current technological limitations.

After creating thefi ctional story, we produced a 3-minute animated video to report it to our participants in a standardized way. The story's characters are Ada, an overwhelmed open-source maintainer, and three members of her team: Ellie, John, and Anne. The *fictional story* that served as a baseline for the video creation and the video are publicly available within the supplemental material[3].

3.2.2 Sandbox Sessions. We conducted sandbox sessions with a small sample of participants to adjust thefi ctional story and the session instrument. We invited three participants who had experience contributing to and maintaining open source projects on GitHub. We asked for feedback on the 3-minute video, verifying whether the participants could capture the intended message of the fictional story. In addition, we validated the script and confirmed whether the session wouldfi t in a 1-hour time slot. The sandbox participants watched the video, answered all the questions, and provided us with feedback about thefl ow of the script. The participants suggested a few minor adjustments, which were incorporated to the instruments. We also analyzed the answers to ensure that they provided data to answer our research question. The data collected during these sandbox sessions were discarded.

3.2.3 Participants Recruitment. We recruited 32 practitioners experienced with OSS bots (contributors, maintainers, bot developers, or researchers). We employed three strategies to recruit participants. First, we leveraged our existing connections to the OSS community (n=20 participants, ≃62.5% of the sample). We also advertised the call on social media platforms frequently used by developers [1, 55, 59], including Twitter, Facebook, and Reddit (n=2, ≃6.5% of the sample). Finally, we asked participants to refer us to other qualified participants (n=10, ≃31% of the sample).

We conducted the designfi ction sessions with 32 participants—identified here as P1–P32. Table 1 shows the demographic attributes of our participants. The majority (28) are men (≃85%), while three are women (≃12%), and one is non-binary (≃3%). Participants are geographically distributed across Europe (EU, ≃44%), North America (NA, ≃34%), and South America (SA, ≃22%). Their experience with open source software development is diverse: between 4 and 5 years (≃28%), 5 and 10 (≃24%), more than 10 (≃18%), 3 years (≃15%), 1 year (≃9%), 2 years (≃3%), and zero (≃3%). When it comes to their experience with bots, 28 (≃87.5%) are experienced with bots as an open-source project maintainer, 25 (≃78.1%) as a contributor, 13 (≃40.6%) as a researcher, and 13 (≃40.6%) as a bot developer.

3.2.4 Design Fiction Sessions. We conducted a series of synchronous designfi ction sessions. The motivation behind this approach, instead of asking the participants to watch the story and write its end [8, 43], was to engage the participants and ask questions during and after the debriefing. The sessions provided thefl exibility to delve deeper into unforeseen information and enabled researchers to explore topics that emerged during the session [27]. Before each session, we shared a consent form with the participants asking for their permission to video record. We also sent our participants a

[3]https://zenodo.org/record/5428540

Table 1: Demographics of Design Fiction Participants (Y = Years of experience with OSS development; L = Location; BD = Bot developer; M = Maintainer; C = Contributor; R = Researcher)

ID	Gender	Y	L	Experienced with bots as			
				BD	M	C	R
P1	Man	5-10	EU	✓	✓	✓	✓
P2	Man	4-5	SA		✓		
P3	Man	> 10	EU		✓	✓	
P4	Woman	1	NA	✓	✓		✓
P5	Man	4-5	SA		✓	✓	
P6	Man	4-5	NA	✓	✓	✓	✓
P7	Man	5-10	EU	✓	✓		
P8	Man	4-5	NA		✓	✓	✓
P9	Man	> 10	SA	✓	✓	✓	✓
P10	Man	3	NA	✓	✓	✓	✓
P11	Non-Binary	> 10	NA		✓	✓	
P12	Man	> 10	NA	✓	✓	✓	
P13	Man	5-10	NA	✓	✓	✓	
P14	Woman	3	EU		✓		
P15	Man	4-5	SA		✓	✓	
P16	Man	> 10	NA	✓	✓	✓	
P17	Man	1	SA			✓	
P18	Man	4-5	EU		✓	✓	✓
P19	Man	3	EU		✓	✓	
P20	Man	2	SA		✓	✓	
P21	Woman	4-5	EU		✓	✓	
P22	Man	5-10	NA	✓		✓	✓
P23	Man	5-10	SA	✓	✓	✓	
P24	Man	5-10	EU		✓	✓	✓
P25	Man	> 10	NA	✓	✓	✓	
P26	Man	3	EU		✓	✓	✓
P27	Man	5-10	EU		✓	✓	✓
P28	Man	4-5	EU		✓	✓	
P29	Man	5-10	EU		✓		
P30	Woman	Zero	EU				✓
P31	Man	1	EU			✓	✓
P32	Man	3	NA	✓	✓		

short survey containing demographic questions to capture their familiarity with open-source development and bots on GitHub.

We started the sessions with a short explanation about the research objectives and guidelines, giving the participant an overview of the Design Fiction approach. The participant then watched the 3-minute *fictional story*'s video. After watching the video, we clarified questions and followed up with four scenarios to explore how they would design the meta-bot to mitigate noise. We created the scenarios based on a theory of how human developers perceive bot behaviors as noise [68]. This theory shows that bot messages are perceived as noisy by newcomers due to their lack of experience ("*Human previous experience*"); developers might be interrupted by notifications ("*Frequency and timing of actions*") – or overloaded by verbose messages ("*Verbosity*"); and bots might present bugs or create spam ("*Perform unrequested tasks*"). Thus, we used these scenarios to characterize problems we aimed to alleviate with the meta-bot. Next, we present the investigated scenarios:

Scenario One (S1) – Newcomers. This scenario describe a situation that occurs when a developer submits theirfi rst contribution to an open-source project. As soon as the newcomer submits a pull request, bots start posting their respective comments. Newcomers might perceive the bot information as noise because of their lack of experience dealing with bots' messages.

Scenario Two (S2) – Notifications' interruptions. This scenario describes when a core developer is working on a priority task and does not want to be interrupted. As described in the noise theory we drew from [68], in some cases the noise leads to a notification overload that interrupts the development workflow at the wrong time.

Scenario Three (S3) – Information overload. This scenario represents the case when bots inflate pull requests with repetitive or verbose messages. According to Wessel et al. [68], this might occur for several reasons, including decisions inherent to the bot design.

Scenario Four (S4) – Unexpected bugs or spam. Similar to the previous scenario, this scenario describes a specific case of information overload when a bot performs an unsolicited action on a pull request because of a bug or spam.

The participants acted as *storytellers*, answering questions to support the conclusion of the *fictional story*. For each scenario, we asked them to describe how they envision the meta-bot in an ideal scenario, not limited by current technology. Depending on the participants' response, we followed up with specific questions: for example, asking for more information about the the features that the participant mentioned. The detailed session script is publicly available[4]. Each session was conducted remotely by thefi rst author and lasted on average 54 minutes. The participants received a 25-dollar gift card as a token of appreciation for their time.

3.2.5Qu alitative Analysis. Each session recording was transcribed by this paper'sfi rst or second author. To qualitatively analyze the session transcripts, we applied open and axial coding procedures [63] throughout multiple rounds of analysis. We started by applying open coding, whereby we identified the envisioned features for the meta-bot or its environment. Thefi rst author of this paper conducted a preliminary analysis, identifying the main codes. More specifically, the researcher performed an iterative process of inductively coding one transcript at a time and built post-formed codes as the analysis progressed and associated them to respective parts of the transcripts. Then, thefi rst and second authors discussed the emergent codes and reached a negotiated agreement [24] in weekly hands-on meetings. During these meetings, the researchers refined the code set by checking the code names, merging codes together, or identifying a different granularity level for a code. These discussions aimed to increase the reliability of the results and mitigate bias [50, 62]. Then, the analysis was presented and discussed with the other authors. During the data analysis process, we employed a constant comparison method [25], wherein we continuously compared the emerging codes from one session with those obtained from the previous ones. Afterward, thefirst author further analyzed and revised the transcripts to identify relationships between concepts that emerged from the open coding analysis (axial coding).

We do not share the session transcripts due to confidentiality reasons. However, we made our complete code book publicly available within the supplemental material. The code book includes all code names, descriptions, and examples of quotes.

[4]https://zenodo.org/record/5428540

4 PHASE I: DESIGN FICTION FINDINGS

In this section, we present the strategies to mitigate bots' noise derived from the analysis of the participatory designfi ction sessions (see Section 3). The participants discussed several design strategies to mitigate noise created by bots, as presented in Table 2. We organized those strategies in terms of the potential features for the meta-bot and improvements for the underlining platform (GitHub). In summary, we found 22 design strategies, organized intofi ve categories: *information management* (IMi), *newcomers' assistance* (NAi), *notification management* (NMi), *spam and failures management* (SMi), and *platform support* (PSi). In the following, we present thesefi ve main categories. We describe the categories in **bold**, and provide the number of participants we assigned to each category (in parentheses).

Table 2: Envisioned Design Strategies for the Meta-bot and the GitHub Platform (# = Number of participants we assigned to each category)

Meta-bot's Design Strategies		#
Information Management	**IM1.** Summarization of bot comments	11
	IM2. Aggregating bot comments	9
	IM3. Prioritization based on tasks	9
	IM4. Prioritization based on issues	7
	IM5. Keep the most recent information	7
	IM6. Categorization of bot comments	5
	IM7. Interacting with users through natural language	4
	IM8. Internationalization	2
Newcomers' Assistance	**NA1.** Explaining rules, instructions, and requirements	8
	NA2. Welcoming message	8
	NA3. Provide information interactively	5
	NA4. Newcomers pull request notification	3
Notification Management	**NM1.** Notify through pre-specified communication channel	8
	NM2. Schedule bot notifications	7
	NM3. Notify developers in their idle times	7
	NM4. Notifying only interested developers	5
	NM5. Do not notify maintainers until the condition is satisfied	2
Spam and Failures Management	**SM1.** Prevent repetitive bot activities	8
	SM2. Spam messages notification	4
	SM3. Bugs report	2
GitHub Interface's Design Strategies		**#**
Platform Support	**PS1.** Separating bot comments	11
	PS2. Bots configuration dashboard	4

Information management. The strategy **summarization of bot comments** (11) was frequently mentioned by the participants as a way to mitigate information overload. Summaries should be concise and report an overview of the pull request status: "*Give me a context report or summary. I expect the meta-bot to be just one particular comment with some points. Just one comment with everything as a conscious report*" [P26]. However, this strategy also imposes some technical challenges when it comes to implementation. According to P31, "*it is difficult to summarize [other bot comments], because, although the message is created by a bot, it's supposedly based on a template.*" In terms of its implementation, the easiest strategy to reduce noise would be **aggregating bot comments** (9). In this specific case, the meta-bot would merge bots outputs into a single comment on a pull request. This strategy is usually mentioned in conjunction with summarizing the bots outputs: "*it could possibly summarize [bot comments] and put them in a single message*" [P6]. In both cases, the meta-bot creates a single output for all bots, however, it does not imply implementing the merging strategy always based on the summarized version of each bot output.

Another strategy concerned the order that the information is presented to the developers. Participants suggest a *prioritization* of bot outputs within the summary the meta-bot provides, such that the meta-bot has the capacity to treat some bot comments as more important than others. Basically, participants mentioned two different types of prioritization: **based on tasks** (9) and **based on issues** (7). For the prioritization based on tasks, the meta-bot would sort the most important bot comments based on the task implemented in the pull request: "*it would also be able to sort of filter out what is useful and what is not useful based on the task the developer is actually working on*" [P10]. The prioritization might also take into account the pull request problems raised by bots, sorting by the level of criticality of bot notifications, as mentioned by P21: "*if any critical problem happens, then I would like to be notified with a specific bot report, I would receive a critical notification.*" To complement prioritization,fi ve participants also suggested the **categorization of bot comments** (5). The Meta-bot would group the bot outputs based on their types (e.g., testing, security, information) before reporting in the pull request. With a categorization of bot comments, developers "*know if [they] need[] to look at [a specific bot comment] or not*" [P27].

To avoid inflating the pull requests with several comments from the meta-bot, one suggested strategy is to **keep the most recent information** (7). Participants suggested that the meta-bot creates a single comment and keeps updating it with new information from other bots: "*the meta-bot just creates one comment and keeps updating it*" [P16]. It should also keep the comment up-to-date: "*if [the developer] commit[s] again, the meta-bot updates the comment. If [the developer]fi xes Lint's errors, for example, the meta-bot will remove the warning from the comment.*" [P23]

Participants also mentioned other additional aspects of the meta-bot communication unrelated to mitigating noise. For example, participants envision the meta-bot **interacting with users through natural language** (4) by providing an interface for communicating with developers to understand their requests and answer their questions. To promote diversity and inclusion, the meta-bot can provide **Internationalization** (2) and support different languages (e.g. German).

Newcomers' assistance. The newcomers' scenario we presented to participants led them to think about how the information presented by the current bots might affect newcomers' perceptions and success. As a result, we found four main strategies that might assist newcomers, of which **explaining rules, instructions, and requirements** (8) was one of the most frequent. According to participants, the meta-bot could guide the newcomers and inform them about the project's rules and the requirements to approve the pull request. For example, P13 explains the importance of providing such explanations: "*[the newcomers] do not understand the rules yet and ... don't understand which rules are important.* " Thus, "*the meta-bot would do an excellent job for a newcomer by explaining why these rules exist*" [P13]. The meta-bot could also refer to the contribution guidelines to assist a newcomer developers'fi rst contribution, as well as include a **welcoming message** in the meta-bot's comment on newcomers' pull request (8). The meta-bot might post a comment, for example, " *'Hi, welcome! I just saw this is yourfirst contribution. Are you aware of the rules of this repository?' or 'the*

rules of this community'?" [P2]. These greetings could be used "*to let [newcomers] know that they are welcome into the community*" [P10]; however, it is also important to keep the message concise and direct: "*the message should be short. If newcomers see the bot several times in different projects, it will annoy them, and they are going to discard the whole information the bot provides.*"

Concerning the strategies to display bot information to newcomers, participants envision the meta-bot **providing information interactively** (5). As mentioned by P5, the meta-bot might guide the new contributor by showing the information from other bots "*step by step.*" Interviewees deemed this strategy a potential solution to reduce the impact of receiving several different bot notifications at once. P20 offered an exemplary case of how this strategy would apply to a real scenario: "*If the newcomer has not updated the README or the documentation, it shows 'You need to update the documentation' and waits for the newcomer to take action. When the newcomerfi xes the documentation by creating a new commit, the bot informs the documentation is now okay and then shows the next message.*" Another strategy regards **notifying maintainers about new pull requests from newcomers** (3). Since newcomers might prefer other humans to interact, a few participants also see the meta-bot as an approach to "*ping a developer to look at it [newcomers' pull request]*" [P1] aiming at "*encouraging more human activity from the maintainer*" [P6].

Notification management. Participants also reported design strategies to allow the meta-bot to control different aspects of bots' notification. First, the meta-bot should **notify developers through a pre-specified channel** (8), which means it would send the notifications wherever the developer wants to receive the notification (e.g., email, GitHub notifications). As mentioned by P1, this strategy would help the meta-bot to send the notifications to "*a channel that the developers do not ignore.*"

Participants also proposed strategies related to the frequency and timing of notifications. One approach would be to **schedule bot notifications** (7), in which the meta-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" [P8] mode. Another approach would be **notify developers in their idle times** (7). The meta-bot would not interrupt the developers during critical tasks: "*Do not notify the developer if [the meta-bot] is aware that [the developer] is currently working. That is also what other humans would not do*" [P1]. In this approach the meta-bot would learn the developers' schedule and adapt to them. Another option is **not to notify maintainers until the condition is satisfied** (2). Therefore, the meta-bot would notify the developers only when the predefined conditions are met, as stated by P16: "*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.*

To avoid overload with unrelated notifications, we found that it is also important to **notify only interested developers** (5). The meta-bot should notify only developers who are interested in monitoring activities related to a particular repository, issue, or pull request. According to P25, for example, maintainers and contributors have different needs when it comes to being notified by bots: "*if I'm a contributor, I want to know that notification about my contribution. But as a maintainer, I don't need to be reminded about every contribution that happened when I release a new version of my project.*" [P25].

Spam and failures management. We also identified three design strategies to provide control over unforeseen problems created by bot interactions. To **prevent repetitive bot activities** (8), the meta-bot bot would detect bots that are generating repetitive outcomes and prevent them from acting on pull requests; this can avoid duplicate messages and spam: "*it has the ability to control which bots comment often, then it would be easy to say 'no, you already have this comment and I see that your next comment is exactly the same' "* [P1]. There would be also a mechanism for **spam messages notification** (5), wherein the meta-bot would notify developers about repetitive bot messages that might be considered spam. And, if there is a bug with a specific bot, the meta-bot can "*contact the bot maintainers*" [P10] to provide a **bug report** (3).

Platform support. Participants envision a few modifications in the platform interface to improve its integration with bots. One potential modification is to **separate bot comments** (11) by relegating them to a space reserved for bot interactions. As stated by P32 "*developers do not like bots to come in the middle of their conversations. So, bots having their own space or their own channel would be the best [option].*" This dedicated space for bots would present the bot messages that are "*dead-ended*" [P2]: that is, the ones that do not require any response from the developer. Additionally, they suggest implementing mechanisms to collapse the bot outputs: "*then, you can collapse all messages. If you want to read a message, you have to expand it*" [P2].

The participants also proposed the implementation of a **bot configuration dashboard** (4), in which developers can customize their preferences for viewing bot interactions. This dashboard would help developers who work on several repositories to have a common interface to monitor bots' actions, as illustrated by P3 "*when [the developer] end[s] up with tons of repositories, and bots are working on it, [the developer] need some overview picture of it.*"

Research Question 1

What design strategies can potentially **reduce the noise created by bots** on pull requests?

As a result of the designfi ction methodology, we identified a series of design strategies regarding the meta-bot and its integration with the social coding platform. More specifically, participants envision strategies for information management, newcomers' assistance, notification management, spam/failure managements, and platform support.

5 PHASE II: SUITABILITY STUDY

In this phase, we aim at validating our interpretation of the strategies proposed by the designfi ction participants and evaluating the proposed implementation. To further refine the strategies elicited via the designfi ction method, we developed a prototype and collected participants' perceptions of it. Understanding the perceptions of the subject-matter experts—practitioners who face the problems

in the daily life and have large experience with software bots—supports evaluation of the suitability of the proposed solutions.

5.1 Phase II: Method

5.1.1 Prototyping. We developed a prototype to receive feedback about the most cited strategies. We applied the prototype to a scenario created from the Reakit[5] project. This project uses four bots to support the pull request review process and all bots report their outputs using comments on the pull request. Each of them is responsible for a different task:

- **CodesandBox** – provides an isolated test environment for the validation of the code modified by the pull request.
- **Compressed-size-action** – reports data referring to the difference in size offi les modified in the pull request.
- **Codecov bot** – provides code coverage metrics, offering tools for comparing reports between pull requests.
- **Reakit bot** – a project-specific bot implemented to report deploy information.

5.1.2 Implemented Prototype. To offer different views for maintainers and newcomers, we split the prototype into two different versions: the experts' pull request interface (see Figure 2), designed to support maintainers and experienced contributors; and the newcomers' pull request interface (see Figure 3). In Figure 2, we show how we mapped the strategies onto the experts' designed interface. First, we used the strategy of *separating bot comments* (PS1) to design a specific place for bots in the pull request. We created a new tab in the pull request interface (see Figure 2-A) called "Bots Conversation". This tab contains all information and events regarding bots in the pull request, including a timeline of bot events. As for bot outputs, we also disambiguate the bot participants from human participants, as shown in Figure 2-D.

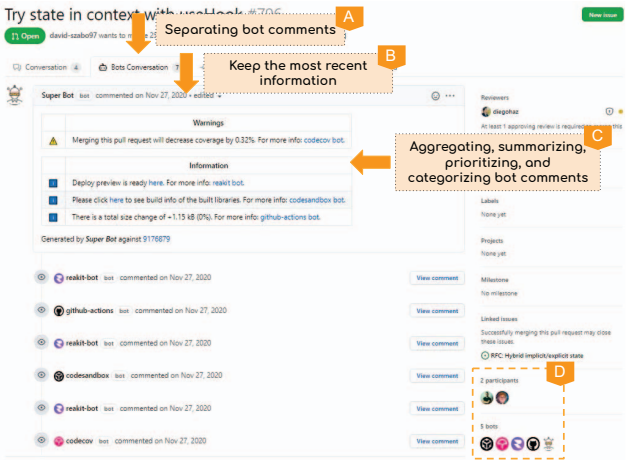

Figure 2: Experts' pull request interface

In relation to the meta-bot comments, we implemented the strategies of *aggregating* (IM2), *summarizing* (IM1), *prioritizing* (IM4), and *categorizing* (IM6) bot comments as we depicted in Figure2-C.

[5]https://github.com/reakit/reakit

First, the meta-bot *aggregates* all bots outputs in one place, and also creates a *summary* with the most important information about each one. It then groups them into *categories*, taking into consideration the *priority*. To *keep the most recent information* (IM5), we include in the summary the latest comment from each bot (Figure2-B). The Reakit bot, for example, posted three comments in the timeline of bot events; however, only one entry appears in the summarized table for that bot. In addition, in the timeline of bot events, it is possible to expand all bot comments.

We also mapped the aforementioned strategies into the newcomers' pull request interface. The differences between those two versions are related to the meta-bot message. Figure 3 highlights the designed interface for newcomers. In addition to the table summarizing the outputs, we added a text-based message to fulfill the requirement of *welcoming newcomers* (NA2), as shown in Figure 3-A. Beyond presenting a welcoming message, designfi ction participants emphasized the importance of *explaining rules, instructions, and requirements* (NA1) for contributors who are new to a project. Thus, we included a link to Reakit's contributing guidelines (see Figure 3-B).

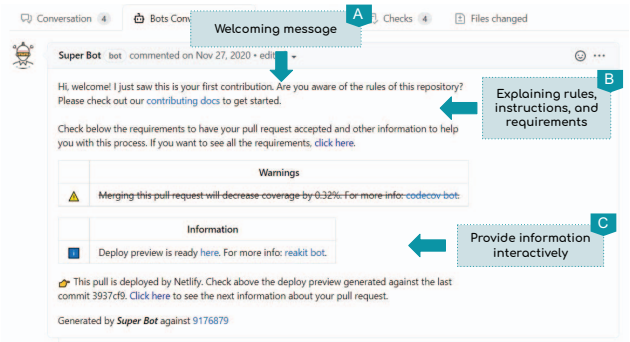

Figure 3: Newcomers' pull request interface

Another important distinction is the way the meta-bot displays the information for newcomers versus experts. In Figure 3-C, we present the interactive process of displaying bots' information. Instead of presenting the complete summary, the meta-bot presents the information *one at a time* (NA3) for newcomers. This approach aims at guiding the newcomers through the requirements for the pull request; we also provided a brief explanation of bot messages for each step. To proceed through the interactive output, the user has to click on the provided links. There is also an option to see all the meta-bot outputs at once.

5.1.3 Interviews. We reached out to our 32 participants via email, inviting them to provide feedback through an online meeting. This process is an opportunity for participants to provide their feedback on particular aspects of ourfi ndings [39], expressing their preferences about the elements of the designed prototype [28]. Fifteen participants provided their feedback: P6, P7, P8, P9, P11, P13, P14, P16, P19, P20, P23, P27, P28, P30, and P31. Each meeting lasted about 30 minutes. During the meeting, we walked them through the prototype, describing how we mapped the envisioned strategies onto the designed interface, and asked for their feedback.

After transcribing the interviews, thefi rst author coded the issues and suggested improvements to the designed interface. The interview analysis process was similar to the participatory design fiction data analysis. For each interviewee, we identified and coded each excerpt that described an issue or an improvement. All researchers met to discuss the results of the coding for each interview to reach a negotiated agreement. All interviewees provided rich feedback, although we reached information saturation after the fourth interview, i.e. after we identified no new suggested improvements to the design interface.

5.1.4 Technology Acceptance Model. To assess the participants perception about the designed interface, we also applied the Technology Acceptance Model (TAM) [18] by conducting a questionnaire immediately after concluding each interview. TAM is a model to assess the user's perception about a technology's usefulness and ease of use, thus determining a user's technology acceptance behavior. This instrument is frequently used in software engineering literature (e.g., [13, 56]).

Table 3: Scale items for measuring usefulness, ease of use and self-predicted future use

Perceived usefulness - (PU)
U1. Using the designed interface would enable me to accomplish tasks more quickly. **U2.** Using the designed interface would improve my performance. **U3.** Using the designed interface would increase my productivity. **U4.** Using the designed interface would increase my effectiveness. **U5.** Using the designed interface would make it easier to do my job. **U6.** I wouldfi nd the designed interface useful.
Perceived ease of use - (PEOU)
E1. Learning to operate the designed interface would be easy for me. **E2.** I wouldfi nd it easy to get the designed interface to do what I want it to do, to mediate the actions of other bots and present it on the pull request. **E3.** My interaction with the designed interface would be clear and understandable. **E4.** It would be easy for me to become skillful at using the designed interface. **E5.** It is easy to remember how to perform tasks using the designed interface. **E6.** I wouldfi nd the designed interface easy to use.
Self-prediction of Future Use (SPFU)
S1. Assuming the designed interface would be available, I predict that I will use it in the future. **S2.** I would prefer using the designed interface to the existing interface.

The questions are organized to measure each of the three main constructs of TAM: *perceived usefulness* (Ui); *ease of use* (Ei), and *self-predicted future use* (Si). Table 3 shows our assessment model which was adapted from previous literature [2, 18]. We used a 5-point Likert scale to measure participants' agreement with each statement, ranging from "Strongly disagree" to "Strongly agree" and including a neutral value.

5.2 Phase II: Results

In the following subsections, we describe the results from Phase II.

5.2.1 Developers' perceptions of the design strategies. The participants who gave feedback were, overall, positive about the prototype. For instance, P11, an experienced open-source maintainer, reported: "*I'm very resistant to bots; however, I liked it a lot for a couple of reasons.*" He explained that he appreciated the creation of a specific place for bots in the pull request, and the "*compressed information*" [P11] displayed by the meta-bot, since he does not "*need to open a CI page to know what happened*" [P11]. According to P30, when bot comments appear in between human comments, it is easy to miss a piece of interesting information. She stated that our approach would help to avoid that. P16 also described our modifications to the pull request interface in a positive light: "*I liked it that you have removed the restraints of what the interface looks like today and just changed them to what would be better.*"

Table 4: Suggested improvements to the prototype (# = Number of participants we assigned to each category).

Suggested Improvements	#
Include timeline references for bots	10
Quoting bot comments on the main conversation	5
Enhance newcomers bot message	4
Move summary to the main conversation	4
Interactive comments as opt-out feature	3
Replace bots tab name	2
Filtering bot interactions	1

In addition to the positive comments, we found that some design elements needed improvements, as shown in Table 4. During the analysis we could identify seven potential points of improvement reported by the participants. Next, we further explain the reasoning behind those suggestions.

Include timeline references for bots. According to the participants, one problem with having a separate tab for bot comments is the *loss of context*. Since we moved all information related to bots to the new tab, developers might lose track of which event triggered the bot action. As stated by P9, the timeline references might be implemented by including a short line in the timeline of the main conversation with a link to the respective bot comment: "*a notification like 'a bot comment has occurred here' so the user can click to switch tabs.*" To avoid noise, P9 also mentioned the creation of *grouped bot references* in the timeline to deal with cases of pull requests with more than one bot comment in a sequence: "*GitHub interface could simply merge them into one: 'there were lots of bot comments here,' since one of the goals is also to remove the noise.*"

Quoting bot comments on the main conversation. Also related to the *loss of context* due to the creation of the bot tab, four interviewees suggested the possibility of *quoting bot comments on human conversation*. Participants mentioned that in some cases a bot comment might trigger a discussion in the human conversation tab. Therefore, it is important to refer to the bot comment, and enable the possibility of including a bot quotation within the human comment.

Enhance newcomers bot message. Participants also suggested a few adjustments in the message the meta-bot shows to newcomers. As cited by P9 and P27, the interactivity we implemented in the comments using a link is not explicit. P9 suggested that replacing the link with a button would be a better option. For P27, an even better option would be showing all steps hidden by default and providing an easy way to expand and collapse them to remove the need to click on links or buttons. In addition, they suggested including more visual clues in the table and in the text to call the user's attention to important points. For example, it is possible to

"*reuse the icons of the bots a little bit and kind of show visually from which bot is the warning coming from*" [P16].

Interactive comments as opt-out feature. According to the interviewees, choosing between the interactive or static versions of the meta-bot message might depend on personal preferences. Therefore, they recommended including an *interactive version of meta-bot summary* as an opt-out feature, as highlighted by P30: "*even if the person is new to the repository, maybe [she] is a contributor who is very used to contributing to other repositories. So then it's good that you can opt-out.*"

Move summary to the main conversation. Another problem that might occur when separating bot comments is that contributors, especially newcomers, might be unaware of the presence of bots on a pull request. To overcome this problem, interviewees proposed *moving the summary provided by the meta-bot to the main conversation.* P27 suggested that the meta-bot should appear in the human conversation "*like a side panel. Then, the summary can always be visible. And all the detailed information could be in the bot conversation.*"

Replace bots tab name. Although less recurrent, two participants recommended *replacing the name of the bots' tab.* As explained by P11, the term "bots conversation" implies a dialog between bots, which is not the case of these bots. For P27, the designed bots' tab "*is more like history.*" They suggested terms such as "bots history", "bots reports", or any other name that includes "automated."

Filtering bot interactions. Still related to the bots tab, P16 suggested offering an option tofi lter out the interactions in the bots' timeline. First, they would like to have access to interface elements that allow them to selectively show interactions of a single bot, for example. It might be helpful if a bot posted multiple comments in the bots' timeline, reducing the workload of searching for them. P16 mentioned that "*if there are multiple comments from Reakit bot, for example, then I would like to see a thread only with chronological comments. Then, I can follow only this bot, and I don't need to go through it manually.*"

5.2.2 Perceived usefulness, ease of use, and potential future use. In the following, we present the results for the TAM questionnaire in terms of the designed interface's perceived usefulness, ease of use, and prediction of future use. As a measure of consistency, we checked the questionnaire items' reliability. A precise, reliable, and valid instrument ensures collection of accurate information. Therefore, we conducted the reliability analysis to ensure the internal validity and consistency of the items used for each factor, using Cronbach's Alpha [3]. Carmines and Zeller [10] suggest that a Cronbach's Alpha reliability level that exceeds a minimum of 0.70 indicates a reliable measure. According to the results, the Alpha values exceeded the threshold, with 0.84 and 0.72 for usefulness and ease of use items, respectively.

Usefulness of the Designed Interface. Most participants found the designed interface useful. We present each item's detailed results in Figure 4. None of the participants disagreed with any item related to the usefulness of the designed interface—all items had more than 50% of agreement or strong agreement. In particular, quickness

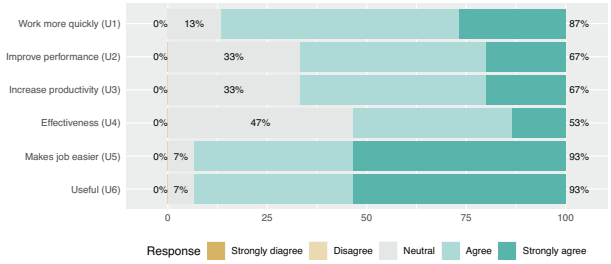

Figure 4: Responses to the 5-point Likert-scale items for Perceived Usefulness

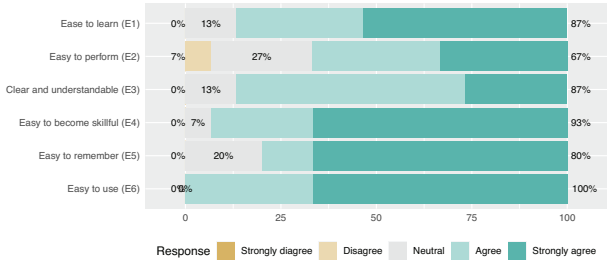

Figure 5: Responses to the 5-point Likert-scale items for Perceived Ease of Use

(U1), easier job (U5), and usefulness (U6) had more than 85% of agreement or strong agreement.

Ease of Use of the Designed Interface. In Figure 5, we can observe the answers' distribution per item related to the ease of use. More than 67% of the participants agreed or strongly agreed with the items. In addition, all participants agreed that the designed interface is easy to use (E6). Only one participant disagreed with the designed interface's ease for performing his desired tasks (E2). In section 5.2.1, we highlighted the suggestions to improve the design interface, which are likely to affect the ease of use positively.

Self-predicted Future Use. Figure 6 reports self-predicted future use of the designed interface. We observe that 14 (93%) participants agreed or strongly agreed that if the designed interface were available in the future, they would use it (S1). Compared to the current approach employed by GitHub, a large number of participants (13) agreed with a preference for the designed interface. Only one participant disagreed, i.e., he preferred the traditional interface.

Research Question 2

How do **participants perceive** the design strategies to reduce the noise created by bots on pull requests?

We found seven potential improvements for our designed interface. Participants perceived the designed interface as a useful and easy to use interface, and would potentially use it in the future, indicating the suitability of the design strategies.

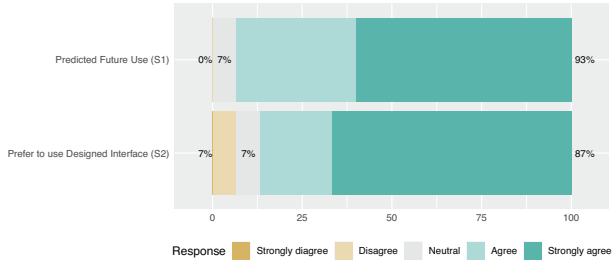

Figure 6: Responses to the 5-point Likert-scale items for Self-predicted Future Use

6 DISCUSSION

Identifying design strategies to reduce the noise created by bots on pull requests is relevant since developers often complain about the information overload caused by repetitive bot behavior on pull requests [7, 21, 41, 51, 68]. Employing Design Fiction as a method to prototype a technology [32], we gained insights to refine the design of a meta-bot and the underlining platform, taking into account the perceptions of practitioners experienced with bots on social coding platforms.

According to our participants, the meta-bot should act as a gatekeeper: a layer between other bots and the users. As a gatekeeper, the meta-bot helps to mitigate information overload by curating and presenting in a structured way the other bots' outputs. By reducing the cognitive effort to process incoming information [40], concise and well-organized information might help developers to leverage bots outputs.

In line with Erlenhov et al. [21]'s results, our study indicates that a combination of three different characteristics appears to be relevant for the meta-bot: intelligence, adaptability, and autonomy. However, intelligence and adaptability are not yet widely present on bots that work on GitHub [66]. Our participants mentioned several strategies for the meta-bot that rely on learning from past experiences and adapting its behavior. One example is the ability to notify developers only on their idle times. To do so, the meta-bot must be smart enough to learn developers' preferences and adapt accordingly. Making smarter decisions (e.g., notifying developers on their idle times) would require bots to be enriched with learning models for the target context. This topic was also explored in other domains. For example, some bots in the educationfield learn from previous interactions and estimate students' interest level [47] or learning styles [33], adapting their interactions to improve collaboration. Similar models could be used in open-source development.

In the following, we discuss how our results lead to practical implications for practitioners and insights and suggestions for researchers.

Implications for Bot Developers: Our study results provide insights for bot developers who want to mitigate noise, laying a foundation for designing better bots. For example, ourfindings indicate the OSS developers would like to customize aspects of the bot interaction (e.g., notifications frequency and timing). Therefore, it is important for bot developers to design a highly customized bot, providing project maintainers control over bot actions. In addition, our research can also help bot designers by providing guidelines and insights to support the design of bot messages. Instead of providing the information aggregated, bot developers should consider other possibilities, such as customizing the message or providing the information interactively. Applying one of those strategies might help developers deal with and interpret the information from bots.

Implications for Researchers: Our results can serve as a reference to guide further research. For example, we found several strategies to present the bot information to developers (e.g., summarization, categorization, prioritizing, interactively). Additional effort is still necessary to investigate how these strategies might influence the way developers interpret the bot comments' content. How developers think, perceive, and remember information (i.e. their cognitive style) is likely to affect how they handle bot messages and learn from them [65]. Future research can further investigate these differences and inform a set of guidelines on how to design effective messages for different developer profiles. Further, our work can inspire researchers to use designfi ction, a method still rarely used in software engineering studies but that has been shown to be effective in other domains.

Implications for Social Coding Platforms: The preliminary implementation of the meta-bot revealed some limitations imposed by the GitHub platform that restrict the design of bots. Wessel et al. [68] already mentioned some examples of those technical challenges in their hierarchical categorization of bot problems. In short, the platform restrictions might limit both the extent of bot actions and the way bots are allowed to communicate. It is essential to provide a morefl exible way for bots to interact on the platform. In addition, 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 of separating bot events by designing a new tab in the pull request interface; this idea can be leveraged to reshape the interface and better accommodate bot interactions.

7 LIMITATIONS

In this section, we discuss the potential threats to the validity of ourfi ndings and how we addressed or mitigated them.

Generalizability of the results. Since we recruited practitioners experienced with bots on the GitHub platform, ourfindings may not necessarily apply to other social coding platforms, such as GitLab and Bitbucket. Although we do not anticipate big differences in these platforms, additional research is necessary to investigate the transferability of the results.

Data representativeness. Although we conducted the participatory designfi ction with a substantial number of practitioners, we likely did not discover all possible strategies or provide full explanations of the strategies. We are aware that each bot as well as each project has its singularities and that the open-source universe is expansive. Our strategies to keep collecting data until reaching information saturation and to consider different practitioner profiles and identify recurrent mentions of design strategies from multiple perspectives aimed to alleviate this issue. Anyway, our

findings reflect the perspective of practitioners experienced with bots. Therefore, we acknowledge that additional research is necessary to consider the perspective of those who do not have any experience with bots on social coding platforms.

Applicability of the results. The strategies presented in this paper came from participats' insights, hence, additional research is necessary to test the strategies in practice. Moreover, conducted both phase I and II with the same participants. However, the participants were exposed to strategies proposed by the other participants, providing an opportunity for them to provide feedback expressing their preferences about the elements of the designed prototype and for all strategies.

Information saturation. We continued recruiting participants and conducting interviews until we came to an agreement that no new significant information was found. As posed by Strauss and Corbin [61], sampling may be discontinued once the collected data is considered sufficiently dense and data collection no longer generates new information. As previously mentioned, we also made sure to interview different groups with different perspectives on bots before deciding whether saturation had been reached. In particular, we interviewed researchers, bot developers, and developers who are contributors and/or maintainers of open-source projects.

Reliability of results. To improve the reliability of ourfindings, we employed a constant comparison method [25]. In this method, each interpretation is constantly compared with existingfindings as it emerges from the qualitative analysis. In addition, we also developed a prototype and collected feedback from the participants. To check the reliability of the TAM instrument, we performed a reliability check on the questionnaire items. Additionally, to direct data collected, we carefully designed a 3-minute animated video and guided participants through four scenarios as a starting point for thinking about the future, constantly reminding them that they were not constrained by current technological limitations.

8 CONCLUSION

In this paper, we took thefi rst steps toward overcoming information overload created by bots. By capturing the expectations of maintainers, contributors, bot developers, and experienced researchers, we elicited design strategies for the creation of a meta-bot. We presented participants with afi ctional story of a meta-bot capable of better supporting developers' interactions on pull requests and operating as a mediator between developers and the existing bots. Participants answered questions to complete the end of thefictional story, raising concerns around the use of bots and discussing the design strategies to mitigate noise.

Grounded in participatory designfi ction, we used the emerged design strategies to implement a prototype of the meta-bot. Participants perceived the prototype as a useful and ease-to-use tool to overcome noise, and indicated a potential future use of the designed interface. Compared to the previous literature, thesefindings provide a comprehensive understanding and exploration of design ideas to enhance the integration between bots, humans, and social coding platforms.

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; the Dutch science foundation NWO through the Vici "TestShift" project (No. VI.C.182.032); the Emerging Leaders in the Americas Program (ELAP); and UTFPR-Campo Mourão. We also thank the Open Source developers and researchers who spent their time to participate in our research.

REFERENCES

[1] Mauricio Aniche, Christoph Treude, Igor Steinmacher, Igor Wiese, Gustavo Pinto, Margaret-Anne Storey, and Marco Aurelio Gerosa. 2018. How modern news aggregators help development communities shape and share knowledge. In *ICSE'18*. 499–510.

[2] Muhammad Ali Babar, Dietmar Winkler, and Stefan Biffl. 2007. Evaluating the usefulness and ease of use of a groupware tool for the software architecture evaluation process. In *First International Symposium on Empirical Software Engineering and Measurement (ESEM 2007)*. IEEE, 430–439.

[3] J Martin Bland and Douglas G Altman. 1997. Statistics notes: Cronbach's alpha. *Bmj* 314, 7080 (1997), 572.

[4] Julian C Bleecker. 2004. *The reality effect of technoscience*. University of California, Santa Cruz.

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

[6] Mark Blythe and Enrique Encinas. 2016. The co-ordinates of designfiction: extrapolation, irony, ambiguity and magic. In *Proceedings of the 19th international conference on supporting group work*. ACM, 345–354.

[7] 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)*.

[8] Heloisa Candello, Mauro Pichiliani, Mairieli Wessel, Claudio Pinhanez, and Michael Muller. 2019. Teaching Robots to Act and Converse in Physical Spaces: Participatory Design Fictions with Museum Guides. In *Proceedings of the Halfway to the Future Symposium 2019*. ACM, 15.

[9] Heloisa Candello, Marisa Vasconcelos, and Claudio Pinhanez. 2017. Evaluating the conversationfl ow and content quality of a multi-bot conversational system. In *Proceedings of the 2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems*.

[10] E.G. Carmines and R.A. Zeller. 1979. *Reliability and Validity Assessment*. Number no. 17 in 07. SAGE Publications. http://books.google.com.br/books?id=BN_MMD9BHogC

[11] 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.

[12] 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.

[13] Hui Hui Chen, Ming Che Lee, Yun Lin Wu, Jing Yao Qiu, Cheng He Lin, Hong Yong Tang, and Ching Hui Chen. 2012. An analysis of moodle in engineering education: The TAM perspective. In *Proceedings of IEEE International Conference on Teaching, Assessment, and Learning for Engineering (TALE) 2012*. H1C–1–H1C–5. https://doi.org/10.1109/TALE.2012.6360324

[14] 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

[15] EunJeong Cheon and Norman Makoto Su. 2018. Futuristic Autobiographies: Weaving Participant Narratives to Elicit Values Around Robots. In *Proceedings of the 2018 ACM/IEEE International Conference on Human-Robot Interaction* (Chicago, IL, USA) *(HRI '18)*. ACM, New York, NY, USA, 388–397. https://doi.org/10.1145/3171221.3171244

[16] Paul Coulton, Joseph Lindley, Miriam Sturdee, and Mike Stead. 2017. Design Fiction as World Building. In *Proceedings of Research Through Design Conference 2017*.

[17] 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.

[18] Fred D Davis. 1989. Perceived usefulness, perceived ease of use, and user acceptance of information technology. *MIS quarterly* (1989), 319–340.

[19] Tapajit Dey, Sara Mousavi, Eduardo Ponce, Tanner Fry, Bogdan Vasilescu, Anna Filippova, and Audris Mockus. 2020. Detecting and Characterizing Bots That Commit Code. In *Proceedings of the 17th International Conference on Mining Software Repositories* (Seoul, Republic of Korea) *(MSR '20)*. Association for Computing Machinery, New York, NY, USA, 209–219. https://doi.org/10.1145/3379597.3387478

[20] Enrique Encinas and Mark Blythe. 2016. The solution printer: magic realist design fiction. In *Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems*. ACM, 387–396.

[21] Linda Erlenhov, Francisco Gomes de Oliveira Neto, and Philipp Leitner. 2016. An Empirical Study of Bots in Software Development–Characteristics and Challenges from a Practitioner's Perspective. In *Proceedings of the 2020 28th ACM SIGSOFT International Symposium on Foundations of Software Engineering (FSE 2020)*.

[22] Laura Forlano and Anijo Mathew. 2014. From designfi ction to design friction: Speculative and participatory design of values-embedded urban technology. *Journal of Urban Technology* 21, 4 (2014), 7–24.

[23] Jonas Fritsch, Morten Breinbjerg, and Ditte Amund Basballe. 2013. Ekkomaten—exploring the echo as a designfi ction concept. *Digital Creativity* 24, 1 (2013), 60–74.

[24] D Randy Garrison, Martha Cleveland-Innes, Marguerite Koole, and James Kappelman. 2006. Revisiting methodological issues in transcript analysis: Negotiated coding and reliability. *The internet and higher education* 9, 1 (2006), 1–8.

[25] Barney G Glaser and Anselm L Strauss. 2017. *Discovery of grounded theory: Strategies for qualitative research*. Routledge.

[26] Ellie Harmon, Chris Bopp, and Amy Voida. 2017. The Design Fictions of Philanthropic IT: Stuck Between an Imperfect Present and an Impossible Future. 7015–7028. https://doi.org/10.1145/3025453.3025650

[27] Siw Elisabeth Hove and Bente Anda. 2005. Experiences from conducting semi-structured interviews in empirical software engineering research. In *11th IEEE International Software Metrics Symposium (METRICS'05)*. IEEE, 10–pp.

[28] Alvin D Jeffery, Laurie L Novak, Betsy Kennedy, Mary S Dietrich, and Lorraine C Mion. 2017. Participatory design of probability-based decision support tools for in-hospital nurses. *Journal of the American Medical Informatics Association* 24, 6 (2017), 1102–1110.

[29] Chaiyakarn Khanan, Worawit Luewichana, Krissakorn Pruktharathikoon, Jirayus Jiarpakdee, Chakkrit Tantithamthavorn, Morakot Choetkiertikul, Chaiyong Ragkhitwetsagul, and Thanwadee Sunetnanta. 2020. JITBot: An Explainable Just-in-Time Defect Prediction Bot. In *Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering* (Virtual Event, Australia) *(ASE '20)*. Association for Computing Machinery, New York, NY, USA, 1336–1339. https://doi.org/10.1145/3324884.3415295

[30] David Kirby. 2010. The future is now: Diegetic prototypes and the role of popular films in generating real-world technological development. *Social Studies of Science* 40, 1 (2010), 41–70.

[31] Ben Kirman, Conor Linehan, Shaun Lawson, and Dan O'Hara. 2013. CHI and the future robot enslavement of humankind: a retrospective. In *CHI'13 Extended Abstracts on Human Factors in Computing Systems*. 2199–2208.

[32] Eva Knutz and Thomas Markussen. 2014. The role offi ction in experiments within design, art & architecture-towards a new typology of designfiction. *Artifact: Journal of Design Practice* 3, 2 (2014), 8–1.

[33] Annabel M Latham, Keeley A Crockett, David A McLean, Bruce Edmonds, and Karen O'Shea. 2010. Oscar: An intelligent conversational agent tutor to estimate learning styles. In *International Conference on Fuzzy Systems*. IEEE, Washington, DC, USA, 1–8.

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

[35] Joseph Lindley, Paul Coulton, and Emmett L Brown. 2016. Peer Review and Design Fiction: "Honestly, they're not just made up.". *CHI Extended Abstracts (Alt. CHI). ACM* (2016).

[36] Conor Linehan, Ben J Kirman, Stuart Reeves, Mark A Blythe, Joshua G Tanenbaum, Audrey Desjardins, and Ron Wakkary. 2014. Alternate endings: using fiction to explore design futures. In *CHI'14 Extended Abstracts on Human Factors in Computing Systems*. ACM, 45–48.

[37] Dongyu Liu, Micah J. Smith, and Kalyan Veeramachaneni. 2020. Understanding User-Bot Interactions for Small-Scale Automation in Open-Source Development. In *Extended Abstracts of the 2020 CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) *(CHI EA '20)*. Association for Computing Machinery, New York, NY, USA, 1–8. https://doi.org/10.1145/3334480.3382998

[38] Ellen Lupton. 2017. *Design is storytelling*.

[39] Sharan B Merriam. 1998. *Qualitative Research and Case Study Applications in Education. Revised and Expanded from" Case Study Research in Education."*. ERIC.

[40] George A Miller. 1956. The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological review* 63, 2 (1956), 81.

[41] 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

[42] 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

[43] Michael Muller and Thomas Erickson. 2018. In the Data Kitchen: A Review (a Design Fiction on Data Science). In *Extended Abstracts of the 2018 CHI Conference on Human Factors in Computing Systems* (Montreal QC, Canada) *(CHI EA '18)*. ACM, New York, NY, USA, Article alt14, 10 pages. https://doi.org/10.1145/3170427.3188407

[44] Michael Muller and Q Vera Liao. 2017. Exploring AI Ethics and Values through Participatory Design Fictions. *Human Computer Interaction Consortium* (2017). https://www.slideshare.net/traincroft/hcic-muller-and-liao-participatory-design-fictions-77345391

[45] Michael J Muller. 2007. Participatory design: the third space in HCI. In *The human-computer interaction handbook*. CRC press, 1087–1108.

[46] Larissa Vivian Nägele, Merja Ryöppy, and Danielle Wilde. 2018. PDFi: participatory designfi ction with vulnerable users. In *Proceedings of the 10th Nordic Conference on Human-Computer Interaction*. 819–831.

[47] Kazuaki Nakamura, Koh Kakusho, Tetsuo Shoji, and Michihiko Minoh. 2012. Investigation of a Method to Estimate Learners' Interest Level for Agent-based Conversational e-Learning. In *International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems*. Springer, Berlin, Heidelberg, 425–433.

[48] Azadeh Nematzadeh, Giovanni Luca Ciampaglia, Yong-Yeol Ahn, and Alessandro Flammini. 2016. Information overload in group communication: From conversation to cacophony in the twitch chat. *Royal Society open science* 6, 10 (2016), 191412.

[49] Renee Noortman, Britta F. Schulte, Paul Marshall, Saskia Bakker, and Anna L. Cox. 2019. HawkEye - Deploying a Design Fiction Probe. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (Glasgow, Scotland Uk) *(CHI '19)*. ACM, New York, NY, USA, Article 422, 14 pages. https://doi.org/10.1145/3290605.3300652

[50] Michael Quinn Patton. 2014. *Qualitative research & evaluation methods: Integrating theory and practice*. Sage publications.

[51] 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

[52] Purit Phan-udom, Naruedon Wattanakul, Tattiya Sakulniwat, Chaiyong Ragkhitwetsagul, Thanwadee Sunetnanta, Morakot Choetkiertikul, and Raula Gaikovina Kula. 2020. Teddy: Automatic Recommendation of Pythonic Idiom Usage For Pull-Based Software Projects. In *2020 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, 806–809.

[53] 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.

[54] Dragos Serban, Bart Golsteijn, Ralph Holdorp, and Alexander Serebrenik. 2021. SAW-BOT: Proposing Fixes for Static Analysis Warnings with GitHub Suggestions. In *Workshop on Bots in Software Engineering*. IEEE Computer Society, United States.

[55] Leif Singer, Fernando Figueira Filho, and Margaret-Anne Storey. 2014. Software engineering at the speed of light: how developers stay current using Twitter. In *36th ICSE*. 211–221.

[56] 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)*.

[57] Bruce Sterling. 2009. Cover Story Designfiction. *interactions* 16, 3 (2009), 20–24.

[58] 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.

[59] Margaret-Anne Storey, Christoph Treude, Arie van Deursen, and Li-Te Cheng. 2010. The impact of social media on software engineering practices and tools. In *FSE/SDP workshop on Future of Softw Eng Research*. 359–364.

[60] 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

[61] Anselm Strauss and Juliet M Corbin. 1997. *Grounded theory in practice*. Sage.

[62] Anselm Strauss and Juliet M. Corbin. 2007. *Basics of Qualitative Research : Techniques and Procedures for Developing Grounded Theory* (3rd ed.). SAGE Publications.

[63] ANSELM L Strauss and JM Corbin. 1998. Basics of qualitative research: Techniques and procedures for developing grounded theory Sage Publications. *SAGE Publications* (1998).

[64] Simon Urli, Zhongxing Yu, Lionel Seinturier, and Martin Monperrus. 2018. How to Design a Program Repair Bot?: Insights from the Repairnator Project. In *Proceedings of the 40th International Conference on Software Engineering: Software*

Engineering in Practice (Gothenburg, Sweden) *(ICSE-SEIP '18)*. ACM, New York, NY, USA, 95–104. https://doi.org/10.1145/3183519.3183540

[65] Mihaela Vorvoreanu, Lingyi Zhang, Yun-Han Huang, Claudia Hilderbrand, Zoe Steine-Hanson, and Margaret Burnett. 2019. From Gender Biases to Gender-Inclusive Design: An Empirical Investigation. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (Glasgow, Scotland Uk) *(CHI '19)*. Association for Computing Machinery, New York, NY, USA, 1–14. https://doi.org/10.1145/3290605.3300283

[66] 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

[67] Mairieli Wessel, Alexander Serebrenik, Igor Scaliante Wiese, Igor Steinmacher, and Marco Aurelio Gerosa. 2020. Effects of Adopting Code Review Bots on Pull Requests to OSS Projects. In *IEEE International Conference on Software Maintenance and Evolution*. IEEE Computer Society.

[68] Mairieli Wessel, Igor Wiese, Igor Steinmacher, and Marco A. Gerosa. 2021. Don't Disturb Me: Challenges of Interacting with Software Bots on Open Source Software Projects. *Procidings of ACM Human-Computer Interaction* CSCW (2021).

[69] 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